Package 'CorporaCoCo'

August 8, 2022

Encoding UTF-8

Type Package

Title Corpora Co-Occurrence Comparison

Version 2.0

Date 2022-08-05

Author Anthony Hennessey [aut],
Viola Wiegand [aut],
Christopher R. Tench [aut],
Jamie Lentin [aut],
Mike Allaway [ctb],
Andrew Edmondson [ctb],
Matthew Henderson [ctb],
Michaela Mahlberg [aut, cre]

Maintainer Michaela Mahlberg <mahlberg.lab@gmail.com>

Description Identifies significant difference in co-occurrence counts for a given node or set of nodes across two corpora, using a Fisher's Exact test.

URL https://github.com/mahlberg-lab/CorporaCoCo/

License GPL (>= 3)

Depends R (>= 3.2.0),

Imports methods, stats, data.table (>= 1.11.2), stringi, RColorBrewer, rlist

Suggests unittest, R.rsp, knitr, rmarkdown, lintr, testthat (>= 3.0.0), markdown

VignetteBuilder knitr

BugReports https://github.com/mahlberg-lab/CorporaCoCo/issues

Config/testthat/edition 3

NeedsCompilation no

Repository CRAN

Date/Publication 2022-08-08 10:30:10 UTC

R topics documented:

`CorporaCoCo-package` *Comparing Co-occurrence between corpora.*

Description

The package implements the method introduced in Wiegand and Hennessey et al. (2017a). It identifies significant difference in co-occurrence counts for a given node or set of nodes across two corpora, using a Fisher's Exact test.

Details

A good place to start is the 'Introduction to CorporaCoCo' vignette. You can open the vignette with `vignette("intro", package = "CorporaCoCo")`.

For a list of all documentation use `library(help="CorporaCoCo")`. For updates on development versions of the package and documentation, please see the GitHub page.

Author(s)

Maintainer: Mahlberg - Lab <mahlberg.lab@gmail.com>.

References

* Wiegand, V., Hennessey, A., Tench, C. R., & Mahlberg, M. (2017a, May 24). *Comparing co-occurrences between corpora*. 38th ICAME conference, Charles University, Prague.

* Wiegand, V., Hennessey, A., Tench, C. R., & Mahlberg, M. (2017b, July 24). *A cookbook of co-occurrence comparison techniques and how they relate to the subtleties in your research question*. 9th International Corpus Linguistics Conference, University of Birmingham, Birmingham.

Examples of how the method has been used can be found in:

* Mahlberg, M., Wiegand, V., & Hennessey, A. (2020). Eye language – body part collocations and textual contexts in the nineteenth-century novel. In L. Fesenmeier & I. Novakova (Eds.), *Phraseology and Stylistics of Literary Language/Phraséologie et Stylistique de la Langue Littéraire* (pp. 143–176). Peter Lang. https://www.academia.edu/45152494/Eye_language_body_part_collocations_and_textual_contexts_in_the_nineteenth_century_novel

* Wiegand, V. (2019). *A Corpus Linguistic Approach to Meaning-Making Patterns in Surveillance Discourse* [PhD, University of Birmingham]. https://etheses.bham.ac.uk/id/eprint/9778

corp_coco *Co-occurrence comparison*

Description

Calculates statistically significant difference in co-occurrence counts.

Usage

```
corp_coco(A, B, nodes, collocates = NULL, fdr = 0.01)

# Deprecated
coco(A, B, nodes, fdr = 0.01, collocates = NULL)
```

Arguments

A	A corp_cooccurrence object. For the deprecated coco function this is a data.frame of co-occurrence counts as returned by corp_get_counts.
B	A corp_cooccurrence object. For the deprecated coco function this is a data.frame of co-occurrence counts as returned by corp_get_counts.
nodes	A character vector of node types or character string representing a single node type.
collocates	A character vector of collocates types or character string representing a single collocate type. The *collocates* essentially act as a filter on the *y* column of the returned data structure. *collocates* should be used to target the testing; reducing the number of tests will reduce the loss of power from the multiple test correction.
fdr	The desired level at which to control the False Discovery Rate. Default value is 0.01.

