Package ‘corpora’

June 10, 2025

Type Package

Title Statistics and Data Sets for Corpus Frequency Data

Version 0.7

Depends R (>= 3.5.0)

Imports methods, stats, utils, grDevices

Date 2025-06-09

Description Utility functions for the statistical analysis of corpus frequency data.
This package is a companion to the open-source course ``Statistical Inference:
A Gentle Introduction for Computational Linguists and Similar Creatures'' ('SIGIL').

License GPL-3

URL http://SIGIL.R-Forge.R-Project.org/

LazyData yes

Encoding UTF-8

NeedsCompilation no

Author Stephanie Evert [cre, aut] (ORCID:
<https://orcid.org/0000-0002-4192-2437>)

Maintainer Stephanie Evert <stephanie.evert@fau.de>

Repository CRAN

Date/Publication 2025-06-10 10:30:02 UTC

Contents

`corpora-package` *corpora: Statistical Inference from Corpus Frequency Data*

Description

The `corpora` package provides a collection of functions for statistical inference from corpus frequency data, as well as some convenience functions and example data sets.

It is a companion package to the open-source course *Statistical Inference: a Gentle Introduction for Linguists and similar creatures* originally developed by Marco Baroni and Stephanie Evert. Statistical methods implemented in the package are described and illustrated in the units of this course.

Starting with version 0.6 the package also includes best-practice implementations of various corpus-linguistic analysis techniques.

Details

An overview of some important functions and data sets included in the `corpora` package. See the package index for a complete listing.

Analysis functions:

- `keyness()` provides reference implementations for best-practice keyness measures, including the recommended LRC measure (Evert 2022)
- `am.score()` computes various standard association measures for collocation analysis (Evert 2004, 2008) as well as user-defined formulae
- `binom.pval()` is a vectorised function that computes p-values of the binomial test more efficiently than `binom.test` (using central p-values in the two-sided case)
- `fisher.pval()` is a vectorised function that efficiently computes p-values of Fisher's exact test on 2×2 contingency tables for large samples (using central p-values in the two-sided case)
- `prop.cint()` is a vectorised function that computes multiple binomial confidence intervals much more efficiently than `binom.test`
- `z.score()` and `z.score.pval()` can be used to carry out a z-test for a single proportion (as an approximation to `binom.test`)
- `chisq()` and `chisq.pval()` are vectorised functions that compute the test statistic and p-value of a chi-squared test for 2×2 contingency tables more efficiently than `chisq.test`

Utility functions:

- `cont.table()` creates 2×2 contingency tables for frequency comparison test that can be passed to `chisq.test` and `fisher.test`
- `sample.df()` extracts random samples of rows from a data frame
- `qw()` splits a string on whitespace or a user-specified regular expression (similar to Perl's `qw//` construct)
- `corpora.palette()` provides some nice colour palettes (better than R's default colours)
- `rowVector()` and `colVector()` convert a vector into a single-row or single-column matrix

Data sets:

- Several data sets based on the British National Corpus, including complete metadata for all 4048 text files (`BNCmeta`), per-text frequency counts for a number of linguistic corpus queries (`BNCqueries`), and relative frequencies of 65 lexico-grammatical features for each text (`BNCbiber`)
- Frequency counts of passive constructions in all texts of the Brown and LOB corpora (`BrownLOBPassives`) for frequency comparison with regression models, complemented by distributional features (`DistFeatBrownFam`) as additional predictors
- A small text corpus of *Very Short Stories* in the form of a data frame `VSS`, with one row for each token in the corpus.
- Small example tables to illustrate frequency comparison of lexical items (`BNCcomparison`) and collocation analysis (`BNCInChargeOf`)
- `KrennPPV` is a data set of German verb-preposition-noun collocation candidates with manual annotation of true positives and pre-computed association scores
- Three functions for generating large synthetic data sets used in the SIGIL course: `simulated.census()`, `simulated.language.course()` and `simulated.wikipedia()`

Author(s)

Stephanie Evert (https://purl.org/stephanie.evert)

References

The official homepage of the corpora package and the SIGIL course is http://SIGIL.R-Forge.R-Project.org/.

---

am.score *Compute association scores for collocation analysis (corpora)*

Description

Compute a wide range of established association measures (Evert 2004, 2008) for collocation analysis based on the co-occurrence frequency of two lexical items as well as their individual "marginal" frequencies. Frequency data can be provided in a number of different ways for convenience, and user-defined measures can be specified as mathematical formulae using the flexible notation introduced by Evert (2004).

Usage

```
am.score(w1, w2, f, f1, f2, N, measure,
         span.size=1, log=FALSE, labels=FALSE,
         conf.level=.95, p.adjust=TRUE,
         param=list())

builtin.am
```

Arguments

w1	a character vector specifying the first lexical item of each co-occurrence pair. If of length 1, it is assumed to be the same for all pairs. May be omitted if f1 is an unnamed vector parallel to f.
w2	a character vector specifying the second lexical item of each co-occurrence pair. If of length 1, it is assumed to be the same for all pairs. May be omitted if f2 is an unnamed vector parallel to f.
f	a numeric vector specifying the co-occurrence frequency of each pair of lexical items (w_1, w_2)
f1	a numeric vector of first marginal frequencies f_1, labelled with the unique strings occurring in w1. Alternatively, an unlabelled numeric vector of the same length as f or of length 1 (implying that all pairs have the same first marginal f_1).
f2	a numeric vector of second marginal frequencies f_2, labelled with the unique strings occurring in w2. Alternatively, an unlabelled numeric vector of the same length as f or of length 1 (implying that all pairs have the same first marginal f_2).

`N`	a numeric vector specifying the sample size N underlying the contingency tables. It is normally a single value, which is the same for all co-occurrence pair, but a vector of the same length as `f` may also be specified.
`measure`	either a string specifying one of the built-in association measures or a function implementing a user-defined measure (see “Details” below)
`span.size`	for surface co-occurrence, specify the total size of each collocational span in tokens (e.g. `span.size=6` for a L3/R3 span) in order to adjust first marginal frequencies as explained by Evert (2008: Sec. 4.1 + 5.1). No adjustment should be specified if contingency tables have already been obtained according to Evert (2004: 68-70).
`log`	if TRUE, apply a signed log transformation $h(x) = \mathrm{sgn}(x) \cdot \log_2(\|x\| + 1)$ to the computed association scores in order to deskew their distribution.
`labels`	if TRUE, add labels to the returned vector of assocation scores, constructed from `w1` and `w2`
`conf.level`	the desired confidence level for association measures based on confidence intervals such as `LRC` (defaults to 95%)
`p.adjust`	if `TRUE`, apply a Bonferroni correction in order to control the family-wise error rate across all statistical tests carried out across the data set. This parameter is used by measures based on confidence intervals and measures that return (log) p-values.
`param`	a list of additional parameters passed to the ‘param‘ argument of the selected association measure, with ‘conf.level‘ and ‘p.adjust‘ added automatically.

Details

This function computes a range of standard association measures for collocation analysis, as well as user-defined measures specified by mathematical formulae. It assumes familiarity with the basic approaches of collocation analysis as explained in Evert (2008) or in Unit 4 of the SIGIL course at `https://SIGIL.R-Forge.R-Project.org/`, as well as with the notation for contingency tables introduced there.

While the main purpose of the function is to determine the most strongly associated items in a large data set of co-occurrence pairs, it can also be applied to the collocates of a single node word. In this case, `w1` and `f1` only consist of a single item (the node word and its marginal frequency). Keep in mind that marginal frequencies `f2` still need to be obtained from the entire corpus, not just from co-occurrences with the node.

Frequency data:

Frequency signatures for all co-occurrence pairs (w_1, w_2) must be provided by the user in terms of a vector of co-occurrence frequencies `f`, as well as vectors of marginal frequencies `f1` (for the lexical items w_1) and `f2` (for the lexical items w_2). The underlying sample size `N` of the contingency tables also has to be specified, which is usually the same for all co-occurrence pairs.

Marginal frequencies are normally given as numeric vectors labelled with the distinct strings from `w1` (for `f1`) and the distinct strings from `w2` (for `f2`), respectively. These vectors are used to look up the appropriate marginal frequencies for each co-occurrence item based on `w1` and `w2`. Alternatively, `f1` and `f2` can be unlabelled vectors of the same length as `f` (or of length 1), in which case `w1` and `w2` may be omitted.

Because of these different invocations, it is recommeded to always use named arguments to `am.score()`.

Contingency tables are automatically derived from the frequency signatures internally, and can be accessed by user-defined measures using the flexible and convenient notation of Evert (2004, 2008).

Types of co-occurrence:

Evert (2008: Sec. 3) distinguishes three types of co-occurrence, which have implications on how frequency signatures and contingency tables have to be constructed. While reading the full description in Evert (2008) is strongly encouraged, concise summaries are provided here for reference.

Syntactic co-occurrence only considers pairs of lexical items in a specific syntactic relation to each other, such as adjective-noun or verb-object. Frequency data are obtained by extracting all instances of this syntactic relation from a corpus, resulting in a list of pair tokens. The sample size N is the total number of such pair tokens, while co-occurrency frequency f is obtained by counting the pairs, and marginal frequencies f_1, f_2 by counting lexical items in the first or second element of the pairs, respectively. For a "word sketch" (or "word profile") collecting multiple syntactic relations, the collocation analysis for each relation has to be carried out separately.

Textual co-occurrence refers to co-occurrence within text segments such as sentences, paragraphs, tweets, or chapters. In this case, the sample size N corresponds to the total number of relevant text segments, and f_1, f_2, f are "document frequencies", i.e. the number of text segments in which w_1, w_2, or both lexical items occur.

Surface co-occurrence usually considers w_1 as the "node" of the analysis and looks for collocates w_2 that occur within a certain number of tokens around the node (the "collocational span"). The span can be symmetric, e.g. L3/R3 (three tokens to the left and right of the node) or asymmetric, e.g. L0/R5 (five tokens to the right of the node only). The span size k is the total number of tokens in each span (6 for L3/R3, and 5 for L0/R5). Evert (2008) recommends to obtain frequency signatures as follows: f from the co-occurrence counts within collocational spans, f_1 as the overall corpus frequency of each distinct node, f_2 as the overall corpus frequency of each distinct collocate, and sample size N as the total number of tokens in the corpus. Contingency tables and expected frequencies then need to be adjusted for the span size (Evert 2008: Sec. 4.1 + 5.1), which can easily be achieved with the argument `span.size=k`.

Built-in association measures:

The `am.score()` function includes a wide selection of built-in association measures, which can be selected via their name in the `measure` argument. Please refer to Evert (2004, 2008) or `http://www.collocations.de/AM/` for mathematical details on these measures and their properties as well as full equations. Formulae shown below use the flexible notation for contingency tables introduced there.

Some measures take additional paramters specified in the `param` argument. Measures that return (log) p-values apply a Bonferroni correction if `p.adjust=TRUE` (the default). Measures based on confidence intervals use the confidence level specified in `conf.level`, which is also adjusted by the Bonferrroni correction.

`MI` (Pointwise) **mutual information**, the binary logarithm of the ratio between observed and expected co-occurrence frequency:

$$\log_2 \frac{O_{11}}{E_{11}}$$

Pointwise MI has a very strong bias towards pairs with low expected co-occurrence frequency (because of E_{11} in the denominator). It should usually be combined with frequency thresholds on f_1 and f_2.

`MI.k` A heuristic variant of pointwise **mutual information** intended to counteract the low-frequency bias by raising O_{11} to power k:

$$\log_2 \frac{(O_{11})^k}{E_{11}}$$

The exponent k can be specified as a user parameter (e.g. `param=list(k=3)` for the popular MI3 measure) and defaults to $k = 2$ otherwise.

