Package: collostructions (via r-universe)

September 20, 2024

Type Package

Title An R Implementation for the Family of Collostructional Methods

Version 0.2.0

Date 2021-02-09

Author Susanne Flach

Maintainer Susanne Flach <susanne.flach@es.uzh.ch>

Description Functions and example data for collostructional or collocational analyses.

License GPL-2

Depends R(>= 3.0.0), stats

LazyData TRUE

Encoding UTF-8

RoxygenNote 7.1.1

URL www.sfla.ch

Repository https://staffanbetner.r-universe.dev

RemoteUrl https://github.com/skeptikantin/collostructions

RemoteRef HEAD

RemoteSha 1a8c827f2ec045bb196aaef8843461b0ce9d00f9

Contents

`beginStart` *Data set (*begin/start-to-*VERB)*

Description

Data set of the *begin/start-to-VERB* pattern in the British National Corpus (BNC), with the frequencies of the verbs in the open slot. Aggregate of `beginToV` and `startToV`, to illustrate the easiest use of `collex.dist()` with aggregated frequency lists, e.g., imported from outside R.

Usage

```
data("beginStart")
```

Format

A data frame with 2163 observations on the following 3 variables.

`WORD` a factor with 2,163 verb types either in *begin to* V and/or *start to* V.

`beginToV` numeric vector, frequencies of verbs with *begin to* V.

`startToV` numeric vector, frequencies of verbs with *start to* V.

Examples

```
## Not run:
## Distinctive Collexeme Analysis

# load data
data("beginStart")

# perform Distinctive Collexeme Analysis (with defaults)
# see ?collex.dist() for more use cases:
x <- collex.dist(beginStart)


## End(Not run)
```

beginToV *Data set (begin-to-VERB)*

Description

Data set of the *begin-to-VERB* construction in the British National Corpus (BNC), with the frequencies of the verbs in the open slot (CQP query: [hw="begin" & class="VERB"] [hw="to"] [pos="V.I"]).

Usage

```
data("beginToV")
```

Format

A data frame with 1957 observations on the following 2 variables.

`WORD` A factor with levels of types in open slot of *begin-to-VERB*

`CXN.FREQ` A numeric vector of the frequencies in V2.

Examples

```
## Not run:

data(beginToV)     # load
str(beginToV)      # inspect structure of object
head(beginToV)     # view head of object

## Calculate Simple Collexeme Analysis

# load required data
data(beginToV)     # load frequency list for construction
data(BNCverbL)     # load frequency list for verb frequencies (lemmas)

# join frequency files to create input for collex()
beginToV.in <- join.freqs(beginToV, BNCverbL, all = FALSE) # only types in cxn
beginToV.in <- join.freqs(beginToV, BNCverbL) # all types, even if not in cxn

# calculate
beginToV.out <- collex(beginToV.in, sum(BNCverbL$CORP.FREQ)) # using logL
beginToV.out <- collex(beginToV.in, sum(BNCverbL$CORP.FREQ), "mi") # mi

# inspect result
head(beginToV.out, 20) # view first 20 lines of calculation
tail(beginToV.out, 20) # view last 20 lines of calculation

## Calculate Distinctive Collexeme Analysis

# load data
```

```
data(beginToV)
data(startToV)

# merge frequency lists
# the first argument to join.freqs() will be the 'default' by which output is
# sorted and Z.DIR is calculated
beginStart.in <- join.freqs(beginToV, startToV) # merge both data frames

# calculate
beginStart.out <- collex.dist(beginStart.in)

# inspect result
head(beginStart.out, 20)
head(beginStart.out, 20)

## End(Not run)
```

BNCverbL *Data set (BNC verb lemma frequency list)*

Description

A data frame with case-insensitive BNC verb lemma frequencies.

Usage

```
data("BNCverbL")
```

Format

A data frame with 35939 observations on the following 2 variables.

`WORD` a factor with levels for all verb lemmas

`CORP.FREQ` a numeric vector with verb lemma frequencies

Details

Lemmas starting in problematic characters have been removed (almost exclusively tokenization problems, e.g. single quotes, slashes, backslashes, hashes, asterisks or square brackets). Please make sure you have a clean file of frequencies if you want to use this data for joining frequency lists to avoid problems in collex().

Source

BNCxml version (CQP query: [class="VERB"])

Examples

```
## Not run:

data(BNCverbL)
str(BNCverbL)
head(BNCverbL)


## End(Not run)
```

---

causInto *Data set (*into-*causative)*

Description

A dataset of the *into*-causative (e.g., *they forced us into thinking...*) from the BNC for the illustration of functions. Contains one observation/token per line.

Usage

```
data("causInto")
```

Format

A data frame with 1,426 observations on the following 3 variables.

`VOICE` a factor with annotation for voice (levels `active`, `passive`, and `reflexive`)

`V1` a factor with levels for the matrix verb, lemmatised (e.g., *they **force**d us into thinking...*)

`V2` a factor with levels for the content verb, lemmatised (e.g., *they forced us into **think**ing...*)

References

Flach, Susanne. 2018. "What's that passive doing in my argument structure construction?" A note on constructional interaction, verb classes and related issues. Talk at the Workshop on constructions, verb classes, and frame semantics, IDS Mannheim.

Examples

```
## Not run:

## E.g., in Co-Varying Collexeme Analysis:

# load data
data(causInto)

# inspect, contains more variables than needed for collex.covar()
head(causInto)
```

```
# subset
into.cca <- subset(causInto, select = c(V1, V2))

# perform CCA
into.cca <- collex.covar(into.cca)

## End(Not run)
```

---

causMake *Data set (*make-*causative)*

Description

A toy sample data set of pronominal *make*-causative (*this makes me feel X*) from the British National Corpus (BNC) to illustrate multiple distinctive co-varying collexeme analysis (see `collex.covar.mult()`).

Usage

```
data("causMake")
```

Format

A data frame with 5000 observations of the following 3 variables.

`MAKE` a factor with 4 levels of different inflectional forms of *make*

`OBJ` a factor with 7 pronoun levels levels (i.e., `her` `him` `it` `me` `them` `us` `you`)

`V2` a factor with levels 633 levels, i.e., the verbs in the complement.

Examples

```
## Not run:
## Multiple Distinctive Collexeme Analysis

# load data
data("causMake")

# perform Multiple Distinctive Co-Varying Collexeme Analysis (with defaults)
# see ?collex.covar.mult for more use cases:
x <- collex.covar.mult(causMake)

## End(Not run)
```

CLMETprog.qc *Data set (Present Progressive by quarter century, CLMET)*

Description

Data based on Johannsen & Flach (2015) with frequencies of all verbs occurring in the present progressive in the *Corpus of Late Modern English Texts* (CLMET-3.1). Contains only occurrences with unambiguous assignment of quarter century.

Usage

```
data("CLMETprog.qc")
```

Format

A data frame with 5,291 observations on the following 3 variables.

`WORD` a factor with levels for all lemmas occurring as types in present progressive

`QUARTCENT` a factor with levels `1700-1724` `1725-1749` ... `1900-1924`

`CXN.FREQ` a numeric vector with the corpus frequencies

Details

CQP version query: [pos="VB[PZ]" & lemma="be"] [class="ADV|PRON"]* [pos="V.G"]

Source

Corpus of Late Modern English Texts, CLMET-3.1 (De Smet, Flach, Tyrkkö & Diller 2015), CQP version.

References

De Smet, Hendrik, Susanne Flach, Jukka Tyrkkö & Hans-Jürgen Diller. 2015. *The Corpus of Late Modern English (CLMET), version 3.1: Improved tokenization and linguistic annotation.* KU Leuven, FU Berlin, U Tampere, RU Bochum.

Johannsen, Berit & Susanne Flach. 2015. Systematicity beyond obligatoriness in the history of the English progressive. Paper presented at ICAME 36, 27–31 May 2015, Universität Trier.

Examples

```
## Not run:
data(CLMETprog.qc)

## End(Not run)
```

`CLMETsimple.qc` *Data set (Present Simple by quarter century, CLMET)*

Description

Data based on Johannsen & Flach (2015) with frequencies of all verbs occurring in the simple present in the *Corpus of Late Modern English Texts* (CLMET-3.1). Contains only occurrences with unambiguous assignment of quarter century.

Usage

```
data("CLMETsimple.qc")
```

Format

A data frame with 24,693 observations on the following 3 variables.

