Unit #4: Collocation analysis in R

Stefan Evert

23 June 2016

Contents

Co-occurrence data & contingency tables

Notation

Collocation analysis is based on the co-occurrence frequencies of word pairs, which are ranked according to some quantitative measure of their **statistical association**. For each candidate word pair (w_1, w_2), the relevant co-occurrence frequency data are collected in the form of a contingency table as shown in the left panel below.

	w_2	$\neg w_2$	
w_1	O_{11}	O_{12}	$= R_1$
$\neg w_1$	O_{21}	O_{22}	$= R_2$
	$= C_1$	$= C_2$	$= N$

	w_2	$\neg w_2$
w_1	$E_{11} = \frac{R_1 C_1}{N}$	$E_{12} = \frac{R_1 C_2}{N}$
$\neg w_1$	$E_{21} = \frac{R_2 C_1}{N}$	$E_{22} = \frac{R_2 C_2}{N}$

Figure 1: Notation for observed and expected contingency tables

$O_{11} = O$ represents the **co-occurrence frequency** of the two words, $R_1 = O_{11} + O_{12}$ is the **marginal frequency** of w_1 and $C_1 = O_{11} + O_{21}$ is the marginal frequency of w_2. The **sample size** N is obtained by adding up all four cells of the contingency table.

Statistical **association measures** (**AM**) compare the observed contingency table O_{ij} with the **expected frequencies** E_{ij} under the null hypothesis of no association between w_1 and w_2, i.e. assuming that the two words co-occur only by chance. Statisticans often speak about the null hypothesis of **independence** of rows and columns, but it may be most intuitive to think of E_{ij} as the contingency table expected for a randomly shuffled corpus. Expected frequencies can be computed from the row and column marginals as shown in the right panel above. Again, the most important value is the top left cell $E_{11} = E$, representing the **expected co-occurrence frequency** of w_1 and w_2.

Contingency tables in R

For a collocation analysis, we will want to compute association scores for a large number of word pairs (w_1, w_2), so it does not make sense to represent each table as a separate 2×2 matrix. It is much better to collect all contingency tables in a single large table (i.e. a `data.frame`), where each row corresponds to the full contingency table of one word pair.

Since an accurate computation of contingency tables (according to Evert 2004, 2008) can be rather tricky, it is often better to use specialized software tools (such as the UCS toolkit) for this purpose and load the resulting data table into R. An exception are syntactic co-occurrences, which can be counted with relative ease based on a list of *pair tokens*.

The `SIGIL` package includes a co-occurrence table for bigrams of adjacent words in the Brown corpus. Start by loading the package and reading the documentation for this data set with `?BrownBigrams`.

```
library(SIGIL)
```

It is often instructive to look at the first few rows of a data frame. You can also use `subset()` to pick out a few interesting words.

```
subset(BrownBigrams, word1 == "learn")
```

```
##          id word1 pos1 word2 pos2 O11 O12   O21    O22
## 10751 10751 learn    V about    I   5  72  1645 908046
## 10752 10752 learn    V  from    I   7  70  4004 905687
## 10753 10753 learn    V  that    I   7  70  6133 903558
## 10754 10754 learn    V   the    D   5  72 58339 851352
## 10755 10755 learn    V    to    T  18  59 24781 884910
```

> **Q:** What are the marginal frequencies of *learn* and *about*? Can you specify the full contingency table for this word pair as a 2-by-2 matrix?

Expected frequencies

As a first step, we need to compute a contingency table of expected frequencies for each word pair. Let us pick out a small subset for practicing:

```
BB <- subset(BrownBigrams, word1 == "learn")
```

In an R markdown document, you can print nicely formatted tables with the `knitr::kable()` function:

```
knitr::kable(BB)
```

	id	word1	pos1	word2	pos2	O11	O12	O21	O22
10751	10751	learn	V	about	I	5	72	1645	908046
10752	10752	learn	V	from	I	7	70	4004	905687
10753	10753	learn	V	that	I	7	70	6133	903558
10754	10754	learn	V	the	D	5	72	58339	851352
10755	10755	learn	V	to	T	18	59	24781	884910

