# Package 'poolr'

May 7, 2025

**Version** 1.2-0

**Date** 2025-05-01

**Title** Methods for Pooling P-Values from (Dependent) Tests

**Depends** R (>= 3.5.0)

**Imports** methods, stats, utils, mathjaxr

**Suggests** testthat

**RdMacros** mathjaxr

**Description**

Functions for pooling/combining the results (i.e., p-values) from (dependent) hypothesis tests. Included are Fisher's method, Stouffer's method, the inverse chi-square method, the Bonferroni method, Tippett's method, and the binomial test. Each method can be adjusted based on an estimate of the effective number of tests or using empirically derived null distribution using pseudo replicates. For Fisher's, Stouffer's, and the inverse chi-square method, direct generalizations based on multivariate theory are also available (leading to Brown's method, Strube's method, and the generalized inverse chi-square method). An introduction can be found in Cinar and Viechtbauer (2022) <doi:10.18637/jss.v101.i01>.

**License** GPL (>=2)

**ByteCompile** TRUE

**LazyData** TRUE

**Encoding** UTF-8

**BuildManual** TRUE

## Contents

---

`poolr-package` *Methods for Pooling P-Values from (Dependent) Tests*

---

## Description

The **poolr** package contains functions for pooling/combining the results (i.e., $p$-values) from (dependent) hypothesis tests. Included are Fisher's method, Stouffer's method, the inverse chi-square method, the Bonferroni method, Tippett's method, and the binomial test. Each method can be adjusted based on an estimate of the effective number of tests or using empirically-derived null distribution using pseudo replicates. For Fisher's, Stouffer's, and the inverse chi-square method, direct generalizations based on multivariate theory are also available (leading to Brown's method, Strube's method, and the generalized inverse chi-square method). For more details, see:

- `fisher`: for Fisher's method (and Brown's method)
- `stouffer`: for Stouffer's method (and Strube's method)
- `invchisq`: for the inverse chi-square method
- `bonferroni`: for the Bonferroni method
- `tippett`: for Tippett's method
- `binomtest`: for the binomial test

Note that you can also read the documentation of the package online at `https://ozancinar.github.io/poolr/` (where it is nicely formatted and the output from all examples is provided).

## Author(s)

Ozan Cinar <ozancinar86@gmail.com>
Wolfgang Viechtbauer <wvb@wvbauer.com>

## References

Brown, M. B. (1975). 400: A method for combining non-independent, one-sided tests of significance. *Biometrics, 31*(4), 987–992. `https://doi.org/10.2307/2529826`

Cinar, O. & Viechtbauer, W. (2022). The poolr package for combining independent and dependent p values. *Journal of Statistical Software*, **101**(1), 1–42. `https://doi.org/10.18637/jss.v101.i01`

Fisher, R. A. (1932). *Statistical Methods for Research Workers* (4th ed.). Edinburgh: Oliver and Boyd.

Lancaster, H. O. (1961). The combination of probabilities: An application of orthonormal functions. *Australian Journal of Statistics, 3*(1), 20–33. https://doi.org/10.1111/j.1467-842X.1961.tb00058.x

Strube, M. J. (1985). Combining and comparing significance levels from nonindependent hypothesis tests. *Psychological Bulletin, 97*(2), 334–341. https://doi.org/10.1037/0033-2909.97.2.334

Tippett, L. H. C. (1931). *Methods of Statistics*. London: Williams Norgate.

Wilkinson, B. (1951). A statistical consideration in psychological research. *Psychological Bulletin, 48*(2), 156–158. https://doi.org/10.1037/h0059111

---

`binomtest` *Binomial Test*

---

## Description

Function to carry out the binomial test.

## Usage

```
binomtest(p, adjust = "none", R, m,
         size = 10000, threshold, side = 2, batchsize, nearpd = TRUE, ...)
```

## Arguments

| | |
|---|---|
| `p` | vector of length $k$ with the (one- or two-sided) p-values to be combined. |
| `adjust` | character string to specify an adjustment method to account for dependence. The default is `"none"`, in which case no adjustment is applied. Methods `"nyholt"`, `"liji"`, `"gao"`, `"galwey"`, or `"chen"` are adjustments based on an estimate of the effective number of tests (see `meff`). Adjustment method `"empirical"` uses an empirically-derived null distribution using pseudo replicates. See 'Details'. |
| `R` | a $k \times k$ symmetric matrix that reflects the dependence structure among the tests. Must be specified if `adjust` is set to something other than `"none"`. See 'Details'. |
| `m` | optional scalar (between 1 and $k$) to manually specify the effective number of tests (instead of estimating it via one of the methods described above). |
| `size` | size of the empirically-derived null distribution. Can also be a numeric vector of sizes, in which case a stepwise algorithm is used. This (and the following arguments) are only relevant when `adjust = "empirical"`. |
| `threshold` | numeric vector to specify the significance thresholds for the stepwise algorithm (only relevant when `size` is a vector). |
| `side` | scalar to specify the sidedness of the $p$-values that are used to simulate the null distribution (2, by default, for two-sided tests; 1 for one-sided tests). |
| `batchsize` | optional scalar to specify the batch size for generating the null distribution. When unspecified (the default), this is done in a single batch. |
| `nearpd` | logical indicating if a negative definite `R` matrix should be turned into the nearest positive definite matrix (only relevant when `adjust = "empirical"`). |
| `...` | other arguments. |

## Details

### Binomial Test

By default (i.e., when `adjust = "none"`), the function applies the binomial test to the $p$-values (Wilkinson, 1951). Letting $p_1, p_2, \ldots, p_k$ denote the individual (one- or two-sided) $p$-values of the $k$ hypothesis tests to be combined, the combined $p$-value is then computed with

$$p_c = \sum_{x=r}^{k} \binom{k}{x} \alpha^x (1-\alpha)^{k-x}$$

where

$$r = \sum_{i=1}^{k} I(p_i \leq \alpha)$$

denotes the number of hypothesis tests that are significant at $\alpha$.

The binomial test assumes that the $p$-values to be combined are independent. If this is not the case, the method can either be conservative (not reject often enough) or liberal (reject too often), depending on the dependence structure among the tests. In this case, one can adjust the method to account for such dependence (to bring the Type I error rate closer to some desired nominal significance level).

### Adjustment Based on the Effective Number of Tests

When `adjust` is set to `"nyholt"`, `"liji"`, `"gao"`, `"galwey"`, or `"chen"`, the binomial test is adjusted based on an estimate of the effective number of tests (see `meff` for details on these methods for estimating the effective number of tests). In this case, argument `R` needs to be set to a matrix that reflects the dependence structure among the tests.

There is no general solution for constructing such a matrix, as this depends on the type of test that generated the $p$-values and the sidedness of these tests. If the $p$-values are obtained from tests whose test statistics can be assumed to follow a multivariate normal distribution and a matrix is available that reflects the correlations among the test statistics, then the `mvnconv` function can be used to convert this correlation matrix into the correlations among the (one- or two-sided) $p$-values, which can then be passed to the `R` argument. See 'Examples'.

Once the effective number of tests, $m$, is estimated based on `R` using one of the methods described above, the combined $p$-value is then computed with

$$p_c = \sum_{x=\tilde{r}}^{m} \binom{m}{x} \alpha^x (1-\alpha)^{m-x}$$

where

$$\tilde{r} = \lfloor r \times m/k \rfloor$$

and $\lfloor \cdot \rfloor$ is the floor function.

Alternatively, one can also directly specify the effective number of tests via the `m` argument (e.g., if some other method not implemented in the **poolr** package is used to estimate the effective number of tests). Argument `R` is then irrelevant and doesn't need to be specified.

### Adjustment Based on an Empirically-Derived Null Distribution

When `adjust = "empirical"`, the combined $p$-value is computed based on an empirically-derived null distribution using pseudo replicates (using the `empirical` function). This is appropriate if the

test statistics that generated the $p$-values to be combined can be assumed to follow a multivariate normal distribution and a matrix is available that reflects the correlations among the test statistics (which is specified via the R argument). In this case, test statistics are repeatedly simulated from a multivariate normal distribution under the joint null hypothesis, converted into one- or two-sided $p$-values (depending on the side argument), and the binomial test is applied. Repeating this process size times yields a null distribution based on which the combined $p$-value can be computed, or more precisely, estimated, since repeated applications of this method will yield (slightly) different results. To obtain a stable estimate of the combined $p$-value, size should be set to a large value (the default is 10000, but this can be increased for a more precise estimate). If we consider the combined $p$-value an estimate of the 'true' combined $p$-value that would be obtained for a null distribution of infinite size, we can also construct a 95% (pseudo) confidence interval based on a binomial distribution.

If batchsize is unspecified, the null distribution is simulated in a single batch, which requires temporarily storing a matrix with dimensions [size,k]. When size*k is large, allocating the memory for this matrix might not be possible. Instead, one can specify a batchsize value, in which case a matrix with dimensions [batchsize,k] is repeatedly simulated until the desired size of the null distribution has been obtained.

One can also specify a vector for the size argument, in which case one must also specify a corresponding vector for the threshold argument. In that case, a stepwise algorithm is used that proceeds as follows. For j = 1, ..., length(size),

1. estimate the combined $p$-value based on size[j]
2. if the combined $p$-value is $\geq$ than threshold[j], stop (and report the combined $p$-value), otherwise go back to 1.

By setting size to increasing values (e.g., size = c(1000, 10000, 100000)) and threshold to decreasing values (e.g., threshold = c(.10, .01, 0)), one can quickly obtain a fairly accurate estimate of the combined $p$-value if it is far from significant (e.g., $\geq$ .10), but hone in on a more accurate estimate for a combined $p$-value that is closer to 0. Note that the last value of threshold should be 0 (and is forced to be inside of the function), so that the algorithm is guaranteed to terminate (hence, one can also leave out the last value of threshold, so threshold = c(.10, .01) would also work in the example above). One can also specify a single threshold (which is replicated as often as necessary depending on the length of size).

## Value

An object of class "poolr". The object is a list containing the following components:

| | |
|---|---|
| p | combined $p$-value. |
| ci | confidence interval for the combined $p$-value (only when adjust = "empirical"; otherwise NULL). |
| k | number of $p$-values that were combined. |
| m | estimate of the effective number of tests (only when adjust is one of "nyholt", "liji", "gao", "galwey", or "chen"; otherwise NULL). |
| adjust | chosen adjustment method. |
| statistic | value of the (adjusted) test statistic. |
| fun | name of calling function. |

## Note

The method underlying `adjust = "empirical"` assumes that the test statistics that generated the $p$-values to be combined follow a multivariate normal distribution. Hence, the matrix specified via `R` must be positive definite. If it is not and `nearpd = TRUE`, it will be turned into one (based on Higham, 2002, and a slightly simplified version of `nearPD` from the **Matrix** package).

