The Stata Journal (2015)
15, Number 4, pp. 1118–1156

Studying coincidences with network analysis and other multivariate tools

Modesto Escobar
Universidad de Salamanca
Salamanca, Spain
modesto@usal.es

Abstract. In this article, I introduce a new user-written command (`coin`) to study data structures. `coin` is founded on a combination of statistical and social network analyses. The purpose of this procedure is to ascertain the most frequent events in a given set of scenarios and to study the relationships between them. `coin` uses a variety of procedures, such as multidimensional scaling, principal components analysis, correspondence analysis, biplot representations, agglomeration techniques, and network analysis algorithms. Because of this, it works with dichotomous variables and can be applied to the exploratory analysis of questionnaires, the study of textual networks, the review of database contents, and the comparison of different statistical analyses of interdependence.

Keywords: st0416, coin, coincidence analysis, multivariate analysis, social network analysis

1 Introduction

The `coin` command facilitates a wide range of statistical analyses and graphs for a set of multiple and interdependent dichotomous variables. With `coin`, the user can obtain conjoint frequencies of each of the variables. It also permits the calculation of diverse correlation coefficients and measures of distances, aiming for discovery of observable system-wide patterns in series of events or discovery of characteristics that are not mutually exclusive across sets of scenarios or cases by using both multivariate statistical tools and social network analysis.

`coin` may be particularly suitable in three cases:

- when getting information from multiresponses or from variables with categories that are not mutually exclusive
- when working with two-mode networks
- when analyzing co-occurrences

Let us discuss each of these cases in turn.

The first case when `coin` may be particularly suitable is when getting information from multiresponses or from variables with categories that are not mutually exclusive, as may occur, for example, when a variable is consistent in the countries visited by a person. The scope of responses in this case may range from no country visited to more than one hundred countries visited for those who travel a lot. Similarly, for statistical programs used by researchers (R, SAS, SPSS, Stata, Statgraphics, etc.), it is likely that many specialists would mention more than one of these programming environments. This problem has been addressed in previous issues of this journal (Cox and Kohler 2003), and there are programs that convert these types of variables from a polytomous format (see table 1) into a dichotomous format (see table 2).[1]

Table 1. Example of polytomous format

Program1	Program2	Program3
Stata		
SAS	Other	
R	Stata	
SPSS	Other	

Table 2. Example of dichotomous format

R	SAS	SPSS	Stata	Other
0	0	0	1	0
0	1	0	0	1
1	0	0	1	0
0	0	1	0	1

Ben Jann's 2005 widely used **`mrtab`** command permits the analysis of variables with nonmutually exclusive values. This command allows both the formation of tables with multiresponse variables and the creation of two-way tables with other single variables, such as gender. However, **`mrtab`**, as well as previously cited programs, is unable to tabulate multiresponse variables with themselves to determine, for example, how many people have visited both Canada and Mexico or how many researchers use both R and Stata. The **`coin`** command is useful in these situations, because it can tabulate a variable whose values are not mutually exclusive with itself.

1. Within the **`coin`** package is another command called **`precoin`**, which converts the polytomous variables into dichotomous ones. Some advantages of this command compared with existing ones are that it is much faster because it is written in Mata, it is not necessary to determine previously the values of the convertible categories, and it allows the indication of a minimum number of positive cases that a category should have to become a dummy and allows the sorting of new variables by the frequencies of positive values. For details of this command, type **`help precoin`** after installation.

A second time when `coin` may be particularly suitable is in the case of two-mode networks. Although Stata was not designed for network analysis, Corten (2011), Miura (2012), and Grund (2015) have developed commands that permit calculations and graphical representations of networks. However, none of them posed the problem of the treatment of two-mode networks. The purpose of social network analysis is to study the interactions among a set of actors (individual or collective). Its starting point is a square matrix of adjacencies with as many rows and columns as there are actors. The matrix cells contain the magnitude of the link (choice, neighborly relationship, nature of personal relationship, similarity, ...) between the actor in the row with the actor in the column. In the famous example of Italian Renaissance families (Padgett and Ansell 1993), two matrices were created for 16 Florentine families: one for marital relations and another for the commercial relations between them. Both were built symmetrically (nondirected networks) insofar as both marital and commercial relationships involve reciprocity.

`coin` is not a command designed to analyze social networks. However, it uses both statistical and graphical representations and therefore may be useful with two-mode networks, in which relations are established between two different sets of actors or events (such as in the example of Galaskiewicz and Wasserman (1989), which analyzed the relationship between a set of corporations and a set of nonprofit organizations).

The starting point of `coin` greatly resembles these two-mode networks. It is a dataset where a set of actors (or scenarios) are listed in the rows, and another set of actors (or events) are allocated to the columns. The first action of `coin` is to transform this two-mode matrix into a one-mode matrix in which the actors (scenarios or cases) located in rows disappear and only those actors (events or variables) located in the columns are shown. Imagine, for example, two people who go to the movies and three who go to the theater. Their representation would be such that we would have two scenarios in rows and five people in the columns (see table 3).

Table 3. Example of a two-mode network

Scenario	a	b	c	d	e
Movies	1	1	0	0	0
Theater	0	0	1	1	1

By applying `coin`, a square matrix of five people is formed (see table 4). People a and b would be connected to indicate that they both saw a movie, and people c, d, and e would interconnect because they all went to the theater.

Table 4. Two-mode network transformed into a one-mode network

	a	b	c	d	e
a	0	1	0	0	0
b	1	0	0	0	0
c	0	0	0	1	1
d	0	0	1	0	1
e	0	0	1	1	0

A third case where `coin` may be particularly useful is the analysis of co-occurrences. Hofmann and Puzicha (1998) considered that the modeling and prediction of the co-occurrence of events is a fundamental application of unsupervised learning techniques. With this aim, it takes as its point of departure a dyadic data structure called co-occurrence data, which consists of two finite sets of elements, $X = \{x_1, \ldots, x_N\}$ and $Y = \{y_1, \ldots, y_M\}$, wherein the elementary observations are the pairs of elements formed by the sets X and Y together with their corresponding frequency. As an example, imagine a set of texts analyzed with another set of words. A co-occurrence matrix (see table 5) would consist of as many rows as there are available texts and as many columns as there are word frequencies to be analyzed. The elements of this matrix would be the frequency of each word in each text (Leydesdorff and Vaughan 2006; Tumminello et al. 2011).

Table 5. Co-occurrence matrix

	Word1	Word2	Word3	Word4
Text1	2	3	2	10
Text2	4	2	0	3
Text3	8	1	5	0
Text4	0	0	6	7
Text5	5	0	0	2

In contrast to co-occurrence analysis, coincidence analysis has as its starting point a more elementary matrix (called an incidence matrix) whose observations concern merely presence or absence. Thus, the co-occurrence matrix above could be converted into an incidence matrix in two ways: by dividing the texts into smaller units (such as paragraphs) or by converting the frequencies into dichotomous values. As an example, the above co-occurrence matrix is converted into an incidence matrix (see table 6) where the words that are present in each of the texts are expressed by 1. As you can see, all four words studied are found in Text1, while Text2 does not contain Word3 and Text4 does not contain Word1 and Word2.

Table 6. Incidence matrix

	Word1	Word2	Word3	Word4
Text1	1	1	1	1
Text2	1	1	0	1
Text3	1	1	1	0
Text4	0	0	1	1
Text5	1	0	0	1

So, in general, the `coin` command can be useful in at least three situations: first, when multiple responses are analyzed, in which case people interviewed would be the cases and the various responses expressed dichotomously would the variables; second, with two-mode social networks, where actors would be the cases and organizations, events, or other actors would be the variables; and third, in the study of co-occurrences, where different texts or scenarios would be the cases while words, events, acts, or people would be the variables.

Specifically, `coin` is useful in the study of media audiences; the content analysis of newspapers, textbooks, party manifestos, blog posts, tweets, etc. (Provalis Research 2015); the quantification of mutual quotes among a number of authors in a series of articles; counting occurrences and co-occurrences of characters in texts, documents, films, or photographs; the study of relations between organizations or associations based on their common members; and the statistical analysis of the behavior of multiple choice in questionnaires.

2 Foundations

The purpose of coincidence analysis is to detect what characters, objects, attributes, characteristics, or events tend to occur together within certain limits (Escobar 2009). These given limits are called scenarios (I) and are considered to be units of analysis. In each scenario i, a series of J events may occur; these events are dependent or independent of each other and will be represented as J dichotomous variables X_j. The objective of the analysis is to find the subset of pairs of categories that are not independent in the set of possible matching events $J(J-1)/2$ in each scenario. The smallest possible analysis is that of a single set, with only one scenario in which there are two possible events denoted X_j and X_k. Only if both events are jointly present in the same scenario is it asserted that both are mere coincident; that is, $(x_{ij} = 1 \wedge x_{ik} = 1) \Rightarrow f_{jk} = 1$.

The analysis becomes more complicated and becomes the subject of statistical analysis when there is a large number of scenarios in one set (n). In this case, data may be arranged in the form of a matrix of dimensions $n \times J$. This $\mathbf{I}$ matrix is an incidence matrix, composed only of 0s and 1s, which indicate for the ith scenario whether the event j has occurred (1) or not (0).

The corresponding frequency matrix $\mathbf{F}$ has dimensions $J \times J$ and may be obtained by means of the product $\mathbf{I'I}$. The elements of this matrix are univariate frequencies (f_{jj}) and bivariate frequencies (f_{jk}) of the events in one set of scenarios.

Take as an example the previous set (I) of five texts. Each text (i) could be considered a scenario whose events are the appearance of a word of interest for the analysis. In each scenario, there would be as many words (that is, variables) as a researcher considered to be important (J). However, only those words (X_j) that are included in each text would be marked with 1 ($x_{ij} = 1$). In this way, the sum of all variables for each text would represent the number of relevant words (f_i) included in it. On its side, each variable (X_j) would be a word, and a summation (f_j or, even better, f_{jj}) would indicate the number of texts in which the words are present.