Details

The corp_coco function implements the method introduced in Wiegand and Hennessey et al. (2017a) (described in more detail from a linguistic perspective in Wiegand, 2019).

fdr indicates the level at which the False Discovery Rate will be controlled because the method carries out a large number of tests. For a description of the form of FDR used see Benjamini and Hochberg (1995). For description of the *p_adjusted* column in the returned structure see p.adjust.

The returned data structure is a data.table. A data.table is also a data.frame and will behave exactly as such if the data.table library is not loaded.

The returned data.table contains details of all the co-occurrences for which there is evidence of a difference in co-occurrence between the two supplied data sets. The effect size is calculated as the log base 2 of the odds ratio. The effects size and its confidence interval are captured in the *effect_size*, *CI_lower* and *CI_upper* columns. The *p_value* column contains the non-adjusted p-value from the Fisher's Exact Test.

Value

A data.table of the form

```
Classes ‘data.table’ and 'data.frame': 11 variables:
 $ x           : chr
 $ y           : chr
 $ H_A         : int
 $ M_A         : int
 $ H_B         : int
 $ M_B         : int
 $ effect_size : num
 $ CI_lower    : num
 $ CI_upper    : num
 $ p_value     : num
 $ p_adjusted  : num
 - attr(*, "sorted")= chr  "x" "y"
 - attr(*, ".internal.selfref")=<externalptr>
 - attr(*, "coco_metadata")=List of 5
  ..$ nodes      : chr
  ..$ collocates : chr
  ..$ fdr        : num
  ..$ PACKAGE_VERSION:Classes 'package_version', 'numeric_version'
  .. ..$ : int
  ..$ date  : Date, format: "2016-11-01"
```

References

Y. Benjamini and Y. Hochberg (1995) *Controlling the False Discovery Rate: A Practical and Powerful Approach to Multiple Testing.* Journal of the Royal Statistical Society. Series B (Methodological) **57 (1)289–300**.

* Wiegand, V., Hennessey, A., Tench, C. R., & Mahlberg, M. (2017a, May 24). *Comparing co-occurrences between corpora*. 38th ICAME conference, Charles University, Prague. * Wiegand, V. (2019). *A Corpus Linguistic Approach to Meaning-Making Patterns in Surveillance Discourse* [PhD, University of Birmingham]. https://etheses.bham.ac.uk/id/eprint/9778

---

corp_concordance *Concordance*

Description

Concordances from corp_text and corp_cooccurrence objects.

Usage

```
corp_concordance(obj, span, nodes, collocates, context)
```

```
  ## S3 method for class 'corp_concordance'
print(x, collocates = attr(x, "collocates"),
                 collocate_marker = "*", as_data_table = FALSE, ...)
```

Arguments

obj	A corp_text or a corp_cooccurrence object.
span	A character string defining the co-occurrence span. See details section in corp_surface. For corp_cooccurrence objects this defaults to the *span* used to create the object.
nodes	A character vector of node types or character string representing a single node type. If supplied, only concordance lines for the specified node types will be displayed. For corp_cooccurrence objects this defaults to the *nodes* used to create the object.
collocates	A character vector of collocate types or character string representing a single collocate type. If supplied, only concordance lines for the specified collocate types will be selected (i.e. act as a filter for corp_concordance). For corp_cooccurrence objects this defaults to the *collocates* used to create the object. For print *collocates* is used to determine which collocate types to highlight; and does not act as a filter. See the "intro" vignette for an example: vignette("intro", package = "CorporaCoCo"). Defaults to the collocates passed to the corp_concordance function.
context	The number of context tokens to be displayed. Default is 3.
x	A corp_concordance object.
collocate_marker	The characters used to highlight the highlighted collocates. Default is '*'.
as_data_table	Print the internal data.table object.
...	Ignored.

