Frequently Asked Questions

CorporaCoCo v1.0-1 (2017-03-26)

Anthony Hennessey
Statistics and Probability, School of Mathematical Sciences
University of Nottingham
anthony.hennessey@nottingham.ac.uk

Viola Wiegand
Centre for Corpus Research, College of Arts and Law
University of Birmingham
v.wiegand@bham.ac.uk

Michaela Mahlberg
Centre for Corpus Research, College of Arts and Law
University of Birmingham
m.a.mahlberg@bham.ac.uk

Christopher R. Tench
Division of Clinical Neurosciences, School of Medicine
University of Nottingham
christopher.tench@nottingham.ac.uk

Jamie Lentin
Shuttle Thread
Manchester
jm@ravingmantis.com

Contents

Contents

1 Co-occurrence

1.1 How do I count the co-occurrences for my corpora?

The not very useful answer is "however you like".
More usefully to get you started we supply the `surface()` function which, given a vector of the tokenized corpus text, will produce a complete set of *surface co-occurrence* hit and miss counts in the form expected by the `coco()` function.

```
library(stringi)
text <- "'One side of WHAT? The other side of WHAT?' thought Alice to herself."
tokens <- unlist(stri_extract_all_words( stri_trans_tolower(text)))
surface(tokens, span = '2R')

          x       y H M
 1:   alice herself 1 1
 2:   alice      to 1 1
 3:      of     the 1 3
 4:      of thought 1 3
 5:      of    what 2 2
 6:     one      of 1 1
 7:     one    side 1 1
 8:   other      of 1 1
 9:   other    side 1 1
10:    side      of 2 2
11:    side    what 2 2
12:     the   other 1 1
13:     the    side 1 1
14: thought   alice 1 1
15: thought      to 1 1
16:      to herself 1 0
17:    what   alice 1 3
18:    what   other 1 3
19:    what     the 1 3
20:    what thought 1 3
```

If you are not sure about or need a reminder of the different types, and interpretations of, the various flavors of co-occurrence refer to Evert (2008).

1.2 Does the package understand co-occurrence barriers?

Yes, but if you are interested in co-occurrence barriers you should probably be using a more sophisticated tool for your co-occurrence counting than our `surface()` function. That being said the `surface()` function does implement a basic idea of co-occurrence boundaries. Any `NA` in the vector of tokens passed to the `surface()` function will be counted as a token when considering token seperation, but counts of co-occurrence with `NA` are ignored; the implication of this is that with a `span = n` any set of `n` consecutive `NA` elements will act as a co-occurrence boundary. For example using the `stringi` package to seperate sentences we can stop co-occurrence across sentence boundaries.

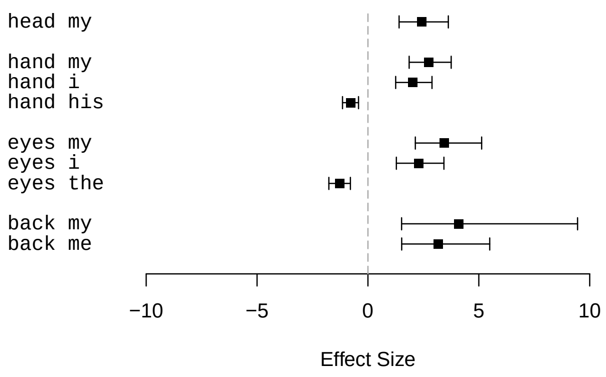
```
sentences  <- unlist(stri_extract_all_boundaries(text, type = 'sentence'))
sentences

[1] "'One side of WHAT? "       "The other side of WHAT?' " "thought Alice to herself."

span <- 2
tokenized_sentences <- stri_extract_all_words(stri_trans_tolower( sentences ))
tokens <- unlist(lapply(tokenized_sentences, function(x) append(x, rep(NA, span))))
tokens

 [1] "one"     "side"    "of"      "what"    NA        NA        "the"     "other"   "side"
[10] "of"      "what"    NA        NA        "thought" "alice"   "to"      "herself" NA
[19] NA

surface(tokens, span = '2R')

          x       y H M
 1:   alice herself 1 1
 2:   alice      to 1 1
 3:      of    what 2 0
 4:     one      of 1 1
 5:     one    side 1 1
 6:   other      of 1 1
 7:   other    side 1 1
 8:    side      of 2 2
 9:    side    what 2 2
10:     the   other 1 1
11:     the    side 1 1
12: thought   alice 1 1
13: thought      to 1 1
14:      to herself 1 0
```

2 Plotting results

2.1 How do I plot my results?

Any object returned by the `coco()` or `surface_coco()` functions can be plotted directly. For example here is how you would plot the results from the ‘Proof of Concept’ vignette.

```
plot(results)
```

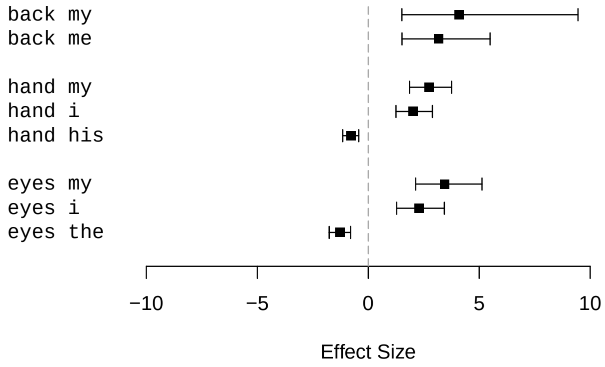


Also remember that the `results` object is just a `data.frame` so you can fashion your own plots directly from the data.