`WORD` a factor with levels for all lemmas occurring as types in present simple

`QUARTCENT` a factor with levels `1700-1724` `1725-1749` ... `1900-1924`

`CORP.FREQ` a numeric vector with the corpus frequencies

Details

CQP version query: [pos="VB[PZ]"] – frequencies for *be* were reduced by their frequency value in `CLMETprog.qc` to avoid them being present in both data sets.

Source

Corpus of Late Modern English Texts, CLMET-3.1 (De Smet, Flach, Tyrkkö & Diller 2015), CQP version.

References

De Smet, Hendrik, Susanne Flach, Jukka Tyrkkö & Hans-Jürgen Diller. 2015. *The Corpus of Late Modern English (CLMET), version 3.1: Improved tokenization and linguistic annotation.* KU Leuven, FU Berlin, U Tampere, RU Bochum.

Johannsen, Berit & Susanne Flach. 2015. Systematicity beyond obligatoriness in the history of the English progressive. Paper presented at ICAME 36, 27–31 May 2015, Universität Trier.

Examples

```
## Not run:
data(CLMETsimple.qc)
str(CLMETsimple.qc)
head(CLMETsimple.qc)

## End(Not run)
```

collex *Function for Simple Collexeme Analysis*

Description

Implementation of Simple Collexeme Analysis (Stefanowitsch & Gries 2003) over a data frame with frequencies of verbs in a construction and their total frequencies in a corpus.

Usage

```
collex(x, corpsize = 1e+08L, am = "logl", reverse = FALSE, decimals = 5,
       threshold = 1, cxn.freq = NULL, str.dir = FALSE, p.fye = FALSE,
       delta.p = FALSE, p.adj = "none")
```

Arguments

x	A data frame with types in a construction in column 1 (WORD), construction frequencies in column 2 (CXN.FREQ) and corpus frequencies in column 3 (CORP.FREQ). The name of the columns in your input file is up to you.
corpsize	The size of the corpus in number of tokens (e.g., verb frequencies for all verb constructions, or total corpus size etc.). If not given, default is 100 million words, roughly the size of the BNC, but you should always provide the appropriate number. (Note: corpsize is different to the argument cxn.freq, which refers to the total number of tokens of the construction, see below.)
am	Association measure to be calculated. Currently available, tested, and conventional in the collostruction literature: "logl" (log likelihood, the default) "fye" (negative decadic log transformed p-value of FYE, the original), can be used in conjunction witht the argument p.fye to calculate the original Fisher-Yates p-value. "fye.ln" (negative natural log transformed p-value of FYE, variant of fye). Experimental, so use with caution (implementing Evert 2004). They are association measures for the investigation of collocations, and most do not make much sense in the context of collostructions, but they may be useful for some purposes: "chisq", "dice", "gmean", "jaccard", "liddell", "mi", "mi3", "ms", "odds", "pois", "t", "z", "z.cor", "random". Based on a chi-square test, there is also "cramersV", which is an effect size for chi-squared.
reverse	If FALSE (default), output will be ordered in descending order of collostruction strength (i.e. descending order of attraction). Use reverse = TRUE for ordering in ascending order of collostruction strength (i.e. descending order of repulsion).
decimals	Number of decimals in the output. Default is 5 decimal places (except for EXP, which is always 1).

`threshold` Frequency threshold of items for which collostruction strength is to be calculated, i.e., if you want to exclude hapaxes from the *output* (they are not excluded from the calculation).

`cxn.freq` Frequency of construction. Use *only* if `x` does not contain all instances of the construction (e.g., if hapaxes have been removed beforehand or if the corpus you queried does not give you access to full frequency information on all tokens/types). The default case is that `collex()` automatically calculates `cxn.freq` from the input in `x`. Note that `cxn.freq` is independent of the setting for `threshold` (which determines how many of your *input* types you want in the *output*).

`str.dir` Do you want a "directed" association measure in the output? For measures that are positive for attracted and repelled items, `str.dir = TRUE` returns negative values for repelled items and allows differentiation of attraction based on this measure. Default is `FALSE`.

`p.fye` Should the traditional Fisher-Yates p-value be calculated? This will not have any effect unless `am = "fye"`.

`delta.p` Should delta*P* be calculated? If yes, both types of delta*P* will be calculated (see below).

`p.adj` If an association measure is chosen that provides significance levels (`"logl"`, `"chisq"`, `"fye"`, `"fye.ln"`), the significance levels in the output are adjusted for multiple hypothesis testing. The default is the Bonferroni correction. You can use any of the adjustment methods (`"holm"`, `"hochberg"`, `"hommel"`, `"BH"`, `"BY"`, `"fdr"`) as used in `p.adjust` (follow link for further details). If you don't want a correction, use `p.adjust = "none"`.

Details

Corpus size: It's highly recommended to specify `corpsize` to a conceptually sensible value. If you do not, the default may give you anything from an error for data from much larger corpora than the BNC to highly inaccuarate results for small corpora or infrequent phenomena. For phenomena from the BNC, this will probably not distort results too much (cf. Gries in multiple discussions of the "Fourth Cell(TM)").

FYE: Note to users of `am = "fye"`: packages versions up to 0.0.10 used the negative natural logarithm for p-value transformation. Versions >= 0.1.0 use the negative decadic logarithm. If you want to continue using the natural logarithm transformation, use `am = "fye.ln"` as an association measure and repeat procedure. (If you see this message, you are using a version >= 0.1.0. It will disappear in versions >= 1.0 once on CRAN).

Association measures: The default `"logl"` as an association measure is due to the fact that for larger datasets from larger corpora the original `"fye"` easily returns `Inf` for the most strongly associated and/or dissociated items, which are then non-rankable (they are ranked by frequency of occurrence if this happens).

Thresholds: The `threshold` argument default of 1 does not remove non-occurring items (although the logic of this argument implies as much). This is a "bug" that I decided to keep for historical reasons. If you do not want to calculate the repulsion for non-occurring items, you need to enter a frequency list that contains only the occurring items.

Value

The output of `collex()` is sorted by collostructional strength, with most attracted to least attracted; for ties, ordering is by frequency (further columns depent on argument settings):

`COLLEX`	The collexemes.
`CORP.FREQ`	Frequency of collexeme in corpus.
`OBS`	Observed frequency of collexeme in construction.
`EXP`	Expected frequency in construction
`ASSOC`	Association of collexeme: `attr` (attracted) or `rep` (repelled), based on the difference between observed and expected.
`COLL.STR./AM/`	Association measure used. `/AM/`, i.e., type of association score should always be reported along with values.
`STR.DIR`	Same as collostruction strength (`COLL.STR`), but "directed", i.e. positive for attraction and negative for repulsion. Only displayed if `str.dir = TRUE`.
`DP1`	Association (word-to-cxn), i.e., delta*P*(w|cxn), see Gries & Ellis (2015: 240)
`DP2`	Association (cxn-to-word), i.e., delta*P*(cxn|w), see Gries & Ellis (2015: 240)
`SIGNIF`	Significance level. `*****` = significant at $p < .00001$, `****` = significant at $p < .0001$, `***` at $p < .001$, `**` at $p < .01$, `*` at $p < .05$, and `ns` is not significant (but tendency is given by the difference between `OBS` and `EXP`, i.e. as returned in `ASSOC`). Association measures without significance tests have `SIGNIF == na`.

Note

The function will abort if your input data frame has items with 'non-sensical' data, that is, if a collexeme has a higher frequency in the *construction* than it has in the *corpus* (which is logically impossible of course). This is a surprisingly common problem with untidy corpus frequency lists derived from messy annotation, especially when the collexemes have been manually cleaned from a rather inclusive query, but the corpus frequencies have different/erroneous part-of-speech tagging (cf. Flach 2015), where a syntactically quirky constructions in *Let's **go party*** was "hand-cleaned", but *party* did not have any frequency as a verb, because it was always tagged as a noun. As of package version 0.2.0, the function aborts with a descriptive error message, and prints a list of the items with non-sensical frequencies. For further input checks, see `input.check()`.

Author(s)

Susanne Flach, susanne.flach@es.uzh.ch

Thanks to Anatol Stefanowitsch, Berit Johannsen, Kirsten Middeke, Volodymyr Dekalo and Robert Daugs for suggestions, debugging, and constructive complaining, and to Stefan Hartmann, who doesn't know what a complaint is, but who provided invaluable feedback when asked how the package could be improved.

References

Evert, Stefan. 2004. *The statistics of word cooccurrences: Word pairs and collocations*. U Stuttgart Dissertation. http://www.collocations.de/AM/

Flach, Susanne. 2015. Let's go look at usage: A constructional approach to formal constraints on go-VERB. In Thomas Herbst & Peter Uhrig (eds.), *Yearbook of the German Cognitive Linguistics Association* (Volume 3), 231-252. Berlin: De Gruyter Mouton. doi:10.1515/gcla-2015-0013.