Value

Returns a corp_concordance object.

---

corp_cooccurrence *Calculate Co-occurrence Counts*

Description

Calculates co-occurrence counts. For each co-occurrence the maximum possible number of co-occurrences is also calculated.

Usage

```
corp_surface(text, span, nodes = NULL, collocates = NULL)

is.corp_cooccurrence(obj)
is.corp_surface(obj)

# deprecated
surface(x, span, nodes = NULL, collocates = NULL)
```

Arguments

text	A corp_text object.
span	A character string defining the co-occurrence span. See Details.
nodes	A character vector of node types or character string representing a single node type. If supplied, only co-occurrences for the specified node types will be calculated. If *nodes* is not supplied co-occurrences will be calculated for the set of all node types. Restricting *nodes* can significantly reduce memory usage and execution times.
collocates	A character vector of collocate types or character string representing a single collocate type. If supplied, only co-occurrences for the specified collocate types will be calculated. If *collocates* is not supplied, co-occurrences will be calculated for all collocate types with a non-zero co-occurrence count. Restricting *collocates* can significantly reduce execution times.
obj	A corp_cooccurrence object as is returned by the corp_surface function.
x	In the deprecated surface function *x* is a vector of tokens. *x* is assumed to be an ordered vector of tokenized text. No processing will be applied to *x* prior to the co-occurrence count calculations.

Details

Surface co-occurrence: 'surface' co-occurrence is easiest to describe with an example. The following is a span of '2LR', that is 2 to the left and 2 to the right.

```
("a", "man", "a", "plan", "a", "cat", "a", "canal", "panama")
      |__________|____|__________|
```

In this example the node "plan" would co-occur once each with the collocates "man" and "cat", and twice with the collocate "a".

Other examples of span:

span = '1L2R'

```
("a", "man", "a", "plan", "a", "cat", "a", "canal", "panama")
             |____|____|__________|
```

span = '2R'

```
("a", "man", "a", "plan", "a", "cat", "a", "canal", "panama")
                   |____|__________|
```

For a detailed description of 'surface' co-occurrence see Evert (2008).

Co-occurrence barriers:

NAs can be used to implement co-occurrence barriers eg if two NA characters are inserted into *x* at each sentence boundary then with span = 2 co-occurrences will not happen across sentences. See Evert (2008) for detailed description of co-occurrence barriers.

Value

corp_surface: Returns a corp_surface object.

The corp_surface object can be interrogated using the corp_get_* accessor functions.

The corp_surface objects are used as arguments to the corp_coco) function.

References

S. Evert (2008) *Corpora and collocations.* Corpus Linguistics: An International Handbook **1212–1248**.

See Also

corp_coco) and corp_concordance).