Expected frequencies are computed in terms of row and column marginals. While it would be possible to spell out E_{ij} directly in terms of the observed frequencies O_{ij}, it is easier and safer to compute the row and column totals first and add them to the table. R's `transform()` function provides a convenient way to compute derived variables for all rows of a data frame.

```
BB <- transform(BB,
                R1 = O11 + O12, R2 = O21 + O22,
                C1 = O11 + O21, C2 = O12 + O22,
                N = O11 + O12 + O21 + O22)
knitr::kable(BB)
```

	id	word1	pos1	word2	pos2	O11	O12	O21	O22	R1	R2	C1	C2	N
10751	10751	learn	V	about	I	5	72	1645	908046	77	909691	1650	908118	909768
10752	10752	learn	V	from	I	7	70	4004	905687	77	909691	4011	905757	909768
10753	10753	learn	V	that	I	7	70	6133	903558	77	909691	6140	903628	909768
10754	10754	learn	V	the	D	5	72	58339	851352	77	909691	58344	851424	909768
10755	10755	learn	V	to	T	18	59	24781	884910	77	909691	24799	884969	909768

In the same way, we can now compute expected frequencies E_{ij} using the equations shown at the top of the document.

```
BB <- transform(BB,
                E11 = R1 * C1 / N, E12 = R1 * C2 / N,
                E21 = R2 * C1 / N, E22 = R2 * C2 / N)
knitr::kable(BB[, c(2:9, 15:18)], digits=2)
```

	word1	pos1	word2	pos2	O11	O12	O21	O22	E11	E12	E21	E22
10751	learn	V	about	I	5	72	1645	908046	0.14	76.86	1649.86	908041.1
10752	learn	V	from	I	7	70	4004	905687	0.34	76.66	4010.66	905680.3
10753	learn	V	that	I	7	70	6133	903558	0.52	76.48	6139.48	903551.5
10754	learn	V	the	D	5	72	58339	851352	4.94	72.06	58339.06	851351.9
10755	learn	V	to	T	18	59	24781	884910	2.10	74.90	24796.90	884894.1

Q: What does a comparison of `O11` and `E11` tell you about the collocational strength of these 5 bigrams?

Association measures (AM)

Simple association measures

Simple association measures are based on a direct comparison of observed co-occurrence frequency $O = O_{11}$ with the expected co-occurrence frequency $E = E_{11}$. They do not take the other three cells of the contingency table into account. A particular simple and intuitive AM is the ratio of observed and expected frequency. Its logarithm corresponds to the information-theoretic concept of pointwise **Mutual Information**:

$$\text{MI} = \log_2 \frac{O}{E}$$

Another popular AM in computational lexicography is **t-score**, derived from a completely inappropriate application of Student's t-test (but let's not get into that).

$$\text{t-score} = \frac{O - E}{\sqrt{O}}$$

Let us add these two measures to our collocation table:

```
BB <- transform(BB,
                MI = log2(O11 / E11),
                t.score = (O11 - E11) / sqrt(O11))
knitr::kable(BB[, c(2:6, 15, 19:20)], digits=2)
```

	word1	pos1	word2	pos2	O11	E11	MI	t.score
10751	learn	V	about	I	5	0.14	5.16	2.17
10752	learn	V	from	I	7	0.34	4.37	2.52
10753	learn	V	that	I	7	0.52	3.75	2.45
10754	learn	V	the	D	5	4.94	0.02	0.03
10755	learn	V	to	T	18	2.10	3.10	3.75

> **Q:** Look up the equations of other simple association measures in the lecture slides or on <www.collocations.de/AM> and add them to the data frame. Do the association scores agree with your intuitive judgement of collocational strength?

Statistical association measures

More sophisticated association measures make use of the full observed and expected contingency tables. Many of these measures are derived from statistical hypothesis tests, e.g. the chi-squared test statistic X^2.

$$X^2 = \sum_{ij} \frac{(O_{ij} - E_{ij})^2}{E_{ij}}$$

Other AMs are based on the conditional probabilities $P(w_2|w_1) \approx O_{11}/R_1$ and $P(w_1|w_2) \approx O_{11}/C_1$. A recent example is the directional measure ΔP:

$$\Delta P_{2|1} = \frac{O_{11}}{R_1} - \frac{O_{21}}{R_2}$$

> **Q:** Add these assocation scores for these two measures to the data table. Find other interesting AMs in the lecture slides and add them to the data table.