2.2 How do I plot a subset of my results?

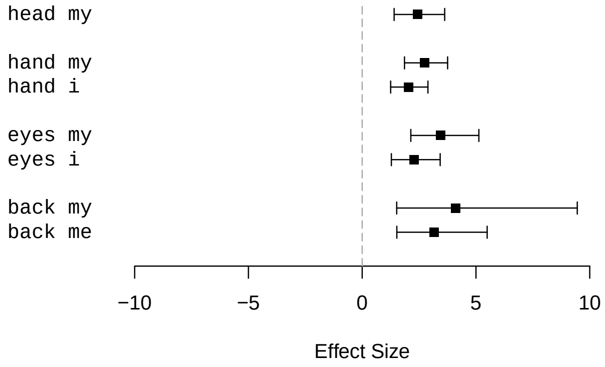
The object returned by the `coco()` and `surface_coco()` functions is a `coco` object. When you `plot()` a `coco` object you can pass an optional `nodes` argument, this can be used to both filter the results set on nodes and to control the order of the nodes in the plot. For example if you only wanted to plot results for the "eyes", "hand", "back" nodes and you wanted to plot them out of alphabetical order

```
plot(results, nodes = c('eyes', 'hand', 'back'))
```



Also remember that the object returned by the `coco()` and `surface_coco()` functions is also just an ordinary `data.frame` so you can filter and order it just like you would any other `data.frame`. This is useful if you want to do some more complex filtering of your results, for example if you are only interested in results with a positive effect size

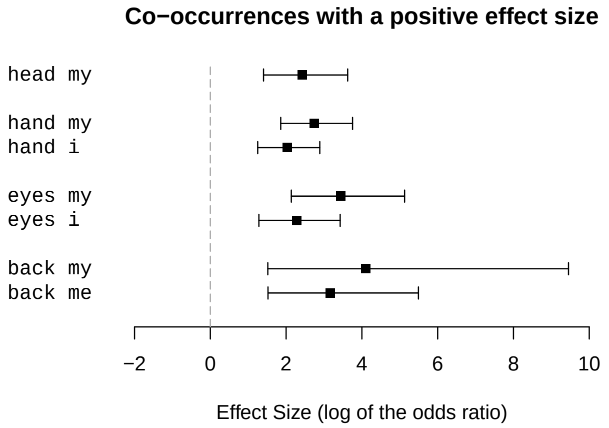
```
plot( with(results, results[effect_size > 0]) )
```



2.3 How do I control what the plot looks like?

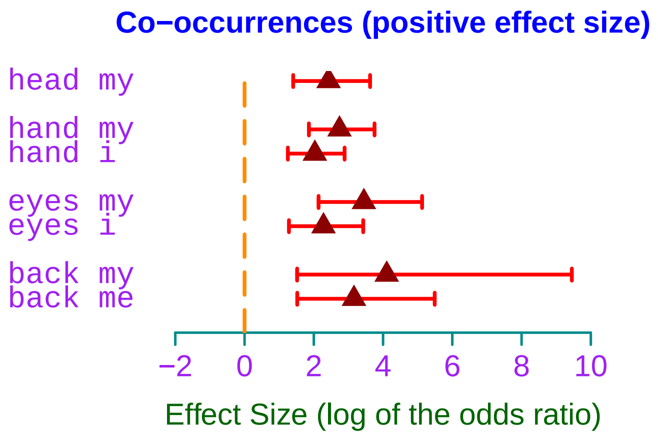
When plotting a `coco` object object there are a limited set of arguments that you can pass through to the underlying `plot` function; these are listed in the help for `plot.coco()`. They can be used something like this

```
plot(
    with(results, results[effect_size > 0]),
    forest_plot_args = list(
        main = 'Co-occurrences with a positive effect size',
        xlim = c(-2, 10),
        xlab = 'Effect Size (log of the odds ratio)'
    )
)
```



Also any graphical parameters set using `par()` will also effect the plot. By combining the two you can effect most of the visual aspects of the plot, for example

```
keep <- par(no.readonly=TRUE)
par(
    cex.main = 1.5,
    col.main = 'blue',
    cex.axis = 1.5,
    col.axis = 'purple',
    cex.lab = 1.5,
    col.lab = 'darkgreen',
    lwd = 3.0,
    col = 'darkred'
)
plot(
    with(results, results[effect_size > 0]),
    forest_plot_args = list(
        main = 'Co-occurrences (positive effect size)',
        xlim = c(-2, 10),
        xlab = 'Effect Size (log of the odds ratio)',
        pch = 17,
        cex.pch = 2,
        lwd.xaxt = 2.0,
        col.xaxt = 'darkcyan',
        col.whisker = 'red',
        col.zero = 'darkorange'
    )
)
par(keep)
```



References

Evert, S. (2008). Corpora and collocations. In A. Lüdeling & M. Kytö (Eds.), *Corpus linguistics: An international handbook* (pp. 1212–1248). Berlin: Mouton de Gruyter.