Examples

```
# =====================
# surface co-occurrence
# =====================

x <- corp_text("A man, a plan, a canal -- Panama!")

y <- corp_surface(x, span = "2R")
corp_get_counts(y)

##         x      y H M
##  1:     a      a 2 4
##  2:     a  canal 1 5
##  3:     a    man 1 5
##  4:     a panama 1 5
##  5:     a   plan 1 5
##  6: canal panama 1 0
##  7:   man      a 1 1
##  8:   man   plan 1 1
##  9:  plan      a 1 1
## 10:  plan  canal 1 1

# filter on nodes
y <- corp_surface(x, span = '2R', nodes = c("canal", "man", "plan"))
corp_get_counts(y)

##         x      y H M
##  1: canal panama 1 0
##  2:   man      a 1 1
##  3:   man   plan 1 1
##  4:  plan      a 1 1
```

```
##  5:  plan  canal 1 1

# filter on nodes and collocates
y <- corp_surface(x, span = '2R', nodes = c("canal", "man", "plan"),
                  collocates = c("panama", "a"))
corp_get_counts(y)

##         x      y H M
##  1: canal panama 1 0
##  2:   man      a 1 1
##  3:  plan      a 1 1

# co-occurrence barrier
tokens_with_barrier <- data.frame(
     type =            c("a", "man", "a", "plan", NA, NA, "a", "canal", "panama"),
    start = as.integer(c( 1,   3,     8,   10,    NA, NA,  16,  18,      27)),
      end = as.integer(c( 1,   5,     8,   13,    NA, NA,  16,  22,      32)),
    stringsAsFactors = FALSE
)
x <- corp_text("A man, a plan, a canal -- Panama!", tokens = tokens_with_barrier)

y <- corp_surface(x, span = '2R')
corp_get_counts(y)

#         x      y H M
#  1:     a      a 1 4
#  2:     a  canal 1 4
#  3:     a    man 1 4
#  4:     a panama 1 4
#  5:     a   plan 1 4
#  6: canal panama 1 0
#  7:   man      a 1 1
#  8:   man   plan 1 1
```

---

corp_get_* *Accessors*

Description

Accessor methods for various corp_* objects.

Usage

```
  ## S3 method for class 'corp_text'
corp_get_text(obj)
  ## S3 method for class 'corp_cooccurrence'
corp_get_text(obj)

  ## S3 method for class 'corp_cooccurrence'
corp_get_text_obj(obj)
```

```
  ## S3 method for class 'corp_text'
corp_get_tokens(obj)
  ## S3 method for class 'corp_cooccurrence'
corp_get_tokens(obj)

  ## S3 method for class 'corp_text'
corp_get_metadata(obj)
  ## S3 method for class 'corp_cooccurrence'
corp_get_metadata(obj)
  ## S3 method for class 'corp_concordance'
corp_get_metadata(obj)
  ## S3 method for class 'corp_coco'
corp_get_metadata(obj)

  ## S3 method for class 'corp_cooccurrence'
corp_get_counts(obj)
```

Arguments

`obj`	A `corp_*` object.

Value

corp_get_text: Returns a `character string` of the text that the co-occurrence counts were calculated against. This comes from the `corp_text` object used to create the `corp_cooccurrence` object.

corp_get_tokens: Returns a `data.table` of the tokenization that the co-occurrence counts were calculated against. This comes from the `corp_text` object used to create the `corp_cooccurrence` object.

corp_get_counts: Returns a **data.table** containing the co-occurrence counts. Note that a `data.table` is also a `data.frame` so if the `data.table` library is not loaded the returned object will behave exactly as a `data.frame`; however, for large data sets there will be significant performance enhancement offered by exploiting **data.table** functionality.

The `data.table` is of the form:

```
Classes ‘data.table’ and 'data.frame': ...
 $ x: chr
 $ y: chr
 $ H: int
 $ M: int
 - attr(*, "sorted")= chr  "x" "y"
 - attr(*, ".internal.selfref")=<externalptr>
```

where `H` is the number of times `x` types co-occurs with `y` types (think *H*its), and `M` is the number of times `x` types fail to co-occur with `y` types when they could have (think *M*isses); hence `H + M` is the maximum number of times that `x` types can co-occur with `y` types.

corp_text *Tokenized text*

Description

Encapsulates the tokenization of a piece of text.

Usage

```
corp_text(text, tokens = NULL)

is.corp_text(obj)

corp_text_rbindlist(x)

## S3 method for class 'corp_text'
corp_type_lookup(obj)
```

Arguments

text	A character string of the text which is the subject of the co-occurrence counting.
tokens	This is a data.frame containing *type*, *start* and *end* variables.

```
$ tokens:Classes 'data.frame': 3 variables:
 ..$ type : chr
 ..$ start: int
 ..$ end  : int
```

tokens captures the types within the text along with their character positions. For example we could represent the types in the text "Do cats eat bats?" with the *tokens* data.frame:

```
   type start end
1:   do     1   2
2: cats     4   7
3:  eat     9  11
4: bats    13  16
```

If the *tokens* argument is not supplied, the *tokens* will be calculated from the supplied *text* argument. The default behaviour is to tokenize on word boundaries according to the Unicode Standard with the *types* being the unique set of lowercased extracted words. This is achieved using the **stringi** CRAN package and will work for any UTF-8 encoded text (in any language).

obj	A corp_text object as is returned by the corp_text function.
x	A list of corp_text objects.