## Author(s)

Ozan Cinar <ozancinar86@gmail.com>
Wolfgang Viechtbauer <wvb@wvbauer.com>

## References

Cinar, O. & Viechtbauer, W. (2022). The poolr package for combining independent and dependent p values. *Journal of Statistical Software*, **101**(1), 1–42. https://doi.org/10.18637/jss.v101.i01

Higham, N. J. (2002). Computing the nearest correlation matrix: A problem from finance. *IMA Journal of Numerical Analysis, 22*(3), 329–343. https://doi.org/10.1093/imanum/22.3.329

Wilkinson, B. (1951). A statistical consideration in psychological research. *Psychological Bulletin, 48*(2), 156–158. https://doi.org/10.1037/h0059111

## Examples

```
# copy p-values and LD correlation matrix into p and r
# (see help(grid2ip) for details on these data)
p <- grid2ip.p
r <- grid2ip.ld

# apply the binomial test
binomtest(p)

# use mvnconv() to convert the LD correlation matrix into a matrix with the
# correlations among the (two-sided) p-values assuming that the test
# statistics follow a multivariate normal distribution with correlation
# matrix r (note: 'side = 2' by default in mvnconv())
mvnconv(r, target = "p", cov2cor = TRUE)[1:5,1:5] # show only rows/columns 1-5

# adjustment based on estimates of the effective number of tests
binomtest(p, adjust = "nyholt", R = mvnconv(r, target = "p", cov2cor = TRUE))
binomtest(p, adjust = "liji",   R = mvnconv(r, target = "p", cov2cor = TRUE))
binomtest(p, adjust = "gao",    R = mvnconv(r, target = "p", cov2cor = TRUE))
binomtest(p, adjust = "galwey", R = mvnconv(r, target = "p", cov2cor = TRUE))
binomtest(p, adjust = "chen",   R = mvnconv(r, target = "p", cov2cor = TRUE))

# setting argument 'm' manually
binomtest(p, m = 12)

# adjustment based on an empirically-derived null distribution (setting the
# seed for reproducibility)
set.seed(1234)
```

```
binomtest(p, adjust = "empirical", R = r)

# generate the empirical distribution in batches of size 100
binomtest(p, adjust = "empirical", R = r, batchsize = 100)

# using the stepwise algorithm
binomtest(p, adjust = "empirical", R = r, size = c(1000, 10000, 100000), threshold = c(.10, .01))
```

---

`bonferroni` *Bonferroni Method*

---

## Description

Function to carry out the Bonferroni method.

## Usage

```
bonferroni(p, adjust = "none", R, m,
           size = 10000, threshold, side = 2, batchsize, nearpd = TRUE, ...)
```

## Arguments

| | |
|---|---|
| `p` | vector of length $k$ with the (one- or two-sided) p-values to be combined. |
| `adjust` | character string to specify an adjustment method to account for dependence. The default is `"none"`, in which case no adjustment is applied. Methods `"nyholt"`, `"liji"`, `"gao"`, `"galwey"`, or `"chen"` are adjustments based on an estimate of the effective number of tests (see `meff`). Adjustment method `"empirical"` uses an empirically-derived null distribution using pseudo replicates. See 'Details'. |
| `R` | a $k \times k$ symmetric matrix that reflects the dependence structure among the tests. Must be specified if `adjust` is set to something other than `"none"`. See 'Details'. |
| `m` | optional scalar (between 1 and $k$) to manually specify the effective number of tests (instead of estimating it via one of the methods described above). |
| `size` | size of the empirically-derived null distribution. Can also be a numeric vector of sizes, in which case a stepwise algorithm is used. This (and the following arguments) are only relevant when `adjust = "empirical"`. |
| `threshold` | numeric vector to specify the significance thresholds for the stepwise algorithm (only relevant when `size` is a vector). |
| `side` | scalar to specify the sidedness of the $p$-values that are used to simulate the null distribution (2, by default, for two-sided tests; 1 for one-sided tests). |
| `batchsize` | optional scalar to specify the batch size for generating the null distribution. When unspecified (the default), this is done in a single batch. |
| `nearpd` | logical indicating if a negative definite `R` matrix should be turned into the nearest positive definite matrix (only relevant when `adjust = "empirical"`). |
| `...` | other arguments. |

## Details

### Bonferroni Method

By default (i.e., when `adjust = "none"`), the function applies the Bonferroni method to the $p$-values. Letting $p_1, p_2, \ldots, p_k$ denote the individual (one- or two-sided) $p$-values of the $k$ hypothesis tests to be combined, the combined $p$-value is then computed with

$$p_c = \min(1, \min(p_1, p_2, \ldots, p_k) \times k).$$

The Bonferroni method does not assume that the $p$-values to be combined are independent. However, if the $p$-values are not independent, the method can become quite conservative (not reject often enough), depending on the dependence structure among the tests. In this case, one can adjust the method to account for such dependence (to bring the Type I error rate closer to some desired nominal significance level).

### Adjustment Based on the Effective Number of Tests

When `adjust` is set to `"nyholt"`, `"liji"`, `"gao"`, `"galwey"`, or `"chen"`, the Bonferroni method is adjusted based on an estimate of the effective number of tests (see `meff` for details on these methods for estimating the effective number of tests). In this case, argument `R` needs to be set to a matrix that reflects the dependence structure among the tests.

There is no general solution for constructing such a matrix, as this depends on the type of test that generated the $p$-values and the sidedness of these tests. If the $p$-values are obtained from tests whose test statistics can be assumed to follow a multivariate normal distribution and a matrix is available that reflects the correlations among the test statistics, then the `mvnconv` function can be used to convert this correlation matrix into the correlations among the (one- or two-sided) $p$-values, which can then be passed to the `R` argument. See 'Examples'.

Once the effective number of tests, $m$, is estimated based on `R` using one of the methods described above, the combined $p$-value is then computed with

$$p_c = \min(1, \min(p_1, p_2, \ldots, p_k) \times m).$$

Alternatively, one can also directly specify the effective number of tests via the `m` argument (e.g., if some other method not implemented in the **poolr** package is used to estimate the effective number of tests). Argument `R` is then irrelevant and doesn't need to be specified.

### Adjustment Based on an Empirically-Derived Null Distribution

When `adjust = "empirical"`, the combined $p$-value is computed based on an empirically-derived null distribution using pseudo replicates (using the `empirical` function). This is appropriate if the test statistics that generated the $p$-values to be combined can be assumed to follow a multivariate normal distribution and a matrix is available that reflects the correlations among the test statistics (which is specified via the `R` argument). In this case, test statistics are repeatedly simulated from a multivariate normal distribution under the joint null hypothesis, converted into one- or two-sided $p$-values (depending on the `side` argument), and the Bonferroni method is applied. Repeating this process `size` times yields a null distribution based on which the combined $p$-value can be computed, or more precisely, estimated, since repeated applications of this method will yield (slightly) different results. To obtain a stable estimate of the combined $p$-value, `size` should be set to a large value (the default is `10000`, but this can be increased for a more precise estimate). If we consider the combined $p$-value an estimate of the 'true' combined $p$-value that would be obtained for a null

distribution of infinite size, we can also construct a 95% (pseudo) confidence interval based on a binomial distribution.

If `batchsize` is unspecified, the null distribution is simulated in a single batch, which requires temporarily storing a matrix with dimensions `[size,k]`. When `size*k` is large, allocating the memory for this matrix might not be possible. Instead, one can specify a `batchsize` value, in which case a matrix with dimensions `[batchsize,k]` is repeatedly simulated until the desired size of the null distribution has been obtained.

One can also specify a vector for the `size` argument, in which case one must also specify a corresponding vector for the `threshold` argument. In that case, a stepwise algorithm is used that proceeds as follows. For `j = 1, ..., length(size)`,

1. estimate the combined $p$-value based on `size[j]`
2. if the combined $p$-value is $\geq$ than `threshold[j]`, stop (and report the combined $p$-value), otherwise go back to 1.

By setting `size` to increasing values (e.g., `size = c(1000, 10000, 100000)`) and `threshold` to decreasing values (e.g., `threshold = c(.10, .01, 0)`), one can quickly obtain a fairly accurate estimate of the combined $p$-value if it is far from significant (e.g., $\geq$ .10), but hone in on a more accurate estimate for a combined $p$-value that is closer to 0. Note that the last value of `threshold` should be 0 (and is forced to be inside of the function), so that the algorithm is guaranteed to terminate (hence, one can also leave out the last value of `threshold`, so `threshold = c(.10, .01)` would also work in the example above). One can also specify a single `threshold` (which is replicated as often as necessary depending on the length of `size`).

## Value

An object of class `"poolr"`. The object is a list containing the following components:

| | |
|---|---|
| `p` | combined $p$-value. |
| `ci` | confidence interval for the combined $p$-value (only when `adjust = "empirical"`; otherwise `NULL`). |
| `k` | number of $p$-values that were combined. |
| `m` | estimate of the effective number of tests (only when `adjust` is one of `"nyholt"`, `"liji"`, `"gao"`, `"galwey"`, or or `"chen"`; otherwise `NULL`). |
| `adjust` | chosen adjustment method. |
| `statistic` | value of the (adjusted) test statistic. |
| `fun` | name of calling function. |

## Note

The method underlying `adjust = "empirical"` assumes that the test statistics that generated the $p$-values to be combined follow a multivariate normal distribution. Hence, the matrix specified via `R` must be positive definite. If it is not and `nearpd = TRUE`, it will be turned into one (based on Higham, 2002, and a slightly simplified version of `nearPD` from the **Matrix** package).