The relevant part of the analysis is to study whether the appearance of each word in the texts follows a particular pattern in relation to the occurrence of other words. A mere coincidence would be defined when two specific words (j and k) are present at least once in the same text. And a pair of words would be likely to be coincident if their frequency of coappearance in the selected texts (f_{jk}) is greater than if both events were independent. For example, if a word has appeared in 10 texts out of 100 and the other word has appeared in 20, only if they coappeared in more than 2 texts can it be said that both words are not independent; to put it in another way, these two words will likely be coincident.

Three probabilistic measures can be derived from the frequency matrices $\mathbf{F}$: marginal, conditional, and joint probabilities.

The marginal probability of X_j, denoted as $\Pr(X_j)$, can be obtained from the ratio between the frequency of each event (f_{jj}) and the total number of scenarios where they could have appeared (n):

$$\Pr(X_j) = \frac{f_{jj}}{n}$$

The joint probability of two events X_j and X_k, expressed as $\Pr(X_{jk})$, is derived from the frequency of occurrence in the same scenario, further divided by the set of scenarios contemplated in a given set:

$$\Pr(X_{jk}) = \frac{f_{jk}}{n}$$

The conditional probabilities, denoted by $\Pr(X_j|X_k)$, express the possibility that a specific event has occurred under the assumption that a second event also occurred. They are obtained by dividing the joint probability by the probability of the conditioning event:

$$\Pr(X_j|X_k) = \frac{\Pr(X_{jk})}{\Pr(X_k)} = \frac{f_{jk}}{f_{kk}}$$

By definition, the two events j and k are independent when the conditional probability of the first given the second is identical to the marginal probability of the first, that is, when the following condition is satisfied:

$$f_{jk} = \frac{f_{jj} f_{kk}}{n}$$

On this basis, those events j and k that meet the following inequality are what we will call probable coincident events:

$$f_{jk} > \frac{f_{jj} f_{kk}}{n}$$

However, because it is generally samples of scenarios that are worked on, this criterion has to be more restrictive. Instead, Haberman residuals (Haberman 1973) can be used as a criterion of significance (r_{jk}), according to the following expression:

$$r_{jk} = \frac{f_{jk} - \frac{f_{jj} f_{kk}}{n}}{\sqrt{\frac{1-f_{jj}}{n} \frac{1-f_{kk}}{n}}}$$

We will call it a statistically probable coincidence when two events j and k have the probability of having a 0 residual less than a fixed quantity (normally, 0.05); that is, in large samples, two events (X_j and X_k) are statistically coincident if their r_{jk} is larger than 1.96, as long as this statistic is normally distributed in random samples. Following this, sets of coincidences or adjacencies are defined by the matrix ($\mathbf{A}$). Each element of this matrix is defined by the rule

$$a_{jk} \begin{cases} = c_{jk} \Leftrightarrow \Pr(r_{jk} <= 0) < p \wedge (j \neq k) \\ = 0 \Leftrightarrow \Pr(r_{jk} <= 0) >= p \vee (j = k) \end{cases}$$

where p is a different number according to the type of coincidence considered [mere (1); probable (0.5); or statistically probable (0.05 or less)] and c_{jk} either is a constant (usually 1) to indicate the existence of a match or is a proximity distance. In this context, the use of r_{jk} is proposed, whose maximum value is known and equal to $\sqrt{n}$, although the use of a constant equal to the unit would be preferable.

Given any matrix ($\mathbf{A}$), a graph with as many nodes as events (J) can be drawn in each set (h), whose connecting lines or arrows may be delineated in case of coincidence with a thickness proportional to the size of the residuals r_{jk}. Similarly, the area of each node can be represented in proportion to the frequency (f_{jj}) of each event (X_j) in a determinate set of scenarios.

3 Features of the coin command

The `coin` program is capable of calculating statistics and creating graphs.

3.1 Statistics

`coin` calculates the following statistics:

1. Empirical frequencies (`frequencies`)

   In general, the statistics generated by `coin` are presented in matrix mode with as many columns and rows as events (or variables) that have been entered into the analysis. The coincidence matrix is the result of multiplying the transpose of **I** by itself. In the diagonal of this matrix appear the frequencies of the events in different scenarios, while nondiagonal elements are frequencies of the coincidences of the events. The first matrix of table 7 (below) reflects the coincidence matrix of the incidence matrix of table 6. In the diagonal axis of the coincidence matrix, we see that Word1 and Word4 appear in four texts, while Word2 and Word3 appear in only three. Word1 appears in the same text as Word2 and as Word4 three times, while the remaining coincidences occur twice.

2. Relative frequencies (`vertical`, `horizontal`, `grelative`)

   `coin` can calculate three relative frequencies, which it converts into percentages. The `vertical` and `horizontal` relative frequencies are conditional (that is, calculated for each specific event) on the assumption that the other has also occurred. `coin`'s two modalities are a function of the conditional event whether it is shown in rows or columns. The third relative frequency is the conjoint probability, calculated on the total number of scenarios analyzed. These three relative frequencies are shown in table 7, converted into percentages. The vertical and horizontal percentages are redundant, because they are the respective transpose matrices.

   Unlike in other tables, the bases of the percentages in table 7 are not calculated by summing all frequencies of a row or column. This is because in these types of tables, frequencies in each category are not mutually exclusive. The bases of the horizontal and vertical percentages are the empirical frequencies present in the corresponding diagonal. In addition, the base of the total percentages is the number of scenarios, which is not equal to the sum of the elements in the diagonal insofar as the events are not mutually exclusive but coincident.

Table 7. Absolute, relative, and expected frequencies

```
. use example

. coin Word*, frequencies vertical grelative expected minimum(3)

5 scenarios. 2 probable coincidences amongst 4 events. Density: 0.33.
> Components: 2.
4 events(n>=3): Word1 Word2 Word3 Word4
          Frequencies | Word1  Word2  Word3  Word4
----------------------+----------------------------
                Word1 |     4
                Word2 |     3      3
                Word3 |     2      2      3
                Word4 |     3      2      2      4
  Rel. frequencies(%t) | Word1  Word2  Word3  Word4
----------------------+----------------------------
                Word1 |  80.0
                Word2 |  60.0   60.0
                Word3 |  40.0   40.0   60.0
                Word4 |  60.0   40.0   40.0   80.0
     Col. percentages | Word1  Word2  Word3  Word4
----------------------+----------------------------
                Word1 | 100.0  100.0   66.7   75.0
                Word2 |  75.0  100.0   66.7   50.0
                Word3 |  50.0   66.7  100.0   50.0
                Word4 |  75.0   66.7   66.7  100.0
 Expected frequencies | Word1  Word2  Word3  Word4
----------------------+----------------------------
                Word1 |   3.2
                Word2 |   2.4    1.8
                Word3 |   2.4    1.8    1.8
                Word4 |   3.2    2.4    2.4    3.2
```

3. Expected frequencies (`expected`)

   Expected frequencies are the values that the frequencies would acquire if the events were independent. Their formulas were given in the previous section. `coin` can display them if used with the `expected` option. In the current example, only the duplets Word2 with Word1 and Word2 with Word3 have a frequency higher than what we expected, so they should be considered probable coincident events.

4. Residuals, both standardized and normalized, with their statistical significances (`residuals`, `standard`, `normalized`, `pnormalized`, `pfisher`)

   Comparisons between empirical and expected frequencies give way to the residuals. Residuals can be standardized by dividing by the square root of the expected frequency or can be normalized using Haberman's formula as discussed in the previous section.

   In addition, a residual's significance can be obtained using the `pnormalized` option. This is valid for high expected frequencies above 5 and is better for those above 30. In the small-sample case, it is preferable to use Fisher's exact test (`pfisher`) to obtain correct significances. With the significance, one can distinguish statistically probable events from those that are not. See table 8.

Table 8. Residuals and their significances

```
. coin Word*, residuals standard normalized pnormalized pfisher minimum(3)
5 scenarios. 2 probable coincidences amongst 4 events. Density: 0.33.
> Components: 2.
4 events(n>=3): Word1 Word2 Word3 Word4
              Residuals | Word1  Word2  Word3  Word4
------------------------+----------------------------
                  Word1 |   0.8
                  Word2 |   0.6    1.2
                  Word3 |  -0.4    0.2    1.2
                  Word4 |  -0.2   -0.4   -0.4    0.8
     Standard residuals | Word1  Word2  Word3  Word4
------------------------+----------------------------
                  Word1 |   0.4
                  Word2 |   0.4    0.9
                  Word3 |  -0.3    0.1    0.9
                  Word4 |  -0.1   -0.3   -0.3    0.4
     Haberman residuals | Word1  Word2  Word3  Word4
------------------------+----------------------------
                  Word1 |   2.2
                  Word2 |   1.4    2.2
                  Word3 |  -0.9    0.4    2.2
                  Word4 |  -0.6   -0.9   -0.9    2.2
  p. Haberman residuals | Word1  Word2  Word3  Word4
------------------------+----------------------------
                  Word1 |  0.04
                  Word2 |  0.11   0.04
                  Word3 |  0.80   0.36   0.04
                  Word4 |  0.70   0.80   0.80   0.04
   p. Fisher exact test | Word1  Word2  Word3  Word4
------------------------+----------------------------
                  Word1 |  0.20
                  Word2 |  0.40   0.10
                  Word3 |     .   0.70   0.10
                  Word4 |     .      .      .   0.20
```

5. Odds ratios, their typical errors, and significances (`odds`, `stodds`, `podds`)

   Instead of residuals, odds ratios can be used to see if the two events or series of events are coincident. An odds ratio of higher than 1 would indicate a probable coincidence. If the standard error of these odds ratios (`stodds`) is calculated, their significance can be obtained (`podds`) and is equivalent to that of Haberman residuals (`pnormalized`).

   With small frequencies, it is important not to use odds ratios. In fact, if the non-coincident frequencies are equal to 0, the value of the odds ratio is undefined. In these cases, `coin` adds a value of 0.5 to these frequencies to obtain the approximate value of this statistic. See table 9.