Gries, Stefan Th. & Nick C. Ellis. 2015. Statistical measures for usage-based linguistics. *Language Learning* 65(S1). 228–255. doi:10.1111/lang.12119.

Stefanowitsch, Anatol & Stefan Th. Gries. 2003. Collostructions: Investigating the interaction of words and constructions. *International Journal of Corpus Linguistics* 8(2). 209-243.

Examples

```
## Not run:

#### Calculate Simple Collexeme Analysis
## Example 1: goVerb (cf. Flach 2015)

# load data
data(goVerb)

# inspect data (optional)
head(goVerb, 15) # displays first 15 lines of data frame

# perform collex
goV.out <- collex(goVerb, 616336708)  # total words in corpus (excl. punct)
goV.out <- collex(goVerb, 93993713)   # total verbs in corpus

# inspect output
head(goV.out, 15)  # first 15 items (strongly attracted)
tail(goV.out, 15)  # last 15 items (strongly repelled)

# clear workspace (remove objects)
rm(goVerb, goV.out)

## Example 2: beginToV (also see help file for ?beginToV)
data(beginToV)     # load data for begin-to-V
data(BNCverbL)     # load a frequency list for verb string frequencies

# merge frequency lists (see ?join.freqs):
beginToV.in <- join.freqs(beginToV, BNCverbL, all = FALSE)

# perform collex
beginToV.out <- collex(beginToV.in, sum(BNCverbL$CORP.FREQ)) # using logL

# inspect output
head(beginToV.out, 30)   # first 30 most strongly associated types
tail(beginToV.out, 20)   # last 20 items least strongly associated types

# clear workspace (remove objects)
rm(beginToV, BNCverbL, beginToV.in, beginToV.out)

##### SPECIAL: IN USE OVER LISTS
```

```
## collex() can be used to perform several Simple Collexeme Analyses
## in one function call. See ?collex.dist for an example of multiple
## analyses across time periods (simple vs. progressive). The procedure with
## collex() is almost identical, except that you should use Map(...),
## because you have to provide corpus frequencies for each period
## (i.e., for each iteration of collex()):

# 1. Create a numeric vector of corpus frequencies:
corpfreqs <- c(corpFreqPeriod1, corpFreqPeriod2, ...)

# 2. Pass 'corpfreqs' vector as an argument to Map() like so:
myList.out <- Map(collex, myList.in, corpsize = corpfreqs, ...)

## End(Not run)
```

`collex.covar` *Function for Covarying Collexeme Analysis*

Description

Implementation of Covarying Collexeme Analysis (Gries & Stefanowitsch 2004, Stefanowitsch & Gries 2005) to investigate the collostructional interaction between two slots of a construction.

Usage

```
collex.covar(x, am = "logl", raw = TRUE, all = FALSE, reverse = FALSE,
            decimals = 5, str.dir = FALSE, p.fye = FALSE, delta.p = FALSE)
```

Arguments

`x`	Input, a data frame. Two options: EITHER as raw, with one observation per line, and with collexeme 1 in column 1 and collexeme 2 in column 2 (in which case `raw = TRUE`, the default) OR as an aggregated frequency list, which must contain a third column with the frequency of the combinations in columns 1 and 2 (in which case you must set `raw = FALSE`).
`am`	Association measure to be calculated. Currently available, tested, and conventional in the collostruction literature: `"logl"` (log likelihood, the default) `"fye"` (negative decadic log transformed p-value of FYE, the original), can be used in conjunction witht the argument `p.fye` to calculate the original Fisher-Yates p-value. `"fye.ln"` (negative natural log transformed p-value of FYE, variant of `fye`). Experimental, so use with caution (implementing Evert 2004). They are primarily collocation measures, and most make not much sense in the context of collostructions, but may be useful for some purposes: `"chisq"`, `"dice"`, `"gmean"`, `"jaccard"`, `"liddell"`, `"mi"`, `"mi3"`, `"ms"`, `"odds"`, `"pois"`, `"t"`, `"z"`, `"z.cor"`, `"random"`.

raw	TRUE (default) or FALSE. If you have one observation per line (i.e. a 'raw' list), use default. If you enter an aggregated frequency list, set raw = FALSE.
all	FALSE (default) or TRUE. The default calculates attested combinations. If all possible combinations of slots 1 and 2 are to be included, set all = TRUE, but this can take a long time for constructions with high type frequencies.
reverse	FALSE (default) or TRUE. If FALSE, output will be ordered in descending order of attraction. Use TRUE for ordering in descending order of repulsion.
decimals	Number of decimals in the output. Default is 5 decimal places (except for EXP, which is always 1).
str.dir	Do you want a "directed" collostruction strength in the output? For measures that are positive for both attracted and repelled items, str.dir = TRUE returns negative values for repelled items. Default is FALSE.
p.fye	If am = "fye", should traditional pFYE-value be calculated? This feature is experimental, and will not have an effect unless am = "fye".
delta.p	Should delta*P* be calculated? If yes, both types of delta*P* will be calculated (see below).

Value

Output is ordered in descending order of association (unless reverse = TRUE):

SLOT1	type/item in slot 1 (e.g. string or lemma...)
SLOT2	type/item in slot 2 (e.g. string or lemma...)
fS1	Total frequency of item 1 in slot 1 of cxn
fS2	Total frequency of item 2 in slot 2 of cxn
OBS	Observed frequency of combination
EXP	Expected frequency of combination
ASSOC	Association of combination (attr or rep).
COLL.STR./AM/	Value of association measure used.
STR.DIR	"Directed" collostruction strength. STR.DIR is positive if item is attracted and negative if repelled. (Only displayed if str.dir = TRUE.)
DP1	Association (slot1-to-slot2), i.e., delta*P*(s2|s1), how predictive is slot 1 of slot 2? See Gries & Ellis (2015: 240)

in the constructional context.

DP2	Association (slot2-to-slot1), i.e., delta*P*(s1|s2), how predictive is slot 2 of slot 1? See Gries & Ellis (2015: 240)

in the constructional context.

SIGNIF	Significance level. ***** = significant at $p < .00001$, **** = significant at $p < .0001$, *** at $p < .001$, ** at $p < .01$, * at $p < .05$, and ns is not significant (but tendency is given by the difference between OBS and EXP, i.e. as returned in ASSOC). Association measures without significance tests have SIGNIF == na.

Note

If you use the function on constructions with a high type frequency, be patient when setting `all = TRUE`. The function needs to perform `Types.In.A` x `Types.In.B` number of tests. Even on a fairly powerful computer it can take about half an hour to perform ~500,000+ tests.

For Multiple Distinctive Collexeme Analysis (MDCA), where you have more than two conditions/constructions, you can use `collex.covar()`. The association scores of CCA (`collex.covar()`) and approximation-based MDCA correlate highly (e.g., for the `modadv` data: Pearson r = .9987841; Spearman's rho = .9999993), suggesting `collex.covar()` is a workable alternative to approximation.

Note to users of `am = "fye"`: packages versions up to 0.0.10 used the negative natural logarithm for p-value transformation. Versions >= 0.1.0 use the negative decadic logarithm. If you want to continue using the natural logarithm transformation, use `am = "fye.ln"` as an association measure and repeat procedure. (If you see this message, you are using a version >= 0.1.0. It will disappear in versions >= 1.0 once on CRAN).

Author(s)

Susanne Flach, susanne.flach@es.uzh.ch

Thanks to Anatol Stefanowitsch, Berit Johannsen, Kirsten Middeke and Volodymyr Dekalo for suggestions, debugging, and constructive complaining.

References

Evert, Stefan. 2004. The statistics of word cooccurrences. Word pairs and collocations. Stuttgart: Universität Stuttgart Doctoral Dissertation. http://www.stefan-evert.de/PUB/Evert2004phd.pdf.

Gries, Stefan Th. & Nick C. Ellis. 2015. Statistical measures for usage-based linguistics. *Language Learning* 65(S1). 228–255. doi:10.1111/lang.12119.

Gries, Stefan Th. & Anatol Stefanowitsch. 2004. Covarying collexemes in the into-causative. In Michel Archard & Suzanne Kemmer (eds.), *Language, culture, and mind*, 225–236. Stanford, CA: CSLI.

Stefanowitsch, Anatol & Stefan Th. Gries. 2005. Covarying collexemes. *Corpus Linguistics and Linguistic Theory* 1(1). 1–43. doi:10.1515/cllt.2005.1.1.1.

