Package 'keyperm'

August 30, 2023

Type Package

Title Keyword Analysis Using Permutation Tests

Version 0.1.1

Description Efficient implementation of permutation tests for keyword analysis in corpus linguistics as described in Mildenberger (2023) <arXiv:2308.13383>.

License GPL (>= 2)

Imports slam, tm

LinkingTo Rcpp

RoxygenNote 7.1.0

NeedsCompilation yes

Author Thoralf Mildenberger [aut, cre]
(<https://orcid.org/0000-0001-7242-1873>)

Maintainer Thoralf Mildenberger <mild@zhaw.ch>

Repository CRAN

Date/Publication 2023-08-30 15:40:01 UTC

R topics documented:

`combine_results` *Combine results of permutation test for keyness*

Description

Combine results of two runs of `keyperm()` with `output = "counts"`, possibly with different subsets of terms.

Usage

```
combine_results(results_1, results_2)
```

Arguments

`results_1`	Results from permutation test. Must be of class `keyperm_results_counts` (obtained by setting `output = "counts"` in `keyperm()`)
`results_2`	Results from permutation test. Must be of class `keyperm_results_counts` and have the same scoretype as `results_1`.

Details

Results of two runs of `keyperm()` with `output = "counts"`, i.e. objects of type `keyperm_results_counts` using can be combined using `combine_results()`. For this to make sense, `scoretype` needs to be the same in both results, but terms in both objects need not be the same.

There are at least two important uses of the function:

Parallelization: `keyperm()` is run several times with the same parameters on different cores, using `parallel::mclapply()` or a similar function.

Screening runs: `keyperm()` is first run using a small to medium number of permutations, but considering all terms. Terms with p-values clearly exceeding some reasonable significance threshold are then excluded, and `keyperm()` is run a second time with a (preferably) large number of permutations but using only the remaining terms. The results of both runs can then be combined into one object. The rationale behind this approach is that in many cases small p-values need to be determined with much greater accurary than larger ones far away from significance, especially if a correction for multiple testing is to be applied or the p-values are used for ranking (although they should not...).

Value

An object of class `keyperm_results_counts`

`create_ifl` *Create an Indexed Frequency List*

Description

The keyperm package stores frequency lists in a special data structure called indexed frequency list. This can currently be created from a tdm object as implemented in the tm package.

Indexed frequency lists are essentially frequency lists stored in a three-column format, similar to the simple triplet matrix internally used by tm to store term-document-matrices. The first column stores number of document `i`, second number of term `j` and the third the frequencies with which the term `j` occurs in document `i`. Zero occurences are omitted. All columns contain integers, and the frequency list is sorted by document.

The object returned is of class `indexed_frequency_list`. In addition to the actual frequency list it contains an index for fast access as well as pre-computed total number of tokens per document and total occurences per term.

Usage

```
create_ifl(
  tdm,
  subset_terms = 1:dim(tdm)[1],
  subset_docs = 1:dim(tdm)[2],
  corpus
)
```

Arguments

`tdm`	a tdm-matrix from the tm package. Currently, this is the only supported input, but others may be added in later versions.
`subset_terms`	vector of terms to be considered. Can be integer (indices) or boolean. Terms not included still are counted for total number of token per document.
`subset_docs`	vector of documents to be considered. Can be integer (indices) or boolean. Documents excluded do not contribute to total number of occurences of a term.
`corpus`	vector indicating which documents belong to corpus A (first corpus). Can be integer (indices) or boolean. Currently, only comparisons of two corpora are supported.

Value

A list with class `indexed_frequency_list` containing the following components:

keyness_scores *Calculate observed keyness scores*

Description

Calculates a vector of observed keyness scores for a given pair of corpora.

Usage

```
keyness_scores(ifl, type = "llr", laplace = 1)
```

Arguments

ifl	Indexed frequency list as generated by `create_ifl()`.
type	The type of keyness measure. One of `llr`, `chisq`, `diff`, `logratio` or `ratio`. See details.
laplace	Parameter of laplace correction. Only relevant for `type = "ratio"` and `type = "logratio"`. See details.

Details

Keyness scores are calculated for an Indexed frequency list from a given pair of corpora as generated by `create_ifl()`.

Currently, the following types of scores are supported:

`llr` The log-likelihood ratio

`chisq` The Chi-Square-Statistic

`diff` Difference of relative frequencies

`logratio` Binary logarithm of the ratio of the relative frequencies, possibly using a laplace correction to avoid infinite values.