Collocation rankings

For a real collocation analysis, we have to annotate the full data table `BrownBigrams` with marginal frequencies, expected frequencies and association scores. Note that this cannot be achieved with a single `transform()` command because intermediate results do not become available immediately.

```
BB <- transform(BrownBigrams,
                R1 = O11 + O12, R2 = O21 + O22,
                C1 = O11 + O21, C2 = O12 + O22,
                N = O11 + O12 + O21 + O22)
BB <- transform(BB,
                E11 = R1 * C1 / N, E12 = R1 * C2 / N,
                E21 = R2 * C1 / N, E22 = R2 * C2 / N)
BB <- transform(BB,
                MI = log2(O11 / E11),
                t.score = (O11 - E11) / sqrt(O11),
                X2 = (O11-E11)^2/E11 + (O12-E12)^2/E12 + (O21-E21)^2/E21 + (O22-E22)^2/E22,
                DP = O11 / R1 - O21 / R2)
BB2 <- BB[, c(2:6, 15, 19:22)] # just the relevant columns
```

In order to find the most strongly associated bigrams, we need to reorder the data frame `BB2` according to one of the association measures. The `order()` function determines a suitable row index:

```
idx <- order(BB2$MI, decreasing=TRUE)
idx10 <- idx[1:10] # top-10 collocations (according to MI)
knitr::kable(BB2[idx10, ], digits=2)
```

	word1	pos1	word2	pos2	O11	E11	MI	t.score	X2	DP
3411	baton	Y	rouge	Y	5	0	17.47	2.24	909768.0	1.00
5939	dolce	Y	vita	Y	5	0	17.47	2.24	909768.0	1.00
12162	neutral	Y	tones	Y	5	0	17.47	2.24	909768.0	1.00
22133	u	Y	thant	Y	5	0	17.47	2.24	909768.0	1.00
5591	deja	F	vue	F	6	0	17.21	2.45	909768.0	1.00
12578	notre	Y	dame	Y	6	0	17.21	2.45	909768.0	1.00
16644	souvanna	Y	phouma	Y	5	0	17.21	2.24	758139.2	0.83
22413	vice	R	versa	R	5	0	17.21	2.24	758139.2	0.83
4006	boa	N	constrictor	N	6	0	16.99	2.45	779800.3	0.86
15744	sante	Y	fe	Y	5	0	16.99	2.24	649832.9	1.00

Q: Compute top-10 or top-20 collocations for other association measures. What do you notice about these lists? Can you describe the differences between AMs?

We can make our life a lot easier by defining a helper function that reorders a data frame according to one of the columns. The function takes two arguments: the data frame to be reordered and the name of the column. A third optional argument allows us to select a specified number of highest-ranked collocations.

```
order.df <- function (x, name, n=Inf) {
  idx <- order(x[[name]], decreasing=TRUE)
  head(x[idx, ], n) # return first n rows
}
```

Q: Can you add a fourth argument to the function that allows a choice between increasing and decreasing ordering?

A top-10 list can now easily be computed and printed with

```
knitr::kable(order.df(BB2, "DP", n=10), digits=2)
```

	word1	pos1	word2	pos2	O11	E11	MI	t.score	X2	DP
3411	baton	Y	rouge	Y	5	0	17.47	2.24	909768.0	1
5591	deja	F	vue	F	6	0	17.21	2.45	909768.0	1
5939	dolce	Y	vita	Y	5	0	17.47	2.24	909768.0	1
8813	hong	Y	kong	Y	11	0	16.34	3.32	909768.0	1
11017	lo	Y	shu	Y	21	0	15.40	4.58	909768.0	1
12162	neutral	Y	tones	Y	5	0	17.47	2.24	909768.0	1
12578	notre	Y	dame	Y	6	0	17.21	2.45	909768.0	1
22133	u	Y	thant	Y	5	0	17.47	2.24	909768.0	1
24164	yugoslav	Y	claims	Y	7	0	16.80	2.65	796046.1	1
15744	sante	Y	fe	Y	5	0	16.99	2.24	649832.9	1