## Author(s)

Ozan Cinar <ozancinar86@gmail.com>
Wolfgang Viechtbauer <wvb@wvbauer.com>

## References

Bland, J. M., & Altman, D. G. (1995). Multiple significance tests: The Bonferroni method. *British Medical Journal, 310*(6973), 170. https://doi.org/10.1136/bmj.310.6973.170

Cinar, O. & Viechtbauer, W. (2022). The poolr package for combining independent and dependent p values. *Journal of Statistical Software*, **101**(1), 1–42. https://doi.org/10.18637/jss.v101.i01

Higham, N. J. (2002). Computing the nearest correlation matrix: A problem from finance. *IMA Journal of Numerical Analysis, 22*(3), 329–343. https://doi.org/10.1093/imanum/22.3.329

## Examples

```
# copy p-values and LD correlation matrix into p and r
# (see help(grid2ip) for details on these data)
p <- grid2ip.p
r <- grid2ip.ld

# apply the Bonferroni method
bonferroni(p)

# use mvnconv() to convert the LD correlation matrix into a matrix with the
# correlations among the (two-sided) p-values assuming that the test
# statistics follow a multivariate normal distribution with correlation
# matrix r (note: 'side = 2' by default in mvnconv())
mvnconv(r, target = "p", cov2cor = TRUE)[1:5,1:5] # show only rows/columns 1-5

# adjustment based on estimates of the effective number of tests
bonferroni(p, adjust = "nyholt", R = mvnconv(r, target = "p", cov2cor = TRUE))
bonferroni(p, adjust = "liji",   R = mvnconv(r, target = "p", cov2cor = TRUE))
bonferroni(p, adjust = "gao",    R = mvnconv(r, target = "p", cov2cor = TRUE))
bonferroni(p, adjust = "galwey", R = mvnconv(r, target = "p", cov2cor = TRUE))
bonferroni(p, adjust = "chen",   R = mvnconv(r, target = "p", cov2cor = TRUE))

# setting argument 'm' manually
bonferroni(p, m = 12)

# adjustment based on an empirically-derived null distribution (setting the
# seed for reproducibility)
set.seed(1234)
bonferroni(p, adjust = "empirical", R = r)

# generate the empirical distribution in batches of size 100
bonferroni(p, adjust = "empirical", R = r, batchsize = 100)

# using the stepwise algorithm
bonferroni(p, adjust = "empirical", R = r, size = c(1000, 10000, 100000), threshold = c(.10, .01))
```

`empirical` *Simulate Empirically-Derived Null Distributions*

## Description

Function to simulate empirically-derived null distributions of various methods for combining $p$-values using pseudo replicates.

## Usage

```
empirical(R, method, side = 2, size = 10000, batchsize, ...)
```

## Arguments

| | |
|---|---|
| `R` | a $k \times k$ symmetric matrix that contains the correlations among the test statistics. |
| `method` | character string to specify for which method to simulate the null distribution (either `"fisher"`, `"stouffer"`, `"invchisq"`, `"binomtest"`, `"bonferroni"`, or `"tippett"`). |
| `side` | scalar to specify the sidedness of the $p$-values that are used to simulate the null distribution (2, by default, for two-sided tests; 1 for one-sided tests). |
| `size` | size of the empirically-derived null distribution that should be generated. |
| `batchsize` | optional scalar to specify the batch size for generating the null distribution. When unspecified (the default), this is done in a single batch. |
| `...` | other arguments. |

## Details

This function simulates the null distribution of a particular method for combining $p$-values when the test statistics that generate the $p$-values to be combined can be assumed to follow a multivariate normal distribution and a matrix is available that reflects the correlations among the test statistics (which is specified via the `R` argument). In this case, test statistics are repeatedly simulated from a multivariate normal distribution under the joint null hypothesis, converted into one- or two-sided $p$-values (depending on the `side` argument), and the chosen method is applied. Repeating this process `size` times yields the null distribution.

If `batchsize` is unspecified, the null distribution is simulated in a single batch, which requires temporarily storing a matrix with dimensions `[size,k]`. When `size*k` is large, allocating the memory for this matrix might not be possible. Instead, one can specify a `batchsize` value, in which case a matrix with dimensions `[batchsize,k]` is repeatedly simulated until the desired size of the null distribution has been obtained.

## Value

A vector of combined $p$-values as simulated under the joint null hypothesis for a given method.

## Note

The R matrix must be positive definite. If it is not, the function uses nearPD to find the nearest positive definite matrix (Higham, 2002) before simulating the null distribution.

## Author(s)

Ozan Cinar <ozancinar86@gmail.com>
Wolfgang Viechtbauer <wvb@wvbauer.com>

## References

Cinar, O. & Viechtbauer, W. (2022). The poolr package for combining independent and dependent p values. *Journal of Statistical Software*, **101**(1), 1–42. https://doi.org/10.18637/jss.v101.i01

Higham, N. J. (2002). Computing the nearest correlation matrix: A problem from finance. *IMA Journal of Numerical Analysis, 22*(3), 329–343. https://doi.org/10.1093/imanum/22.3.329

## Examples

```
# create an example correlation matrix with constant positive correlations
R <- matrix(0.6, nrow = 10, ncol = 10)
diag(R) <- 1

# generate null distribution for Fisher's method (setting the seed for reproducibility)
set.seed(1234)
psim <- empirical(R, method = "fisher")

# Fisher's method is liberal in this scenario (i.e., its actual Type I error
# rate is around .14 instead of the nominal significance level of .05)
mean(psim <= .05)

# estimate the actual Type I error rate of the other methods in this scenario
psim <- empirical(R, method = "stouffer")
mean(psim <= .05)
psim <- empirical(R, method = "invchisq")
mean(psim <= .05)
psim <- empirical(R, method = "binomtest")
mean(psim <= .05)
psim <- empirical(R, method = "bonferroni")
mean(psim <= .05)
psim <- empirical(R, method = "tippett")
mean(psim <= .05)

# Stouffer's and the inverse chi-square method also have clearly inflated
# Type I error rates and the binomial test just barely. As expected, the
# Bonferroni method is overly conservative and so is Tippett's method.
```

`fisher` *Fisher's Method*

## Description

Function to carry out Fisher's method.

## Usage

```
fisher(p, adjust = "none", R, m,
       size = 10000, threshold, side = 2, batchsize, nearpd = TRUE, ...)
```

## Arguments

| | |
|---|---|
| `p` | vector of length $k$ with the (one- or two-sided) p-values to be combined. |
| `adjust` | character string to specify an adjustment method to account for dependence. The default is `"none"`, in which case no adjustment is applied. Methods `"nyholt"`, `"liji"`, `"gao"`, `"galwey"`, or `"chen"` are adjustments based on an estimate of the effective number of tests (see `meff`). Adjustment method `"empirical"` uses an empirically-derived null distribution using pseudo replicates. Finally, method `"generalized"` uses a generalization of Fisher's method based on multivariate theory. See 'Details'. |
| `R` | a $k \times k$ symmetric matrix that reflects the dependence structure among the tests. Must be specified if `adjust` is set to something other than `"none"`. See 'Details'. |
| `m` | optional scalar (between 1 and $k$) to manually specify the effective number of tests (instead of estimating it via one of the methods described above). |
| `size` | size of the empirically-derived null distribution. Can also be a numeric vector of sizes, in which case a stepwise algorithm is used. This (and the following arguments) are only relevant when `adjust = "empirical"`. |
| `threshold` | numeric vector to specify the significance thresholds for the stepwise algorithm (only relevant when `size` is a vector). |
| `side` | scalar to specify the sidedness of the $p$-values that are used to simulate the null distribution (2, by default, for two-sided tests; 1 for one-sided tests). |
| `batchsize` | optional scalar to specify the batch size for generating the null distribution. When unspecified (the default), this is done in a single batch. |
| `nearpd` | logical indicating if a negative definite `R` matrix should be turned into the nearest positive definite matrix (only relevant when `adjust = "empirical"` or `adjust = "generalized"`). |
| `...` | other arguments. |

## Details

### Fisher's Method

By default (i.e., when `adjust = "none"`), the function applies Fisher's method to the $p$-values (Fisher, 1932). Letting $p_1, p_2, \ldots, p_k$ denote the individual (one- or two-sided) $p$-values of the $k$ hypothesis tests to be combined, the test statistic is then computed with

$$X^2 = -2 \sum_{i=1}^{k} \ln(p_i).$$

Under the joint null hypothesis, the test statistic follows a chi-square distribution with $2k$ degrees of freedom which is used to compute the combined $p$-value.

Fisher's method assumes that the $p$-values to be combined are independent. If this is not the case, the method can either be conservative (not reject often enough) or liberal (reject too often), depending on the dependence structure among the tests. In this case, one can adjust the method to account for such dependence (to bring the Type I error rate closer to some desired nominal significance level).

### Adjustment Based on the Effective Number of Tests

When `adjust` is set to `"nyholt"`, `"liji"`, `"gao"`, `"galwey"`, or `"chen"`, Fisher's method is adjusted based on an estimate of the effective number of tests (see `meff` for details on these methods for estimating the effective number of tests). In this case, argument `R` needs to be set to a matrix that reflects the dependence structure among the tests.

There is no general solution for constructing such a matrix, as this depends on the type of test that generated the $p$-values and the sidedness of these tests. If the $p$-values are obtained from tests whose test statistics can be assumed to follow a multivariate normal distribution and a matrix is available that reflects the correlations among the test statistics, then the `mvnconv` function can be used to convert this correlation matrix into the correlations among the (one- or two-sided) $p$-values, which can then be passed to the `R` argument. See 'Examples'.

Once the effective number of tests, $m$, is estimated based on `R` using one of the methods described above, the test statistic of Fisher's method can be modified with

$$\tilde{X}^2 = \frac{m}{k} \times X^2$$

which is then assumed to follow a chi-square distribution with $2m$ degrees of freedom.

Alternatively, one can also directly specify the effective number of tests via the `m` argument (e.g., if some other method not implemented in the **poolr** package is used to estimate the effective number of tests). Argument `R` is then irrelevant and doesn't need to be specified.

### Adjustment Based on an Empirically-Derived Null Distribution

When `adjust = "empirical"`, the combined $p$-value is computed based on an empirically-derived null distribution using pseudo replicates (using the `empirical` function). This is appropriate if the test statistics that generated the $p$-values to be combined can be assumed to follow a multivariate normal distribution and a matrix is available that reflects the correlations among the test statistics (which is specified via the `R` argument). In this case, test statistics are repeatedly simulated from a multivariate normal distribution under the joint null hypothesis, converted into one- or two-sided $p$-values (depending on the `side` argument), and Fisher's method is applied. Repeating this process `size` times yields a null distribution based on which the combined $p$-value can be computed, or more precisely, estimated, since repeated applications of this method will yield (slightly) different

results. To obtain a stable estimate of the combined $p$-value, `size` should be set to a large value (the default is `10000`, but this can be increased for a more precise estimate). If we consider the combined $p$-value an estimate of the 'true' combined $p$-value that would be obtained for a null distribution of infinite size, we can also construct a 95% (pseudo) confidence interval based on a binomial distribution.

If `batchsize` is unspecified, the null distribution is simulated in a single batch, which requires temporarily storing a matrix with dimensions `[size,k]`. When `size*k` is large, allocating the memory for this matrix might not be possible. Instead, one can specify a `batchsize` value, in which case a matrix with dimensions `[batchsize,k]` is repeatedly simulated until the desired size of the null distribution has been obtained.

One can also specify a vector for the `size` argument, in which case one must also specify a corresponding vector for the `threshold` argument. In that case, a stepwise algorithm is used that proceeds as follows. For `j = 1, ..., length(size)`,

1. estimate the combined $p$-value based on `size[j]`
2. if the combined $p$-value is $\geq$ than `threshold[j]`, stop (and report the combined $p$-value), otherwise go back to 1.