Table 9. Odds ratios, standard error, and significance

```
. coin Word*, odds stodds podds minimum(3)
5 scenarios. 2 probable coincidences amongst 4 events. Density: 0.33.
> Components: 2.
4 events(n>=3): Word1 Word2 Word3 Word4
             Odd ratios | Word1  Word2  Word3  Word4
------------------------+----------------------------
                  Word1 |     .
                  Word2 |   6.0      .
                  Word3 |   0.5    2.0      .
                  Word4 |   1.5    0.5    0.5      .
    se. ln(odd ratios) | Word1  Word2  Word3  Word4
------------------------+----------------------------
                  Word1 |     .
                  Word2 |   2.1      .
                  Word3 |   2.0    1.9      .
                  Word4 |   2.1    2.0    2.0      .
         p. odd ratios | Word1  Word2  Word3  Word4
------------------------+----------------------------
                  Word1 |     .
                  Word2 |  0.21      .
                  Word3 |  0.63   0.36      .
                  Word4 |  0.43   0.63   0.63      .
```

6. Tetrachoric correlation matrix (`tetrachoric`)

   Because the data are binary, it would not be appropriate to work with correlation coefficients. Instead, `coin` works with tetrachoric correlation to carry out a principal components analysis (see section 3.2). With the `tetrachoric` option, this matrix can be displayed on the screen.

7. Matrix of distances between events (`distance(`*measure*`)`)

   One way to study the proximity of events is to use proximity or distance measures. Among them, binary proximity measures (Hubálek 1982; Gower 1985) are especially relevant. These measures have a maximum value of 1 when two dichotomous events are completely coincident and 0 when they are independent of one another. They can sometimes take negative values, where −1 is the minimum value, in the case of two fully antagonistic events; that is, when one event occurs, the other is absent and vice versa.

   Stata allows the calculation of 14 proximity measures that can be classified into four types. Included in the first category is the `matching` distance [also known as the Rogers and Tanimoto coefficient (`rogers`)]. The common characteristic of measures in the first category is that they are ratios between numerators in which positive coincidences (when two events appear in the same setting, a) and negative coincidences (when two events are absent in the same setting, d) appear, and denominators in which all settings are considered with different weights. Measures belonging to this category are the Sneath and Sokal coefficient (`sneath`), the

Rogers and Tanimoto coefficient (`rogers`), the Anderberg coefficient (`anderberg`), and the binary similarity coefficient (`gower2`).

The second category is that of Jaccard distances (`jaccard`). Jaccard distances measure events in which neither of the two events whose level of coincidence is sought to be measured are excluded (d). Those settings that do not include any of the events are thus not included in either the numerator or the denominator. The measures of Dice and anti-Dice (`dice` and `antidice`), Ochiai (`ochiai`), and Kulczynski (`kulczynski`) use the same criteria.

The third category of binary similarity measure is found uniquely in the work of Russell (`russell`). This measure is characterized by considering as similar only those settings in which both events occur (a). It excludes (in the numerator of the ratio) those settings that do not list any event (d), considering that this circumstance does not indicate that the settings are similar. However, in contrast to the similarity measures of `jaccard`, all possible settings appear in the denominator of the ratio.

Finally, the fourth category includes all the measures where the numerator has a comparison between the coincident frequencies (settings in which either the phenomena appear or the phenomena are absent) and the noncoincident frequencies (settings in which one phenomenon appears but the other is absent). As a result, these measures may be either positive (if coincident events predominate) or negative (if noncoincident events predominate), which events do not coincide are more numerous. The well-known Pearson (`pearson`) and Yule (`yule`) measures belong in this section as does that of Hamann (`hamann`).

In the first elaboration, the above measures are all considered similarities. To transform them into measures of distance, they have to be converted in accordance with the following expression:

$$\text{distance} = 1 - \text{similarity}$$

If the measure has a range of 0 to 1, this range is preserved but with a different meaning, because the 0 indicates complete coincidence. However, if the measure possesses a range of -1 to $+1$, the new dissimilarity or distance measure will be between 0 and 2, and a value of 1 will indicate complete independence. As such, coefficients that are greater than this quantity are an expression of two events that coincide with a lower frequency than would be implied by chance. See table 10.

Table 10. Distances among events

```
. coin Word*, dmatrix distance(matching) minimum(3)
5 scenarios. 2 probable coincidences amongst 4 events. Density: 0.33.
> Components: 2.
4 events(n>=3): Word1 Word2 Word3 Word4
    Matching distances | Word1  Word2  Word3  Word4
-----------------------+----------------------------
                 Word1 |   0.0
                 Word2 |   0.2    0.0
                 Word3 |   0.6    0.4    0.0
                 Word4 |   0.4    0.6    0.6    0.0
. coin Word*, dmatrix distance(pearson) minimum(3)
5 scenarios. 2 probable coincidences amongst 4 events. Density: 0.33.
> Components: 2.
4 events(n>=3): Word1 Word2 Word3 Word4
     Pearson distances | Word1  Word2  Word3  Word4
-----------------------+----------------------------
                 Word1 |   0.0
                 Word2 |   0.4    0.0
                 Word3 |   1.4    0.8    0.0
                 Word4 |   1.2    1.4    1.4    0.0
. coin Word*, dmatrix distance(haberman) minimum(3)
5 scenarios. 2 probable coincidences amongst 4 events. Density: 0.33.
> Components: 2.
4 events(n>=3): Word1 Word2 Word3 Word4
    Haberman distances | Word1  Word2  Word3  Word4
-----------------------+----------------------------
                 Word1 |   0.0
                 Word2 |   0.9    0.0
                 Word3 |   3.1    1.9    0.0
                 Word4 |   2.8    3.1    3.1    0.0
. coin Word*, dmatrix distance(geodesic) minimum(3)
5 scenarios. 2 probable coincidences amongst 4 events. Density: 0.33.
> Components: 2.
4 events(n>=3): Word1 Word2 Word3 Word4
    Geodesic distances | Word1  Word2  Word3  Word4
-----------------------+----------------------------
                 Word1 |   0.0
                 Word2 |   1.0    0.0
                 Word3 |   2.0    1.0    0.0
                 Word4 |     .      .      .    0.0
```

Aside from these measures recognized by statistical literature, Haberman's (1973) normalized residual is proposed here as an indicator of concurrence (similarity) between two possible events. Its most outstanding property is that it is symmetrical, which implies that the residuals of a given event and another are the same independently of their order. A second noteworthy property is that it possesses as maximum and minimum values the square root of n. The first property is important where one event necessarily occurs after the other. The minimum value in the second property only occurs when one of the events is present while the other is absent; however, in the case where both events are lacking in at least one setting, this method does not take the minimum value because there is some coincidence in the nonappearance in a given setting.

It would be very easy to convert the normalized residuals of Haberman in a distance measure, if it is subtracted from the square root of n. If so, two entirely concurrent events would have the value 0, two independent events would take the value $\sqrt{n}$, and two comprehensively antagonistic events would take the value $2\sqrt{n}$.

Below, when we discuss the different spatial representations of the incidents, we will use the concept of distance, and we will select different metrics to obtain these representations. I must mention here a) the geodesic distance between incidents, such as the number of vertices to be covered to get from one incident to another; b) the distance χ^2, which is used in the representation of events based on correspondence analysis; and c) tetrachoric correlation, which is used as a measure of similarity in principal components analysis.

8. Adjacency matrix (`adjacencies`)

An adjacency matrix is the central element of the `coin` program. It is constructed from the significances that have been chosen and can be used for the development of graphs. This matrix (like that of incidences) is composed of only 0s and 1s, with 1s indicating the coincidence between two events or a series of events; this coincidence, as discussed above, may be a mere coincidence, a probable coincidence, or a statistically probable coincidence.

In the simple example that follows, the coincidence matrices will be different matrices depending upon how the coincidence is classified (see table 11). All possible pairs are merely coincident because they appear together at least once in the text. Word3 and Word4, for example, do not have a probable coincidence because the probability that Word3 will appear is 66.6%, and it appears in only 50% of cases in which Word4 appears. Finally, no coincidence between the words is statistically probable given that the sample size of texts is so small.

Table 11. Adjacency matrices

```
. coin Word*, adjacencies minimum(3) pvalue(1)
5 scenarios. 6 mere coincidences amongst 4 events. Density: 1.00.
> Components: 1.
4 events(n>=3): Word1 Word2 Word3 Word4
      Adjacency matrix | Word1  Word2  Word3  Word4
-----------------------+----------------------------
                 Word1 |   0.0
                 Word2 |   1.0    0.0
                 Word3 |   1.0    1.0    0.0
                 Word4 |   1.0    1.0    1.0    0.0
. coin Word*, adjacencies minimum(3)
5 scenarios. 2 probable coincidences amongst 4 events. Density: 0.33.
> Components: 2.
4 events(n>=3): Word1 Word2 Word3 Word4
      Adjacency matrix | Word1  Word2  Word3  Word4
-----------------------+----------------------------
                 Word1 |   0.0
                 Word2 |   1.0    0.0
                 Word3 |   0.0    1.0    0.0
                 Word4 |   0.0    0.0    0.0    0.0
. coin Word*, adjacencies minimum(3) pvalue(.05)
5 scenarios. 0 statistically probable(p<=.05) coincidences amongst 4 events.
> Density: 0.00. Components: 4.
4 events(n>=3): Word1 Word2 Word3 Word4
Warning: These variables haven´t statistically probable(p<=.05) coincidences.
      Adjacency matrix | Word1  Word2  Word3  Word4
-----------------------+----------------------------
                 Word1 |   0.0
                 Word2 |   0.0    0.0
                 Word3 |   0.0    0.0    0.0
                 Word4 |   0.0    0.0    0.0    0.0
```

To improve the analysis, the events described in the above matrices can be arranged by frequency. The `descending` option will arrange dichotomous variables from highest to lowest frequency, while the `ascending` option will arrange them from lowest to highest frequency. The default is to present variables in the order in which they appear in the variables list.