In practical work, you will often want to apply various filters, e.g. to find the most strongly associated noun-noun pairs

```
NN <- subset(BB2, pos1 == "N" & pos2 == "N")
knitr::kable(order.df(NN, "X2", n=10), digits=2)
```

	word1	pos1	word2	pos2	O11	E11	MI	t.score	X2	DP
4006	boa	N	constrictor	N	6	0	16.99	2.45	779800.3	0.86
4533	carbon	N	tetrachloride	N	18	0	14.86	4.24	534955.6	0.62
5417	cu.	N	ft.	N	5	0	16.63	2.24	505424.4	1.00
5412	crown	N	gall	N	7	0	16.09	2.65	489871.8	0.54
14211	oxidation	N	pond	N	13	0	14.45	3.61	290632.9	0.57
14212	oxidation	N	ponds	N	7	0	15.27	2.65	276881.0	0.30
15672	rupee	N	equivalent	N	6	0	15.27	2.45	237326.4	1.00
6028	drainage	N	ditch	N	5	0	15.47	2.24	227437.0	0.50
22281	urethane	N	foams	N	11	0	14.09	3.32	192439.1	0.42
2578	anode	N	holder	N	19	0	13.19	4.36	177509.0	0.26

or collocations of a particular node word

```
time.x <- subset(BB2, word2 == "time")
knitr::kable(order.df(time.x, "X2", n=10), digits=2)
```

	word1	pos1	word2	pos2	O11	E11	MI	t.score	X2	DP
15734	same	J	time	N	95	1.11	6.43	9.63	7997.24	0.14
6965	first	J	time	N	66	1.66	5.31	7.92	2501.86	0.06
16659	spare	J	time	N	7	0.02	8.24	2.64	2105.28	0.50
11049	long	J	time	N	39	0.70	5.80	6.13	2100.77	0.09
20913	this	D	time	N	115	8.13	3.82	9.97	1415.31	0.02

	word1	pos1	word2	pos2	O11	E11	MI	t.score	X2	DP
17485	that	D	time	N	65	3.55	4.20	7.62	1069.06	0.03
16234	short	J	time	N	16	0.27	5.86	3.93	902.17	0.09
11981	much	J	time	N	17	0.54	4.97	3.99	498.55	0.05
20748	third	J	time	N	10	0.27	5.22	3.08	354.06	0.06
16561	some	D	time	N	33	2.63	3.65	5.29	351.33	0.02

As a last example, let us take a look at strongly associated bigrams of two consecutive determiners. In the Penn treebank tagset, this includes expressions such as *such a* and *all the*.

```
II <- subset(BB2, pos1 == "D" & pos2 == "D")
knitr::kable(order.df(II, "X2", n=10), digits=2)
```

	word1	pos1	word2	pos2	O11	E11	MI	t.score	X2	DP
20910	this	D	the	D	9	315.07	-5.13	-102.02	319.43	-0.06
11301	many	D	a	D	15	0.69	4.44	3.69	303.73	0.46
8010	half	D	an	D	6	0.31	4.29	2.32	105.75	0.07
20754	this	D	a	D	5	109.29	-4.45	-46.64	102.34	-0.02
17061	such	D	a	D	24	4.07	2.56	4.07	99.80	0.11
429	a	D	half	D	13	2.13	2.61	3.02	57.00	0.00
17062	such	D	an	D	6	0.66	3.18	2.18	43.19	0.03
1496	all	D	those	D	7	1.60	2.13	2.04	18.22	0.00
1490	all	D	the	D	82	128.33	-0.65	-5.12	17.91	-0.02
1487	all	D	that	D	12	4.04	1.57	2.30	15.71	0.00

Q: Are there any surprising entries in the top-10 list? Can you make a guess what's going on?

Keyword analysis

Keywords can be seen as a special case of collocation analysis. Instead of presence or absence of w_2, the columns of the contingency table correspond to the two (sub-)corpora to be compared (see SIGIL Unit #2). For example, a word w that occurs k_1 times in a corpus of n_1 tokens and k_2 times in a second corpus of n_2 tokens would be represented by the contingency table

	Corp 1	Corp 2
w	k_1	k_2
$\neg w$	$n_1 - k_1$	$n_2 - k_2$

We can apply the association measures introduced above to such contingency tables: the highest-ranked "collocations" resulting from this analysis are keywords for corpus 1.