`G2` The G^2 statistic of a likelihood ratio test for independence of rows and columns in a contingency table, which is very popular in computational linguistics under the name **log-likelihood**:

$$\pm 2 \left(\sum_{ij} O_{ij} \cdot \log \frac{O_{ij}}{E_{ij}} \right)$$

This implementation computes *signed* association scores, which are negative iff $O_{11} < E_{11}$. Log-likelihood has a strong bias towards high co-occurrence frequency and often produces a highly skewed distribution of scores. It may therefore be advisable to combine it with an additional log transformation (`log=TRUE`).

`G2.pv` The **p-values** corresponding to the G^2 scores of the likelihood ratio test. In order to achieve sensible scaling and ensure that larger values correspond to higher collocability, the negative base-10 logarithm $-\log_{10} p$ is returned, with negative sign iff $O_{11} < E_{11}$. Family-wise error rates are controlled with a Bonferroni correction to the p-values if `p.adjust=TRUE` (the default). Note that scores above 2 correspond to a significance level $p < .01$.

`Fisher.pv` The **p-values** of a one-sided **Fisher's exact test** for independence of rows and columns in a contingency table, conditioned on the marginal frequencies. As usual for p-values, the negative base-10 logarithm $-\log_{10} p$ is returned, which is always non-negative (keep in mind that the one-sided test only detects positive association). Family-wise error rates are controlled with a Bonferroni correction to the p-values if `p.adjust=TRUE` (the default).

`simple.ll` Simple **log-likelihood** (Evert 2008: 1225):

$$\pm 2 \left(O_{11} \cdot \log \frac{O_{11}}{E_{11}} - (O_{11} - E_{11}) \right)$$

This measure provides a good approximation to the full log-likelihood measure (Evert 2008: 1235), but can be computed much more efficiently. Like `G2`, this measure computes *signed* association scores and has a strong bias towards high co-occurrence frequency.

`t` The **t-score** association measure, which is popular for collocation identification in computational lexicography:

$$\frac{O_{11} - E_{11}}{\sqrt{O_{11}}}$$

T-score is known to filter out low-frequency data effectively.

`X2` The X^2 statistic of Pearson's **chi-squared** test for independence of rows and columns in a contingency table, with Yates's correction applied:

$$\pm \frac{N\big(|O_{11}O_{22} - O_{12}O_{21}| - N/2\big)^2}{R_1 R_2 C_1 C_2}$$

This implementation computes *signed* association scores, which are negative iff $O_{11} < E_{11}$. The formula above gives a more compact form of Yates's correction than the familiar sum over the four cells of the contingency table. See also `chisq`.

`z` The **z-score** association measure, based on a normal approximation to the binomial distribution of co-occurrence by chance:

$$\frac{O_{11} - E_{11}}{\sqrt{E_{11}}}$$

Z-score has a strong bias towards pairs with low expected co-occurrence frequency (because of E_{11} in the denominator). Like pointwise MI, it should usually be combined with frequency thresholds on f_1 and f_2.

`Dice` The **Dice coefficient** of association, which corresponds to the harmonic mean of the conditional probabilities $P(w_2|w_1)$ and $P(w_1|w_2)$:

$$\frac{2O_{11}}{R_1 + C_1}$$

`odds.ratio` Discounted log **odds ratio**, an effect-size measure that is sensitive to small marginal frequencies:

$$\log \frac{(O_{11} + \frac{1}{2})(O_{22} + \frac{1}{2})}{(O_{12} + \frac{1}{2})(O_{21} + \frac{1}{2})}$$

`DP` The asymmetric **Delta P** measure $\Delta P_{2|1} = P(w_2|w_1) - P(w_2|\neg w_1)$ proposed by Gries (2013: 143-144):

$$\frac{O_{11}}{R_1} - \frac{O_{21}}{R_2}$$

`LRC` The **conservative LogRatio** (LRC) keyness measure (see `keyness`) can also be applied as an association measure. The implementation here always computes `PositiveLRC`, i.e. the lower boundary of a one-sided confidence interval for $\log_2 r$, in order to ensure a sensible and consistent scaling of the association score. Please keep in mind that negative scores do not necessarily indicate negative association ($O_{11} < E_{11}$), but rather lack of significant evidence for a positive association.
Confidence intervals are determined at confidence level `conf.level` and adjusted with a Bonferroni correction if `p.adjust=TRUE` (the default). Note that LRC is applied to the rows of the contingency table rather than its columns, in order to obtain relative risk for w_2 given w_1.

User-defined association measures:

User-defined association measures can be applied by passing a suitable function in the `measure` argument. This function needs to be fully vectorised and will be applied to all co-occurrence pairs in the data set.

It can use any of following arguments to access the contingency tables of observed and expected frequencies, following the notation of Evert (2008):

`O, E` observed and expected co-occurrence frequency

`R1, R2, C1, C2` the row and column marginals of the contingency table

`N` sample size

`f, f1, f2` the frequency signature of each co-occurrence pair, a different notation for $f = O$, $f_1 = R_1$ and $f_2 = C_1$

`O11, O12, O21, O22` the contingency table of observed frequencies

`E11,E12,E21,E22` the contingency table of expected frequencies

`param` a list with additional user-specified parameters, always including `conf.level` and `p.adjust`

`...` must be specified to ignore unused arguments

Except for `param`, all these arguments will be numeric vectors of the same length, and the function must return a numeric vector of the same length.

For example, the built-in MI measure could also be implemented with the user function

```
my.MI <- function (O, E, ...) log2(O / E)
```

As a matter of fact, all built-in association measures are implemented in this way. The corresponding formulae can be obtained from the list `builtin.am`.

Bonferroni correction:

If `p.adjust=TRUE`, statistical inference is corrected for multiple testing in order to control **family-wise error rates**. This applies in particular to association measures based on confidence intervals (such as `LRC`) and versions that return (log) p-values (such as `G2-pv`). Note that the `G2` scores themselves are never adjusted.

Family size m is automatically determined from the number of co-occurrence pairs processed in a single function call. Alternatively, the family size can be specified explicitly in the `p.adjust` argument, e.g. if a large data set is processed in multiple batches, or `p.adjust=FALSE` can be used to disable the correction.

For the adjustment, a highly conservative Bonferroni correction $\alpha' = \alpha/m$ is applied to significance levels. Since the large candidate sets and sample sizes often found in corpus linguistics tend to produce large numbers of false positives, this conservative approach is considered to be useful.

Value

A numeric vector of the same length as `f` containing the desired association scores for all co-occurrence pairs. The vector is labelled with the corresponding co-occurrence pairs if `label=TRUE`.

For all standard association measures, larger values indicate higher collocability. There is usually no meaningful scale for interpretation of the scores, though many measures return positive scores for higher than expected co-occurrence frequency and negative scores for lower than expected co-occurrence frequency.

P-values are returned on a negative log-10 scale ($-\log_{10} p$), so a score of 2 corresponds to $p = .01$ and a score of 3 to $p = .001$. The score for $p = .05$ is approximately 1.3. These cutoffs can be used to implement a significance filter for collocation candidates.

Author(s)

Stephanie Evert (`https://purl.org/stephanie.evert`)

References

`http://www.collocations.de/AM/`

Evert, S. (2004). *The Statistics of Word Cooccurrences: Word Pairs and Collocations*. Dissertation, Institut für maschinelle Sprachverarbeitung, University of Stuttgart. Published in 2005, URN urn:nbn:de:bsz:93-opus-23714. Available from `http://www.collocations.de/phd.html`.

Evert, S. (2008). Corpora and collocations. In Lüdeling, A. and Kytö, M., editors, *Corpus Linguistics. An International Handbook*, chapter 58, pages 1212–1248. Mouton de Gruyter, Berlin, New York. (extended manuscript (PDF))

Evert, S. (2022). Measuring keyness. In *Digital Humanities 2022: Conference Abstracts*, pages 202-205, Tokyo, Japan / online. `https://osf.io/cy6mw/`

Gries, S. T. (2013). 50-something years of work on collocations: What is or should be next .... *International Journal of Corpus Linguistics*, 18(1):137–165.

See Also

`prop.cint`, which is used by some of the association measures based on confidence intervals; `keyness` for details on the LRC association measure (known as `PositiveLRC` there).

For a gentle introduction to assocation measures, contingency tables, and different notions of co-occurrence see Evert (2008) or Unit 4 of the SIGIL course at `https://SIGIL.R-Forge.R-Project.org/`. The unit also includes a worked example carrying out several collocation analyses with real-life corpus data.

Examples

```
## surface collocations with L2/R2 window
head(SurfaceColloc$cooc, 10) # table of word pairs with co-occurrence frequencies
head(SurfaceColloc$f1) # tables of marginal frequencies
head(SurfaceColloc$f2)

# add association scores (MI and p-values from log-likelihood test) to data frame
# keep in mind that we need to adjust expected frequencies for the total span size of 4 tokens
colloc <- transform(
  SurfaceColloc$cooc,
  MI = am.score(w1, w2, f, SurfaceColloc$f1, SurfaceColloc$f2, SurfaceColloc$N,
               "MI", span.size=4),
  pval = am.score(w1, w2, f, SurfaceColloc$f1, SurfaceColloc$f2, SurfaceColloc$N,
                 "G2.pv", span.size=4)
)
colloc <- colloc[order(colloc$MI, decreasing=TRUE), ] # ordered by MI scores
colloc

# apply significance filter
subset(colloc, pval > 3) # significant positive association at p < .001

# can also obtain labelled vector of scores
with(SurfaceColloc$cooc,
     am.score(w1, w2, f, SurfaceColloc$f1, SurfaceColloc$f2, SurfaceColloc$N,
              "z", span.size=4, labels=TRUE))


## syntactic collocations from pre-computed frequency signatures (must specify named arguments!)
head(KrennPPV)
colloc <- transform(
  KrennPPV[, 1:9], # omit pre-computed association scores
  t = am.score(f=freq, f1=f.PP, f2=f.verb, N=N, measure="t"),
  OR = am.score(f=freq, f1=f.PP, f2=f.verb, N=N, measure="odds.ratio"))
```

```
all.equal(colloc$t, KrennPPV$t.score) # compare with pre-computed scores

colloc <- colloc[order(colloc$t, decreasing=TRUE), ]
head(colloc, 20) # top collocates (among all pairs)
subset(colloc, verb == "legen") # top collocates of verb "legen"


## collocations of "in charge of" with LRC as an association measure
colloc <- transform(
  BNCInChargeOf, # reconstruct frequency signature from O11, O12, C1, C2
  f = f.in,
  f1 = N.in,
  f2 = f.in + f.out,
  N = N.in + N.out
)
colloc <- transform(
  colloc,
  LRC = am.score("in charge of", collocate, f, f1, f2, N, "LRC"))
colloc <- colloc[order(colloc$LRC, decreasing=TRUE), ]
head(colloc, 20)
```

binom.pval *P-values of the binomial test for frequency counts (corpora)*

Description

This function computes the p-value of a binomial test for frequency counts. In the two-sided case, a "central" p-value (Fay 2010) provides better numerical efficiency than the likelihood-based approach of `binom.test` and is always consistent with confidence intervals.

Usage

```
binom.pval(k, n, p = 0.5,
           alternative = c("two.sided", "less", "greater"))
```

Arguments

`k`	frequency of a type in the corpus (or an integer vector of frequencies)
`n`	number of tokens in the corpus, i.e. sample size (or an integer vector specifying the sizes of different samples)
`p`	null hypothesis, giving the assumed proportion of this type in the population (or a vector of proportions for different types and/or different populations)
`alternative`	a character string specifying the alternative hypothesis; must be one of `two.sided` (default), `less` or `greater`

Details

For `alternative="two.sided"` (the default), a “central” p-value is computed (Fay 2010: 53f), which differs from the likelihood-based two-sided p-value determined by `binom.test` (the “minlike” method in Fay’s terminology). This approach has two advantages: (i) it is numerically robust and efficient, even for very large samples and frequency counts; (ii) it is always consistent with Clopper-Pearson confidence intervals (see examples below).