See Also

Use `reshape.cca` to transform the output of `collex.covar` from the 'long' format to a 'wide' format (i.e., cross-tabulate association scores.)

Examples

```
## Not run:

### Example I: Attested combinations (only)
data(caus.into)
```

```
# inspect
head(caus.into)

# subset, because caus.into contains too many variables
into.vrbs <- caus.into[, c(2,3)]    # CCA between V1 and V2
into.voice <- caus.into[, c(1,2)]   # CCA between VOICE and V1

# perform Co-Varying Collexeme Analysis
into.vrbs.cca <- collex.covar(into.vrbs)
into.voice.cca <- collex.covar(into.voice)

# clear workspace (remove objects)
rm(into.voice, into.vrbs.cca, into.voice.cca)

### Example 2: If you want to test all possible combinations
# Depending on your machine, this may take a while, because it needs
# to perform 199*426 = 84,774 tests for the 199 slot 1 types and the
# 426 slot 2 types (rather than the 1,076 tests for attested combinations).
into.vrbs.cca.all <- collex.covar(into.vrbs, all = TRUE)

### Example 3: An aggregated list
# set raw = FALSE, otherwise the function will abort
# (output wouldn't make any sense):
data(modadv)
head(modadv, 12)
modadv.cca <- collex.covar(modadv, raw = FALSE)

## End(Not run)
```

`collex.covar.mult` *Function for Multiple Covarying Collexeme Analysis*

Description

Implementation of Multiple/Distinctive Covarying Collexeme Analysis (Stefanowitsch & Flach 2020) to investigate the collostructional association between three or more slots/conditions of a construction.

Usage

```
collex.covar.mult(x, am = "t", raw = TRUE, all = FALSE, reverse = FALSE,
                  threshold = 1, decimals = 5)
```

Arguments

x	A data frame with each (categorical) condition in one column. In principle, these conditions can be anything: open slots in a construction (verbs, nouns, prepositions, etc.), constructions (e.g., ditransitive/prep-dative, negation [y/n]), annotated variables, genre, time periods...). The function assumes a raw list by

	default, i.e., with one observation per line. If you have an aggregated list, the last column must contain the frequencies (in which case set `raw = FALSE`).
`am`	Association measure to be calculated. Currently available (though very experimental, see below): `"t"` (*t*-score, the default), `"pmi"` (positive mutual information), `"poiss"` (poisson-stirling), `"z"` (*z*-score), `"locmi"` (local mutual information) and `"tmi"` (true mutual information). See below.
`raw`	Does the data frame contain raw observations, i.e., one observation per line? Default is `raw = TRUE`.
`all`	Should association be calculated for all possible combinations? The default is `FALSE`, which calculates association only for attested combinations. Please read the notes below for when `TRUE` makes sense and when it should be avoided.
`reverse`	The default sorting is in descending order by positive attraction. Set to `TRUE` for reverse sorting.
`threshold`	Set this to a positive integer if the *output* should only contain combinations with a frequency above a certain threshold. Please read the notes below.
`decimals`	The number of decimals in the association measure.

Details

General: This function uses a code section from the function `scfa` from the R package `cfa` (Mair & Funke 2017) for the calculation of expected values. Multiple Covarying Collexeme Analysis is conceptually essentially Configural Frequency Analysis (CFA; van Eye 1990) in a collostructional context. If you use this method, you can cite Stefanowitsch & Flach (2020).

Note that this function can only provide measures based on the counts/observations in the data frame. That is, different to the other collexeme functions, there is no option to supply overall (corpus) frequencies if your data frame contains only a sample of all corpus counts. For instance, if you have removed hapax combinations from your data frame, then the frequencies of all types that occur in hapax types will not be included.

All combinations: While the calculation of association measures for all possible combinations of all conditions `all = TRUE` is necessary if you want to assess the relevance of the *absence* of a combination (negative association value for most measures), you need to be aware of the consequences: for use cases with high type frequencies this will involve the calculation of a huge number of *n*-types, which can break the R session. It is doubtful if it is linguistically relevant anyway: most high(er) frequent combinations that are linguistically interesting will have at least a few observations. **Also note** that if you supply an aggregated data frame (with a frequency column, i.e., `raw = FALSE`), `all = TRUE` currently has no effect (and it's doubtful that this will be implemented). In this case, you can 'untable' your original data frame, see examples below.

Threshold: You can restrict the output such that only combinations that occur an *n* number of times are shown. This might be useful if you have a large data frame.

Association measures: The implemented measures are relatively simple (as they only involve observed and expected values), but they do the trick of separating idiomatic, highly associated types from less strongly associated ones. Most measures are based on Evert (2004), with the exception

of `tmi`. As there are, as of yet, no sufficient number of studies it is difficult to advise, but the `t-score` appears relatively robust (hence the default). However, since an observed value of 0 (if `all = TRUE`) would result in `-Inf`, 0.5 is added to the observed value of unattested combinations before the `t-value` is calculated.

Value

The output is sorted in decending order of attraction.

`\emph{CONDITION}...`	The first *n* columns are the conditions you originally entered.
`OBS`	Observed frequency of the combination
`EXP`	Expected frequency of the combination
`\emph{AM}`	The chosen association measure.

Author(s)

Susanne Flach, susanne.flach@es.uzh.ch

References

Patrick Mair and Stefan Funke (2017). cfa: Configural Frequency Analysis (CFA). R package version 0.10-0. https://CRAN.R-project.org/package=cfa

Eye, A. von (1990). *Introduction to configural frequency analysis. The search for types and antitypes in cross-classification*. Cambridge: CUP.

Stefanowitsch, Anatol & Susanne Flach. 2020. *Too big to fail* but *big enough to pay for their mistakes*: A collostructional analysis of the patterns [too ADJ to V] and [ADJ enough to V]. In Gloria Corpas & Jean-Pierre Colson (eds.), *Computational Phraseology*, 248–272. Amsterdam: John Benjamins.

Examples

```
## Not run:
## Multiple Distinctive Collexeme Analysis

## Case 1: Raw list of observations
# load data

## Case 3: You only have an aggregated list with attested values,
## but want to measure the relevance of the absence (i.e., dissociation measures)
library(reshape)
# Untable (where df stands for your original data frame, minus the last column with the frequencies)
df_new <- untable(df[,-ncol(df)], num = df[, ncol(df)])

## End(Not run)
```

collex.dist *Function for Distinctive Collexeme Analysis*

Description

Implementation of Distinctive Collexeme Analysis (Gries & Stefanowitsch 2004) to be run over a data frame with frequencies of items in two alternating constructions OR, more generally, for keyword analysis over a data frame with frequency lists of two corpora OR, most generally, over a data frame with comparison of frequencies of two conditions. Note: if you want to perform Multiple Distinctive Collexeme Analysis, use `collex.covar()`, see **Notes** below.

Usage

```
collex.dist(x, am = "logl", raw = FALSE, reverse = FALSE, decimals = 5,
            threshold = 1, cxn.freqs = NULL, str.dir = FALSE, p.fye = FALSE,
            delta.p = FALSE)
```

Arguments

`x`	Input, a data frame. Two options: EITHER as aggregated frequencies with the types in column A (`WORD`), and the frequencies of `WORD` in the first construction in column 2 and in the frequencies of `WORD` in the second construction in column 3, OR as raw data, i.e., one observation per line, where column 1 must contain the construction types and column 2 must contain the collexeme (see `head(future)` or `?future` for an example of this format). The names of the columns is up to you, but you must set `raw = TRUE` if the input data frame has one observation per line, otherwise the function will abort with an error message to that effect (see below).
`am`	Association measure to be calculated. Currently available, tested, and conventional in the collostruction literature: `"logl"` (log likelihood, the default) `"fye"` (negative decadic log transformed p-value of FYE, the original), can be used in conjunction witht the argument `p.fye` to calculate the original Fisher-Yates p-value. `"fye.ln"` (negative natural log transformed p-value of FYE, variant of `fye`). Experimental, so use with caution (implementing Evert 2004). They are primarily collocation measures, and most make not much sense in the context of collostructions, but may be useful for some purposes: `"chisq"`, `"dice"`, `"gmean"`, `"jaccard"`, `"liddell"`, `"mi"`, `"mi3"`, `"ms"`, `"odds"`, `"pois"`, `"t"`, `"z"`, `"z.cor"`, `"random"`. Based on a chi-square test, there is also `"cramersV"`, which is an effect size for chi-squared.
`raw`	Does input data frame contain a raw list of occurrences? Leave default `raw = FALSE`, if your input contains aggregated frequencies with collexeme types in column 1 and the frequencies in the alternating constructions in columns 2 and 3, respectively. Set this argument to `raw = TRUE`, if you have one observation per

line, where the type of construction is in column 1 and the collexeme in column 2 (see ?future data set for an example). The function will then produce the combined frequency list before performing the distinctive collexeme analysis.