By setting `size` to increasing values (e.g., `size = c(1000, 10000, 100000)`) and `threshold` to decreasing values (e.g., `threshold = c(.10, .01, 0)`), one can quickly obtain a fairly accurate estimate of the combined $p$-value if it is far from significant (e.g., $\geq$ .10), but hone in on a more accurate estimate for a combined $p$-value that is closer to 0. Note that the last value of `threshold` should be 0 (and is forced to be inside of the function), so that the algorithm is guaranteed to terminate (hence, one can also leave out the last value of `threshold`, so `threshold = c(.10, .01)` would also work in the example above). One can also specify a single `threshold` (which is replicated as often as necessary depending on the length of `size`).

**Adjustment Based on Multivariate Theory**

When `adjust = "generalized"`, Fisher's method is computed based on a Satterthwaite approximation that accounts for the dependence among the tests, assuming that the test statistics that generated the $p$-values follow a multivariate normal distribution. In that case, `R` needs to be set equal to a matrix that contains the covariances among the $-2\ln(p_i)$ values. If a matrix is available that reflects the correlations among the test statistics, this can be converted into the required covariance matrix using the `mvnconv` function. See 'Examples'.

This generalization of Fisher's method is sometimes called Brown's method, based on Brown (1975), although the paper only describes the method for combining one-sided p-values. Both one- and two-sided versions of Brown's method are implemented in **poolr**.

## Value

An object of class `"poolr"`. The object is a list containing the following components:

| | |
|---|---|
| `p` | combined $p$-value. |
| `ci` | confidence interval for the combined $p$-value (only when `adjust = "empirical"`; otherwise `NULL`). |
| `k` | number of $p$-values that were combined. |
| `m` | estimate of the effective number of tests (only when `adjust` is one of `"nyholt"`, `"liji"`, `"gao"`, `"galwey"`, or `"chen"`; otherwise `NULL`). |

| | |
|---|---|
| adjust | chosen adjustment method. |
| statistic | value of the (adjusted) test statistic. |
| fun | name of calling function. |

## Note

The methods underlying adjust = "empirical" and adjust = "generalized" assume that the test statistics that generated the $p$-values to be combined follow a multivariate normal distribution. Hence, the matrix specified via R must be positive definite. If it is not and nearpd = TRUE, it will be turned into one (based on Higham, 2002, and a slightly simplified version of nearPD from the **Matrix** package).

## Author(s)

Ozan Cinar <ozancinar86@gmail.com>
Wolfgang Viechtbauer <wvb@wvbauer.com>

## References

Brown, M. B. (1975). 400: A method for combining non-independent, one-sided tests of significance. *Biometrics, 31*(4), 987–992. https://doi.org/10.2307/2529826

Cinar, O. & Viechtbauer, W. (2022). The poolr package for combining independent and dependent p values. *Journal of Statistical Software*, **101**(1), 1–42. https://doi.org/10.18637/jss.v101.i01

Fisher, R. A. (1932). *Statistical Methods for Research Workers* (4th ed.). Edinburgh: Oliver and Boyd.

Higham, N. J. (2002). Computing the nearest correlation matrix: A problem from finance. *IMA Journal of Numerical Analysis, 22*(3), 329–343. https://doi.org/10.1093/imanum/22.3.329

## Examples

```
# copy p-values and LD correlation matrix into p and r
# (see help(grid2ip) for details on these data)
p <- grid2ip.p
r <- grid2ip.ld

# apply Fisher's method
fisher(p)

# use mvnconv() to convert the LD correlation matrix into a matrix with the
# correlations among the (two-sided) p-values assuming that the test
# statistics follow a multivariate normal distribution with correlation
# matrix r (note: 'side = 2' by default in mvnconv())
mvnconv(r, target = "p", cov2cor = TRUE)[1:5,1:5] # show only rows/columns 1-5

# adjustment based on estimates of the effective number of tests
fisher(p, adjust = "nyholt", R = mvnconv(r, target = "p", cov2cor = TRUE))
fisher(p, adjust = "liji",   R = mvnconv(r, target = "p", cov2cor = TRUE))
fisher(p, adjust = "gao",    R = mvnconv(r, target = "p", cov2cor = TRUE))
```

```
fisher(p, adjust = "galwey", R = mvnconv(r, target = "p", cov2cor = TRUE))
fisher(p, adjust = "chen",   R = mvnconv(r, target = "p", cov2cor = TRUE))

# setting argument 'm' manually
fisher(p, m = 12)

# adjustment based on an empirically-derived null distribution (setting the
# seed for reproducibility)
set.seed(1234)
fisher(p, adjust = "empirical", R = r)

# generate the empirical distribution in batches of size 100
fisher(p, adjust = "empirical", R = r, batchsize = 100)

# using the stepwise algorithm
fisher(p, adjust = "empirical", R = r, size = c(1000, 10000, 100000), threshold = c(.10, .01))

# use mvnconv() to convert the LD correlation matrix into a matrix with the
# covariances among the (two-sided) '-2ln(p_i)' values assuming that the
# test statistics follow a multivariate normal distribution with correlation
# matrix r (note: 'side = 2' by default in mvnconv())
mvnconv(r, target = "m2lp")[1:5,1:5] # show only rows/columns 1-5

# adjustment based on generalized method
fisher(p, adjust = "generalized", R = mvnconv(r, target = "m2lp"))

# when using mvnconv() inside fisher() with adjust = "generalized", the
# 'target' argument is automatically set and doesn't need to be specified
fisher(p, adjust = "generalized", R = mvnconv(r))
```

---

`grid2ip` *Results from testing the association between depressive symptoms and 23 SNPs in the GRID2IP gene*

---

## Description

Results from testing the association between depressive symptoms (as measured with the CES-D scale) and 23 single-nucleotide polymorphisms (SNPs) in the GRID2IP gene based on a sample of 886 adolescents (Van Assche et al., 2017).

## Usage

```
grid2ip.p
grid2ip.ld
grid2ip.geno
grid2ip.pheno
```

## Format

Object `grid2ip.p` is a vector with the 23 $p$-values of the tests (two-sided). Object `grid2ip.ld` contains a matrix with the linkage disequilibrium (LD) correlations among the 23 SNPs. `grid2ip.geno` is a matrix that contains the genotypes of the adoloscents for the 23 SNPs. `grid2ip.pheno` is a vector with the phenotype for the adoloscents (the log-transformed CES-D scale values).

## References

Van Assche, E., Moons, T., Cinar, O., Viechtbauer, W., Oldehinkel, A. J., Van Leeuwen, K., Verschueren, K., Colpin, H., Lambrechts, D., Van den Noortgate, W., Goossens, L., Claes, S., & van Winkel, R. (2017). Gene-based interaction analysis shows GABAergic genes interacting with parenting in adolescent depressive symptoms. *Journal of Child Psychology and Psychiatry, 58*(12), 1301–1309. https://doi.org/10.1111/jcpp.12766

---

`invchisq` *Inverse Chi-Square Method*

---

## Description

Function to carry out the inverse chi-square method.

## Usage

```
invchisq(p, adjust = "none", R, m,
         size = 10000, threshold, side = 2, batchsize, nearpd = TRUE, ...)
```

## Arguments

| | |
|---|---|
| `p` | vector of length $k$ with the (one- or two-sided) p-values to be combined. |
| `adjust` | character string to specify an adjustment method to account for dependence. The default is `"none"`, in which case no adjustment is applied. Methods `"nyholt"`, `"liji"`, `"gao"`, `"galwey"`, or `"chen"` are adjustments based on an estimate of the effective number of tests (see `meff`). Adjustment method `"empirical"` uses an empirically-derived null distribution using pseudo replicates. Finally, method `"generalized"` uses a generalization of the inverse chi-square method based on multivariate theory. See 'Details'. |
| `R` | a $k \times k$ symmetric matrix that reflects the dependence structure among the tests. Must be specified if `adjust` is set to something other than `"none"`. See 'Details'. |
| `m` | optional scalar (between 1 and $k$) to manually specify the effective number of tests (instead of estimating it via one of the methods described above). |
| `size` | size of the empirically-derived null distribution. Can also be a numeric vector of sizes, in which case a stepwise algorithm is used. This (and the following arguments) are only relevant when `adjust = "empirical"`. |
| `threshold` | numeric vector to specify the significance thresholds for the stepwise algorithm (only relevant when `size` is a vector). |

| | |
|---|---|
| side | scalar to specify the sidedness of the $p$-values that are used to simulate the null distribution (2, by default, for two-sided tests; 1 for one-sided tests). |
| batchsize | optional scalar to specify the batch size for generating the null distribution. When unspecified (the default), this is done in a single batch. |
| nearpd | logical indicating if a negative definite R matrix should be turned into the nearest positive definite matrix (only relevant when adjust = "empirical" or adjust = "generalized"). |
| ... | other arguments. |

## Details

### Inverse Chi-Square Method

By default (i.e., when adjust = "none"), the function applies the inverse chi-square method to the $p$-values. Letting $p_1, p_2, \ldots, p_k$ denote the individual (one- or two-sided) $p$-values of the $k$ hypothesis tests to be combined, the test statistic is then computed with

$$X^2 = \sum_{i=1}^{k} F^{-1}(1 - p_i, 1)$$

where $F^{-1}(\cdot, 1)$ denotes the inverse of the cumulative distribution function of a chi-square distribution with one degree of freedom. Under the joint null hypothesis, the test statistic follows a chi-square distribution with $k$ degrees of freedom which is used to compute the combined $p$-value.

The inverse chi-square method assumes that the $p$-values to be combined are independent. If this is not the case, the method can either be conservative (not reject often enough) or liberal (reject too often), depending on the dependence structure among the tests. In this case, one can adjust the method to account for such dependence (to bring the Type I error rate closer to some desired nominal significance level).

### Adjustment Based on the Effective Number of Tests

When adjust is set to "nyholt", "liji", "gao", "galwey", or "chen", the inverse chi-square method is adjusted based on an estimate of the effective number of tests (see meff for details on these methods for estimating the effective number of tests). In this case, argument R needs to be set to a matrix that reflects the dependence structure among the tests.

There is no general solution for constructing such a matrix, as this depends on the type of test that generated the $p$-values and the sidedness of these tests. If the $p$-values are obtained from tests whose test statistics can be assumed to follow a multivariate normal distribution and a matrix is available that reflects the correlations among the test statistics, then the mvnconv function can be used to convert this correlation matrix into the correlations among the (one- or two-sided) $p$-values, which can then be passed to the R argument. See 'Examples'.