Value

The p-value of a binomial test applied to the given data (or a vector of p-values).

Author(s)

Stephanie Evert (`https://purl.org/stephanie.evert`)

References

Fay, Michael P. (2010). Two-sided exact tests and matching confidence intervals for discrete data. *The R Journal*, **2**(1), 53-58.

See Also

`z.score.pval`, `prop.cint`

Examples

```
# inconsistency btw likelihood-based two-sided binomial test and confidence interval
binom.test(2, 10, p=0.555)

# central two-sided test as implemented by binom.pval is always consistent
binom.pval(2, 10, p=0.555)
prop.cint(2, 10, method="binomial")
```

BNCbiber *Biber’s (1988) register features for the British National Corpus*

Description

This data set contains a table of the relative frequencies (per 1000 words) of 65 linguistic features (Biber 1988, 1995) for each text document in the British National Corpus (Aston & Burnard 1998).

Biber (1988) introduced these features for the purpose of a multidimensional register analysis. Variables in the data set are numbered according to Biber’s list (see e.g. Biber 1995, 95f).

Feature frequencies were automatically extracted from the British National Corpus using query patterns based on part-of-speech tags (Gasthaus 2007). Note that features 60 and 65 had to be omitted because they cannot be identified with sufficient accuracy by the automatic methods. For further information on the extraction methodology, see Gasthaus (2007, 20-21). Unfortunately, the original data set and the Python scripts used for feature extraction do not seem to be publicly available any more; the version included here contains some bug fixes.

Usage

```
BNCbiber
```

Format

A numeric matrix with 4048 rows and 65 columns, specifying the relative frequencies (per 1000 words) of 65 linguistic features. Documents are listed in the same order as the metadata in `BNCmeta` and rows are labelled with text IDs, so it is straightforward to combine the two data sets.

	A. Tense and aspect markers
`f_01_past_tense`	Past tense
`f_02_perfect_aspect`	Perfect aspect
`f_03_present_tense`	Present tense
	B. Place and time adverbials
`f_04_place_adverbials`	Place adverbials (e.g., *above, beside, outdoors*)
`f_05_time_adverbials`	Time adverbials (e.g., *early, instantly, soon*)
	C. Pronouns and pro-verbs
`f_06_first_person_pronouns`	First-person pronouns
`f_07_second_person_pronouns`	Second-person pronouns
`f_08_third_person_pronouns`	Third-person personal pronouns (excluding *it*)
`f_09_pronoun_it`	Pronoun *it*
`f_10_demonstrative_pronoun`	Demonstrative pronouns (*that, this, these, those* as pronouns)
`f_11_indefinite_pronoun`	Indefinite pronounes (e.g., *anybody, nothing, someone*)
`f_12_proverb_do`	Pro-verb *do*
	D. Questions
`f_13_wh_question`	Direct *wh*-questions
	E. Nominal forms
`f_14_nominalization`	Nominalizations (ending in *-tion, -ment, -ness, -ity*)
`f_15_gerunds`	Gerunds (participial forms functioning as nouns)
`f_16_other_nouns`	Total other nouns
	F. Passives
`f_17_agentless_passives`	Agentless passives
`f_18_by_passives`	*by*-passives
	G. Stative forms
`f_19_be_main_verb`	*be* as main verb
`f_20_existential_there`	Existential *there*
	H. Subordination features
`f_21_that_verb_comp`	*that* verb complements (e.g., *I said that he went.*)
`f_22_that_adj_comp`	*that* adjective complements (e.g., *I'm glad that you like it.*)
`f_23_wh_clause`	*wh*-clauses (e.g., *I believed what he told me.*)
`f_24_infinitives`	Infinitives
`f_25_present_participle`	Present participial adverbial clauses (e.g., *Stuffing his mouth with cookies, Joe ran out th*
`f_26_past_participle`	Past participial adverbial clauses (e.g., *Built in a single week, the house would stand for*
`f_27_past_participle_whiz`	Past participial postnominal (reduced relative) clauses (e.g., *the solution produced by thi*
`f_28_present_participle_whiz`	Present participial postnominal (reduced relative) clauses (e.g., *the event causing this de*
`f_29_that_subj`	*that* relative clauses on subject position (e.g., *the dog that bit me*)
`f_30_that_obj`	*that* relative clauses on object position (e.g., *the dog that I saw*)

`f_31_wh_subj`	*wh* relatives on subject position (e.g., *the man who likes popcorn*)
`f_32_wh_obj`	*wh* relatives on object position (e.g., *the man who Sally likes*)
`f_33_pied_piping`	Pied-piping relative clauses (e.g., *the manner in which he was told*)
`f_34_sentence_relatives`	Sentence relatives (e.g., *Bob likes fried mangoes, which is the most disgusting thing I've*
`f_35_because`	Causative adverbial subordinator (*because*)
`f_36_though`	Concessive adverbial subordinators (*although, though*)
`f_37_if`	Conditional adverbial subordinators (*if, unless*)
`f_38_other_adv_sub`	Other adverbial subordinators (e.g., *since, while, whereas*)
	I. Prepositional phrases, adjectives and adverbs
`f_39_prepositions`	Total prepositional phrases
`f_40_adj_attr`	Attributive adjectives (e.g., *the big horse*)
`f_41_adj_pred`	Predicative adjectives (e.g., *The horse is big.*)
`f_42_adverbs`	Total adverbs
	J. Lexical specificity
`f_43_type_token`	Type-token ratio (including punctuation)
`f_44_mean_word_length`	Average word length (across tokens, excluding punctuation)
	K. Lexical classes
`f_45_conjuncts`	Conjuncts (e.g., *consequently, furthermore, however*)
`f_46_downtoners`	Downtoners (e.g., *barely, nearly, slightly*)
`f_47_hedges`	Hedges (e.g., *at about, something like, almost*)
`f_48_amplifiers`	Amplifiers (e.g., *absolutely, extremely, perfectly*)
`f_49_emphatics`	Emphatics (e.g., *a lot, for sure, really*)
`f_50_discourse_particles`	Discourse particles (e.g., sentence-initial *well, now, anyway*)
`f_51_demonstratives`	Demonstratives
	L. Modals
`f_52_modal_possibility`	Possibility modals (*can, may, might, could*)
`f_53_modal_necessity`	Necessity modals (*ought, should, must*)
`f_54_modal_predictive`	Predictive modals (*will, would, shall*)
	M. Specialized verb classes
`f_55_verb_public`	Public verbs (e.g., *assert, declare, mention*)
`f_56_verb_private`	Private verbs (e.g., *assume, believe, doubt, know*)
`f_57_verb_suasive`	Suasive verbs (e.g., *command, insist, propose*)
`f_58_verb_seem`	*seem* and *appear*
	N. Reduced forms and dispreferred structures
`f_59_contractions`	Contractions
n/a	Subordinator *that* deletion (e.g., *I think [that] he went.*)
`f_61_stranded_preposition`	Stranded prepositions (e.g., *the candidate that I was thinking of*)
`f_62_split_infinitve`	Split infinitives (e.g., *He wants to convincingly prove that ...*)
`f_63_split_auxiliary`	Split auxiliaries (e.g., *They were apparently shown to ...*)
	O. Co-ordination
`f_64_phrasal_coordination`	Phrasal co-ordination (N *and* N; Adj *and* Adj; V *and* V; Adv *and* Adv)
n/a	Independent clause co-ordination (clause-initial *and*)
	P. Negation
`f_66_neg_synthetic`	Synthetic negation (e.g., *No answer is good enough for Jones.*)
`f_67_neg_analytic`	Analytic negation (e.g., *That's not likely.*)

Author(s)

Stephanie Evert (`https://purl.org/stephanie.evert`); feature extractor by Jan Gasthaus (2007).

References

Aston, Guy and Burnard, Lou (1998). *The BNC Handbook.* Edinburgh University Press, Edinburgh. See also the BNC homepage at `http://www.natcorp.ox.ac.uk/`.

Biber, Douglas (1988). *Variations Across Speech and Writing.* Cambridge University Press, Cambridge.

Biber, Douglas (1995). *Dimensions of Register Variation: A cross-linguistic comparison.* Cambridge University Press, Cambridge.

Gasthaus, Jan (2007). *Prototype-Based Relevance Learning for Genre Classification.* B.Sc.\ thesis, Institute of Cognitive Science, University of Osnabrück.

See Also

`BNCmeta`

`BNCcomparison` *Comparison of written and spoken noun frequencies in the British National Corpus*

Description

This data set compares the frequencies of 60 selected nouns in the written and spoken parts of the British National Corpus, World Edition (BNC). Nouns were chosen from three frequency bands, namely the 20 most frequent nouns in the corpus, 20 nouns with approximately 1000 occurrences, and 20 nouns with approximately 100 occurrences.

See Aston & Burnard (1998) for more information about the BNC, or go to `http://www.natcorp.ox.ac.uk/`.

Usage

```
BNCcomparison
```

Format

A data frame with 61 rows and the following columns:

`noun`**:** lemmatised noun (aka stem form)

`written`**:** frequency in the written part of the BNC

`spoken`**:** frequency in the spoken part of the BNC

Details

In addition to the 60 nouns, the data set contains a row labelled `OTHER`, which represents the total frequency of all other nouns in the BNC. This value is needed in order to calculate the sample sizes of the written and spoken part for frequency comparison tests.

Author(s)

Stephanie Evert (https://purl.org/stephanie.evert)

References

Aston, Guy and Burnard, Lou (1998). *The BNC Handbook.* Edinburgh University Press, Edinburgh. See also the BNC homepage at http://www.natcorp.ox.ac.uk/.

`BNCdomains` *Distribution of domains in the British National Corpus (BNC)*

Description

This data set gives the number of documents and tokens in each of the 18 domains represented in the British National Corpus, World Edition (BNC). See Aston & Burnard (1998) for more information about the BNC and the domain classification, or go to http://www.natcorp.ox.ac.uk/.

Usage

```
BNCdomains
```

Format

A data frame with 19 rows and the following columns:

`domain`: name of the respective domain in the BNC

`documents`: number of documents from this domain

`tokens`: total number of tokens in all documents from this domain

Details

For one document in the BNC, the domain classification is missing. This document is represented by the code `Unlabeled` in the data set.

Author(s)

Marco Baroni <<baroni@sslmit.unibo.it>>

References

Aston, Guy and Burnard, Lou (1998). *The BNC Handbook.* Edinburgh University Press, Edinburgh. See also the BNC homepage at http://www.natcorp.ox.ac.uk/.

`BNCInChargeOf` *Collocations of the phrase "in charge of" (BNC)*

Description

This data set lists collocations (in the sense of Sinclair 1991) of the phrase *in charge of* found in the British National Corpus, World Edition (BNC). A span size of 3 and a frequency threshold of 5 were used, i.e. all words that occur at least five times within a distance of three tokens from the key phrase *in charge of* are listed as collocates. Note that collocations were not allowed to cross sentence boundaries.

See Aston & Burnard (1998) for more information about the BNC, or go to `http://www.natcorp.ox.ac.uk/`.