`reverse` FALSE (default) or TRUE. If FALSE, output will be ordered in descending order of collostruction strength relative to condition A (i.e. attraction to condition A). Use `reverse = TRUE` for ordering in descending order of collostruction strength relative to condition B (i.e. repulsion to condition A).

`decimals` Number of decimals in the output. Default is 5 decimal places (except for EXP, which is always 1).

`threshold` Frequency threshold for items you want to calculate association measures for. By default, this is 1 (= calculated for all items). Any number higher than 1 will exclude items that have a combined frequency lower than the threshold. For instance, `threshold = 2` will exclude hapaxes, `threshold = 3` will exclude items occurring only once or twice, etc. Note: the threshold refers to overall frequency, i.e., if an item occurs four times in one condition, but never in the other, this item is excluded at `threshold = 5`, whereas items that occur twice in condition A and three times in condition B (total of 5) is included, so make sure this is what you want.

`cxn.freqs` A numeric vector or list of length 2. This option lets you enter construction frequencies 'manually' if x is a reduced dataset (e.g., from `join.freqs()` with a threshold higher than 1). For full data in x, the function will extract the required construction frequencies for calulations of association directly from x, but if x contains a reduced data set, you *must* provide corpus frequencies or else your results will be wrong (although they may look reasonable). Note that `cxn.freqs` is independent of the setting for `threshold` (which refers to the output).

`str.dir` Do you want a "directed" association measure in the output? For measures that are positive for attracted and repelled items, `str.dir = TRUE` returns negative values for repelled items and allows differentiation of attraction based on this measure. Default is FALSE.

`p.fye` If `am = "fye"`, should traditional pFYE-value be calculated? This feature is experimental, and will not have an effect unless `am = "fye"`.

`delta.p` Should delta*P* be calculated? If yes, both types of delta*P* will be calculated (see below).

Details

FYE: Note to users of `am = "fye"`: packages versions up to 0.0.10 used the negative natural logarithm for p-value transformation. Versions >= 0.1.0 use the negative decadic logarithm. If you want to continue using the natural logarithm transformation, use `am = "fye.ln"` as an association measure and repeat procedure. (If you see this message, you are using a version >= 0.1.0. It will disappear in versions >= 1.0 once on CRAN).

Association measures: The default `"logl"` as an association measure is due to the fact that for larger datasets from larger corpora the original `"fye"` easily returns `Inf` for the most strongly associated and/or dissociated items, which are then non-rankable (they are ranked by frequency of occurrence if this happens).

Thresholds: The `threshold` argument default of 1 does not remove non-occurring items (although the logic of this argument implies as much). This is a "bug" that I decided to keep for historical reasons (and to avoid problems with `collex()`. Use primarily to leave out low-frequency items in the *output*.

Value

OUTPUT ordered in descending order of association to cxn/condition A (unless `reverse = TRUE`):

`COLLEX`	type/item (e.g. string, lemma...)
`O.CXN1`	Observed frequency in cxn/condition A
`E.CXN1`	Expected frequency in cxn/condition A
`O.CXN2`	Observed frequency in cxn/condition B
`E.CXN2`	Expected frequency in cxn/condition B
`ASSOC`	Name of cxn/condition with which the COLLEX type is associated.
`COLL.STR./AM/`	Association measure used. `/AM/`, i.e., type of association score should always be reported along with values.
`STR.DIR`	"Directed" collostruction strength. `STR.DIR` is positive if item is attracted to condition A and negative if attracted to condition B (i.e. 'negatively attracted' to condition A). Only displayed if `str.dir = TRUE`.
`DP1`	Association (word-to-cxn), i.e., delta$P$(w|cxn), see Gries & Ellis (2015: 240)
`DP2`	Association (cxn-to-word), i.e., delta$P$(cxn|w), see Gries & Ellis (2015: 240)
`SIGNIF`	Significance level. `*****` = significant at $p < .00001$, `****` = significant at $p < .0001$, `***` at $p < .001$, `**` at $p < .01$, `*` at $p < .05$, and `ns` is not significant (but tendency is given by the difference between `OBS` and `EXP`, i.e. as returned in `ASSOC`). Association measures without significance tests have `SIGNIF == na`.
`SHARED`	Is `COLLEX` a shared type between cxn/condition A and B? Returns `Y` (yes) or `N` (no).

Note

Multiple Distinctive Collexeme Analysis: If you want to perform a Multiple Distinctive Collexeme Analysis (MDCA) for more than two levels of your construction, you cannot use `collex.dist()`, as it can only handle two levels of cxn. Instead, use `collex.covar()` for Co-Varying Collexeme Analysis (CCA), which can handle more than two levels of the first condition. The main difference between MDCA and CCA is conceptual (and arguably historical), but they are mathematically the same thing; the association scores of CCA and approximation-based MDCA correlate highly (e.g., for the `modadv` data: Pearson $r = .9987841$; Spearman's rho = .9999993).

Author(s)

Susanne Flach, susanne.flach@es.uzh.ch

Thanks to Anatol Stefanowitsch, Berit Johannsen, Kirsten Middeke and Volodymyr Dekalo for discussion, suggestions, debugging, and constructive complaining.

References

Evert, Stefan. 2004. *The statistics of word cooccurrences: Word pairs and collocations*. U Stuttgart Dissertation. http://www.collocations.de/AM/

Gries, Stefan Th. & Nick C. Ellis. 2015. Statistical measures for usage-based linguistics. *Language Learning* 65(S1). 228–255. doi:10.1111/lang.12119.

Gries, Stefan Th. & Anatol Stefanowitsch. 2004. Extending collostructional analysis: A corpus-based perspective on "alternations." *International Journal of Corpus Linguistics* 9(1). 97-129.

Hilpert, Martin. 2011. Diachronic collostructional analysis: How to use it and how to deal with confounding factors. In Kathryn Allan & Justyna A. Robinson (eds.), *Current methods in historical semantics*, 133–160. Berlin & Boston: De Gruyter.

Johannsen, Berit & Susanne Flach. 2015. Systematicity beyond obligatoriness in the history of the English progressive. Paper presented at ICAME 36, 27–31 May 2015, Universität Trier.

See Also

See `freq.list()` for an easy function to create frequency lists from vectors of characters in preparation for Distinctive Collexeme Analysis.For use with incomplete data, see example in `ditrdat_pub`.