Once the effective number of tests, $m$, is estimated based on R using one of the methods described above, the test statistic of the inverse chi-square method can be modified with

$$\tilde{X}^2 = \frac{m}{k} \times X^2$$

which is then assumed to follow a chi-square distribution with $m$ degrees of freedom.

Alternatively, one can also directly specify the effective number of tests via the m argument (e.g., if some other method not implemented in the **poolr** package is used to estimate the effective number of tests). Argument R is then irrelevant and doesn't need to be specified.

### Adjustment Based on an Empirically-Derived Null Distribution

When `adjust = "empirical"`, the combined $p$-value is computed based on an empirically-derived null distribution using pseudo replicates (using the `empirical` function). This is appropriate if the test statistics that generated the $p$-values to be combined can be assumed to follow a multivariate normal distribution and a matrix is available that reflects the correlations among the test statistics (which is specified via the `R` argument). In this case, test statistics are repeatedly simulated from a multivariate normal distribution under the joint null hypothesis, converted into one- or two-sided $p$-values (depending on the `side` argument), and the inverse chi-square method is applied. Repeating this process `size` times yields a null distribution based on which the combined $p$-value can be computed, or more precisely, estimated, since repeated applications of this method will yield (slightly) different results. To obtain a stable estimate of the combined $p$-value, `size` should be set to a large value (the default is `10000`, but this can be increased for a more precise estimate). If we consider the combined $p$-value an estimate of the 'true' combined $p$-value that would be obtained for a null distribution of infinite size, we can also construct a 95% (pseudo) confidence interval based on a binomial distribution.

If `batchsize` is unspecified, the null distribution is simulated in a single batch, which requires temporarily storing a matrix with dimensions `[size,k]`. When `size*k` is large, allocating the memory for this matrix might not be possible. Instead, one can specify a `batchsize` value, in which case a matrix with dimensions `[batchsize,k]` is repeatedly simulated until the desired size of the null distribution has been obtained.

One can also specify a vector for the `size` argument, in which case one must also specify a corresponding vector for the `threshold` argument. In that case, a stepwise algorithm is used that proceeds as follows. For `j = 1, ..., length(size)`,

1. estimate the combined $p$-value based on `size[j]`
2. if the combined $p$-value is $\geq$ than `threshold[j]`, stop (and report the combined $p$-value), otherwise go back to 1.

By setting `size` to increasing values (e.g., `size = c(1000, 10000, 100000)`) and `threshold` to decreasing values (e.g., `threshold = c(.10, .01, 0)`), one can quickly obtain a fairly accurate estimate of the combined $p$-value if it is far from significant (e.g., $\geq$ .10), but hone in on a more accurate estimate for a combined $p$-value that is closer to 0. Note that the last value of `threshold` should be 0 (and is forced to be inside of the function), so that the algorithm is guaranteed to terminate (hence, one can also leave out the last value of `threshold`, so `threshold = c(.10, .01)` would also work in the example above). One can also specify a single `threshold` (which is replicated as often as necessary depending on the length of `size`).

### Adjustment Based on Multivariate Theory

When `adjust = "generalized"`, the inverse chi-square method is computed based on a Satterthwaite approximation that accounts for the dependence among the tests, assuming that the test statistics that generated the $p$-values follow a multivariate normal distribution. In that case, `R` needs to be set equal to a matrix that contains the covariances among the $F^{-1}(1 - p_i, 1)$ values. If a matrix is available that reflects the correlations among the test statistics, this can be converted into the required covariance matrix using the `mvnconv` function. See 'Examples'.

## Value

An object of class `"poolr"`. The object is a list containing the following components:

| | |
|---|---|
| p | combined $p$-value. |
| ci | confidence interval for the combined $p$-value (only when adjust = "empirical"; otherwise NULL). |
| k | number of $p$-values that were combined. |
| m | estimate of the effective number of tests (only when adjust is one of "nyholt", "liji", "gao", "galwey", or or "chen"; otherwise NULL). |
| adjust | chosen adjustment method. |
| statistic | value of the (adjusted) test statistic. |
| fun | name of calling function. |

## Note

The methods underlying adjust = "empirical" and adjust = "generalized" assume that the test statistics that generated the $p$-values to be combined follow a multivariate normal distribution. Hence, the matrix specified via R must be positive definite. If it is not and nearpd = TRUE, it will be turned into one (based on Higham, 2002, and a slightly simplified version of nearPD from the **Matrix** package).

## Author(s)

Ozan Cinar <ozancinar86@gmail.com>
Wolfgang Viechtbauer <wvb@wvbauer.com>

## References

Cinar, O. & Viechtbauer, W. (2022). The poolr package for combining independent and dependent p values. *Journal of Statistical Software*, **101**(1), 1–42. https://doi.org/10.18637/jss.v101.i01

Higham, N. J. (2002). Computing the nearest correlation matrix: A problem from finance. *IMA Journal of Numerical Analysis, 22*(3), 329–343. https://doi.org/10.1093/imanum/22.3.329

Lancaster, H. O. (1961). The combination of probabilities: An application of orthonormal functions. *Australian Journal of Statistics, 3*(1), 20–33. https://doi.org/10.1111/j.1467-842X.1961.tb00058.x

## Examples

```
# copy p-values and LD correlation matrix into p and r
# (see help(grid2ip) for details on these data)
p <- grid2ip.p
r <- grid2ip.ld

# apply the inverse chi-square method
invchisq(p)

# use mvnconv() to convert the LD correlation matrix into a matrix with the
# correlations among the (two-sided) p-values assuming that the test
# statistics follow a multivariate normal distribution with correlation
# matrix r (note: 'side = 2' by default in mvnconv())
mvnconv(r, target = "p", cov2cor = TRUE)[1:5,1:5] # show only rows/columns 1-5
```

```
# adjustment based on estimates of the effective number of tests
invchisq(p, adjust = "nyholt", R = mvnconv(r, target = "p", cov2cor = TRUE))
invchisq(p, adjust = "liji",   R = mvnconv(r, target = "p", cov2cor = TRUE))
invchisq(p, adjust = "gao",    R = mvnconv(r, target = "p", cov2cor = TRUE))
invchisq(p, adjust = "galwey", R = mvnconv(r, target = "p", cov2cor = TRUE))
invchisq(p, adjust = "chen",   R = mvnconv(r, target = "p", cov2cor = TRUE))

# setting argument 'm' manually
invchisq(p, m = 12)

# adjustment based on an empirically-derived null distribution (setting the
# seed for reproducibility)
set.seed(1234)
invchisq(p, adjust = "empirical", R = r)

# generate the empirical distribution in batches of size 100
invchisq(p, adjust = "empirical", R = r, batchsize = 100)

# using the stepwise algorithm (not run, because this takes slightly longer
# and can lead to a note when running the standard package checks)

invchisq(p, adjust = "empirical", R = r, size = c(1000, 10000, 100000), threshold = c(.10, .01))


# use mvnconv() to convert the LD correlation matrix into a matrix with the
# covariances among the (two-sided) 'F(1-p_i,1)' values assuming that the
# test statistics follow a multivariate normal distribution with correlation
# matrix r (note: 'side = 2' by default in mvnconv())
mvnconv(r, target = "chisq1")[1:5,1:5] # show only rows/columns 1-5

# adjustment based on generalized method
invchisq(p, adjust = "generalized", R = mvnconv(r, target = "chisq1"))

# when using mvnconv() inside invchisq() with adjust = "generalized", the
# 'target' argument is automatically set and doesn't need to be specified
invchisq(p, adjust = "generalized", R = mvnconv(r))
```

---

`meff` *Estimate the Effective Number of Tests*

---

## Description

Estimate the effective number of tests.

## Usage

```
meff(R, eigen, method, ...)
```

## Arguments

| | |
|---|---|
| R | a $k \times k$ symmetric matrix that reflects the correlation structure among the tests. |
| eigen | optional vector to directly supply the eigenvalues to the function (instead of computing them from the matrix given via R). |
| method | character string to specify the method to be used to estimate the effective number of tests (either "nyholt", "liji", "gao", "galwey", or "chen"). See 'Details'. |
| ... | other arguments. |

## Details

The function estimates the effective number of tests based on one of five different methods. All methods except the one by Chen and Liu (2011) work by extracting the eigenvalues from the $R$ matrix supplied via the R argument (or from the eigenvalues directly passed via the eigen argument). Letting $\lambda_i$ denote the $i$th eigenvalue of this matrix (with $i = 1, \ldots, k$) in decreasing order, the effective number of tests ($m$) is estimated as follows.

**Method by Nyholt (2004)**

$$m = 1 + (k - 1)\left(1 - \frac{\text{Var}(\lambda)}{k}\right)$$

where $\text{Var}(\lambda)$ is the observed sample variance of the $k$ eigenvalues.

**Method by Li & Ji (2005)**

$$m = \sum_{i=1}^{k} f(|\lambda_i|)$$

where $f(x) = I(x \geq 1) + (x - \lfloor x \rfloor)$ and $\lfloor \cdot \rfloor$ is the floor function.

**Method by Gao et al. (2008)**

$$m = \min(x) \text{ such that } \frac{\sum_{i=1}^{x} \lambda_i}{\sum_{i=1}^{k} \lambda_i} > C$$

where $C$ is a pre-defined parameter which is set to 0.995 by default, but can be adjusted (see 'Note').

**Method by Galwey (2009)**

$$m = \frac{\left(\sum_{i=1}^{k} \sqrt{\lambda'_i}\right)^2}{\sum_{i=1}^{k} \lambda'_i}$$

where $\lambda'_i = \max[0, \lambda_i]$.

**Method by Chen & Liu (2011)**

$$m = \sum_{i=1}^{k} \frac{1}{R_i}$$

where $R_i = \sum_{j=1}^{k} |r_{ij}|^C$ for $i = 1, \ldots, k$ and $r_{ij}$ denotes the element in the $R$ matrix in row $i$ and column $j$. By default, the value of $C$ is set to 7, but can be adjusted (see 'Note').

**Note:** For all methods that can yield a non-integer estimate (all but the method by Gao et al., 2008), the resulting estimate $m$ is rounded down to the nearest integer.

**Specifying the R Matrix**

The $R$ matrix should reflect the dependence structure among the tests. There is no general solution on how such a matrix should be constructed, as this depends on the type of test and the sidedness of these tests. For example, we can use the correlations among related but changing elements across the analyses/tests, or a function thereof, as a proxy for the dependence structure. For example, when conducting $k$ analyses with the same dependent variable and $k$ different independent variables, the correlations among the independent variables could serve as such a proxy. Analogously, if analyses are conducted for $k$ dependent variables with the same set of independent variables, the correlations among the dependent variables could be used instead.