When there are many events to analyze, all previous matrices may be limited in the number of columns that are displayed by using the `head()` option. All variables will appear in the rows of the result, while only those mentioned in `head()` will appear in the columns.

Sometimes, it may be best to limit the analysis to only those events that have a certain number of incidences. This can be controlled with the `minimum(#)` option. The default is `minimum(5)`, which means that no results are presented of those variables or events that have not happened in at least five scenarios. This amount may change with an absolute or a relative frequency. If a number between 0 and 1 is expressed, it shall be construed as the percentage of scenarios that must be present for an event to be analyzed.

Similarly, though more complex, is the `support(#)` option. With this option, a frequency or relative frequency can be shown, affecting the construction of the adjacency matrix. `support()` can be understood as the minimum number of coincidences needed so that the matrix element corresponding to a pair of adjacent events appears as a value of 1.

Another way of presenting the aforementioned statistics is with the `list` option. This generates, by default, the ordered set of Haberman positive residuals. This list can be modified in two ways: with the `key(`*key*`)` option, the statistics that are shown can be changed, and with the `lmin(#)` option, the minimum value displayed can be changed (by default, it is set at 0). Similarly, the `ledges(#)` option can be used to limit the list to the # higher values.

9. Centrality measures (`centrality`)

   The statistics enumerated up to now are all matrix values; that is, they have a different value for each pair of events considered. The vast majority of them, including frequencies, are symmetrical, but others (such as vertical or horizontal percentages) are not. In contrast, centrality measures, derived from the adjacency matrix, do not refer to a pair of events, but they take account of the set of relations that each event presents. The degree index shows the number of coincidences (in percentages) of each event. The closeness index takes as an average the geodesic distance (see item 7 above) between an event and those that are directly or indirectly connected. The betweenness expresses the connecting strength that each event has. Similarly, the information index reflects the percentage of coincidences involved in each event in question. See table 12.

Table 12. Centrality measures

```
. coin Word*, centrality minimum(3)
5 scenarios. 2 probable coincidences amongst 4 events. Density: 0.33.
> Components: 2.
4 events(n>=3): Word1 Word2 Word3 Word4
   Centrality measures |     Degree       Close     Between      Inform
-----------------------+------------------------------------------------
                 Word1 |       0.33        0.67        0.00        0.25
                 Word2 |       0.67        1.00        1.00        0.50
                 Word3 |       0.33        0.67        0.00        0.25
                 Word4 |       0.00           .        0.00           .
```

3.2 Graphics

Much of the statistical information obtained through `coin` can be displayed graphically. The `coin` command is capable of producing seven different types of charts.

1. Bar plots (`bar`)

   The most basic charts are bar plots, obtained with the `bar` option. The incidence percentages of each of the events are ordered from the greatest to the least present and are displayed horizontally. See figure 1 for an example.

2. Coincidence bar plots (`cbar(`*varlist*`)`)

   Coincidence bar plots are specific for each event. In each graph of this type, incidences of all the events analyzed appear in light colored bars, and the coincidences of the event represented by the set appear in a darker color. Thus, in these types of graphs, the relative frequencies of the diagonal of the matrix (the incidences of the events) are represented with light colored bars, while the relative frequencies of occurrence in the column of the event are represented with darker colored bars. With this kind of graph, you can detect with ease what events coincide in absolute terms with a particular event by ordering the bars according to their corresponding percentages of coincidence. See figure 1 for an example.

3. Conditional coincidence bar plots (`ccbar(`*varlist*`)`)

   These graphs are also specific for each event. With conditional coincidence bar plots, the probability of every event is conditional to the event that is represented; consequently, the event chosen to be represented does not appear in the plot. The bar with the darker color represents the conditional probability. Both left and right bars, which are lighter, may also appear and indicate the expected value in the case of independence between events. Thus, if the darker bar appears on the left, the event represented by the bar is probably coincidental; if the darker bar appears on the right, however, the corresponding event is less likely. See figure 1 for an example.

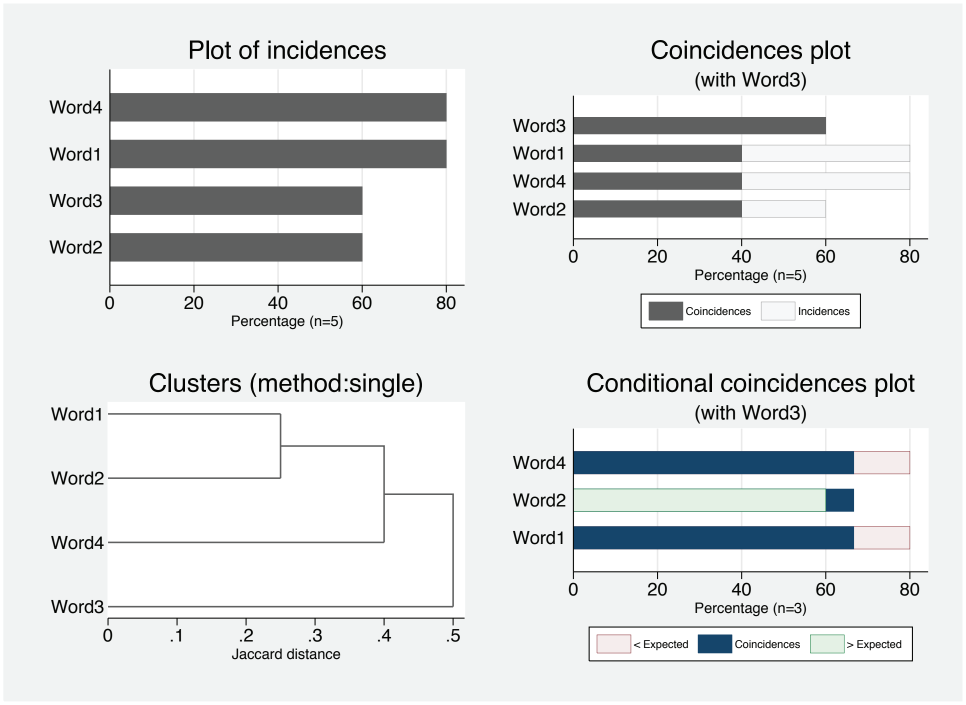

Figure 1. (Co)incidence bar plots and dendrogram

4. Standardized residual graphs (`rgraph(`*varlist*`)`)

   Standardized residual graphs are also specific graphics for each event. In this case, all the events are represented by symbols proportional to their frequency. A horizontal line is drawn from the symbol with a length equivalent to 1.64 times the typical error of the residual. This is done to check the unilateral hypothesis that the events are statistically probable under the assumption that they are normally distributed.

   The central value of the horizontal axis of these graphs is 0, corresponding to the null hypothesis. Two events that are not mutually probable generate a symbol to the left of the central axis, and the confidence interval is plotted from the symbol toward the central axis. If the central axis is not reached, it is assumed that, with a 95% confidence level, neither event is probable in the population. When the two events are probable, the corresponding symbol is to the right of the axis and the confidence interval is plotted to the left. If the vertical axis is exceeded, both events can be considered statistically probable with a significance level of 0.05. See figure 2 for an example.

5. Graphs of odds ratios (`ograph(`*varlist*`)`)

   These graphs are similar to the above, but instead show odds ratios with their corresponding confidence intervals. Thus, graphs of odds ratios have on the horizontal axis a central value of 1. Here the null hypothesis is that the odds ratio is equal to 1, because in this case the events are independent.

   As with the standardized residual graphs, if the symbol is to the left of the corresponding event, it will not be probable with the event that is represented in the whole graph. If the confidence interval reaches the line representing 1, then it may be likely in a population with a confidence interval of 95%. If these two events are mutually probable, the symbol of the odds ratio will be located to the right, at a greater distance than the confidence intervals if they were statistically probable. See figure 2 for an example.

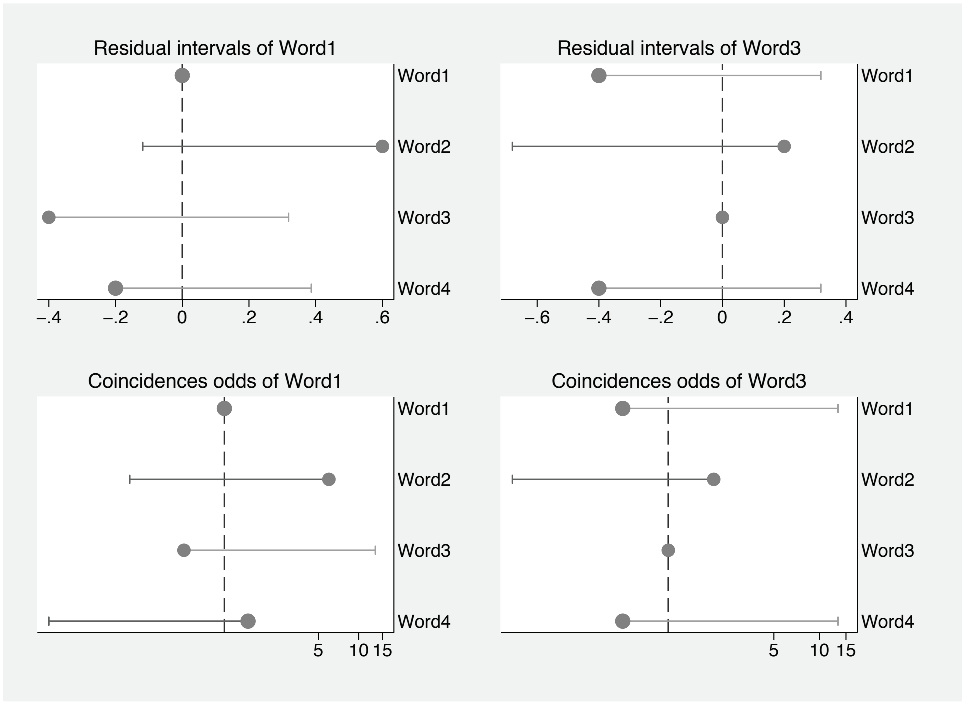

Figure 2. Residuals and odds ratios graphs

6. Dendrograms (`dendrogram(`*linkage*`)`)

   Dendrograms can be created to display the extent of coincidence in a series of events, producing hierarchical clustering algorithms. Several procedures permit the linking of more than two events on the basis of their distances. In the simple method (`single`), events are linked by the distance between the closest events. The complete method (`complete`) links events by the distance between the least coincident events. Events can also be linked by using the median of the distances (`median`) or by performing a weighted average (`waverage`) or a nonweighted mean (`average`). Finally, analysis can also be conducted by comparing the centroids (`centroid`) of each pair of groups of events or by trying to minimize the variance within each cluster of events with the Ward method (`ward`).