`ratio` ratio of the relative frequencies, possibly using a laplace correction to avoid infinite values.

`llr` and `chisq` are the test-statistics for a two-by-two contingency table.

	corpus A	corpus B	TOTAL
term of interest	o_{11}	o_{12}	r_1
other tokens	o_{21}	o_{22}	r_2
TOTAL	c_1	c_2	N

Both measure deviations from equal proportions but do not indicate the direction. For `llr`, the correct version using terms for all four fields of the table is used, not the version using only two terms that is sometimes used in corpus linguistics:

$$llr = -2 * (o11 * log(o11/e11) + o12 * log(o12/e12) + o21 * log(o21/e21) + o22 * log(o22/e22))$$

where $oij * log(oij/eij) = 0$ if $oij = 0$.

`chisq` is the usual Chi-Square statistic for a test of independece / homogeneity:

$$chisq = (o11 - e11)^2/e11 + (o12 - e12)^2/e12 + (o21 - e21)^2/e21 + (o22 - e22)^2/e22$$

Here, oij are the observed counts as given above and eij are the correpsonding expected values under an independence / homogeneity assumption.

`diff` and `logratio` are measures of the effect size, but using the permutation approach implemented here a p-value can be calculated as well. Both indicate the direction of the effect, and can be used for one- or two-sided tests.

$$diff = o11/c1 - o12/c2$$

`logratio` is based on a ratio of ratios and would be infinite when a term does not occur in either of the two corpora, irrespective of number of occurences in the other corpus. Hence, we use a laplace correction adding a (not neccesarily integer) number k of ficticious occurences to both corpora:

$$logratio = log2(((o11 + k)/(c1 + k))/((o12 + k)/(c2 + k)))$$

where $o11$ and $o12$ are the number of occurences of the term of interest in Corpora A and B and $c1$ and $c2$ are the total numbers of tokens in A and B. Setting k to zero corresponds to the usual logratio (which may be infinite). k is given by the `laplace` argument and defaults to one, meaning one ficticious occurence is added to either corpus. Doing so prevents infinite values but has little effect when the number of occurences is large.

`ratio` is the same as `logratio` but omits the logarithm:

$$ratio = ((o11 + k)/(c1 + k))/((o12 + k)/(c2 + k))$$

This leads to the same p-values but is faster to compute.

Value

a numerical vector of the scores, one for each term. Terms are stored in the names attribute.

`keyperm` *Calculate the permutation distribution for a keyness measure*

Description

Calculate the permutation distributions of a given keyness measure for each term by shuffeling the copus labels. Number of documents per corpus is kept constant.

Usage

```
keyperm(ifl, observed, type = "llr", laplace = 1, output = "counts", nperm)
```

Arguments

`ifl`	Indexed frequency list as generated by `create_ifl()`.
`observed`	The vector of observed values of the keyness scores as generarted by `keyness_scores()`
`type`	The type of keyness measure. One of `llr`, `chisq`, `diff`, `logratio` or `ratio`. See details.
`laplace`	Parameter of laplace correction. Only relevant for `type = "ratio"` and `type = "logratio"`. See details.
`output`	The type of output. For `output = "full"` a matrix with all generated scores is returned, for `output = "counts"` a matrix with three columns counting the number of permutations for which the score is strictly smaller than, equal to or strictly larger than the observed value.
`nperm`	The number of permutations to generate.

Details

While usually keyness scores are judged by reference to a limiting null distribution under a token-by-token-sampling model, this implementation approximates the null distribution under a document-by-document sampling model. The permutation distributions of a given keyness measure for each term is calculated by repeatedly shuffeling the copus labels. Number of documents per corpus is kept constant.

Currently, the following types of scores are supported:

`llr` The log-likelihood ratio

`chisq` The Chi-Square-Statistic

`diff` Difference of relative frequencies

`logratio` Binary logarithm of the ratio of the relative frequencies, possibly using a laplace correction to avoid infinite values.

`ratio` ratio of the relative frequencies, possibly using a laplace correction to avoid infinite values.

`llr` and `chisq` are the test-statistics for a two-by-two contingency table.

	corpus A	corpus B	TOTAL
term of interest	o_{11}	o_{12}	r_1
other tokens	o_{21}	o_{22}	r_2
TOTAL	c_1	c_2	N

Both measure deviations from equal proportions but do not indicate the direction. For `llr`, the correct version using terms for all four fields of the table is used, not the version using only two terms that is sometimes used in corpus linguistics:

$$llr = -2 * (o11 * log(o11/e11) + o12 * log(o12/e12) + o21 * log(o21/e21) + o22 * log(o22/e22))$$

where $oij * log(oij/eij) = 0$ if $oij = 0$.