If the tests of interest have test statistics that can be assumed to follow a multivariate normal distribution and a matrix is available that reflects the correlations among the test statistics (which might be approximated by the correlations among the interchanging independent or dependent variables), then the `mvnconv` function can be used to convert this correlation matrix into the correlations among the (one- or two-sided) $p$-values, which in turn can then be passed to the `R` argument. See 'Examples'.

**Non-Positive Semi-Definite R**

Depending on the way $R$ was constructed, it may happen that this matrix is not positive semi-definite, leading to negative eigenvalues. The methods given above can all still be carried out in this case. However, another possibility is to handle such a case by using an algorithm that finds the nearest positive (semi-)definite matrix (e.g., Higham 2002) before passing this matrix to the function (see `nearPD` from the **Matrix** package for a corresponding implementation).

## Value

A scalar giving the estimate of the effective number of tests.

## Note

For `method = "gao"`, `C = 0.995` by default, but a different value of `C` can be passed to the function via `...` (e.g., `meff(R, method = "gao", C = 0.95)`). For `method = "chen"`, `C = 7` by default, but a different value of `C` can be passed to the function via `...` (e.g., `meff(R, method = "chen", C = 6)`).

## Author(s)

Ozan Cinar <ozancinar86@gmail.com>
Wolfgang Viechtbauer <wvb@wvbauer.com>

## References

Chen, Z. X., & Liu, Q. Z. (2011). A new approach to account for the correlations among single nucleotide polymorphisms in genome-wide association studies. *Human Heredity*, **72**(1), 1–9. https://doi.org/10.1159/000330135

Cinar, O. & Viechtbauer, W. (2022). The poolr package for combining independent and dependent p values. *Journal of Statistical Software*, **101**(1), 1–42. https://doi.org/10.18637/jss.v101.i01

Gao, X., Starmer, J., & Martin, E. R. (2008). A multiple testing correction method for genetic association studies using correlated single nucleotide polymorphisms. *Genetic Epidemiology, 32*(4), 361–369. https://doi.org/10.1002/gepi.20310

Galwey, N. W. (2009). A new measure of the effective number of tests, a practical tool for comparing families of non-independent significance tests. *Genetic Epidemiology, 33*(7), 559–568. https://doi.org/10.1002/gepi.20408

Higham, N. J. (2002). Computing the nearest correlation matrix: A problem from finance. *IMA Journal of Numerical Analysis, 22*(3), 329–343. https://doi.org/10.1093/imanum/22.3.329

Li, J., & Ji, L. (2005). Adjusting multiple testing in multilocus analyses using the eigenvalues of a correlation matrix. *Heredity, 95*(3), 221–227. https://doi.org/10.1038/sj.hdy.6800717

Nyholt, D. R. (2004). A simple correction for multiple testing for single-nucleotide polymorphisms in linkage disequilibrium with each other. *American Journal of Human Genetics, 74*(4), 765–769. https://doi.org/10.1086/383251

## Examples

```
# copy LD correlation matrix into r (see help(grid2ip) for details on these data)
r <- grid2ip.ld

# estimate the effective number of tests based on the LD correlation matrix
meff(r, method = "nyholt")
meff(r, method = "liji")
meff(r, method = "gao")
meff(r, method = "galwey")
meff(r, method = "chen")

# use mvnconv() to convert the LD correlation matrix into a matrix with the
# correlations among the (two-sided) p-values assuming that the test
# statistics follow a multivariate normal distribution with correlation
# matrix r (note: 'side = 2' by default in mvnconv())
mvnconv(r, target = "p", cov2cor = TRUE)[1:5,1:5] # show only rows/columns 1-5

# use this matrix instead for estimating the effective number of tests
meff(mvnconv(r, target = "p", cov2cor = TRUE), method = "nyholt")
meff(mvnconv(r, target = "p", cov2cor = TRUE), method = "liji")
meff(mvnconv(r, target = "p", cov2cor = TRUE), method = "gao")
meff(mvnconv(r, target = "p", cov2cor = TRUE), method = "galwey")
meff(mvnconv(r, target = "p", cov2cor = TRUE), method = "chen")
```

---

mvnconv *Convert Correlations Among Multivariate Normal Test Statistics to Covariances for Various Target Statistics*

---

## Description

Function to convert a matrix with the correlations among multivariate normal test statistics to a matrix with the covariances among various target statistics.

## Usage

```
mvnconv(R, side = 2, target, cov2cor = FALSE)
```

## Arguments

| | |
|---|---|
| R | a $k \times k$ symmetric matrix that contains the correlations among the test statistics. |
| side | scalar to specify the sidedness of the $p$-values that are obtained from the test statistics (2, by default, for two-sided tests; 1 for one-sided tests). |
| target | the target statistic for which the covariances are calculated (either "p", "m2lp", "chisq1", or "z"). See 'Details'. |
| cov2cor | logical to indicate whether to convert the covariance matrix to a correlation matrix (default is FALSE). |

## Details

The function converts a matrix with the correlations among multivariate normal test statistics to a matrix with the covariances among various target statistics. In particular, assume

$$\begin{bmatrix} t_i \\ t_j \end{bmatrix} \sim \text{MVN}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 & \rho_{ij} \\ \rho_{ij} & 1 \end{bmatrix}\right)$$

is the joint distribution for test statistics $t_i$ and $t_j$. For side = 1, let $p_i = 1-\Phi(t_i)$ and $p_j = 1-\Phi(t_j)$ where $\Phi(\cdot)$ denotes the cumulative distribution function of a standard normal distribution. For side = 2, let $p_i = 2(1 - \Phi(|t_i|))$ and $p_j = 2(1 - \Phi(|t_j|))$. These are simply the one- and two-sided $p$-values corresponding to $t_i$ and $t_j$.

If target = "p", the function computes $\text{Cov}[p_i, p_j]$.

If target = "m2lp", the function computes $\text{Cov}[-2\ln(p_i), -2\ln(p_j)]$.

If target = "chisq1", the function computes $\text{Cov}[F^{-1}(1-p_i, 1), F^{-1}(1-p_j, 1)]$, where $F^{-1}(\cdot, 1)$ denotes the inverse of the cumulative distribution function of a chi-square distribution with one degree of freedom.

If target = "z", the function computes $\text{Cov}[\Phi^{-1}(1-p_i), \Phi^{-1}(1-p_j)]$, where $\Phi^{-1}(\cdot)$ denotes the inverse of the cumulative distribution function of a standard normal distribution.

## Value

The function returns the covariance matrix (or the correlation matrix if cov2cor = TRUE).

## Note

Since computation of the covariances requires numerical integration, the function doesn't actually compute these covariances on the fly. Instead, it uses the mvnlookup lookup table, which contains the covariances.

## Author(s)

Ozan Cinar <ozancinar86@gmail.com>
Wolfgang Viechtbauer <wvb@wvbauer.com>

## References

Cinar, O. & Viechtbauer, W. (2022). The poolr package for combining independent and dependent p values. *Journal of Statistical Software*, **101**(1), 1–42. https://doi.org/10.18637/jss.v101.i01

## Examples

```
# illustrative correlation matrix
R <- matrix(c(   1,  0.8,  0.5,  0.3,
               0.8,    1,  0.2,  0.4,
               0.5,  0.2,    1,  0.7,
               0.3,  0.4,  0.7,    1), nrow = 4, ncol = 4)

# convert R into covariance matrices for the chosen targets
mvnconv(R, target = "p")
mvnconv(R, target = "m2lp")
mvnconv(R, target = "chisq1")
mvnconv(R, target = "z")

# convert R into correlation matrices for the chosen targets
mvnconv(R, target = "p",      cov2cor = TRUE)
mvnconv(R, target = "m2lp",   cov2cor = TRUE)
mvnconv(R, target = "chisq1", cov2cor = TRUE)
mvnconv(R, target = "z",      cov2cor = TRUE)
```

---

# mvnlookup *Lookup Table for the mvnconv() Function*

---

## Description

Lookup table for the mvnconv function.

## Usage

```
mvnlookup
```

## Format

The data frame contains the following columns:

| | | |
|---|---|---|
| **rhos** | numeric | correlations among the test statistics |
| **m2lp_1** | numeric | $\text{Cov}[-2\ln(p_i), -2\ln(p_j)]$ (for one-sided tests) |
| **m2lp_2** | numeric | $\text{Cov}[-2\ln(p_i), -2\ln(p_j)]$ (for two-sided tests) |
| **z_1** | numeric | $\text{Cov}[\Phi^{-1}(1-p_i), \Phi^{-1}(1-p_j)]$ (for one-sided tests) |
| **z_2** | numeric | $\text{Cov}[\Phi^{-1}(1-p_i), \Phi^{-1}(1-p_j)]$ (for two-sided tests) |
| **chisq1_1** | numeric | $\text{Cov}[F^{-1}(1-p_i, 1), F^{-1}(1-p_j, 1)]$ (for one-sided tests) |
| **chisq1_2** | numeric | $\text{Cov}[F^{-1}(1-p_i, 1), F^{-1}(1-p_j, 1)]$ (for two-sided tests) |
| **p_1** | numeric | $\text{Cov}[p_i, p_j]$ (for one-sided tests) |
| **p_2** | numeric | $\text{Cov}[p_i, p_j]$ (for two-sided tests) |

## Details

Assume

$$\begin{bmatrix} t_i \\ t_j \end{bmatrix} \sim \text{MVN}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 & \rho_{ij} \\ \rho_{ij} & 1 \end{bmatrix}\right)$$

is the joint distribution for test statistics $t_i$ and $t_j$. For one-sided tests, let $p_i = 1 - \Phi(t_i)$ and $p_j = 1 - \Phi(t_j)$ where $\Phi(\cdot)$ denotes the cumulative distribution function of a standard normal distribution. For two-sided tests, let $p_i = 2(1 - \Phi(|t_i|))$ and $p_j = 2(1 - \Phi(|t_j|))$. These are simply the one- and two-sided $p$-values corresponding to $t_i$ and $t_j$.

Columns `p_1` and `p_2` contain the values for $\text{Cov}[p_i, p_j]$.

Columns `m2lp_1` and `m2lp_2` contain the values for $\text{Cov}[-2\ln(p_i), -2\ln(p_j)]$.

Columns `chisq1_1` and `chisq1_2` contain the values for $\text{Cov}[F^{-1}(1-p_i, 1), F^{-1}(1-p_j, 1)]$, where $F^{-1}(\cdot, 1)$ denotes the inverse of the cumulative distribution function of a chi-square distribution with one degree of freedom.

Columns `z_1` and `z_2` contain the values for $\text{Cov}[\Phi^{-1}(1-p_i), \Phi^{-1}(1-p_j)]$, where $\Phi^{-1}(\cdot)$ denotes the inverse of the cumulative distribution function of a standard normal distribution.

Computation of these covariances required numerical integration. The values in this table were precomputed.