Examples

```
## Not run:
##### Calculate Distinctive Collexeme Analysis
## This is a little lengthy, because there are multiple ways to provide
## input to DCA (Case 1, Case 2, and Case 3). There are also use cases
## to run multiple DCA over a list of files (see below).

### Case 1: An aggregated frequency list
## The easiest use case: Words in col1, and their frequencies
## in cxns A and B in col2 and col3, respectively:
# load data
data(beginStart)
# perform DCA (no settings necessary, more as required):
beginStart.dca1 <- collex.dist(beginStart)
beginStart.dca2 <- collex.dist(beginStart, am = "fye")
beginStart.dca3 <- collex.dist(beginStart, am = "fye", str.dir = TRUE)

# inspect:
head(beginStart.dca1, 15)  # 15 most strongly attracted items to cxn A
tail(beginStart.dca1, 25)  # 20 most strongly attracted items to cxn B

# cleanup (remove objects from workspace):
rm(beginStart.dca1, beginStart.dca2, beginStart.dca3)

### Case 2: Two separate aggregated frequency lists
## Like Case 1, but with separate lists of cxns A and B that need to combined:
# load data
```

```
data(beginToV)
data(startToV)

# I. Merge the lists
beginStart.in <- join.freqs(beginToV, startToV)
head(beginStart.in, 12)

# II. Calculate association
beginStart.out <- collex.dist(beginStart.in)

# III. Inspect
head(beginStart.out, 15)   # 15 most strongly attracted items to cxn A
tail(beginStart.out, 20)   # 20 most strongly attracted items to cxn B

# cleanup (remove objects from workspace):
rm(beginToV, startToV, beginStart.in, beginStart.out)

### Case 3: A list with one observation per line (i.e. raw = TRUE)
# where the cxns are in col1 and the collexemes are in col2:
# load & inspect the will/going-to-V alternation:
data(future)
head(future, 12)

# Calculate:
future.out <- collex.dist(future, raw = TRUE)
head(future.out, 6)
tail(future.out, 6)

# cleanup (remove objects from workspace):
rm(future, future.out)

##### IF YOU HAVE INCOMPLETE DATA SETS
## Illustrate the application of the cxn.freq argument if you do not have all
## types; this is *not* a sample of a larger data set, but rather as if lines
## from an aggregate frequency lists were unavailable. To illustrate, we'll
## recreate the dative alternation from Gries & Stefanowitsch (2004).
data(ditrdat_pub)

# The data is from Gries & Stefanowitsch (2004: 106), ie. the top collexemes for
# the ditransitive vs. the to-dative. That is, the low-frequent items are not
# in their results list in the table.
# So the following would lead to (linguistically) wrong results, because
# collex.dist() calculates the cxn frequencies from an incomplete data set,
# when in fact they are much higher:

collex.dist(ditrdat_pub, am = "fye")

# However, you can recreate the results table by specifying the cxn frequencies
# provided in the publication (as cxn.freq), as a vector, where the first
# element contains the total of the first cxn and the second contains the total
# of the second cxn. You can also get the traditional Fisher-Yates p-value as
# in the original publication:
```

```
ditrdat.dca <- collex.dist(ditrdat_pub, am = "fye", p.fye = TRUE,
                           cxn.freqs = c(1035, 1919), decimals = 3)
# Inspect:
head(ditrdat.dca, 20)  # left side of Table 2 (Gries & Stefanowitsch 2004: 106)
tail(ditrdat.dca, 19)  # right side of Table 2 (Gries & Stefanowitsch 2004: 106)
# the right side of Table 2 is "upside down", because collex.dist() orders by
# collostructional continuum. Run the above again with reverse = T.


# NB: If you have a raw input file, make sure you pass a vector with the correct
# frequencies for whatever R recognizes as the first element (usually
# alphabetically if column 1 is a factor); this behavior has to be checked
# carefully in R4.x, as R now standardly reads in characters as characters.

##### IN USE OVER LISTS, e.g.,
## We performed several Distictive Collexeme Analyses for (present) progressive
## vs. (simple) present over 10 25-yr periods in CLMET (Johannsen & Flach 2015).
## Note that although using historical data, this is quite different to
## Diachronic Distinctive Collexeme Analysis (e.g., Hilpert 2011), where periods
## are conditions in *one* DCA and thus mathematically *not* independent of
## each other. The sample data below runs one DCA *per period*, so the DCAs are
## mathematically independent of each other. The conditions are still
## two alternating constructions as in 'ordinary' DCA.

## So this means 'multiple' DCAs in the sense of 'several' DCAs, not in the
## sense of 'Multiple Distinctive Collexeme Analysis' (MDCA).

## load data
data(CLMETprog.qc)
data(CLMETsimple.qc)

### I. Prepare

# split data by time period and drop redudant period column
prog <- split(CLMETprog.qc[, c(1,3)], CLMETprog.qc$QUARTCENT)
simple <- split(CLMETsimple.qc[, c(1,3)], CLMETsimple.qc$QUARTCENT)

# combine frequencies for progressive & simple as input to collex.dist()
dist.in <- join.lists(prog, simple)
dist.in <- lapply(dist.in, droplevels)

### II. Perform several DCA (returns a list of DCA output for list)
# if only defaults of arguments are used, use lapply() like so:
dist.out <- lapply(dist.in, collex.dist)

# if you want to override default arguments, use Map() like so:
dist.out <- Map(collex.dist, dist.in, am = "fye")
dist.out <- Map(collex.dist, dist.in, decimals = 7)

### III. Inspect output
str(dist.out)     # structure of list
str(dist.out[1])  # structure of first item in list
```

```
## VI. Export (works if you have installed package 'openxlsx')
# Will write each DCA in a separate Excel worksheet
openxlsx::write.xlsx(dist.out, "ProgSimpleDistColl_CLMET.xlsx")

## End(Not run)
```

`ditrdat_pub` *Data set ditransitive-dative (subset)*

Description

A subset of the ditransitive-dative alternation in ICE-GB, taken from Gries & Stefanowitsch (2004: 106).

Usage

```
data("ditrdat_pub")
```

Format

A data frame with 39 observations on the following 3 variables.

`VERB` a factor variable with the top attracted collexemes to the ditransitive and to-dative, respectively

`DITR` a numeric variable, containing the frequency of `VERB` in the ditransitive

`DAT` a numeric variable, containing the frequency of `VERB` in the to-dative

Details

Data to illustrate the use of `collex.dist()` with incomplete data sets. For full datasets, `collex.dist()` determines the construction totals from the files directly, but this is unavailable if you have incomplete data sets (i.e., with types and their frequencies missing. For example, in their publication, Gries & Stefanowitsch list only data that amounts to 957 (ditransitive) and 813 (to-dative) data points, while the construction totals are 1,035 and 1,919, respectively. In cases of incomplete data, but where construction totals are known, the construction totals need to be passed to the function, see below for an example.

Source

Recreated from Table 2 in Gries & Stefanowitsch (2004: 106).

References

Gries, Stefan Th. & Anatol Stefanowitsch. 2004. Extending collostructional analysis: A corpus-based perspective on "alternations." *International Journal of Corpus Linguistics* 9(1). 97–129.

Examples

```
## Not run:

## 1 Inspect the data: a data frame with 3 columns and the aggregated
## frequencies of verbs in the ditransitive (DITR) and to-dative (DAT).
head(ditrdat_pub)

## 2 Recreate the results in Gries & Stefanowitsch (2004: 106), with
#    the construction frequencies given as the cxn.freqs argument:

# with standards in collex.dist(), i.e., log-likelihood (G2):
collex.dist(ditrdat_pub, cxn.freqs = c(1035, 1919), decimals = 3)

# with p.fye = TRUE, to recreate the traditional p-value (Fisher-Yates)
# (note that p.fye = TRUE will only make sense if am = "fye"):
collex.dist(ditrdat_pub, am = "fye", p.fye = TRUE, cxn.freqs = c(1035, 1919))

## See ?collex.dist() for further examples.

## End(Not run)
```

---

freq.list *Function to create frequency list from a character vector*

Description

Sometimes it is handy to create a frequency list if working with some annotated data rather than importing frequency lists from external programs. Many R functions can create frequency lists, this is just a handy alternative that creates lists that are directly compatible the `join.freqs` function. The output is a data frame and is sorted in descending order of frequency, which may be one or two steps quicker than R functions such as `table()`.

Usage

```
freq.list(x, convert = TRUE, asFactor = TRUE)
```

Arguments

`x`	A factor or character vector.
`convert`	Should the elements in `x` be converted to lowercase? Default is `convert = TRUE`, meaning that case will be ignored and aggregated.
`asFactor`	Should the types be converted to `as.factor()`? Default is `asFactor = TRUE`, because R>=4x has `stringsAsFactors = FALSE` as a default, and several other functions in `collostructions` my otherwise not work properly.

Value

`WORD`	The types
`FREQ`	The frequencies.

See Also

Can be used in preparation for `join.freqs`.

Examples

```
## Not run:

## From a list of raw observations:
head(caus.into, 12)
# so to get a frequency list of the V1 in the into-causative:
freq.list(caus.into$V1)

# or from the future-time expressions:
head(future, 12)
freq.list(future$COLLEXEME)


## End(Not run)
```

---

future *Data set (*will/going-to-*VERB)*

Description

Data set (sample) of 10,000 random tokens of the *will-* vs. *going to*-VERB alternation in the spoken section of the BNC2014, for illustration in `collex.dist()` with raw input.

Usage

```
data("future")
```

Format

A data frame with 10000 observations on the following 2 variables.

`CXN` a factor with levels `going.to` and `will`

`COLLEXEME` a factor with 599 verb types in either the *going to V* or *will* cxn.

Examples

```
## Not run:
## Distinctive Collexeme Analysis

# load data
data("future")

# perform Distinctive Co-Varying Collexeme Analysis (with defaults)
# see ?collex.dist() for more use cases:
x <- collex.dist(future, raw = TRUE)
```

```
# If you do not set raw = TRUE, function aborts:
x <- collex.dist(future)


## End(Not run)
```

---

goVerb *Data set (go-VERB)*

Description

Data set from Flach (2015) containing the construction and corpus frequencies of all verbs that occur as V2 in the *go*-VERB construction in the ENCOW14AX01 corpus (Schäfer & Bildhauer 2012). See Flach (2015:§4.2) for data extraction procedure.