Usage

```
BNCInChargeOf
```

Format

A data frame with 250 rows and the following columns:

`collocate`**:** a collocate of the key phrase *in charge of* (word form)

`f.in`**:** occurrences of the collocate within a distance of 3 tokens from the key phrase, i.e. *inside* the span

`N.in`**:** total number of tokens inside the span

`f.out`**:** occurrences of the collocate *outside* the span

`N.out`**:** total number of tokens outside the span

Details

Punctuation, numbers and any words containing non-alphabetic characters (except for `-`) were not considered as potential collocates. Likewise, the number of tokens inside / outside the span given in the columns `N.in` and `N.out` only includes simple alphabetic word forms.

Author(s)

Stephanie Evert (`https://purl.org/stephanie.evert`)

References

Aston, Guy and Burnard, Lou (1998). *The BNC Handbook.* Edinburgh University Press, Edinburgh. See also the BNC homepage at `http://www.natcorp.ox.ac.uk/`.

Sinclair, John (1991). *Corpus, Concordance, Collocation.* Oxford University Press, Oxford.

BNCmeta *Metadata for the British National Corpus (XML edition)*

Description

This data set provides complete metadata for all 4048 texts of the British National Corpus (XML edition). See Aston & Burnard (1998) for more information about the BNC, or go to `http://www.natcorp.ox.ac.uk/`.

The data have automatically been extracted from the original BNC source files. Some transformations were applied so that all attribute names and their values are given in a human-readable form. The Perl scripts used in the extraction procedure are available from `https://cwb.sourceforge.io/install.php#other`.

Usage

```
BNCmeta
```

Format

A data frame with 4048 rows and the columns listed below. Unless specified otherwise, columns are coded as factors.

`id`: BNC document ID; character vector

`title`: Title of the document; character vector

`n_words`: Number of words in the document; integer vector

`n_tokens`: Total number of tokens (including punctuation and deleted material); integer vector

`n_w`: Number of w-units (words); integer vector

`n_c`: Number of c-units (punctuation); integer vector

`n_s`: Number of s-units (sentences); integer vector

`publication_date`: Publication date

`text_type`: Text type

`context`: Spoken context

`respondent_age`: Age-group of respondent

`respondent_class`: Social class of respondent (NRS social grades)

`respondent_sex`: Sex of respondent

`interaction_type`: Interaction type

`region`: Region

`author_age`: Author age-group

`author_domicile`: Domicile of author

`author_sex`: Sex of author

`author_type`**:** Author type

`audience_age`**:** Audience age

`domain`**:** Written domain

`difficulty`**:** Written difficulty

`medium`**:** Written medium

`publication_place`**:** Publication place

`sampling_type`**:** Sampling type

`circulation`**:** Estimated circulation size

`audience_sex`**:** Audience sex

`availability`**:** Availability

`mode`**:** Text mode (written/spoken)

`derived_type`**:** Text class

`genre`**:** David Lee's genre classification

Author(s)

Stephanie Evert (`https://purl.org/stephanie.evert`)

References

Aston, Guy and Burnard, Lou (1998). *The BNC Handbook.* Edinburgh University Press, Edinburgh. See also the BNC homepage at `http://www.natcorp.ox.ac.uk/`.

`BNCqueries` *Per-text frequency counts for a selection of BNCweb corpus queries*

Description

This data set contains a table of frequency counts obtained with a selection of BNCweb (Hoffmann et al. 2008) queries for each text document in the British National Corpus (Aston & Burnard 1998).

Usage

```
BNCqueries
```

Format

A data frame with 4048 rows and 12 columns. The first column (`id`) contains a character vector of text IDs, the remaining columns contain integer vector of the corresponding per-text frequency counts for various BNCweb queries. Column names ending in `.S` indicate sentence counts rather than token counts.

The list below shows the BNCweb query used for each feature in CEQL syntax (Hoffmann et al. 2008, Ch. 6).

- `id`: text ID
- `split.inf.S`: number of sentences containing a split infinitive with *-ly* adverb; query: `_TO0 +ly_AV0 _V?I`
- `adv.inf.S`: number of sentences containing a non-split infinitive with *-ly* adverb; query: `+ly_AV0 _TO0 _V?I`
- `superlative.S`: number of sentences containing a superlative adjective; query: `the (_AJS | most _AJ0)`
- `past.S`: number of sentences containing a paste tense verb; query: `_V?D`
- `wh.question.S`: number of wh-questions; query: `<s> _[PNQ,AVQ] _{V}`
- `stop.to`: frequency of the expression *stop to* + verb; query: `{stop/V} to _{V}`
- `time`: frequency of the noun *time*; query: `{time/N}`
- `click`: frequency of the verb *to click*; query: `{click/V}`
- `noun`: frequency of common nouns; query: `_NN?`
- `nominalization`: frequency of nominalizations; query: `+[tion,tions,ment,ments,ity,ities]_NN?`
- `downtoner`: frequency of downtoners; query: `[almost,barely,hardly,merely,mildly,nearly,only,partially,part`

Author(s)

Stephanie Evert (https://purl.org/stephanie.evert)

References

Aston, Guy and Burnard, Lou (1998). *The BNC Handbook.* Edinburgh University Press, Edinburgh. See also the BNC homepage at http://www.natcorp.ox.ac.uk/.

Hoffmann, Sebastian; Evert, Stefan; Smith, Nicholas; Lee, David; Berglund Prytz, Ylva (2008). *Corpus Linguistics with BNCweb – a Practical Guide*, volume 6 of English Corpus Linguistics. Peter Lang, Frankfurt am Main. See also http://corpora.lancs.ac.uk/BNCweb/.

See Also

BNCmeta

BrownBigrams *Bigrams of adjacent words from the Brown corpus*

Description

This data set contains bigrams of adjacent word forms from the Brown corpus of written American English (Francis & Kucera 1964). Co-occurrence frequencies are specified in the form of an observed contingency table, using the notation suggested by Evert (2008).

Only bigrams that occur at least 5 times in the corpus are included.

Usage

```
BrownBigrams
```

Format

A data frame with 24167 rows and the following columns:

`id`: unique ID of the bigram entry

`word1`: the first word form in the bigram (character)

`pos1`: part-of-speech category of the first word (factor)

`word2`: the second word form in the bigram (character)

`pos2`: part-of-speech category of the second word (factor)

`O11`: co-occurrence frequency of the bigram (numeric)

`O12`: occurrences of the first word without the second (numeric)

`O21`: occurrences of the second word without the first (numeric)

`O22`: number of bigram tokens containing neither the first nor the second word (numeric)

Details

Part-of-speech categories are identified by single-letter codes, corresponding of the first character of the Penn tagset.

Some important POS codes are `N` (noun), `V` (verb), `J` (adjective), `R` (adverb or particle), `I` (preposition), `D` (determiner), `W` (wh-word) and `M` (modal).

Author(s)

Stephanie Evert (`https://purl.org/stephanie.evert`)

References

Evert, Stefan (2008). Corpora and collocations. In A. Lüdeling and M. Kytö (eds.), *Corpus Linguistics. An International Handbook*, chapter 58, pages 1212–1248. Mouton de Gruyter, Berlin, New York.

Francis, W.~N. and Kucera, H. (1964). Manual of information to accompany a standard sample of present-day edited American English, for use with digital computers. Technical report, Department of Linguistics, Brown University, Providence, RI.

`BrownLOBPassives` *Frequency counts of passive verb phrases in the Brown and LOB corpora*

Description

This data set contains frequency counts of passive verb phrases for selected texts from the Brown corpus of written American English (Francis & Kucera 1964) and the comparable LOB corpus of written British English (Johansson *et al.* 1978).

Usage

```
BrownLOBPassives
```

Format

A data frame with 622 rows and the following columns:

`id`**:** a unique ID for each text (character)

`passive`**:** number of passive verb phrases

`n_w`**:** total number of words in the genre category

`n_s`**:** total number of sentences in the genre category

`cat`**:** genre category code (`A`...`R`; factor)

`genre`**:** descriptive label for the genre category (factor)

`lang`**:** descriptive label for the genre category

Author(s)

Stephanie Evert (https://purl.org/stephanie.evert)

References

Francis, W.~N. and Kucera, H. (1964). Manual of information to accompany a standard sample of present-day edited American English, for use with digital computers. Technical report, Department of Linguistics, Brown University, Providence, RI.

Johansson, Stig; Leech, Geoffrey; Goodluck, Helen (1978). Manual of information to accompany the Lancaster-Oslo/Bergen corpus of British English, for use with digital computers. Technical report, Department of English, University of Oslo, Oslo.

See Also

`BrownPassives`, `LOBPassives`

`BrownPassives` *Frequency counts of passive verb phrases in the Brown corpus*

Description

This data set contains frequency counts of passive verb phrases in the Brown corpus of written American English (Francis & Kucera 1964), aggregated by genre category.

Usage

```
BrownPassives
```

Format

A data frame with 15 rows and the following columns:

`cat`**:** genre category code (`A` . . . `R`)

`passive`**:** number of passive verb phrases

`n_w`**:** total number of words in the genre category

`n_s`**:** total number of sentences in the genre category

`name`**:** descriptive label for the genre category

Author(s)

Stephanie Evert (`https://purl.org/stephanie.evert`)

References

Francis, W.~N. and Kucera, H. (1964). Manual of information to accompany a standard sample of present-day edited American English, for use with digital computers. Technical report, Department of Linguistics, Brown University, Providence, RI.

See Also

`LOBPassives`, `BrownLOBPassives`

BrownStats — *Basic statistics of texts in the Brown corpus*

Description

This data set provides some basic quantiative measures for all texts in the Brown corpus of written American English (Francis & Kucera 1964),

Usage

```
BrownStats
```

Format

A data frame with 500 rows and the following columns:

`ty`: number of distinct types

`to`: number of tokens (including punctuation)

`se`: number of sentences

`towl`: mean word length in characters, averaged over tokens

`tywl`: mean word length in characters, averaged over types

Author(s)

Marco Baroni <`<baroni@sslmit.unibo.it>`>

References

Francis, W.~N. and Kucera, H. (1964). Manual of information to accompany a standard sample of present-day edited American English, for use with digital computers. Technical report, Department of Linguistics, Brown University, Providence, RI.

See Also

`LOBStats`

chisq — *Pearson's chi-squared statistic for frequency comparisons (corpora)*

Description

This function computes Pearson's chi-squared statistic (often written as X^2) for frequency comparison data, with or without Yates' continuity correction. The implementation is based on the formula given by Evert (2004, 82).

Usage

```
chisq(k1, n1, k2, n2, correct = TRUE, one.sided=FALSE)
```

Arguments

`k1`	frequency of a type in the first corpus (or an integer vector of type frequencies)
`n1`	the sample size of the first corpus (or an integer vector specifying the sizes of different samples)
`k2`	frequency of the type in the second corpus (or an integer vector of type frequencies, in parallel to `k1`)
`n2`	the sample size of the second corpus (or an integer vector specifying the sizes of different samples, in parallel to `n1`)
`correct`	if `TRUE`, apply Yates' continuity correction (default)
`one.sided`	if `TRUE`, compute the *signed square root* of X^2 as a statistic for a one-sided test (see details below; the default value is `FALSE`)

Details

The X^2 values returned by this function are identical to those computed by `chisq.test`. Unlike the latter, `chisq` accepts vector arguments so that a large number of frequency comparisons can be carried out with a single function call.

The one-sided test statistic (for `one.sided=TRUE`) is the signed square root of X^2. It is positive for $k_1/n_1 > k_2/n_2$ and negative for $k_1/n_1 < k_2/n_2$. Note that this statistic has a *standard normal distribution* rather than a chi-squared distribution under the null hypothesis of equal proportions.

Value

The chi-squared statistic X^2 corresponding to the specified data (or a vector of X^2 values). This statistic has a *chi-squared distribution* with $df = 1$ under the null hypothesis of equal proportions.