Value

corp_text: Returns a `corp_text` object.

The `corp_text` object can be interrogated using the `corp_get_*` accessor functions.

A concordance can be generated from the `corp_text` object using the `corp_concordance` function.

The `corp_text` objects are used as arguments to the `corp_cooccurrence` function.

corp_type_lookup: Returns a **data.table** that can be used to lookup the tokens associated with each type. See example.

corp_text_rbindlist: Returns a `corp_text` object which is an ordered combination of the given list of `corp_text` objects.

TODO: Currently the text is concatenated with a single space.

summary: Prints a summary of the token and type counts for the *text*.

See Also

`corp_cooccurrence` and `corp_concordance`.

Examples

```
x <- "A man, a plan, a canal -- Panama!"

y <- corp_text(x)

corp_get_tokens(y)

##      type start end  token idx
## 1:      a     1   1      A   1
## 2:    man     3   5    man   2
## 3:      a     8   8      a   3
## 4:   plan    10  13   plan   4
## 5:      a    16  16      a   5
## 6:  canal    18  22  canal   6
## 7: panama    27  32 Panama   7

corp_get_text(y)

## [1] "A man, a plan, a canal -- Panama!"

corp_type_lookup(y)

##      type tokens
## 1:      a   A, a
## 2:  canal  canal
## 3:    man    man
## 4: panama Panama
## 5:   plan   plan
```

plot.corp_coco *plot.corp_coco*

Description

Plotting corp_coco objects.

Usage

```
  ## S3 method for class 'corp_coco'
plot(x, as_matrix = FALSE, nodes = NULL, forest_plot_args = NULL, ...)
```

Arguments

x	An corp_coco object.
as_matrix	If as_matrix is set to TRUE a matrix plot rather than a forest plot is produced.
nodes	If a vector of nodes is supplied this will be used to filter the set of results that are plotted. If nodes are supplied the plot will use the nodes order.
forest_plot_args	This is a list of arguments that is passed to the plot.default that produces the foundation of the forest plot. The list may contain a subset or all of the following documented arguments; any arguments that are not documented here will be ignored. A description of each argument can be found in the help for the plot.default function. Available arguments are • xlim Default: Calculated from the ranges of the confidence intervals. • xlab Default: 'Effect Size' • main Default: NULL • sub Default: NULL • asp Default: NA • pch Default: 15 • cex.pch Default: 1 • lwd.xaxt Default: 1 • col.xaxt Default: 'black' • col.whisker Default: 'black' • col.zero Default: 'darkgray' • length.wisker_end Default: 0.05
...	Other arguments will be ignored.

Details

An object of class corp_coco is returned by corp_coco().

Value

No return value, called to plot a corp_coco object

surface_coco *Deprecated – Surface co-occurrence comparison*

Description

Convenience function that combined the functionality of the surface and coco functions.

Usage

```
# Deprecated
surface_coco(a, b, span, nodes, fdr = 0.01, collocates = NULL)
```

Arguments

a	A character vector.
b	A character vector.
span	A character string defining the co-occurrence span. See surface function for details.
nodes	A character vector of nodes or character string representing a single node.
fdr	The desired level at which to control the False Discovery Rate.
collocates	A character vector of collocates or character string representing a single collocate.

Details

See surface and coco.

Value

A data.table of the form returned by the coco functions.

Index