7. Network graphs (`graph(`*layout*`)`)

   Another way to represent the coincidences between events in a given analysis of scenarios is by using graphs suitable for social network analysis. The matrix of incidences could be represented by a two-mode graph. However, this would not be practical in the great majority of situations, because it is common to have a large set of scenarios, for example, more than 1,000. For this reason, it would be more convenient to carry out a mode-1 graph constructed by means of the adjacency matrix between events, leaving aside the scenarios.

   The lines connecting events (nodes) in network graphs can be represented with a thickness proportional to the value of the standardized residuals (`edgessized`) or by adopting up to three different patterns of lines: continuous, dashed, and dotted, according to the three levels of significance of the aforementioned residuals (`levels(`*# # #*`)`).

   The scale of the nodes can be changed with the `mmultiplier(`*#*`)` option. If # is less than 1, the nodes become smaller; if # is greater than 1, they become larger. It is also possible to change the color or form of nodes, but an auxiliary file must be used (see section 3.3).

   To examine a particular area of the graph in greater detail, the `ego(`*varlist*`)` option can be used to omit events or variables that are unrelated to those expressed in the list of variables. This option is complemented by the `lego(`*#*`)` option, which has a default of 1. This number denotes the number of elements in the path. A number greater than 1 shows events that are not immediately related to the variables in `ego()`. The larger the number specified in `lego()`, the greater the number of nodes that will be represented in the network of coincidences plotted graphically. An alternative to control of the graphs of network events is the `edges()` option.

Graph coordinates

The principal difficulty with these graphs is deciding where to place each node. Although there are multiple algorithms to build the nodes layout, only the most statistically based has been implemented in `coin`. These are the following:

1. Circular coordinates (`graph(circle)`)

   In coincidence analysis, circular coordinates place nodes equidistant in an imaginary circle. If the nodes are intended to represent J vertexes, the 360 degrees of a circle is divided into J parts and each event is located within a division. The abscissa coordinates are obtained using the cosine function, while the ordinates are calculated using the sine function. These calculations can take several forms depending on the order in which the events are placed.

2. Multidimensional scaling (`graph(mds)`)

   Multidimensional scaling (MDS) is "a generic term for a class of techniques that attempt to construct a low-dimensional geometrical representation of a proximity matrix for a set of stimuli, with the aim of making any structure in the data as transparent as possible" (Everitt 2003, 252). MDS can also be considered a "dimension-reduction and visualization technique. Dissimilarities (for instance, Euclidean distances) between observations in a high-dimensional space are represented in a lower-dimensional space (typically two dimensions) so that the Euclidean distance in the lower-dimensional space approximates the dissimilarities in the higher-dimensional space" (StataCorp 2015, 471).

   To represent the adjacency matrix according to the principles of a dimensional scale, it must be converted into a set of distances (dissimilarities). The most appropriate method would be to convert the adjacency matrix into its corresponding geodesic distances matrix containing either the number of nodes that separate two events or the sum of the shortest distances if they are different from 1. Next, a modern scaling method with stress loss function, normalized by the squared Euclidean distances, is used to obtain the coordinates of the nodes. Modern MDS (the `mdsmat` command) is applied to the matrix of geodesic distances, using as initial location the circular coordinates of the nodes by the order of introduction or modified according to ascending or descending frequency. By default, a maximum of 1,000 iterations is used to converge, but this limit can be changed with the `iteration(#)` option.

   The effect of applying the mechanics of MDS to the representation of vertices, nodes, or events is twofold. First, two correlated events will tend to be close in space. Second, the events with the highest number of matches will tend to be located in the center of the diagram. As a consequence, less correlated events will tend to be located on the fringes of the space. See figure 3, below, for an example.

3. Principal components analysis (`graph(pca)`)

   Historically, this is the first statistical tool for the spatial representation of variables based on their correlations. Principal components analysis (PCA) is "a procedure for analyzing multivariate data which transforms the original variables into new ones that are uncorrelated and account for decreasing proportions of the variance in the data. The aim of the method is to reduce the dimensionality of the data" (Everitt 2003, 296). "The leading eigenvectors from the eigen decomposition of the correlation or the covariance matrix of the variables describe a series of uncorrelated linear combinations of the variables that contain most of the variance" (StataCorp 2015, 723).

The principal problem with the factorial representation of correlated events is that their factorial coordinates are derived from a correlation matrix whose dichotomous variables are inappropriate for representing such events. To resolve this problem, tetrachoric correlations are proposed for the construction of the input matrix of the subsequent factorial analysis. The `pcamat` command is used and forced to return only two components. Nonrotated eigenvalues are considered to be coordinates of the representation of the nodes.

In this case, the spatial representation will be such that the more highly correlated two events are, the closer they will be to each other. (They occur or do not occur simultaneously.) However, unlike the representation of MDS, events with high mutual correlations will be located at one extremity of the space, generally sharing coordinates of the first (or second) dimension.

Notably, in PCA, the number of noncoincidences is taken into account (as negative correlations). When two events fail to reach a requisite threshold, the coincidence is determined to be 0. In factor analysis, such negative correlations are taken into account and also quantified. See figure 3 for an example.

4. Correspondence analysis (`graph(ca)`)

   Correspondence analysis (CA) is "a method for displaying the relationships between categorical variables in a type of scatterplot diagram" (Everitt 2003, 94). It "offers a geometric representation of the rows and columns of a two-way frequency table that is helpful in understanding the similarities between the categories of variables and the association between the variables. [...] In some respects, CA can be thought of as an analogue to principal components for nominal variables" (StataCorp 2015, 35). With this analysis, similar categories (events or incidences) can be represented together in a Euclidean space of $\min\{(R-1),(C-1)\}$ dimensions according to the pattern of concurrence. To obtain the coordinates by this method, the command `camat` is used. The source matrix is formed with scenarios as rows and events as columns. Only two factors are used to obtain the coordinates of the nodes.

   In the case of CA, χ^2 distances are also used. Using this analysis, the columns (C, whether event or incidence) are compared with each other to discover if they show patterns of similar occurrence in the set of scenarios (R rows) in which they may be included. In a similar fashion to factor analysis, sets of incidents that cooccur in the same scenarios will tend to be located closer together in the graph. Infrequent events will tend to go on the extremities, occupying the greater part of the inertia (variance) of the graph. See figure 3 for an example.

5. Biplot analysis (`graph(biplot)`)

   As Gabriel (1971) comments, "any matrix of rank two can be displayed as a biplot which consists of a vector for each row and a vector for each column, chosen so that any element of the matrix is exactly the inner product of the vectors corresponding to its row and to its column. If a matrix is of higher rank, one may display it approximately by a biplot of a matrix of rank two which approximates the original matrix." The `biplot` command is used, and the variable coordinates are used to represent the nodes.

The result of applying biplot graphing techniques is a graph where both observations (scenarios, in the case of coincidence analysis) and variables (events, in this context) are combined. These are represented in a peculiar way because they are drawn as vectors emanating from the center of the graph with their length proportional to their variance and at an angle equal to their correlation.[2]

With this method (in contract to CA), where the variables are not standardized, the frequent and infrequent events will appear in the center of the graph, while those with a probability close to 50% will be located at the extremities of the graph, provided that they demonstrate high correlation with other events. See figure 3 for an example.

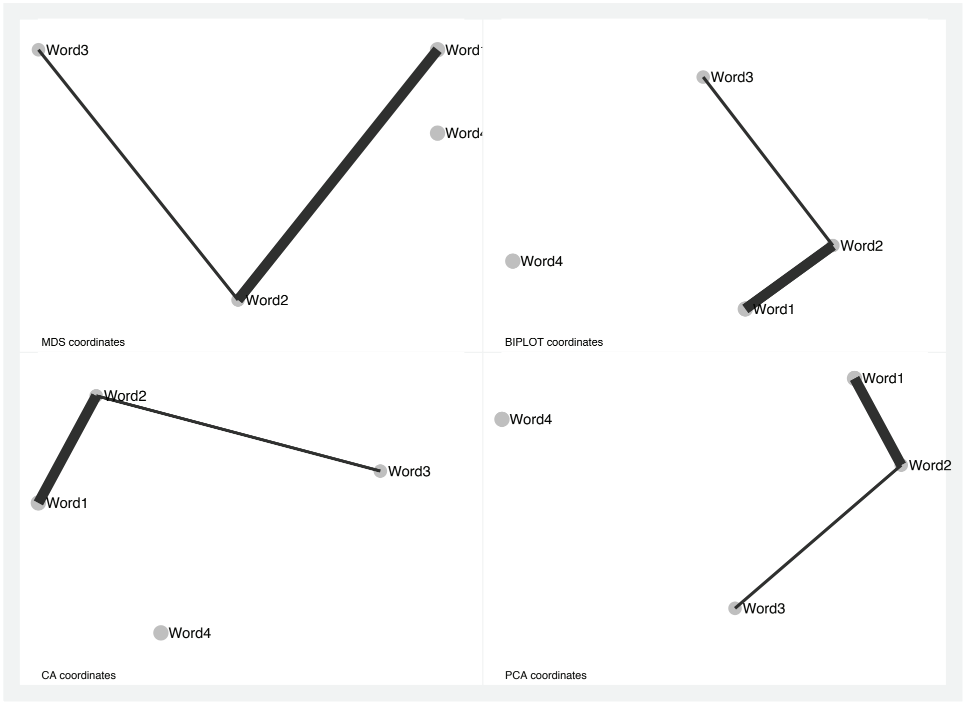

Figure 3. Coincidence network graphs

2. From a practical standpoint, biplot is a PCA that does not hold more than two factors. However, there are two differences. The first is that Pearson correlations are used in biplots, while tetrachoric correlation coefficients are used in PCA. The second is that biplots operate with the mean difference, while in PCA the default is to standardize variables.