---

`print.poolr` *Print Method for 'poolr' Objects*

---

## Description

Print method for objects of class `"poolr"`.

## Usage

```
## S3 method for class 'poolr'
print(x, digits=3, ...)
```

## Arguments

| | |
|---|---|
| `x` | an object of class `"poolr"`. |
| `digits` | integer specifying the number of (significant) digits for rounding/presenting the results. |
| `...` | other arguments. |

## Details

The output shows the combined $p$-value (with the specified number of significant digits), the test statistic (and its assumed null distribution), and the adjustment method that was applied.

## Value

The function does not return an object.

## Author(s)

Ozan Cinar <ozancinar86@gmail.com>
Wolfgang Viechtbauer <wvb@wvbauer.com>

---

`stouffer` *Stouffer's Method*

---

## Description

Function to carry out Stouffer's method.

## Usage

```
stouffer(p, adjust = "none", R, m,
         size = 10000, threshold, side = 2, batchsize, nearpd = TRUE, ...)
```

## Arguments

| | |
|---|---|
| `p` | vector of length $k$ with the (one- or two-sided) p-values to be combined. |
| `adjust` | character string to specify an adjustment method to account for dependence. The default is `"none"`, in which case no adjustment is applied. Methods `"nyholt"`, `"liji"`, `"gao"`, `"galwey"`, or `"chen"` are adjustments based on an estimate of the effective number of tests (see `meff`). Adjustment method `"empirical"` uses an empirically-derived null distribution using pseudo replicates. Finally, method `"generalized"` uses a generalization of Stouffer's method based on multivariate theory. See 'Details'. |
| `R` | a $k \times k$ symmetric matrix that reflects the dependence structure among the tests. Must be specified if `adjust` is set to something other than `"none"`. See 'Details'. |
| `m` | optional scalar (between 1 and $k$) to manually specify the effective number of tests (instead of estimating it via one of the methods described above). |
| `size` | size of the empirically-derived null distribution. Can also be a numeric vector of sizes, in which case a stepwise algorithm is used. This (and the following arguments) are only relevant when `adjust = "empirical"`. |
| `threshold` | numeric vector to specify the significance thresholds for the stepwise algorithm (only relevant when `size` is a vector). |
| `side` | scalar to specify the sidedness of the $p$-values that are used to simulate the null distribution (2, by default, for two-sided tests; 1 for one-sided tests). |
| `batchsize` | optional scalar to specify the batch size for generating the null distribution. When unspecified (the default), this is done in a single batch. |

| | |
|---|---|
| nearpd | logical indicating if a negative definite R matrix should be turned into the nearest positive definite matrix (only relevant when adjust = "empirical" or adjust = "generalized"). |
| ... | other arguments. |

## Details

### Stouffer's Method

By default (i.e., when adjust = "none"), the function applies Stouffer's method to the $p$-values (Stouffer et al., 1949). Letting $p_1, p_2, \ldots, p_k$ denote the individual (one- or two-sided) $p$-values of the $k$ hypothesis tests to be combined, the test statistic is then computed with

$$z = \sum_{i=1}^{k} z_i / \sqrt{k}$$

where $z_i = \Phi^{-1}(1 - p_i)$ and $\Phi^{-1}(\cdot)$ denotes the inverse of the cumulative distribution function of a standard normal distribution. Under the joint null hypothesis, the test statistic follows a standard normal distibution which is used to compute the combined $p$-value.

Stouffer's method assumes that the $p$-values to be combined are independent. If this is not the case, the method can either be conservative (not reject often enough) or liberal (reject too often), depending on the dependence structure among the tests. In this case, one can adjust the method to account for such dependence (to bring the Type I error rate closer to some desired nominal significance level).

### Adjustment Based on the Effective Number of Tests

When adjust is set to "nyholt", "liji", "gao", "galwey", or "chen", Stouffer's method is adjusted based on an estimate of the effective number of tests (see meff for details on these methods for estimating the effective number of tests). In this case, argument R needs to be set to a matrix that reflects the dependence structure among the tests.

There is no general solution for constructing such a matrix, as this depends on the type of test that generated the $p$-values and the sidedness of these tests. If the $p$-values are obtained from tests whose test statistics can be assumed to follow a multivariate normal distribution and a matrix is available that reflects the correlations among the test statistics, then the mvnconv function can be used to convert this correlation matrix into the correlations among the (one- or two-sided) $p$-values, which can then be passed to the R argument. See 'Examples'.

Once the effective number of tests, $m$, is estimated based on R using one of the methods described above, the test statistic of Stouffer's method can be modified with

$$\tilde{z} = \sqrt{\frac{m}{k}} \times z$$

which is then assumed to follow a standard normal distibution.

Alternatively, one can also directly specify the effective number of tests via the m argument (e.g., if some other method not implemented in the **poolr** package is used to estimate the effective number of tests). Argument R is then irrelevant and doesn't need to be specified.

### Adjustment Based on an Empirically-Derived Null Distribution

When adjust = "empirical", the combined $p$-value is computed based on an empirically-derived null distribution using pseudo replicates (using the empirical function). This is appropriate if the

test statistics that generated the $p$-values to be combined can be assumed to follow a multivariate normal distribution and a matrix is available that reflects the correlations among the test statistics (which is specified via the `R` argument). In this case, test statistics are repeatedly simulated from a multivariate normal distribution under the joint null hypothesis, converted into one- or two-sided $p$-values (depending on the `side` argument), and Stouffer's method is applied. Repeating this process `size` times yields a null distribution based on which the combined $p$-value can be computed, or more precisely, estimated, since repeated applications of this method will yield (slightly) different results. To obtain a stable estimate of the combined $p$-value, `size` should be set to a large value (the default is `10000`, but this can be increased for a more precise estimate). If we consider the combined $p$-value an estimate of the 'true' combined $p$-value that would be obtained for a null distribution of infinite size, we can also construct a 95% (pseudo) confidence interval based on a binomial distribution.

If `batchsize` is unspecified, the null distribution is simulated in a single batch, which requires temporarily storing a matrix with dimensions `[size,k]`. When `size*k` is large, allocating the memory for this matrix might not be possible. Instead, one can specify a `batchsize` value, in which case a matrix with dimensions `[batchsize,k]` is repeatedly simulated until the desired size of the null distribution has been obtained.

One can also specify a vector for the `size` argument, in which case one must also specify a corresponding vector for the `threshold` argument. In that case, a stepwise algorithm is used that proceeds as follows. For `j = 1, ..., length(size)`,

1. estimate the combined $p$-value based on `size[j]`
2. if the combined $p$-value is $\geq$ than `threshold[j]`, stop (and report the combined $p$-value), otherwise go back to 1.

By setting `size` to increasing values (e.g., `size = c(1000, 10000, 100000)`) and `threshold` to decreasing values (e.g., `threshold = c(.10, .01, 0)`), one can quickly obtain a fairly accurate estimate of the combined $p$-value if it is far from significant (e.g., $\geq$ .10), but hone in on a more accurate estimate for a combined $p$-value that is closer to 0. Note that the last value of `threshold` should be 0 (and is forced to be inside of the function), so that the algorithm is guaranteed to terminate (hence, one can also leave out the last value of `threshold`, so `threshold = c(.10, .01)` would also work in the example above). One can also specify a single `threshold` (which is replicated as often as necessary depending on the length of `size`).

**Adjustment Based on Multivariate Theory**

When `adjust = "generalized"`, Stouffer's method is computed based on a multivariate normal distribution that accounts for the dependence among the tests, assuming that the test statistics that generated the $p$-values follow a multivariate normal distribution. In that case, `R` needs to be set equal to a matrix that contains the covariances among the $z_i$ values. If a matrix is available that reflects the correlations among the test statistics, this can be converted into the required covariance matrix using the `mvnconv` function. See 'Examples'.

This generalization of Stouffer's method is sometimes called Strube's method, based on Strube (1986), although the paper only describes the method for combining one-sided $p$-values. Both one- and two-sided versions of Strube's method are implemented in **poolr**, but caution must be exercised when applying it to two-sided $p$-values (even if the test statistics follow a multivariate normal distribution, $[z_1, z_2, \ldots, z_k]$ is then not multivariate normal, but this is implicitly assumed by the method).

## Value

An object of class "poolr". The object is a list containing the following components:

| | |
|---|---|
| p | combined $p$-value. |
| ci | confidence interval for the combined $p$-value (only when adjust = "empirical"; otherwise NULL). |
| k | number of $p$-values that were combined. |
| m | estimate of the effective number of tests (only when adjust is one of "nyholt", "liji", "gao", "galwey", or "chen"; otherwise NULL). |
| adjust | chosen adjustment method. |
| statistic | value of the (adjusted) test statistic. |
| fun | name of calling function. |

## Note

The methods underlying adjust = "empirical" and adjust = "generalized" assume that the test statistics that generated the $p$-values to be combined follow a multivariate normal distribution. Hence, the matrix specified via R must be positive definite. If it is not and nearpd = TRUE, it will be turned into one (based on Higham, 2002, and a slightly simplified version of nearPD from the **Matrix** package).

## Author(s)

Ozan Cinar <ozancinar86@gmail.com>
Wolfgang Viechtbauer <wvb@wvbauer.com>

## References

Cinar, O. & Viechtbauer, W. (2022). The poolr package for combining independent and dependent p values. *Journal of Statistical Software*, **101**(1), 1–42. https://doi.org/10.18637/jss.v101.i01

Higham, N. J. (2002). Computing the nearest correlation matrix: A problem from finance. *IMA Journal of Numerical Analysis, 22*(3), 329–343. https://doi.org/10.1093/imanum/22.3.329

Stouffer, S. A., Suchman, E. A., DeVinney, L. C., Star, S. A., & Williams, R. M., Jr. (1949). *The American Soldier: Adjustment During Army Life (Vol. 1)*. Princeton, NJ: Princeton University Press.