Usage

```
data("goVerb")
```

Format

A data frame with 752 observations on the following 3 variables.

`WORD` A factor with the levels for each of the 725 verbs that occurs in the construction

`CXN.FREQ` A numeric vector, containing the observed frequency of V2 in *go*-VERB

`CORP.FREQ` A numeric vector, containing the string frequency of V2 in the corpus

References

Flach, Susanne. 2015. Let's go look at usage: A constructional approach to formal constraints on go-VERB. In Thomas Herbst & Peter Uhrig (eds.), *Yearbook of the German Cognitive Linguistics Association (Volume 3)*, 231-252. Berlin: De Gruyter Mouton. doi:10.1515/gcla-2015-0013.

Schäfer, Roland & Felix Bildhauer. 2012. Building large corpora from the web using a new efficient tool chain. In Nicoletta Calzolari, Khalid Choukri, Thierry Declerck, Mehmet Uğur Doğan, Bente Maegaard, Joseph Mariani, Jan Odijk & Stelios Piperidis (eds.), *Proceedings of the Eighth International Conference on Language Resources and Evaluation* (LREC'12), 486–493. Istanbul: ELRA. Available at http://webcorpora.org

Examples

```
## Not run:

data(goVerb)     # load
str(goVerb)      # inspect structure of object
head(goVerb)     # view head of object

collex(goVerb, 616113708) # used in Flach (2015), all tokens minus punctuation
collex(goVerb, 93993713)  # could/should probably be used, all verb tokens
collex(goVerb, 93993713, "chisq")  # returning chisquare statistic
## End(Not run)
```

input.check *Function to help detect issues in input data frames*

Description

Input data from corpora can be messy. This function checks if your input to `collex()`, `collex.dist` and `collex.covar()` has one of three frequent, potential issues: (a) does it contain leading and trailing whitespace, (b) does it contain types that are essentially identical, but not case sensitive (e.g., *fine, Fine*, and/or (c) does it contain missing values?

Beware, though, that the input types to all functions vary a lot and errors can be plentiful, so this function is (still) somewhat experimental and may show issues where they may not be any (and, sadly, vice versa). Function-specific errors are usually also reflected in error messages, but do let me know if you find bugs or have suggestions for more improved checks and error messages.

Usage

```
input.check(x, raw = FALSE)
```

Arguments

`x`	The data frame that you would pass as `x` to `collex()`, `collex.dist()` and `collex.covar()`.
`raw`	Does `x` contain aggregated data, i.e., a frequency list (`raw = FALSE`) or one observation per line (`raw = TRUE`)? Note that the function tries to guess: if `x` has exactly two colums with text in it, it will assume that it is a raw data file like you use in `collex.dist` and `collex.covar` with `raw = TRUE`, and will abort, unless you set `raw = TRUE`.

Details

DISCLAIMER: this function is quite experimental, so there is of course no guarantee that your data frame is free of technical or conceptual error if the function output does not report any issues. (In some cases, for example, case-sensitivity might even be wanted.) But function should detect the most common problems that arise from messy corpus data or data processing that I have come across in my own work, teaching, and workshops.

Note also that the function does not check for problems specific to a particular collex-function: for example, the check whether corpus frequencies are lower than construction frequencies for simple collexeme analysis is run during the execution of `collex()`. If you have suggestions of how to improve the checker function, let me know.

I'd recommend any errors are fixed in the source files, e.g., in Excel or csv files before import into R.

Value

Returns an attempt at diagnostics for leading/trailing whitespace, duplicate types (e.g., due to capitalisation inconsistency) and/or empty/missing frequencies and/or types.

Author(s)

Report bugs and suggestions to:
Susanne Flach, susanne.flach@es.uzh.ch

Examples

```
## Not run:
## dirty:
df <- data.frame(WORD  = c("Hello", "hello", "hello ", " hello", "Hi ", "  hi", "HI", "", "G'day"),
                 CXN = c(23, NA, 2, 1, 30, 59, NA, 10, 3),
                 CRP = c(569, 3049, 930, 394, 2930, 87, 9, 23, 40))
df
input.check(df)

## a little dirty:
df <- data.frame(WORD  = c("Hello", "Hi", "hi", "HI", "", "G'day"),
                 CXN = c(23, 12, 2, 1, 30, 59),
                 CRP = c(569, 3049, 930, 394, 2930, 28))
df
input.check(df)

## clean:
df <- data.frame(WORD = c("Hello", "Hi", "Goodmorning", "Goodafternoon"),
                 CXN = c(234, 139, 86, 74),
                 CRP = c(23402, 2892, 893, 20923))
input.check(df)


## End(Not run)
```

`join.freqs` *Function to merge two frequency lists*

Description

Function to merge two data frames of frequency lists into a combined data frame of frequencies.

Usage

```
join.freqs(x, y, all = TRUE, threshold = 1)
```

Arguments

x	A data frame with frequencies for condition A, with WORD in first colum and frequencies in the second. This could be (i) the frequency list of types in the construction under investigation (for collex()) or the frequency list of types in construction A when comparing two constructions (for collex.dist()).
y	A data frame with (i) corpus frequencies of an item in a construction (collex()) or the frequecies of types in the second construction (collex.dist()).
all	logical. If TRUE (the default), then all types from both x and y are merged, returning 0 in the condition for which a type is not attested. This is recommended especially for collex.dist(), but is useful for negative evidence calculation for items in collex(). If FALSE, the output will contain only the types in x with frequencies in both x and y, but types that only occur in y will be ignored. The latter is recommended only for joining frequency lists for use in collex().
threshold	Numerical. How many times must an item occur overall to be included in the joint list? Default is 1, which means all items are included. If all = FALSE, setting threshold = number will mean that items in x are only included if they occur at least number times.

Details

Output suitable for collex() and collex.dist(). The column names of the output data frame (cols 2 and 3) will be identical to the names of the objects of x and y. Header for column one will be WORD.

The difference to join.lists() is that join.freqs() two frequency lists are joined, whereas join.lists() joins two *lists* (i.e., 'list', the R object type) of frequency lists, which may contain, e.g., freuency lists from different periods.

Note

The behaviour of join.freqs() deviates from that of merge(): If you merge construction frequencies with a list of corpus frequencies with all = FALSE, and a construction item does not occur in the corpus frequencies list, the item will occur in the output with a corpus frequency of 0. While this will throw an error if you use such a list in collex(), it will allow you to identify faulty data rather than silently dropping it.

Also, you can merge a frequency list of two columns with a third frequency list, but then you will have to manually adjust the column headers afterwards.

Examples

```
## Not run:

#### Example for Simple Collexeme Analysis
# Using the verb lemma frequencies from the BNC
# begin to rain, begin to think, begin to blossom...
data(beginToV)
```

```
## I. Prepare
# merge by lemmas, only types that occur in the construction:
begin1.in <- join.freqs(beginToV, BNCverbL, all = FALSE)
# merge by lemmas, all types, including unattested for 'negative evidence'
begin2.in <- join.freqs(beginToV, BNCverbL)

## II. Perform SCA
# second argument is taken directly from the source data
begin1.out <- collex(begin1.in, sum(BNCverbL$CORP.FREQ))
begin2.out <- collex(begin2.in, sum(BNCverbL$CORP.FREQ))

### Example for Distinctive Collexeme Analysis
# Comparing begin to rain, start to go,...
data(beginToV)  # load set 1
data(startToV)  # load set 2

## I. Prepare
beginStart <- join.freqs(beginToV, startToV) # merge lists (all types)

## II Perform
beginStart.out <- collex.dist(beginStart)


## End(Not run)
```

`join.lists` *Function to merge two lists of frequency lists*

Description

Merges two lists of data frames pair-wise. Both lists need to be of equal length (i.e. identical number of data.frames), and where all data.frames have two columns (for `WORD` and `FREQ`); merging is by items in x[,1]. All pair-wise data frames must have an identical ID column name to match by (using `WORD` as a column name is recommended). Returns a list of the same length with data frames of length 3, named `WORD` (or name of ID column), `name.of.df1` and `name.of.df2`; the latter two contain the frequencies of `WORD`. Suitable to handle the output of `split()`, e.g. if a data frame was split by time period.

Usage

```
join.lists(x, y, all = TRUE, threshold = 1)
```

Arguments

`x`	List 1 containing data frames of frequencies for construction A.
`y`	List 2 containing data frames of (i) corpus frequiences for `WORD` in construction A or (ii) frequencies of `WORD` in construction B.

all	If all = TRUE (the default), each data frame contains all joint types of both columns (esp. for collex.dist()). Set to all = FALSE, if only the types in the first condition (of the first data frame list) should be included (this is often the case for collex()).
threshold	Numerical. How many times must an item occur overall to be included in the joint list? Default is 1, which means all items are included (per list split, e.g., time period). If all = FALSE, setting threshold = number will mean that items in x are only included if they occur at least number times.