Author(s)

Stephanie Evert (`https://purl.org/stephanie.evert`)

References

Evert, Stefan (2004). *The Statistics of Word Cooccurrences: Word Pairs and Collocations.* Ph.D. thesis, Institut f?r maschinelle Sprachverarbeitung, University of Stuttgart. Published in 2005, URN urn:nbn:de:bsz:93-opus-23714. Available from http://www.collocations.de/phd.html.

See Also

`chisq.pval`, `chisq.test`, `cont.table`

Examples

```
chisq.test(cont.table(99, 1000, 36, 1000))
chisq(99, 1000, 36, 1000)
```

---

chisq.pval — *P-values of Pearson's chi-squared test for frequency comparisons (corpora)*

Description

This function computes the p-value of Pearsons's chi-squared test for the comparison of corpus frequency counts (under the null hypothesis of equal population proportions). It is based on the chi-squared statistic X^2 implemented by the `chisq` function.

Usage

```
chisq.pval(k1, n1, k2, n2, correct = TRUE,
           alternative = c("two.sided", "less", "greater"))
```

Arguments

`k1`	frequency of a type in the first corpus (or an integer vector of type frequencies)
`n1`	the sample size of the first corpus (or an integer vector specifying the sizes of different samples)
`k2`	frequency of the type in the second corpus (or an integer vector of type frequencies, in parallel to `k1`)
`n2`	the sample size of the second corpus (or an integer vector specifying the sizes of different samples, in parallel to `n1`)
`correct`	if `TRUE`, apply Yates' continuity correction (default)
`alternative`	a character string specifying the alternative hypothesis; must be one of `two.sided` (default), `less` or `greater`

Details

The p-values returned by this functions are identical to those computed by `chisq.test` (two-sided only) and `prop.test` (one-sided and two-sided) for two-by-two contingency tables.

Value

The p-value of Pearson's chi-squared test applied to the given data (or a vector of p-values).

Author(s)

Stephanie Evert (`https://purl.org/stephanie.evert`)

See Also

`chisq`, `fisher.pval`, `chisq.test`, `prop.test`

Examples

```
chisq.test(cont.table(99, 1000, 36, 1000))
chisq.pval(99, 1000, 36, 1000)
```

`cont.table` *Build contingency tables for frequency comparison (corpora)*

Description

This is a convenience function which constructs 2x2 contingency tables needed for frequency comparisons with `chisq.test`, `fisher.test` and similar functions.

Usage

```
cont.table(k1, n1, k2, n2, as.list=NA)
```

Arguments

`k1`	frequency of a type in the first corpus, a numeric scalar or vector
`n1`	the size of the first corpus (sample size), a numeric scalar or vector
`k2`	frequency of the type in the second corpus, a numeric scalar or vector
`n2`	the size of the second corpus (sample size), a numeric scalar or vector
`as.list`	whether multiple contingency tables can be constructed and are returned as a list (see "Details" below)

Details

If all four arguments `k1` `n1` `k2` `n2` are scalars (vectors of length 1), `cont.table` constructs a single contingency table, i.e. a 2x2 matrix. If at least one argument has length > 1, shorter vectors are replicated as necessary, and a list of 2x2 contingency tables is constructed.

With `as.list=TRUE`, the return value is always a list, even if it contains just a single contingency table. With `as.list=FALSE`, only scalar arguments are accepted and the return value is guaranteed to be a 2x2 matrix.

Value

A numeric matrix containing a two-by-two contingency table for the specified frequency comparison, or a list of such matrices (see "Details").

Author(s)

Stephanie Evert (`https://purl.org/stephanie.evert`)

See Also

`chisq.test`, `fisher.test`

Examples

```
ct <- cont.table(42, 100, 66, 200)
ct

chisq.test(ct)
```

`corpora.palette` *Colour palettes for linguistic visualization (corpora)*

Description

Several useful colour palettes for plots and other visualizations.

The function `alpha.col` can be used to turn colours (partially) translucent for used in crowded scatterplots.

Usage

```
corpora.palette(name=c("seaborn", "muted", "bright", "simple"),
                n=NULL, alpha=1)

alpha.col(col, alpha)
```

Arguments

`name`	name of the desired colour palette (see Details below)
`n`	optional: number of colours to return. The palette will be shortened or recycled as necessary.
`col`	a vector of R colour specifications (as accepted by `col2rgb`)
`alpha`	alpha value between 0 and 1; values below 1 make the colours translucent

Details

Every colour palette starts with the colours black, red, green and blue in this order.

`seaborn`, `muted` and `bright` are 7-colour palettes inspired by the seaborn data visualization library, but add a shade of dark grey as first colour.

`simple` is a 10-colour palette based on R's default palette.

Value

A character vector with colour names or hexadecimal RGB specifications.

Author(s)

Stephanie Evert (`https://purl.org/stephanie.evert`)

See Also

`rgb` for R colour specification formats, `palette` for setting the default colour palette

Examples

```
par.save <- par(mfrow=c(2, 2))
for (name in qw("seaborn muted bright simple")) {
  barplot(rep(1, 10), col=corpora.palette(name, 10), main=name)
}
par(par.save)
```

`DistFeatBrownFam` *Latent dimension scores from a distributional analysis of the Brown Family corpora*

Description

This data frame provides unsupervised distributional features for each text in the extended Brown Family of corpora (Brown, LOB, Frown, FLOB, BLOB), covering edited written American and British English from 1930s, 1960s and 1990s (see Xiao 2008, 395–397).

Latent topic dimensions were obtained by a method similar to Latent Semantic Indexing (Deerwester et al. 1990), applying singular value decomposition to bag-of-words vectors for the 2500 texts in the extended Brown Family. Register dimensions were obtained with the same methodology, using vectors of part-of-speech frequencies (separately for all verb-related tags and all other tags).

Usage

```
DistFeatBrownFam
```

Format

A data frame with 2500 rows and the following 23 columns:

`id`: A unique ID for each text (also used as row name)

`top1`, `top2`, `top3`, `top4`, `top5`, `top6`, `top7`, `top8`, `top9`: latent dimension scores for the first 9 topic dimensions

`reg1`, `reg2`, `reg3`, `reg4`, `reg5`, `reg6`, `reg7`, `reg8`, `reg9`: latent dimension scores for the first 9 register dimensions (excluding verb-related tags)

`vreg1`, `vreg2`, `vreg3`, `vreg4`: latent dimension scores for the first 4 register dimensions based only on verb-related tags

Details

TODO

Author(s)

Stephanie Evert (`https://purl.org/stephanie.evert`)

References

Deerwester, Scott; Dumais, Susan T.; Furnas, George W.; Landauer, Thomas K.; Harshman, Richard (1990). Indexing by latent semantic analysis. *Journal of the American Society For Information Science*, **41**(6), 391–407.

Xiao, Richard (2008). Well-known and influential corpora. In A. Lüdeling and M. Kytö (eds.), *Corpus Linguistics. An International Handbook*, chapter 20, pages 383–457. Mouton de Gruyter, Berlin.

`fisher.pval` *P-values of Fisher's exact test for frequency comparisons (corpora)*

Description

This function computes the p-value of Fisher's exact test (Fisher 1934) for the comparison of corpus frequency counts (under the null hypothesis of equal population proportions). In the two-sided case, a "central" p-value (Fay 2010) provides better numerical efficiency than the likelihood-based approach of `fisher.test` and is always consistent with confidence intervals.

Usage

```
fisher.pval(k1, n1, k2, n2,
            alternative = c("two.sided", "less", "greater"),
            log.p = FALSE)
```

Arguments

`k1`	frequency of a type in the first corpus (or an integer vector of type frequencies)
`n1`	the sample size of the first corpus (or an integer vector specifying the sizes of different samples)
`k2`	frequency of the type in the second corpus (or an integer vector of type frequencies, in parallel to `k1`)
`n2`	the sample size of the second corpus (or an integer vector specifying the sizes of different samples, in parallel to `n1`)
`alternative`	a character string specifying the alternative hypothesis; must be one of `two.sided` (default), `less` or `greater`
`log.p`	if TRUE, the natural logarithm of the p-value is returned

Details

For `alternative="two.sided"` (the default), the p-value of the "central" Fisher's exact test (Fay 2010) is computed, which differs from the more common likelihood-based method implemented by `fisher.test` (and referred to as the "two-sided Fisher's exact test" by Fay). This approach has two advantages: (i) it is numerically robust and efficient, even for very large samples and frequency counts; (ii) it is consistent with Clopper-Pearson type confidence intervals (see examples below).

For one-sided tests, the p-values returned by this function are identical to those computed by `fisher.test` on two-by-two contingency tables.

Value

The p-value of Fisher's exact test applied to the given data (or a vector of p-values).

Author(s)

Stephanie Evert (`https://purl.org/stephanie.evert`)

References

Fay, Michael P. (2010). Confidence intervals that match Fisher's exact or Blaker's exact tests. *Biostatistics*, **11**(2), 373-374.

Fisher, R. A. (1934). *Statistical Methods for Research Workers*. Oliver & Boyd, Edinburgh, 2nd edition (1st edition 1925, 14th edition 1970).

See Also

`fisher.test`, `chisq.pval`

Examples

```
## Fisher's Tea Drinker (see ?fisher.test)
TeaTasting <-
matrix(c(3, 1, 1, 3),
       nrow = 2,
       dimnames = list(Guess = c("Milk", "Tea"),
                       Truth = c("Milk", "Tea")))
print(TeaTasting)
##  - the "corpora" consist of 4 cups of tea each (n1 = n2 = 4)
##     => columns of TeaTasting
##  - frequency counts are the number of cups selected by drinker (k1 = 3, k2 = 1)
##     => first row of TeaTasting
##  - null hypothesis of equal type probability = drinker makes random guesses
fisher.pval(3, 4, 1, 4, alternative="greater")
fisher.test(TeaTasting, alternative="greater")$p.value # should be the same

fisher.pval(3, 4, 1, 4)         # central Fisher's exact test is equal to
fisher.test(TeaTasting)$p.value # standard two-sided Fisher's test for symmetric distribution

# inconsistency btw likelihood-based two-sided Fisher's test and confidence interval
# for 4/15 vs. 50/619 successes
fisher.test(cbind(c(4, 11), c(50, 619)))

# central Fisher's exact test is always consistent
fisher.pval(4, 15, 50, 619)
```

---

keyness — *Compute best-practice keyness measures (corpora)*

Description

Compute best-practice keyness measures (according to Evert 2022) for the frequency comparison of lexical items in two corpora. The function is fully vectorised and should be applied to a complete data set of candidate items (so statistical analysis can be adjusted to control the family-wise error rate).

Usage

```
keyness(f1, n1, f2, n2,
      measure=c("LRC", "PositiveLRC", "G2", "LogRatio", "SimpleMaths", "Lockwords"),
         conf.level=.95, alpha=NULL, p.adjust=TRUE, lambda=1)
```

Arguments

`f1`	a numeric vector specifying the frequencies of candidate items in corpus A (target corpus)
`n1`	sample size of target corpus, i.e. the total number of tokens in corpus A (usually a scalar, but can also be a vector parallel to `f1`)
`f2`	a numeric vector parallel to `f1`, specifying the frequencies of candidate items in corpus B (reference corpus)
`n2`	sample size of reference corpus, i.e. the total number of tokens in corpus B (usually a scalar, but can also be a vector parallel to `f2`)

`measure`	the keyness measure to be computed (see “Details” below)
`conf.level`	the desired confidence level for the `LRC` and `PositiveLRC` measures (defaults to 95%)
`alpha`	if specified, filter out candidate items whose frequency difference between f_1 and f_2 is not significant at level α. This is achieved by setting the score of such candidates to 0.
`p.adjust`	if `TRUE`, apply a Bonferroni correction in order to control the family-wise error rate across all tests carried out in a single function call (i.e. the common length of `f1` and `f2`). Alternatively, the desired family size can be specified instead of `TRUE` (useful if a larger data set is processed in batches). The adjustment applied both the the significance filter (`alpha`) and the confidence intervals (`conf.level`) underlying `LRC` and `PositiveLRC` measures.
`lambda`	parameter λ of the `SimpleMaths` measure.