3.3 The auxiliary file

An auxiliary file in `coin` can have two purposes: to make a selection using determinate criteria other than the size of events in the analysis and to give color and shape to network graphs. Below, the form that an auxiliary file must have will be explained and then its two uses will be recounted.

An auxiliary file is a type of Stata dataset in which events are to be included as cases. The fundamental characteristic of this type of file is that it must have a string variable called `variable` whose values must match the names of the variables in the main file. This auxiliary file could therefore be called a "secondary file". In addition to the `variable` variable, the secondary file may contain other variables of an integer numeric type, preferably labeled, which will help select events or graphs to represent them differently.

`coin` facilitates the creation of the auxiliary file with the `varsave(`*filename*`)` option, which can be accompanied with the `replace` option to overwrite an existing file. `varsave()` generates a file with as many cases as variables that are specified in the *varlist* of `coin`. In addition to generating the `variable` variable described above, `coin` will generate the variable `label`, consisting of the label of each variable, and the variable `frequency`, consisting of the number of events present in the original database. This file must be later edited to add new variables to fulfill the purpose of its creation.

Once the auxiliary file has been created, the data must be loaded into memory with the primary file. You can do this with the `coin` command by typing `using` *filename*. If you desire to select in the analysis the events with certain determinate values in the available variables, you must add the `where(`*exp*`)` option, where *exp* must contain a logical expression in which the names of the variables in the secondary file must be used. For example, if `where(c1 == 1)` is used, Word1 and Word4 are the only events to be analyzed.

An auxiliary file can also allow the variation of the attributes of the nodes of a graph. If it is desired to use different colors, symbols, or both, the options `color(`*varname*`)` or `form(`*varname*`)` must be included, as well as including `using` *filename*. If the `legend` option is added, a legend will also appear in the graph, giving the meaning of the colors and the symbols. Together with `color()` or `form()`, the options `colors(`*colorlist*`)` or `symbols(`*symbollist*`)` can also be used.

3.4 Other possibilities offered by coin

The `coin` command allows data (scenarios) to be weighted by weight (`fweight`, `aweight`, `iweight`, and `pweight`). The `in` *range* and `if` *exp* clauses can also be used to select the scenarios that are encountered in a particular position in the dataset or that satisfy a condition.

The `over(`*varlist*`)` option permits further possibilities for analysis. Using it, the analysis of coincidences can be repeated, controlling each of the events expressed in the option. A limitation that this command possesses (like those commands that draw partial graphs) is that in the specified list any variable that is not found in the main list of variables cannot be included. The result is as many matrices, tables, or graphs as there are variables specified in the list. Also possible is the keyword `_all` to control each of the events analyzed. Even after having specified the `over()` option, `coin` ends with a comprehensive analysis of coincidences covering all the events.

Finally, `coin`'s statistical procedures and ability to convert a two-mode network to a one-mode network can be used in more specialized network mapping and analysis programs because `coin` optionally saves its results to five formats: comma-separated values, Microsoft Excel, UCINET (Borgatti, Everett, and Freeman 2002), Pajek (De Nooy, Mrvar, and Batagelj 2011), and Network Analysis Using Stata (`nwcommands`; Grund 2015). The `export(`*filename*`,` *type*`)` option is used to do this, where *type* can be any of the following (corresponding to the formats just mentioned): `csv`, `xls`, `dl`, `pjk`, or `nw`. The `replace` option must be added to overwrite an existing file.

4 The coin command

4.1 Syntax

`coin` *varlist* [*if*] [*in*] [*weight*] [`using` *filename*] [`,` *options*]

`fweight`s, `aweight`s, `iweight`s, and `pweight`s are allowed; see [U] **11.1.6 weight**.

See the following section for the list of available *options*.

4.2 Options

Options can be classified into the following groups:

Outputs (to produce or not produce some results of the analysis)

`frequencies` writes a table of absolute observed frequencies of coincidences.

`grelative` writes a table of relative frequencies (%) of coincidences.

`crelative` writes a table of relative frequencies (%) of coincidences in the case of controlled variables; see `over(`*varlist*`)`.

`mrelative` writes a table of relative frequencies (%) including missing variables.

`vertical` specifies to display the conditional frequencies vertically calculated (column percentages).

`horizontal` specifies to display the conditional frequencies horizontally calculated (row percentages).

`expected` specifies the expected values of the number of coincidences under the independence hypothesis.

`residuals` specifies to display the residuals, that is, differences between observed and expected frequencies.

`standard` specifies to display the standard residuals, that is, residuals divided by the square root of the expected frequency.

`normalized` specifies to display the Haberman residuals, that is, the adjusted standard residuals.

`pnormalized` specifies to display the significance or p-value of the Haberman residuals.

`odds` specifies to display the odds ratios.

`stodds` specifies to display the standard error of ln(odds ratios).

`podds` specifies to display the p-value of the odds ratios.

`pfisher` specifies to display the p-value of the Fischer exact test.

`tetrachoric` specifies to display the tetrachoric correlations of binary events.

`adjacencies` specifies to display the adjacency matrix, which is obtained comparing z and p.

`dmatrix` specifies to use the distance matrix with the distance measure specified in `distance()`.

`similarity` specifies to use the similarity matrix with the distance measure specified in `distance()`. All the distances are scaled to have a minimum of 0 and a maximum of 1.

`distance(`*measurename*`)` specifies the measure to display or to graph as a dendrogram. *measurename* can be one of the following: `haberman`, `geodesic`, `matching`, `sneath`, `rogers`, `anderberg`, `gower2`, `jaccard`, `dice`, `antidice`, `ochiai`, `kulczynski`, `russell`, `hamann`, `yule`, or `pearson`. The default is `distance(haberman)`.

`mminimum(`#`)` specifies the minimum value of the elements of the similarity matrices. Values below # are changed to 0.

`list` specifies the ordered list of a symmetrical statistic. The default statistic is Haberman.

`key(`*statistic*`)` specifies the statistic to be shown in the ordered list. *statistic* can be one of the following: `frequencies`, `grelative`, `vertical`, `horizontal`, `expected`, `odds`, `residuals`, `standard`, `normalized`, `pnormalized`, `podds`, `pfisher`, `tetrachoric`, `adjacencies`, or `dmatrix`.

`lminimum(`#`)` specifies the minimum value of the ordered symmetrical statistic to list. # can be negative.

`centrality` specifies to display the centrality measures for each event (degree, closeness, betweenness, and information).

`all` specifies to display all previous statistics.

`x` specifies the table of coordinates for each event. This option may be specified only if the `xy()` option is also specified.

`xy(mds | pca | ca | biplot | all)` specifies the table of coordinates for each event.

Controls (apply filters to data)

`over(`*varlist*`)` controls the output for those variables in *varlist*.

`where(`*exp*`)` selects events according to their characteristics specified in *exp*. `using` *filename* must be used with `where()`.

`head(`*varlist*`)` shows only the specified variables in columns.

`width(`#`)` specifies the size of the label column. The default is `width(20)`. The maximum # is 30 characters.

`variable(`*varname*`)` displays the specified variable in the first column and the last row.

`ascending` shows the events in ascending order of appearance.

`descending` shows the events in descending order of appearance.

`minimum(`#`)` specifies the minimum frequency of an event to be analyzed. The default is `minimum(5)`. If 0<#<1, the percentage of scenarios where the event occurs is assumed.

`support(`#`)` specifies the minimum frequency of a coincidence to be represented as an arrow in the graph. The default is `support(1)`. If 0<#<1, the percentage of scenarios where the coincidence occurs is assumed.

`pvalue(`#`)` specifies the minimum p of the Haberman's residual to establish an adjacency. The default is `pvalue(0.5)`, meaning that the coincidence is probable. If `pvalue(1)` is specified, a simple coincidence is considered. For statistically probable coincidences, set the value at 0.05 or less.

`bonferroni` specifies to correct the significance of the p-value by Bonferroni criteria: events(events − 1)/2.

`ledges(`#`)` specifies the maximum number of edges to list.

`edges(`#`)` specifies the maximum number of edges to represent in the graph.

`iterations(`#`)` specifies the maximum number of iterations to obtain coordinates in the MDS graph option.

Plots (graphic data representation)

There are four classes of graphic data representation: bar plots, residual plots, dendrograms or cluster plots, and network graphs.

1. A bar plot can be a simple, coincidence, or conditional coincidence bar plot.

   `bar` plots incidences of all events.

   `cbar(`*varlist*`)` plots coincidences of events in *varlist*.

   `ccbar(`*varlist*`)` plots conditional coincidences of events in *varlist*.

2. Residual plots can be drawn with two different units: standardized residuals or odds ratios.

   `rgraph(`*varlist*`)` draws a standardized residual graph.

   `ograph(`*varlist*`)` draws an odds ratios graph.

3. Dendrograms represent each event as a line that is as long as the minimal distance to another event or set of events. One of the available statistics to measure the distance between two events must be set in the `distance()` option (see above). By default, Haberman's distance is chosen (`distance(haberman)`).

   `dendrogram(`*linkage*`)` plots a dendrogram. *linkage* may be one of the following: `single`, `average`, `complete`, `waverage`, or `wards`. You may also specify `median` or `centroid`, but they cannot handle dendrogram reversals.