The `SIGIL` package includes a table with frequencies of 60 nouns in the written and spoken part of the British National Corpus. The last row of the table provides the cumulative frequency of all other nouns, which is required to determine the respective sample sizes.

```
knitr::kable(head(BNCcomparison))
```

noun	written	spoken
time	156300	21720
year	146308	15593
people	94568	20939
way	93325	14037
man	88370	6460
day	80062	11011

First, we need to compute the sample sizes of the written and spoken BNC by adding up all frequency counts in the respective column. We take the spoken BNC as the first sample so we can find keywords of spoken language. For technical reasons, the frequency counts must be converted to floating-point numbers first.

```
BNCcomparison <- transform(BNCcomparison,
                           spoken = as.numeric(spoken), written=as.numeric(written))
n1 <- sum(BNCcomparison$spoken)
n2 <- sum(BNCcomparison$written)
```

Now construct a data frame with the full contingency tables as shown above.

```
KW <- transform(BNCcomparison,
                O11 = spoken, O21 = n1 - spoken,
                O12 = written, O22 = n2 - written)
knitr::kable(tail(KW))
```

	noun	written	spoken	O11	O21	O12	O22
56	dynamite	94	6	6	1274741	94	19277836
57	tracer	90	10	10	1274737	90	19277840
58	domicile	100	0	0	1274747	100	19277830
59	barbarism	98	2	2	1274745	98	19277832
60	broadside	100	0	0	1274747	100	19277830
61	OTHER	17813315	1107974	1107974	166773	17813315	1464615

For simplicity, we will only apply simple AMs and compute the expected co-occurrence frequency directly using the equation $E_{11} = (O_{11} + O_{12}) * (O_{11} + O_{21})/(n_1 + n_2)$.

```
KW <- transform(KW, E11 = (O11 + O12) * (O11 + O21) / (n1 + n2))
```

Keywords according to the t-score measure:

```
KW <- transform(KW, t.score = (O11 - E11) / sqrt(O11))
knitr::kable(order.df(KW, "t.score", 10), digits=2)
```

	noun	written	spoken	O11	O21	O12	O22	E11	t.score
7	thing	51670	23171	23171	1251576	51670	19226260	4641.89	121.73
3	people	94568	20939	20939	1253808	94568	19183362	7164.14	95.19
1	time	156300	21720	21720	1253027	156300	19121630	11041.41	72.46
4	way	93325	14037	14037	1260710	93325	19184605	6658.96	62.27
6	day	80062	11011	11011	1263736	80062	19197868	5648.66	51.10

	noun	written	spoken	O11	O21	O12	O22	E11	t.score
2	year	146308	15593	15593	1259154	146308	19131622	10041.65	44.46
14	number	53194	6551	6551	1268196	53194	19224736	3705.59	35.16
20	house	50634	5587	5587	1269160	50634	19227296	3487.02	28.09
18	area	52957	5156	5156	1269591	52957	19224973	3604.37	21.61
16	work	53636	5077	5077	1269670	53636	19224294	3641.58	20.15

Keywords according to Mutual Information:

```
KW <- transform(KW, MI = log2(O11 / E11))
knitr::kable(order.df(KW, "MI", 10), digits=2)
```

	noun	written	spoken	O11	O21	O12	O22	E11	t.score	MI
7	thing	51670	23171	23171	1251576	51670	19226260	4641.89	121.73	2.32
52	banger	77	23	23	1274724	77	19277853	6.20	3.50	1.89
48	sultana	78	22	22	1274725	78	19277852	6.20	3.37	1.83
3	people	94568	20939	20939	1253808	94568	19183362	7164.14	95.19	1.55
47	paymaster	82	18	18	1274729	82	19277848	6.20	2.78	1.54
4	way	93325	14037	14037	1260710	93325	19184605	6658.96	62.27	1.08
27	bulb	875	131	131	1274616	875	19277055	62.40	5.99	1.07
24	spider	884	123	123	1274624	884	19277046	62.46	5.46	0.98
1	time	156300	21720	21720	1253027	156300	19121630	11041.41	72.46	0.98
6	day	80062	11011	11011	1263736	80062	19197868	5648.66	51.10	0.96

Q: Find 10 "anti-keywords" with the lowest MI scores. Are they keywords of written language? Do you notice any problems?

Q: Compute keywords based on other association measures. Then carry out a second keyword analysis for written against spoken analysis. How do you obtain suitable contingency tables for this second analysis?