Details

This function computes a range of best-practice keyness measures comparing the relative frequencies π_1 and π_2 of lexical items in populations (i.e. sublanguages) A and B, based on the observed sample frequencies f_1, f_2 and the corresponding sample sizes n_1, n_2. The function is fully vectorised with respect to arguments `f1`, `f2`, `n1` and `n2`, but only a single keyness measure can be selected for each function call. All implemented measures are robust for the corner cases $f_1 = 0$ and $f_2 = 0$, but $f_1 = f_2 = 0$ is not allowed.

Most of the keyness measures are **directional**, i.e. positive scores indicate positive keyness in A ($\pi_1 > \pi_2$) and negative scores indicate negative keyness in A ($\pi_1 < \pi_2$). By contrast, the **one-sided** measures `PositiveLRC` and `SimpleMaths` only detect positive keyness in A, returning small (and possibly negative) scores otherwise, i.e. in case of insufficient evidence for $\pi_1 > \pi_2$ and in case of strong evidence for $\pi_1 < \pi_2$. One-sided measures can be useful for a ranking of the entire data set as positive keyword candidates.

Hardie (2014) and other authors recommend to combine effect-size measures (in particular `LogRatio`) with a **significance filter** in order to weed out candidate items for which there is no significant evidence against the null hypothesis $H_0 : \pi_1 = \pi_2$. Such a filter is activated by specifying the desired significance level `alpha`, and can be combined with all keyness measures. In this case, the scores of all non-significant candidate items are set to 0. The decision is based in the likelihood-ratio test implemented by the `G2` measure and its asymptotic χ^2_1 distribution under H_0.

Note that the significance filter can also be applied to the `G2` measure itself, setting all scores below the critical value for the significance test to 0. For one-sided measures (`PositiveLRC` and `SimpleMaths`), candidates with significant evidence for negative keyness are also filtered out (i.e. their scores are set to 0) in order to ensure a consistent ranking.

By default, statistical inference corrects for multiple testing in order to control **family-wise error rates**. This applies to the significance filter as well as to the confidence intervals underlying `LRC` and `PositiveLRC`. Note that the `G2` scores themselves are never adjusted (only the critical value for the significance filter is modified).

Family size m is automatically determined from the number of candidate items processed in a single function call. Alternatively, the family size can be specified explicitly in the `p.adjust` argument, e.g. if a large data set is processed in multiple batches, or `p.adjust=FALSE` can be used to disable the correction.

For the adjustment, a highly conservative Bonferroni correction $\alpha' = \alpha/m$ is applied to significance levels. Since the large candidate sets and sample sizes often found in corpus linguistics tend to produce large numbers of false positives, this conservative approach is considered to be useful.

See Evert (2022) and its supplementary materials for a more detailed discussion of the implemented best-practice measures and some alternatives.

Keyness Measures:

`G2` The **log-likelihood** measure (Rayson & Garside 2003: 3) computes the score G^2 of a likelihood-ratio test for $H_0 : \pi_1 = \pi_2$. This test is two-sided and always returns positive values, so the sign of its score is inverted for $f_1/n_1 < f_2/n_2$ in order to obtain a directional keyness measure. As a pure significance measure, it tends to prefer high-frequency candidates with large f_1.

`LogRatio` A point estimate of the log **relative risk** $\log_2(\pi_1/\pi_2)$, which has been suggested as an intuitive keyness measure under the name **LogRatio** by Hardie (2014: 45). The implementation uses Walter's (1975) adjusted estimator

$$\log_2 \frac{f_1 + \frac{1}{2}}{n_1 + \frac{1}{2}} - \log_2 \frac{f_2 + \frac{1}{2}}{n_2 + \frac{1}{2}}$$

which is less biased and robust against $f_i = 0$. As a pure effect-size measure, LogRatio tends to assign spuriously high scores to low-frequency candidates with small f_1 and f_2 (due to sampling variation). Combination with a significance filter is strongly recommended.

`LRC` **(default)** A **conservative** estimate for **LogRatio** recommended by Evert (2022) in order to combine and balance the advantages of effect-size and significance measures. A confidence interval (according to the specified `conf.level`) for relative risk $r = \pi_1/\pi_2$ is obtained from an exact conditional Poisson test (Fay 2010: 55), adjusted for multiple testing by default. If a candidate is not significant (i.e. the confidence interval includes $H_0 : r = 1$) its score is set to 0. Otherwise the boundary of the confidence interval closer to 1 is taken as a conservative directional estimate of r and its $\log_2$ is returned.

`PositiveLRC` A **one-sided** variant of **LRC**, which returns the lower boundary of a one-sided confidence interval for $\log_2 r$. Scores ≤ 0 indicate that there is no significant evidence for positive keyness. The directional version of LRC is recommended for general use, but PositiveLRC may be preferred if the hermeneutic interpretation should also consider non-significant candidates (especially with small data sets).

`SimpleMaths` The **simple maths** keyness measure (Kilgarriff 2009) used by the commercial corpus analysis platform **Sketch Engine**:

$$\frac{10^6 \cdot \frac{f_1}{n_1} + \lambda}{10^6 \cdot \frac{f_2}{n_2} + \lambda}$$

Its frequency bias can be adjusted with the user parameter $\lambda > 0$. The scaling factor 10^6 was chosen so that $\lambda = 1$ is a practical default value.

There does not appear to be a convincing mathematical justification behind this measure. It is included here only because of the popularity of the Sketch Engine platform.

`Lockwords` This measure is designed for identifying so-called **lockwords**, whose frequencies are remarkably stable across different corpora (Baker 2011: 73). It is based on the same confidence intervals for $\log_2 r$ as the LRC measure. Here, the maximum over all $|\log_2 r|$ values inside the confidence interval is taken as a conservative upper bound for the true log ratio of

the relative frequencies (which satisfies all the requirements on such a measure discussed by Hardie 2014).
In line with its purpose, the Lockwords measure does not distinguish between positive and negative differences and always returns a positive value. For example, a Lockwords value of 2 means that we have good evidence that the candidate item is at most 4 times as frequent in one population than in the other. In other words, the true relative risk $r = \pi_1/\pi_2$ falls into the range $[\frac{1}{4}, 4]$. Note that the Lockwords value will be $+\infty$ if $f_1 = 0$ or $f_2 = 0$.

Value

A numeric vector of the same length as `f1` and `f2`, containing keyness scores for all candidate lexical items. For most measures, positive scores indicate positive keywords (i.e. higher frequency in the population underlying corpus A) and negative scores indicate negative keywords (i.e. higher frequency in the population underlying corpus B). If `alpha` is specified, non-significant candidates always have a score of 0.

Author(s)

Stephanie Evert (`https://purl.org/stephanie.evert`)

References

Baker, P. (2011). Times may change, but we will always have money: Diachronic variation in recent British English. *Journal of English Linguistics*, **39**(1):65-88.

Evert, S. (2022). Measuring keyness. In *Digital Humanities 2022: Conference Abstracts*, pages 202-205, Tokyo, Japan / online. `https://osf.io/cy6mw/`

Fay, Michael P. (2010). Two-sided exact tests and matching confidence intervals for discrete data. *The R Journal*, **2**(1), 53-58.

Hardie, A. (2014). A single statistical technique for keywords, lockwords, and collocations. Internal CASS working paper no. 1, unpublished.

Kilgarriff, A. (2009). Simple maths for keywords. In *Proceedings of the Corpus Linguistics 2009 Conference*, Liverpool, UK.

Rayson, P. and Garside, R. (2000). Comparing corpora using frequency profiling. In *Proceedings of the ACL Workshop on Comparing Corpora*, pages 1-6, Hong Kong.

Walter, S. D. (1975). The distribution of Levin's measure of attributable risk. *Biometrika*, **62**(2): 371-374.

See Also

`prop.cint`, which is used by the exact conditional Poisson test of the LRC measure

A gentle and detailed introduction to keyness measures can be found in Unit 4 of the SIGIL course at `https://SIGIL.R-Forge.R-Project.org/`, which explains the intuition of LRC visually. The unit also includes a worked example carrying out several keyword analyses with real-life corpus data as well as visualisation in the form of "scattertext" or semantic maps.

Examples

```
# compute all keyness measures for a single candidate item with f1=7, f2=2 and n1=n2=1000
keyness(7, 1000, 2, 1000, measure="G2") # log-likelihood
keyness(7, 1000, 2, 1000, measure="LogRatio")
keyness(7, 1000, 2, 1000, measure="LogRatio", alpha=0.05) # with significance filter
keyness(7, 1000, 2, 1000, measure="LRC") # the default measure
keyness(7, 1000, 2, 1000, measure="PositiveLRC")
keyness(7, 1000, 2, 1000, measure="SimpleMaths")

# a practical example: keywords of spoken British English (from BNC corpus)
n1 <- sum(BNCcomparison$spoken) # sample sizes
n2 <- sum(BNCcomparison$written)
kw <- transform(BNCcomparison,
  G2 = keyness(spoken, n1, written, n2, measure="G2"),
  LogRatio = keyness(spoken, n1, written, n2, measure="LogRatio"),
  LRC = keyness(spoken, n1, written, n2))
kw <- kw[order(-kw$LogRatio), ]
head(kw, 20)              # top LogRatio keywords

kw <- transform(kw,
  Lock = keyness(spoken, n1, written, n2, measure="Lockwords"))
kw <- kw[order(kw$Lock), ] # note increasing sort
head(kw, 20)              # top lockwords

# collocations of "in charge of" with LRC as an association measure
colloc <- transform(BNCInChargeOf,
  PosLRC = keyness(f.in, N.in, f.out, N.out, measure="PositiveLRC"))
colloc <- colloc[order(-colloc$PosLRC), ]
head(colloc, 30)
```

KrennPPV *German PP-Verb collocation candidates annotated by Brigitte Krenn (2000)*

Description

This data set lists 5102 frequent combinations of verbs and prepositional phrases (PP) extracted from a German newspaper corpus. The collocational status of each PP-verb combination was manually annotated by Brigitte Krenn (2000). In addition, pre-computed scores of several standard association measures are provided as well as the underlying frequency signatures.

The KrennPPV candidate set forms part of the data used in the evaluation study of Evert & Krenn (2005).

Usage

```
KrennPPV
```

Format

A data frame with 5102 rows and the following columns:

`PP`: the prepositional phrase, represented by preposition and lemma of the nominal head (character). Preposition-article fusion is indicated by a `+` sign. For example, the prepositional phrase *im letzten Jahr* would appear as `in:Jahr` in the data set.

`verb`: the verb lemma (character). Separated particle verbs have been recombined.

`is.colloc`: whether the PP-verb combination is a lexical collocation (logical)

`is.SVC`: whether a PP-verb collocation is a support verb construction (logical)

`is.figur`: whether a PP-verb-collocation is a figurative expression (logical)

`freq`: co-occurrence frequency of the PP-verb combination within clauses (integer)

`f.PP`: marginal frequency of the PP (integer)

`f.verb`: marginal frequency of the verb (integer)

`N`: sample size of the data set, which is the same for all items (integer)

`MI`: Mutual Information association measure

`Dice`: Dice coefficient association measure

`z.score`: z-score association measure

`t.score`: t-score association measure

`chisq`: chi-squared association measure (without Yates' continuity correction)

`chisq.corr`: chi-squared association measure (with Yates' continuity correction)

`log.like`: log-likelihood association measure

`Fisher`: Fisher's exact test as an association measure (negative logarithm of one-sided p-value)

More information on the compilation and annotation of the data set can be found in Evert & Krenn (2005). See Evert (2008) and `http://www.collocations.de/AM/` for details on the pre-computed association measures.