4. Network graphs represent each event as a node with a size proportional to its occurrence and connected to any other event with which it is coincident. The placing of every node will depend on the method used. At this moment, `coin` is able to produce the five different algorithms already explained above: circular position (`circle`), MDS (`mds`), PCA (`pca`), CA (`ca`), and biplots (`biplot`). If `all` is specified, Stata will draw a combined graph using the latter four forms.

   `graph(circle|mds|pca|ca|biplot|all)` draws a network graph using different types of coordinates.

   `levels(`# [#] [#]`)` specifies sorted cutpoints (a minimum of one and a maximum of three) to draw different line styles (dot, dashed, or solid).

   `positions(`*matrixname*`)` specifies a matrix of the number of nodes (or columns) to define the clock position of the labels in the graph.

   `mmultiplier(`#`)` specifies the markers scale. The default is `mmultiplier(1)`. This option is especially useful in graphs with characteristics.

   `ego(`*varlist*`)` specifies the graph's ego-net with the *varlist* events.

   `lego(`#`)` specifies the maximum length of `ego()`'s path to represent.

   Special characteristics can also be added to network graphs:

   `goptions(`*graph_options*`)` specifies any graph options that affect the characteristics of graphs, for example, `goptions(title("Network Graph"))`. See [G-3] ***twoway_options***.

Network graphs can be enhanced with different colors and forms to its nodes. This can be done in two ways:

a. With a two-dimensions matrix.

   character(*matrix*) introduces a (nodes x 2) characteristics matrix conforming to symbol and color.

b. With a filename, where a so-called string variable coincides with the name of the variables that represent events in the main file. using *filename* specifies a file with events as first variables and other variables as attributes of events.

   form(*varname*) specifies the variable name with which to configure the form of the nodes.

   symbols(*symbollist*) specifies the symbols to use for the nodes. using *filename* and form(*varname*) must also be specified with symbols().

   color(*varname*) specifies the variable name with which to configure the color of the nodes.

   colors(*colorlist*) specifies the colors of the symbols of the nodes. using *filename* and form(*varname*) must also be specified with colors().

   labels(*varname*) specifies a string variable from within using *filename* with which to configure the labels of the nodes. using *filename* and form() must also be specified with labels(). By default, event labels are used.

   legend indicates that a legend with the symbols and colors of the nodes be included. Characteristics of events should be labeled in the characteristics file.

   groups(*#*) is an alternative to specifying options form() and color(). groups() represents events according to their distances in such a way that similar events appear with same colors. # specifies the number of event conglomerates. See dendrogram() and distance() above.

Exportations (saved files)

varsave(*filename*) saves a Stata dataset (.dta) with variable names as cases to begin the creation of the characteristics file.

export(*filename*[, csv | xls | nw | pjk | dl]) specifies to export data to one or two files, depending on the optional specification. csv creates two files: *file*_E.csv contains edges information, and *file*_N.csv (the default) contains nodes information. xls writes two tabs in the same Excel file (.xls) with nodes and edges information. nw saves a Stata dataset (.dta) that can be read with nwcommands (Grund 2015). pjk saves information so that it can be read with the Pajek program. dl saves a file to be read with UCINET.

replace specifies to overwrite existing files.

4.3 Stored results

`coin` stores the following in `r()`:

Scalars			
`r(n)`	number of observations	`r(e)`	number of selected events
`r(c)`	number of coincidences	`r(d)`	density
Macros			
`r(events)`	list of events	`r(selected)`	list of selected events
Matrices			
`r(L)`	vector of frequencies	`r(Q)`	vector of selected events
`r(F)`	frequencies	`r(G)`	relative frequencies
`r(V)`	vertical percentages	`r(H)`	horizontal percentages
`r(E)`	expected frequencies	`r(R)`	residuals
`r(S)`	standardized residuals	`r(N)`	normalized residuals (Haberman)
`r(pH)`	significance (Haberman)	`r(O)`	odds ratios
`r(sO)`	standard error (odds ratios)	`r(pO)`	significance (odds ratios)
`r(pF)`	significance (Fisher test)	`r(T)`	tetrachoric correlations
`r(A)`	adjacency matrix	`r(C)`	centrality measures
`r(X)`	nodes coordinates	`r(Node)`	nodes table
`r(Edge)`	edges table	`r(oEdge)`	ordered edges table
`r(`*Distance*`)`	distance matrix	`r(`*distance*`)`	similarity matrix
`r(Degree)`	degree vector	`r(Close)`	closeness vector
`r(Between)`	betweenness vector	`r(Inform)`	information centrality vector

5 An example: Spanish unemployed

This example comes from the Spanish Labor Force Survey[3] and seeks to explore the ways in which job seekers in Spain look for work. This survey is prepared by the National Institute of Statistics every three months on a sample of 60,000 households. Here we use the survey for the second quarter of 2013. It contains information about 171,909 people who are aged from 0 to 105 years old, among whom 24,485 were seeking work. To make the example a size that could be downloaded and used quickly with the `coin` command, 1,000 unemployed people were selected at random. This is, therefore, a two-phase sample.

The survey asked about the methods the unemployed were using to find work. The responses were coded into 13 categories, which have been grouped into 6 classes of job seekers: a) those using employment agencies, whether public or private; b) those using contacts with employers or other types of contact; c) those using job advertisements; d) those seeking to establish themselves as self-employed; e) those who restrict themselves, waiting for results or offers; and f) those who cannot be classified in the other categories. In `search.dta`, there are 13 indicator variables (with values that are either 0 or 1), which express whether the selected individuals used any of the 13 categories mentioned.

3. Data were obtained from http://www.ine.es.

A primary examination of similarities in the major job search methods used by the unemployed can be obtained using the `frequencies` option of the `coin` command (see table 13). The results table is displayed by categories.

Table 13. Table of coincidences of ways of looking for jobs

```
. use search

. coin search1-search13, frequencies minimum(.20)

1000 scenarios. 58 probable coincidences amongst 13 events. Density: 0.74
8 events(n>=200): search1 search2 search3 search4 search5 search6 search9 search
> 10
(28 probable coincidences amongst 8 selected events. Density: 1.00. Components:
> 1)

          Frequencies | se~h1  se~h2  se~h3  sea~4  sea~5  sea~6  sea~9  se~10
----------------------+--------------------------------------------------------
Agency                |
               public |   555
              private |   205    310
----------------------+--------------------------------------------------------
Contacts              |
            employers |   452    279    801
             informal |   495    281    726    866
----------------------+--------------------------------------------------------
Ads                   |
              placing |   227    174    367    368    405
           looking at |   401    256    612    637    403    706
----------------------+--------------------------------------------------------
Waiting               |
               offers |   232    118    262    291    141    232    315
              results |   148    105    227    229    152    214    136    250
```

Because there are too many categories (13) to fit horizontally on a page, nonfrequent events have been omitted from the table. To this end, the option `minimum(#)` is used, and in this case, a value of 0.20 is used to discard events (ways of looking for jobs) with less than 20% of occurrences (unemployed people using these forms).[4]

To obtain the graph (figure 4), this minimum has been changed to represent events with more than 1.5% of incidence. We use an auxiliary file (`searchc.dta`) so that the different categories are represented by distinct symbols.

```
use search
matrix U=J(1,11,6)
coin search1-search13 using searchc, frequencies graph(mds)           ///
        levels(.05 .01 .001) form(type) minimum(.015)                 ///
        goptions(title(Ways of looking for jobs) name(jobs, replace)) ///
        positions(U)
```

4. Because we introduced a value less than 1 into the option `minimum()`, the program automatically considers this to be a proportion.

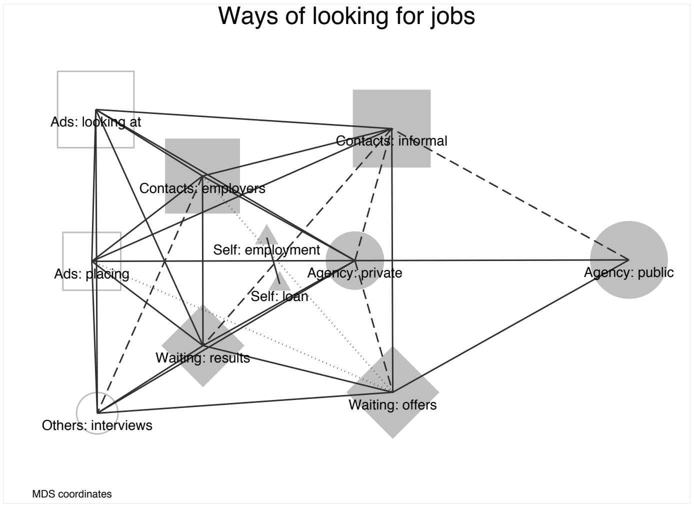

Figure 4. Links between different ways of looking for jobs

The result is a graph in which three methods of job search stand out because of their frequency. First and second are those who use their contacts, be they informal or with employers. Third are those who look at job advertisements. These three along with placing advertisements form a strong network of options very frequently used by unemployed people.

On the other hand, a high degree of centrality in the search method of using private agencies can be observed, which shows that those who use this method often also used other means to look for a job. This contrasts with those who seek employment using public agencies because they only regularly use the method of waiting for job offers. Finally, the low frequency of those who are seeking to create their own employment or asking for a loan can be perceived, but it is clear and obvious that both are coincident among a small proportion of job seekers.

All of these categories are present in the same question in the questionnaire because it is asked in a multiple choice format. Each subject may use more than one method to find work, so coincidences can be identified and counted. In this example, when working with the sample only, the statistically probable coincidences were represented. The representation of different thicknesses of line corresponds to different degrees of significance. The dotted lines correspond to a $p < 0.05$; the discontinuous lines to $p < 0.01$; and the continuous lines to $p < 0.001$.

Another important function that coincidence analysis can perform is to explore the relationships between categories of different variables. To give one example of this utility, we add three age groups to the previous example with 13 categories of job search: young people under age 40, adults between ages 40 and 55, and those older than age 55. To properly introduce these characteristics, we use factor variables to introduce all the categories of a nominal variable.