Strube, M. J. (1985). Combining and comparing significance levels from nonindependent hypothesis tests. *Psychological Bulletin, 97*(2), 334–341. https://doi.org/10.1037/0033-2909.97.2.334

## Examples

```
# copy p-values and LD correlation matrix into p and r
# (see help(grid2ip) for details on these data)
p <- grid2ip.p
r <- grid2ip.ld

# apply Stouffer's method
```

```
stouffer(p)

# use mvnconv() to convert the LD correlation matrix into a matrix with the
# correlations among the (two-sided) p-values assuming that the test
# statistics follow a multivariate normal distribution with correlation
# matrix r (note: 'side = 2' by default in mvnconv())
mvnconv(r, target = "p", cov2cor = TRUE)[1:5,1:5] # show only rows/columns 1-5

# adjustment based on estimates of the effective number of tests
stouffer(p, adjust = "nyholt", R = mvnconv(r, target = "p", cov2cor = TRUE))
stouffer(p, adjust = "liji",   R = mvnconv(r, target = "p", cov2cor = TRUE))
stouffer(p, adjust = "gao",    R = mvnconv(r, target = "p", cov2cor = TRUE))
stouffer(p, adjust = "galwey", R = mvnconv(r, target = "p", cov2cor = TRUE))
stouffer(p, adjust = "chen",   R = mvnconv(r, target = "p", cov2cor = TRUE))

# setting argument 'm' manually
stouffer(p, m = 12)

# adjustment based on an empirically-derived null distribution (setting the
# seed for reproducibility)
set.seed(1234)
stouffer(p, adjust = "empirical", R = r)

# generate the empirical distribution in batches of size 100
stouffer(p, adjust = "empirical", R = r, batchsize = 100)

# using the stepwise algorithm
stouffer(p, adjust = "empirical", R = r, size = c(1000, 10000, 100000), threshold = c(.10, .01))

# use mvnconv() to convert the LD correlation matrix into a matrix with the
# covariances among the (two-sided) 'z_i' values assuming that the
# test statistics follow a multivariate normal distribution with correlation
# matrix r (note: 'side = 2' by default in mvnconv())
mvnconv(r, target = "z")[1:5,1:5] # show only rows/columns 1-5

# adjustment based on generalized method
stouffer(p, adjust = "generalized", R = mvnconv(r, target = "z"))

# when using mvnconv() inside stouffer() with adjust = "generalized", the
# 'target' argument is automatically set and doesn't need to be specified
stouffer(p, adjust = "generalized", R = mvnconv(r))
```

---

`tippett` *Tippett's Method*

---

## Description

Function to carry out Tippett's method.

## Usage

```
tippett(p, adjust = "none", R, m,
        size = 10000, threshold, side = 2, batchsize, nearpd = TRUE, ...)
```

## Arguments

| | |
|---|---|
| p | vector of length $k$ with the (one- or two-sided) p-values to be combined. |
| adjust | character string to specify an adjustment method to account for dependence. The default is "none", in which case no adjustment is applied. Methods "nyholt", "liji", "gao", "galwey", or "chen" are adjustments based on an estimate of the effective number of tests (see meff). Adjustment method "empirical" uses an empirically-derived null distribution using pseudo replicates. See 'Details'. |
| R | a $k \times k$ symmetric matrix that reflects the dependence structure among the tests. Must be specified if adjust is set to something other than "none". See 'Details'. |
| m | optional scalar (between 1 and $k$) to manually specify the effective number of tests (instead of estimating it via one of the methods described above). |
| size | size of the empirically-derived null distribution. Can also be a numeric vector of sizes, in which case a stepwise algorithm is used. This (and the following arguments) are only relevant when adjust = "empirical". |
| threshold | numeric vector to specify the significance thresholds for the stepwise algorithm (only relevant when size is a vector). |
| side | scalar to specify the sidedness of the $p$-values that are used to simulate the null distribution (2, by default, for two-sided tests; 1 for one-sided tests). |
| batchsize | optional scalar to specify the batch size for generating the null distribution. When unspecified (the default), this is done in a single batch. |
| nearpd | logical indicating if a negative definite R matrix should be turned into the nearest positive definite matrix (only relevant when adjust = "empirical"). |
| ... | other arguments. |

## Details

### Tippett's Method

By default (i.e., when adjust = "none"), the function applies the Tippett's method to the $p$-values (Tippett, 1931). Letting $p_1, p_2, \ldots, p_k$ denote the individual (one- or two-sided) $p$-values of the $k$ hypothesis tests to be combined, the combined $p$-value is then computed with

$$p_c = 1 - (1 - \min(p_1, p_2, \ldots, p_k))^k.$$

Tippett's method asumes that the $p$-values to be combined are independent. If this is not the case, the method can either be conservative (not reject often enough) or liberal (reject too often), depending on the dependence structure among the tests. In this case, one can adjust the method to account for such dependence (to bring the Type I error rate closer to some desired nominal significance level).

### Adjustment Based on the Effective Number of Tests

When adjust is set to "nyholt", "liji", "gao", "galwey", or "chen", Tippett's method is adjusted based on an estimate of the effective number of tests (see meff for details on these methods

for estimating the effective number of tests). In this case, argument R needs to be set to a matrix that reflects the dependence structure among the tests.

There is no general solution for constructing such a matrix, as this depends on the type of test that generated the $p$-values and the sidedness of these tests. If the $p$-values are obtained from tests whose test statistics can be assumed to follow a multivariate normal distribution and a matrix is available that reflects the correlations among the test statistics, then the mvnconv function can be used to convert this correlation matrix into the correlations among the (one- or two-sided) $p$-values, which can then be passed to the R argument. See 'Examples'.

Once the effective number of tests, $m$, is estimated based on R using one of the methods described above, the combined $p$-value is then computed with

$$p_c = 1 - (1 - \min(p_1, p_2, \ldots, p_k))^m.$$

Alternatively, one can also directly specify the effective number of tests via the m argument (e.g., if some other method not implemented in the **poolr** package is used to estimate the effective number of tests). Argument R is then irrelevant and doesn't need to be specified.

**Adjustment Based on an Empirically-Derived Null Distribution**

When adjust = "empirical", the combined $p$-value is computed based on an empirically-derived null distribution using pseudo replicates (using the empirical function). This is appropriate if the test statistics that generated the $p$-values to be combined can be assumed to follow a multivariate normal distribution and a matrix is available that reflects the correlations among the test statistics (which is specified via the R argument). In this case, test statistics are repeatedly simulated from a multivariate normal distribution under the joint null hypothesis, converted into one- or two-sided $p$-values (depending on the side argument), and Tippett's method is applied. Repeating this process size times yields a null distribution based on which the combined $p$-value can be computed, or more precisely, estimated, since repeated applications of this method will yield (slightly) different results. To obtain a stable estimate of the combined $p$-value, size should be set to a large value (the default is 10000, but this can be increased for a more precise estimate). If we consider the combined $p$-value an estimate of the 'true' combined $p$-value that would be obtained for a null distribution of infinite size, we can also construct a 95% (pseudo) confidence interval based on a binomial distribution.

If batchsize is unspecified, the null distribution is simulated in a single batch, which requires temporarily storing a matrix with dimensions [size,k]. When size*k is large, allocating the memory for this matrix might not be possible. Instead, one can specify a batchsize value, in which case a matrix with dimensions [batchsize,k] is repeatedly simulated until the desired size of the null distribution has been obtained.

One can also specify a vector for the size argument, in which case one must also specify a corresponding vector for the threshold argument. In that case, a stepwise algorithm is used that proceeds as follows. For j = 1, ..., length(size),

1. estimate the combined $p$-value based on size[j]
2. if the combined $p$-value is $\geq$ than threshold[j], stop (and report the combined $p$-value), otherwise go back to 1.

By setting size to increasing values (e.g., size = c(1000, 10000, 100000)) and threshold to decreasing values (e.g., threshold = c(.10, .01, 0)), one can quickly obtain a fairly accurate estimate of the combined $p$-value if it is far from significant (e.g., $\geq$ .10), but hone in on a more

accurate estimate for a combined $p$-value that is closer to 0. Note that the last value of `threshold` should be 0 (and is forced to be inside of the function), so that the algorithm is guaranteed to terminate (hence, one can also leave out the last value of `threshold`, so `threshold = c(.10, .01)` would also work in the example above). One can also specify a single `threshold` (which is replicated as often as necessary depending on the length of `size`).

## Value

An object of class `"poolr"`. The object is a list containing the following components:

| | |
|---|---|
| `p` | combined $p$-value. |
| `ci` | confidence interval for the combined $p$-value (only when `adjust = "empirical"`; otherwise `NULL`). |
| `k` | number of $p$-values that were combined. |
| `m` | estimate of the effective number of tests (only when `adjust` is one of `"nyholt"`, `"liji"`, `"gao"`, `"galwey"`, or `"chen"`; otherwise `NULL`). |
| `adjust` | chosen adjustment method. |
| `statistic` | value of the (adjusted) test statistic. |
| `fun` | name of calling function. |

## Note

The method underlying `adjust = "empirical"` assumes that the test statistics that generated the $p$-values to be combined follow a multivariate normal distribution. Hence, the matrix specified via `R` must be positive definite. If it is not and `nearpd = TRUE`, it will be turned into one (based on Higham, 2002, and a slightly simplified version of `nearPD` from the **Matrix** package).

## Author(s)

Ozan Cinar <ozancinar86@gmail.com>
Wolfgang Viechtbauer <wvb@wvbauer.com>

## References

Cinar, O. & Viechtbauer, W. (2022). The poolr package for combining independent and dependent p values. *Journal of Statistical Software*, **101**(1), 1–42. https://doi.org/10.18637/jss.v101.i01

Higham, N. J. (2002). Computing the nearest correlation matrix: A problem from finance. *IMA Journal of Numerical Analysis, 22*(3), 329–343. https://doi.org/10.1093/imanum/22.3.329

Tippett, L. H. C. (1931). *Methods of Statistics*. London: Williams Norgate.

## Examples

```
# copy p-values and LD correlation matrix into p and r
# (see help(grid2ip) for details on these data)
p <- grid2ip.p
r <- grid2ip.ld
```

```
# apply Tippett's method
tippett(p)

# use mvnconv() to convert the LD correlation matrix into a matrix with the
# correlations among the (two-sided) p-values assuming that the test
# statistics follow a multivariate normal distribution with correlation
# matrix r (note: 'side = 2' by default in mvnconv())
mvnconv(r, target = "p", cov2cor = TRUE)[1:5,1:5] # show only rows/columns 1-5

# adjustment based on estimates of the effective number of tests
tippett(p, adjust = "nyholt", R = mvnconv(r, target = "p", cov2cor = TRUE))
tippett(p, adjust = "liji",   R = mvnconv(r, target = "p", cov2cor = TRUE))
tippett(p, adjust = "gao",    R = mvnconv(r, target = "p", cov2cor = TRUE))
tippett(p, adjust = "galwey", R = mvnconv(r, target = "p", cov2cor = TRUE))
tippett(p, adjust = "chen",   R = mvnconv(r, target = "p", cov2cor = TRUE))

# setting argument 'm' manually
tippett(p, m = 12)

# adjustment based on an empirically-derived null distribution (setting the
# seed for reproducibility)
set.seed(1234)
tippett(p, adjust = "empirical", R = r)

# generate the empirical distribution in batches of size 100
tippett(p, adjust = "empirical", R = r, batchsize = 100)

# using the stepwise algorithm
tippett(p, adjust = "empirical", R = r, size = c(1000, 10000, 100000), threshold = c(.10, .01))
```

# Index