Author(s)

Stephanie Evert (`https://purl.org/stephanie.evert`)

References

Evert, Stefan (2008). Corpora and collocations. In A. Lüdeling and M. Kytö (eds.), *Corpus Linguistics. An International Handbook*, chapter 58, pages 1212–1248. Mouton de Gruyter, Berlin, New York.

Evert, Stefan and Krenn, Brigitte (2005). Using small random samples for the manual evaluation of statistical association measures. *Computer Speech and Language*, **19**(4), 450–466.

Krenn, Brigitte (2000). *The Usual Suspects: Data-Oriented Models for the Identification and Representation of Lexical Collocations*, volume~7 of *Saarbrücken Dissertations in Computational Linguistics and Language Technology*. DFKI & Universität des Saarlandes, Saarbrücken, Germany.

`LOBPassives` *Frequency counts of passive verb phrases in the LOB corpus*

Description

This data set contains frequency counts of passive verb phrases in the LOB corpus of written British English (Johansson *et al.* 1978), aggregated by genre category.

Usage

```
BrownPassives
```

Format

A data frame with 15 rows and the following columns:

`cat`**:** genre category code (`A` ... `R`)

`passive`**:** number of passive verb phrases

`n_w`**:** total number of words in the genre category

`n_s`**:** total number of sentences in the genre category

`name`**:** descriptive label for the genre category

Author(s)

Stephanie Evert (`https://purl.org/stephanie.evert`)

References

Johansson, Stig; Leech, Geoffrey; Goodluck, Helen (1978). Manual of information to accompany the Lancaster-Oslo/Bergen corpus of British English, for use with digital computers. Technical report, Department of English, University of Oslo, Oslo.

See Also

`BrownPassives`, `BrownLOBPassives`

`LOBStats` *Basic statistics of texts in the LOB corpus*

Description

This data set provides some basic quantiative measures for all texts in the LOB corpus of written British English (Johansson *et al.* 1978).

Usage

```
LOBStats
```

Format

A data frame with 500 rows and the following columns:

`ty`: number of distinct types

`to`: number of tokens (including punctuation)

`se`: number of sentences

`towl`: mean word length in characters, averaged over tokens

`tywl`: mean word length in characters, averaged over types

Author(s)

Marco Baroni <`<baroni@sslmit.unibo.it>`>

References

Johansson, Stig; Leech, Geoffrey; Goodluck, Helen (1978). Manual of information to accompany the Lancaster-Oslo/Bergen corpus of British English, for use with digital computers. Technical report, Department of English, University of Oslo, Oslo.

See Also

`BrownStats`

`PassiveBrownFam` *By-text frequencies of passive verb phrases in the Brown Family corpora.*

Description

This data set specifies the number of passive and active verb phrases for each text in the extended Brown Family of corpora (Brown, LOB, Frown, FLOB, BLOB), covering edited written American and British English from 1930s, 1960s and 1990s (see Xiao 2008, 395–397).

Verb phrase and passive/active aspect counts are based on a fully automatic analysis of the texts, using the Pro3Gres parser (Schneider et al. 2004).

Usage

```
PassiveBrownFam
```

Format

A data frame with 2499 rows and the following 11 columns:

`id`**:** A unique ID for each text (also used as row name)

`corpus`**:** Corpus, a factor with five levels `BLOB`, `Brown`, `LOB`, `Frown`, `FLOB`

`section`**:** Genre, a factor with fifteen levels `A`, ..., `R` (Brown section codes)

`genre`**:** Genre labels, a factor with fifteen levels (e.g. `press reportage`)

`period`**:** Date of publication, a factor with three levels (`1930`, `1960`, `1990`)

`lang`**:** Language variety / region, a factor with levels `AmE` (U.S.) and `BrE` (UK)

`n.words`**:** Number of word tokens, an integer vector

`act`**:** Number of active verb phrases, an integer vector

`pass`**:** Number of passive verb phrases, an integer vector

`verbs`**:** Total number of verb phrases, an integer vector

`p.pass`**:** Percentage of passive verb phrases in the text, a numeric vector

Details

No frequency data could be obtained for text `N02` in the Frown corpus. This entry has been omitted from the table.

Acknowledgements

Frequency information for this data set was kindly provided by Gerold Schneider, University of Zurich (https://www.cl.uzh.ch/de/about-us/people/team/compling/gschneid.html).

Author(s)

Stephanie Evert (https://purl.org/stephanie.evert)

References

Schneider, Gerold; Rinaldi, Fabio; Dowdall, James (2004). Fast, deep-linguistic statistical dependency parsing. In G.-J. M. Kruijff and D. Duchier (eds.), *Proceedings of the COLING 2004 Workshop on Recent Advances in Dependency Grammar*, pages 33-40, Geneva, Switzerland. https://files.ifi.uzh.ch/cl/gschneid/parser/

Xiao, Richard (2008). Well-known and influential corpora. In A. Lüdeling and M. Kytö (eds.), *Corpus Linguistics. An International Handbook*, chapter 20, pages 383–457. Mouton de Gruyter, Berlin.

`prop.cint` *Confidence interval for proportion based on frequency counts (corpora)*

Description

This function computes a confidence interval for a population proportion from the corresponding frequency count in a sample. It either uses the Clopper-Pearson method (inverted exact binomial test) or the Wilson score method (inversion of a z-score test, with or without continuity correction).

Usage

```
prop.cint(k, n, method = c("binomial", "z.score"), correct = TRUE, p.adjust=FALSE,
          conf.level = 0.95, alternative = c("two.sided", "less", "greater"))
```

Arguments

`k`	frequency of a type in the corpus (or an integer vector of frequencies)
`n`	number of tokens in the corpus, i.e. sample size (or an integer vector specifying the sizes of different samples)
`method`	a character string specifying whether to compute a Clopper-Pearson confidence interval (`binomial`) or a Wilson score interval (`z.score`)
`correct`	if `TRUE`, apply Yates' continuity correction for the z-score test (default)
`p.adjust`	if `TRUE`, apply a Bonferroni correction to ensure a family-wise confidence level over all tests carried out in a single function call (i.e. the length of `k`). Alternatively, the desired family size can be specified instead of `TRUE`.
`conf.level`	the desired confidence level (defaults to 95%)
`alternative`	a character string specifying the alternative hypothesis, yielding a two-sided (`two.sided`, default), lower one-sided (`less`) or upper one-sided (`greater`) confidence interval

Details

The confidence intervals computed by this function correspond to those returned by `binom.test` and `prop.test`, respectively. However, `prop.cint` accepts vector arguments, allowing many confidence intervals to be computed with a single function call in a computationally efficient manner.

The **Clopper-Pearson** confidence interval (`binomial`) is obtained by inverting the exact binomial test at significance level α = 1 - `confidence.level`. In the two-sided case, the p-value of the test is computed using the "central" method Fay (2010: 53), i.e. as twice the tail probability of the matching tail. This corresponds to the algorithm originally proposed by Clopper & Pearson (1934).

The limits of the confidence interval are computed in an efficient and numerically robust manner via (the inverse of) the incomplete Beta function.

The **Wilscon score** confidence interval (`z.score`) is computed by solving the equation of the z-score test

$$\frac{k - np}{\sqrt{np(1-p)}} = A$$

for p, where A is the z-value corresponding to the chosen confidence level (e.g. ± 1.96 for a two-sided test with 95% confidence). This leads to the quadratic equation

$$p^2(n + A^2) + p(-2k - A^2) + \frac{k^2}{n} = 0$$

whose two solutions correspond to the lower and upper boundary of the confidence interval.

When Yates' continuity correction is applied, the value k in the numerator of the z-score equation has to be replaced by k^*, with $k^* = k - 1/2$ for the *lower* boundary of the confidence interval (where $k > np$) and $k^* = k + 1/2$ for the *upper* boundary of the confidence interval (where $k < np$). In each case, the corresponding solution of the quadratic equation has to be chosen (i.e., the solution with $k > np$ for the lower boundary and vice versa).

If a **Bonferroni** correction is applied, the significance level α of the underlying test is divided by the number m of tests carried out (specified explicitly by the user or given implicitly by `length(k)`): $\alpha' = \alpha/m$.

Value

A data frame with two columns, labelled `lower` for the lower boundary and `upper` for the upper boundary of the confidence interval. The number of rows is determined by the length of the longest input vector (`k`, `n` and `conf.level`).

Author(s)

Stephanie Evert (`https://purl.org/stephanie.evert`)

References

Clopper, C. J. & Pearson, E. S. (1934). The use of confidence or fiducial limits illustrated in the case of the binomial. *Biometrika*, **26**(4), 404-413.

Fay, Michael P. (2010). Two-sided exact tests and matching confidence intervals for discrete data. *The R Journal*, **2**(1), 53-58.

`https://en.wikipedia.org/wiki/Binomial_proportion_confidence_interval`

See Also

`z.score.pval`, `prop.test`, `binom.pval`, `binom.test`

Examples

```
# Clopper-Pearson confidence interval
binom.test(19, 100)
prop.cint(19, 100, method="binomial")

# Wilson score confidence interval
prop.test(19, 100)
prop.cint(19, 100, method="z.score")
```

`qw` *Split string into words, similar to qw() in Perl (corpora)*

Description

This function splits one or more character strings into words. By default, the strings are split on whitespace in order to emulate Perl's `qw()` (quote words) functionality.

Usage

```
qw(s, sep="\\s+", names=FALSE)
```

Arguments

`s`	one or more strings to be split (a character vector)
`sep`	PCRE regular expression on which to split (defaults to whitespace)
`names`	if TRUE, the resulting character vector is labelled with itself, which is convenient for `lapply` and similar functions

Value

A character vector of the resulting words. Multiple strings in `s` are flattened into a single vector.

If `names=TRUE`, the words are used both as values and as labels of the character vectors, which is convenient when iterating over it with `lapply` or `sapply`.

Author(s)

Stephanie Evert (`https://purl.org/stephanie.evert`)

Examples

```
qw(c("alpha beta gamma", "42 111" ))
qw("alpha beta gamma", names=TRUE)
qw("words with blanks,  sep by commas", sep="\\s*,\\s*")
```

`rowColVector` *Propagate vector to single-row or single-column matrix (corpora)*

Description

This utility function converts a plain vector into a row or column vector, i.e. a single-row or single-column matrix.

Usage

```
rowVector(x, label=NULL)
colVector(x, label=NULL)
```

Arguments

`x`	a (typically numeric) vector
`label`	an optional character string specifying a label for the single row or column returned

Value

A single-row or single-column matrix of the same data type as `x`. Labels of `x` are preserved as column/row names of the matrix.

See `matrix` for details on how non-atomic objects are handled.

Author(s)

Stephanie Evert (`https://purl.org/stephanie.evert`)

Examples

```
rowVector(1:5, "myvec")
colVector(c(A=1, B=2, C=3), label="myvec")
```

`sample.df` *Random samples from data frames (corpora)*

Description

This function takes a random sample of rows from a data frame, in analogy to the built-in function `sample` (which sadly does not accept a data frame).

Usage

```
sample.df(df, size, replace=FALSE, sort=FALSE, prob=NULL)
```

Arguments

`df`	a data frame to be sampled from
`size`	positive integer giving the number of rows to choose
`replace`	Should sampling be with replacement?
`sort`	Should rows in sample be sorted in original order?
`prob`	a vector of probability weights for obtaining the elements of the vector being sampled

Details

Internally, rows are selected with the function `sample.int`. See its manual page for details on the arguments (except for `sort`) and implementation.