`chisq` is the usual Chi-Square statistic for a test of independece / homogeneity:

$$chisq = (o11 - e11)^2/e11 + (o12 - e12)^2/e12 + (o21 - e21)^2/e21 + (o22 - e22)^2/e22$$

Both `llr` and `chisq` asymptotically follow a Chi-Square-Distribution with 1 degree of freedom if the null hypothesis of equal frequencies in both populations is true and the corpora are drawn iid token-by-token. In contrast, In contrast, the p-values calculated here are obtained based on a document-by-document sampling model, which is arguably more realistic in many cases.

Here, oij are the observed counts as given above and eij are the correpsonding expected values under an independence / homogeneity assumption.

`diff` and `logratio` are measures of the effect size, but using the permutation approach implemented here a p-value can be calculated as well. Both indicate the direction of the effect, and can be used for one- or two-sided tests.

$$diff = o11/c1 - o12/c2$$

`logratio` is based on a ratio of ratios and would be infinite when a term does not occur in either of the two corpora, irrespective of number of occurences in the other corpus. Hence, we use a laplace correction adding a (not neccesarily integer) number k of ficticious occurences to both corpora:

$$logratio = log2(((o11 + k)/(c1 + k))/((o12 + k)/(c2 + k)))$$

where $o11$ and $o12$ are the number of occurences of the term of interest in Corpora A and B and $c1$ and $c2$ are the total numbers of tokens in A and B. Setting k to zero corresponds to the usual logratio (which may be infinite). k is given by the `laplace` argument and defaults to one, meaning one ficticious occurence is added to either corpus. Doing so prevents infinite values but has little effect when the number of occurences is large.

`ratio` is the same as `logratio` but omits the logarithm:

$$ratio = ((o11 + k)/(c1 + k))/((o12 + k)/(c2 + k))$$

This leads to the same p-values but is faster to compute.

Value

A numeric matrix with number of rows equal to the number of terms. The columns contain either all permutation values of the keyness score (`output = "full"`) or the number of permutations for which the score is strictly smaller than, equal to or strictly larger than the observed value (`output = "counts"`).

`p_value` *Convert results of permutation test for keyness to p-values*

Description

Calculate p-values from the results of `keyperm()` with `output = "counts"`.

Usage

```
p_value(results, alternative = NULL)
```

Arguments

`results`	results from permutation test. Must be of class `keyperm_results_counts` (obtained by setting `output = "counts"` in `keyperm()`)
`alternative`	direction of p-value to calculate, one of `"two.sided"`, `"greater"`, `"less"`. Defaults depend on the scores used. See details.

Details

Valid (slightly conservative) p-values are calculated from an object of class `keyperm_results_counts` that is obtained by running `keyperm()` with `output = "counts"`. `keyperm_results_counts` is a matrix with three columns that contain the counts of generated permutations that resulted in a score strictly less than, equal to and strictly greater that the observed score.

For a one-sided p-value we use

$$pvalue_greater = (no.greater + no.equal + 1)/(no.ofperms + 1)$$

or

$$pvalue_less = (no.less + no.equal + 1)/(no.ofperms + 1)$$

Adding 1 in both the numerator and denominator amounts to including the observed values. This results in a slightly conservative p-value, but guarantees that the test is valid for any number of random permutations. It also means that never a p-value of zero is returned but the minimum possible p-value is $1/(no.perms + 1)$.

The two-sided p-value is calculated by

$$pvalue_twosided = 2 * min(pvalue_less, pvalue_greater)$$

(values larger than 1 are set to 1).

If `alternative` is not specified by the user, different defaults are used depending on the scoretype (which is included as an attribute in the `keyperm_results_counts` object). Since for `llr` and `chisq`, large values indicate a great deviation from equal frequencies without indicating the direction, `alternative == "greater"` is basically the only alternative of interest and is used as a default. For `diff` and `logratio` large absolute values indicate a great deviation from equal frequencies, and positive values correspond to higher frequencies in A, negative frequencies correspond to a higher frequency in B. For these scoretypes, the default is `alternative = "two.sided"`. If only "positive" keywords for A with respect to B are desired, use `alternative = "less"`.

Value

a numeric vector of p-values.

Index