Examples

```
## Not run:

### We performed a series of Distictive Collexeme Analyses
### for (present) progressive vs. (simple) present over ten
### 25-year periods in CLMET (Johannsen & Flach 2015).
### Note that although working with historical data, this is something
### very different to Diachronic Distinctive Collexeme Analysis
### (e.g., Hilpert 2011), where periods are conditions in *one* DCA
### and thus mathematically not independent of each other.
### The sample data below runs one DCA per period, which are
### mathematically independent of each other. The conditions are still
### two alternating constructions as in 'ordinary' DCA.

### Also note that this means 'multiple' DCAs in the sense of 'several' DCAs,
### not in the sense of 'Multiple Distinctive Collexeme Analysis' (MDCA).

# Load data
data(CLMETprog.qc)
data(CLMETsimple.qc)
head(CLMETprog.qc, 10)
head(CLMETsimple.qc, 10)

### I. Prepare
## Make frequency lists by decade of class list, i.e.,
## split constructions by period,
## keep ITEM & FREQ only, droplevels
prog <- split(CLMETprog.qc[, c(1,3)], CLMETprog.qc$QUARTCENT)
prog <- lapply(prog, droplevels)
simp <- split(CLMETsimple.qc[, c(1,3)], CLMETsimple.qc$QUARTCENT)
simp <- lapply(simp, droplevels(x)

dist.in <- join.lists(prog, simp)
dist.in <- lapply(dist.in, droplevels)
# Cosmetics:
dist.in <- lapply(dist.in, setNames, c("WORD", "progressive", "simple"))

#### CALCULATE COLLEXEMES
dist.out.log <- lapply(prog.collexDist.in, function(x) collex.dist(x))
dist.out.fye <- Map(collex.dist, dist.in, am="fye")

### EXPORT
```

```
## Note: for this strategy, you need to install and load library(openxlsx)
write.xlsx(dist.out.log, "progCollexDistLL.xlsx")
write.xlsx(dist.out.log, "progCollexDistFYE.xlsx")

## End(Not run)
```

---

modadv *Data set (modal-adverb combinations)*

Description

Data set of 792 modal-adverb pairs in the BNC-BABY, such as *would possibly*, *may well* or *can hardly*.

Usage

```
data("modadv")
```

Format

A data frame with 792 observations on the following 3 variables.

`MODAL` A factor with 11 levels of the core modals and contractions, i.e., `\'d`, `\'ll`, `can`, `could`, `may`, `might`, `must`, `shall`, `should` `will`, `would`

`ADVERB` A factor with 280 levels, for each adverb types following modal verbs, e.g., `certainly`, `essentially`, `even`, `lawfully`, `publicly`, `quickly`, and `well`

`FREQ` The frequency of the combination.

Source

BNC-BABY; [hw="will|would|can|could|may|might|must|shall|should" & pos="VM0"] [pos="AV0"]; cf. Flach (2020) with COCA data.

References

Flach, Susanne. 2020. Beyond modal idioms and modal harmony: A corpus-based analysis of gradient idiomaticity in MOD+ADV collocations. *English Language and Linguistics*. aop.

Examples

```
## Not run:
data(modadv)

## Inspect:
# This is an aggregated frequency list:
head(modadv, 12)

### Perform co-varying collexeme analysis
```

```
## ?collex.covar()
# since it's aggregated, you must set raw = FALSE, or it will make no sense.
cca.att <- collex.covar(modadv, am="fye", raw = FALSE, all = FALSE) # only attested combinations
cca.all <- collex.covar(modadv, am="fye", raw = FALSE, all = TRUE) # all combinations


## Reshape the cca output by association measure:
# ?reshape.cca
cca.wide.att <- reshape.cca(cca.att)
cca.wide.all <- reshape.cca(cca.all)


## End(Not run)
```

`reshape.cca` *Function to transform collex.covar() long to wide format*

Description

The output of `collex.covar` is a so-called 'long format', where each row represents a slot1~slot2 pair (or, more generally, a cond1~cond2 pair). This function cross-tabulates the association measures in a 'wide format', where one condition occurs as rows and the other represents the columns and the cells contain the row~col association measure.

Usage

```
reshape.cca(x, cond = "shorter", str.dir = TRUE, value = "COLL.STR", abs.dev = FALSE,
            max.assoc = FALSE, sorton = "abs.dev", decimals = 5)
```

Arguments

`x`	A data frame containing the output of `collex.covar`.
`cond`	Which of the two conditions in `collex.covar` should be the columns of the reshaped format? The default `shorter` uses that condition which has the fewer number of types. But you can override this by setting with `cond = 1` for the first condition (usually slot 1) or `cond = 2` for the second condition (usually slot 2).
`str.dir`	Should the values in the cells indicate the direction of association? `TRUE` is the default and should be used if the association measure you used in `collex.covar` is one that contains only positive values (e.g., `logl`, `fye`, or `fye.ln`). For other association measures, set `str.dir = FALSE`, if the values range from - to +.
`value`	Which value should be cross-tabulated, i.e., put in the cells? For cond1~cond2 pairs that do not occur in `x`, `NA` are returned (see details below). The default cross-tabulates the association measure, but it can also be `'OBS'` for a co-occurrence matrix. Note, since for it does not make sense to calculate a directed value for `'OBS'`, the function will abort if you leave the default for `str.dir = TRUE` unchanged when using `'OBS'`.

abs.dev	The function sums all absolute association measures row-wise Should this be included in the output as an extra column? The default is FALSE. Note that if your x only contains attested values, the summed deviations ignore NA values.
max.assoc	Should the col-condition of maximum assocation per row-condition be included in the output? If TRUE, this column will contain the col-type with which the row-type has the highest positive or negative association (i.e., 'most deviant' relationship).
sorton	By default the output is sorted in descending order of abs.dev. You can also sort alphabetically (sorton = "alphabetical").
decimals	Rounding of cell values. If this is set to a higher value than what was used for this argument in collex.covar, it will of course be ignored.

Details

The function makes most sense for a collex.covar that was run for all possible combinations. If association scores were only calculated for attested combinations, the output of reshape.cca contains NA in the cells of unattested combinations and it is up to the user to decide what to do with it. Both abs.dev and max.assoc can still be calculated and displayed, but they are based on observed combinations only. Since association measures for unobserved combinations can be read as 'negative evidence', the abs.dev will be and the max.assoc type may be different, depending on the strength of (potential) 'negative association'. See examples below for the case of unattested *.

Value

Returns cross-tabulated association scores or observed values.

Author(s)

Susanne Flach, susanne.flach@es.uzh.ch

See Also

collex.covar

Examples

```
## Not run:
data(modadv)

## Inspect:
# This is an aggregated frequency list:
head(modadv, 12)

### Perform co-varying collexeme analysis
## ?collex.covar()
# since it's aggregated, you must set raw = FALSE, or it will make no sense.
cca.att <- collex.covar(modadv, am="fye", raw = FALSE, all = FALSE) # only attested combinations
cca.all <- collex.covar(modadv, am="fye", raw = FALSE, all = TRUE) # all combinations
```

```
## Reshape the cca output by association measure:
# ?reshape.cca
cca.wide.att <- reshape.cca(cca.att)
View(cca.wide.att)
cca.wide.all <- reshape.cca(cca.all)
View(cca.wide.all)

#### Co-occurrence of observations
modadv.obs <- reshape.cca(cca.att, value = "OBS", str.dir = FALSE) # you must set false in this case
# since we ran this on only the attested values, you can replace NA with 0:
modadv.obs[is.na(modadv.obs)] <- 0
View(modadv.obs)


## End(Not run)
```

`startToV` *Data set (start-to-VERB)*

Description

Data set of the *start-to*-VERB construction in the British National Corpus (BNC), with the frequencies of the verbs in the open slot ([hw="start" & class="VERB"] [hw="to"] [pos="V.I"]).

Usage

```
data("startToV")
```

Format

A data frame with 1168 observations on the following 2 variables.

`WORD` a factor with levels of types start-to-V

`CXN.FREQ` a numeric vector of the frequencies in V2.

Examples

```
## Not run:

data(startToV)     # load
str(startToV)      # inspect structure of object
head(startToV)     # view head of object

## End(Not run)
```

Index