Value

A data frame containing the sampled rows of `df`, either their original order (`sort=TRUE`) or shuffled randomly (`sort=FALSE`).

Author(s)

Stephanie Evert (`https://purl.org/stephanie.evert`)

Examples

```
sample.df(BrownLOBPassives, 20, sort=TRUE)
```

`simulated.census` *Simulated census data for examples and illustrations (corpora)*

Description

This function generates a large simulated census data frame with body measurements (height, weight, shoe size) for male and female inhabitants of a highly fictitious country.

The generated data set is usually named `FakeCensus` (see code examples below) and is used for various exercises and illustrations in the SIGIL course.

Usage

```
simulated.census(N=502202, p.male=0.55, seed.rng=42)
```

Arguments

`N`	population size, i.e. number of inhabitants of the fictitious country
`p.male`	proportion of males in the country
`seed.rng`	seed for the random number generator, so data sets with the same parameters (`N`, `p.male`, etc.) are reproducible

Details

The default population size corresponds to the estimated populace of Luxembourg on 1 January 2010 (according to `https://en.wikipedia.org/wiki/Luxembourg`).

Further parameters of the simulation (standard deviation, correlations, non-linearity) will be exposed as function arguments in future releases.

Value

A data frame with `N` rows corresponding to inhabitants and the following columns:

`height`**:** body height in cm

`height`**:** body weight in kg

`shoe.size`**:** shoe size in Paris points (Continental European scale)

`sex`**:** sex, either `m` or `f`

Author(s)

Stephanie Evert (`https://purl.org/stephanie.evert`)

Examples

```
FakeCensus <- simulated.census()
summary(FakeCensus)
```

---

simulated.language.course

Simulated study on effectiveness of language course (corpora)

Description

This function generates simulated results of a study measuring the effectiveness of a new corpus-driven foreign language teaching course.

The generated data set is usually named `LanguageCourse` (see code examples below) and is used for various exercises and illustrations in the SIGIL course.

Usage

```
simulated.language.course(n=c(15,20,10,10,14,18,15), mean=c(60,50,30,70,55,50,60),
                          effect=c(5,8,12,-4,2,6,-5), sd.subject=15, sd.effect=5,
                            seed.rng=42)
```

Arguments

`n`	number of participants in each class
`mean`	average score of each class before the course
`effect`	improvement of each class during the course
`sd.subject`	inter-subject variability, may be different in each class
`sd.effect`	inter-subject variability of effect size, may also be different in each class
`seed.rng`	seed for the random number generator, so data sets with the same parameters are reproducible

Details

TODO

Value

A data frame with `sum(n)` rows corresponding to individual subjects participating in the study and the following columns

`id`: unique ID code of subject

`class`: name of the teaching class

`pre`: score in standardized language test before the course (*pre-test*)

`post`: score in standardized language test after the course (*post-test*)

Author(s)

Stephanie Evert (`https://purl.org/stephanie.evert`)

Examples

```
LanguageCourse <- simulated.language.course()
head(LanguageCourse, 20)
summary(LanguageCourse)
```

`simulated.wikipedia` *Simulated type and token counts for Wikipedia articles (corpora)*

Description

This function generates type and token counts, token-type ratios (TTR) and average word length for simulated articles from the English Wikipedia. Simulation paramters are based on data from the Wackypedia corpus.

The generated data set is usually named `WackypediaStats` (see code examples below) and is used for various exercises and illustrations in the SIGIL course.

Usage

```
simulated.wikipedia(N=1429649, length=c(100,1000), seed.rng=42)
```

Arguments

`N`	population size, i.e. total number of Wikipedia articles
`length`	a numeric vector of length 2, specifying the typical range of Wikipedia article lengths
`seed.rng`	seed for the random number generator, so data sets with the same parameters (`N` and `lenght`) are reproducible

Details

The default population size corresponds to the subset of the Wackypedia corpus from which the simulation parameters were obtained. This excludes all articles with extreme type-token statistics (very short, very long, extremely long words, etc.).

Article lengths are sampled from a lognormal distribution which is scaled so that the central 95% of the values fall into the range specified by the `length` argument.

The simulated data are surprising close to the original Wackypedia statistics.

Value

A data frame with `N` rows corresponding to Wikipedia articles and the following columns:

`tokens`: number of word tokens in the article

`types`: number of distinct word types in the article

`ttr`: token-type ratio (TTR) for the article

`avglen`: average word length in characters (averaged across tokens)

Author(s)

Stephanie Evert (`https://purl.org/stephanie.evert`)

References

The Wackypedia corpus can be obtained from `https://wacky.sslmit.unibo.it/doku.php?id=corpora`.

Examples

```
WackypediaStats <- simulated.wikipedia()
summary(WackypediaStats)
```

stars.pval — *Show p-values as significance stars (corpora)*

Description

A simple utility function that converts p-values into the customary significance stars.

Usage

```
stars.pval(x)
```

Arguments

`x`	a numeric vector of non-negative p-values

Value

A character vector with significance stars corresponding to the p-values.

Significance levels are `***` ($p < .001$), `**` ($p < .01$), `*` ($p < .05$) and `.` ($p < .1$). For non-significant p-values ($p \geq .1$), an empty string is returned.

Author(s)

Stephanie Evert (`https://purl.org/stephanie.evert`)

Examples

```
stars.pval(c(0, .007, .01, .04, .1))
```

SurfaceColloc — *A small data set of surface collocations from the English Wikipedia*

Description

This data set demonstrates how co-occurrence and marginal frequencies can be provided for collocation analysis with `am.score`. It contains surface co-occurrence counts for 7 English nouns as nodes and 7 selected collocates. The counts are based on a collocational span of two tokens to the left and right of the node (L2/R2) in the WP500 corpus. Marginal frequencies for the nodes are overall corpus frequencies of the nouns, so expected co-occurrence frequency needs to be adjusted with the total span size of 4 tokens.

Usage

```
SurfaceColloc
```

Format

A list with the following components:

`cooc`**:** A data frame with 34 rows and the following columns:

- `w1`: node word (noun)
- `w2`: collocate
- `f`: co-occurrence frequency within L2/R2 span

`f1`**:** Labelled integer vector of length 7 specifying the marginal frequencies of the node nouns.

`f2`**:** Labelled integer vector of length 7 specifying the marginal frequencies of the collocates.

`N`**:** Sample size, i.e. the total number of tokens in the WP500 corpus.

Author(s)

Stephanie Evert (`https://purl.org/stephanie.evert`)

See Also

`am.score`

Examples

```
head(SurfaceColloc$cooc, 10)
SurfaceColloc$f1
SurfaceColloc$f2
SurfaceColloc$N
```

VSS *A small corpus of very short stories with linguistic annotations*

Description

This data set contains a small corpus (8043 tokens) of short stories from the collection *Very Short Stories* (VSS, see `http://www.schtepf.de/History/pages/stories.html`). The text was automatically segmented (tokenised) and annotated with part-of-speech tags (from the Penn tagset) and lemmas (base forms), using the IMS TreeTagger (Schmid 1994) and a custom lemmatizer.

Usage

```
VSS
```

Format

A data set with 8043 rows corresponding to tokens and the following columns:

`word`: the word form (or surface form) of the token

`pos`: the part-of-speech tag of the token (Penn tagset)

`lemma`: the lemma (or base form) of the token

`sentence`: number of the sentence in which the token occurs (integer)

`story`: title of the story to which the token belongs (factor)

Details

The Penn tagset defines the following part-of-speech tags:

`CC`	Coordinating conjunction
`CD`	Cardinal number
`DT`	Determiner
`EX`	Existential *there*
`FW`	Foreign word
`IN`	Preposition or subordinating conjunction
`JJ`	Adjective
`JJR`	Adjective, comparative
`JJS`	Adjective, superlative
`LS`	List item marker
`MD`	Modal
`NN`	Noun, singular or mass
`NNS`	Noun, plural
`NP`	Proper noun, singular
`NPS`	Proper noun, plural
`PDT`	Predeterminer
`POS`	Possessive ending
`PP`	Personal pronoun
`PP$`	Possessive pronoun
`RB`	Adverb
`RBR`	Adverb, comparative
`RBS`	Adverb, superlative
`RP`	Particle
`SYM`	Symbol
`TO`	*to*
`UH`	Interjection
`VB`	Verb, base form
`VBD`	Verb, past tense
`VBG`	Verb, gerund or present participle
`VBN`	Verb, past participle
`VBP`	Verb, non-3rd person singular present
`VBZ`	Verb, 3rd person singular present
`WDT`	Wh-determiner
`WP`	Wh-pronoun

`WP$`	Possessive wh-pronoun
`WRB`	Wh-adverb

Author(s)

Stephanie Evert (https://purl.org/stephanie.evert)

References

Schmid, Helmut (1994). Probabilistic part-of-speech tagging using decision trees. In: *Proceedings of the International Conference on New Methods in Language Processing (NeMLaP)*, pages 44-49.

`z.score` *The z-score statistic for frequency counts (corpora)*

Description

This function computes a z-score statistic for frequency counts, based on a normal approximation to the correct binomial distribution under the random sampling model.

Usage

```
z.score(k, n, p = 0.5, correct = TRUE)
```

Arguments

`k`	frequency of a type in the corpus (or an integer vector of frequencies)
`n`	number of tokens in the corpus, i.e. sample size (or an integer vector specifying the sizes of different samples)
`p`	null hypothesis, giving the assumed proportion of this type in the population (or a vector of proportions for different types and/or different populations)
`correct`	if `TRUE`, apply Yates' continuity correction (default)

Details

The z statistic is given by

$$z := \frac{k - np}{\sqrt{np(1-p)}}$$

When Yates' continuity correction is enabled, the *absolute value* of the numerator $d := k - np$ is reduced by $1/2$, but clamped to a non-negative value.

Value

The z-score corresponding to the specified data (or a vector of z-scores).

Author(s)

Stephanie Evert (https://purl.org/stephanie.evert)

See Also

`z.score.pval`

Examples

```
# z-test for H0: pi = 0.15 with observed counts 10..30 in a sample of n=100 tokens
k <- c(10:30)
z <- z.score(k, 100, p=.15)
names(z) <- k
round(z, 3)

abs(z) >= 1.96  # significant results at p < .05
```

`z.score.pval` *P-values of the z-score test for frequency counts (corpora)*

Description

This function computes the p-value of a z-score test for frequency counts, based on the z-score statistic implemented by `z.score`.

Usage

```
z.score.pval(k, n, p = 0.5, correct = TRUE,
             alternative = c("two.sided", "less", "greater"))
```

Arguments

`k`	frequency of a type in the corpus (or an integer vector of frequencies)
`n`	number of tokens in the corpus, i.e. sample size (or an integer vector specifying the sizes of different samples)
`p`	null hypothesis, giving the assumed proportion of this type in the population (or a vector of proportions for different types and/or different populations)
`correct`	if `TRUE`, apply Yates' continuity correction (default)
`alternative`	a character string specifying the alternative hypothesis; must be one of `two.sided` (default), `less` or `greater`

Value

The p-value of a z-score test applied to the given data (or a vector of p-values).

Author(s)

Stephanie Evert (https://purl.org/stephanie.evert)

See Also

z.score, binom.pval, prop.cint

Examples

```
# compare z-test for H0: pi = 0.15 against binomial test
# with observed counts 10..30 in a sample of n=100 tokens
k <- c(10:30)
p.compare <- rbind(
  z.score = z.score.pval(k, 100, p=.15),
  binomial = binom.pval(k, 100, p=.15))
colnames(p.compare) <- k
round(p.compare, 4)
```

Index