The most basic table for the analysis of this relationship is obtained by means of vertical percentages (see table 14).[5] By examining the first three columns, it can be deduced that there is a monotonic relationship between age and the job search method used: older people rely more on public employment offices, while younger people rely more on private offices. Younger people are more prone to use contacts with employers than older people; in the case of informal contacts, the opposite pattern can be observed. With regard to advertisements, young people both place them and examine them more than older people. However, older people are more likely to receive job offers, while the young are more likely to wait for results.

To test whether these relationships are significant, recourse is made to statistical significance derived from normalized residuals (Haberman residuals). Examining table 14, the following hypothesis is tested: that young people are really more prone to search by means of business contacts, looking at advertisements and to a lesser extent, to place advertisements concerning themselves. For their part, older people stand out in the category concerned with public employment agencies and to a lesser degree, the use of informal contacts and job offers.

5. In table 14, note the option `head()` that limits the columns to be shown. In this concrete case, `search9` (waiting offers) and `search10` (waiting results) are not in `head()`, but they are in rows to avoid a table that is too wide.

Table 14. Table of coincidences of ways of looking for jobs (vertical percentages)

```
. coin young-older search*, vertical pnormalized minimum(.15)
> head(young-older search1-search6) width(12)
1000 scenarios. 79 probable coincidences amongst 16 events. Density: 0.66
11 events(n>=150): young adult older search1 search2 search3 search4 search5
> search6 search9 search10
(40 probable coincidences amongst 11 selected events. Density: 0.73)
Col. percent |  young  adult  older  se~h1  se~h2  se~h3  sea~4  sea~5  sea~6
-------------+---------------------------------------------------------------
Age          |
       young |  100.0    0.0    0.0   37.3   44.2   45.1   40.4   45.4   46.5
       adult |    0.0  100.0    0.0   40.2   42.3   38.0   40.0   41.7   39.2
       older |    0.0    0.0  100.0   22.5   13.5   17.0   19.6   12.8   14.3
-------------+---------------------------------------------------------------
Agency       |
      public |   49.3   56.6   67.2  100.0   66.1   56.4   57.2   56.0   56.8
     private |   32.6   33.2   22.6   36.9  100.0   34.8   32.4   43.0   36.3
-------------+---------------------------------------------------------------
Contacts     |
   employers |   86.0   77.2   73.1   81.4   90.0  100.0   83.8   90.6   86.7
    informal |   83.3   87.8   91.4   89.2   90.6   90.6  100.0   90.9   90.2
-------------+---------------------------------------------------------------
Ads          |
     placing |   43.8   42.9   28.0   40.9   56.1   45.8   42.5  100.0   57.1
  looking at |   78.1   70.3   54.3   72.3   82.6   76.4   73.6   99.5  100.0
-------------+---------------------------------------------------------------
Waiting      |
      offers |   29.0   30.2   39.8   41.8   38.1   32.7   33.6   34.8   32.9
     results |   27.6   24.6   19.9   26.7   33.9   28.3   26.4   37.5   30.3
p. Haberman  |  young  adult  older  se~h1  se~h2  se~h3  sea~4  sea~5  sea~6
-------------+---------------------------------------------------------------
Age          |
       young |   0.00   1.00   1.00   1.00   0.17   0.00   0.99   0.03   0.00
       adult |   1.00   0.00   1.00   0.29   0.11   0.97   0.18   0.11   0.57
       older |   1.00   1.00   0.00   0.00   1.00   1.00   0.02   1.00   1.00
-------------+---------------------------------------------------------------
Agency       |
      public |   1.00   0.29   0.00   0.00   0.00   0.12   0.00   0.39   0.10
     private |   0.17   0.11   1.00   0.00   0.00   0.00   0.01   0.00   0.00
-------------+---------------------------------------------------------------
Contacts     |
   employers |   0.00   0.97   1.00   0.12   0.00   0.00   0.00   0.00   0.00
    informal |   0.99   0.18   0.02   0.00   0.01   0.00   0.00   0.00   0.00
-------------+---------------------------------------------------------------
Ads          |
     placing |   0.03   0.11   1.00   0.39   0.00   0.00   0.00   0.00   0.00
  looking at |   0.00   0.57   1.00   0.10   0.00   0.00   0.00   0.00   0.00
-------------+---------------------------------------------------------------
Waiting      |
      offers |   0.92   0.76   0.00   0.00   0.00   0.05   0.00   0.03   0.08
     results |   0.05   0.59   0.96   0.09   0.00   0.00   0.00   0.00   0.00
```

As was the case in all previous examples, these quantities can be transferred to a network graph showing the frequencies of each of the categories as well as the coincidences or relationships between them. In this way, it can be observed by examining the age categories represented by empty triangles that the group of adults does not have any statistically significant relationship with the mode of job search, while the categories of young and old do have such a relationship with three different search strategies (see figure 5).

```
coin young-older search* using searchc, minimum(.02)  ///
      graph(mds) levels(.05 .01 .001) form(type) ///
      goptions(title("Ways of looking for jobs") ///
      caption("Source: INE(2013)") note("MDS coordinates"))
```

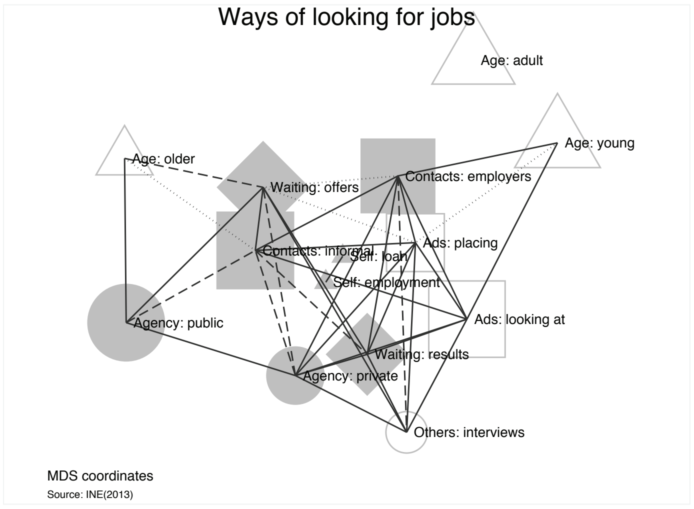

Figure 5. Links between age and different ways of looking for jobs

6 Performance

To test the performance of `coin`, a random dataset was built with 100,000 cases and 500 events with a density of 0.50 between them. Tests were done with 10, 50, 100, 200, 300, and 500 events crossed with 100, 1,000, and 100,000 scenarios.[6] Short execution times were obtained with experiments using up to 100 events. Simple tasks exhibit fast performance. For example, it took just 60 seconds to get all cross frequencies of 500 events with $n = 100000$ scenarios (see figure 6).

6. These tests were performed in Stata 14 on an iMac (OS X 10.10.3) with an Intel Core i7 2.93 GHz and 8 GB of DDR3-1.333 MHz memory.

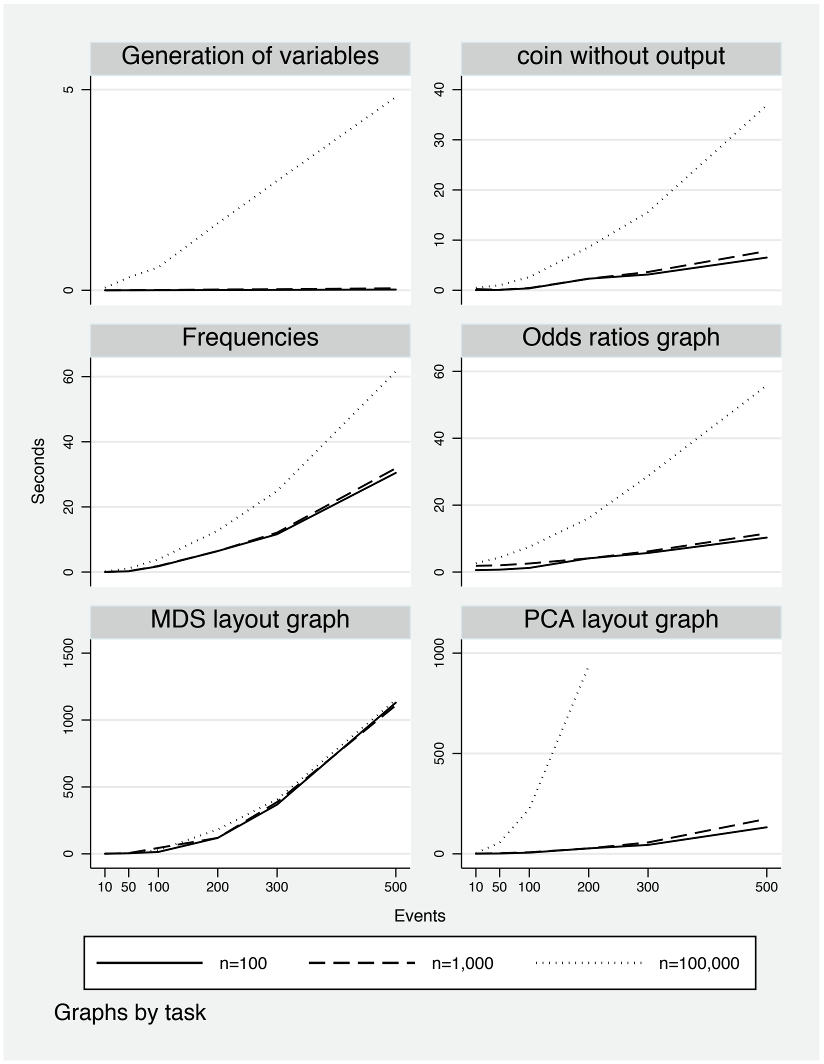

Figure 6. Computation time by task, number of events, and sample size (scenarios)

`coin` is not intended for large numbers of events. When 100 variables or events were analyzed with a large number of records, only slow results were present in the PCA graph case and in obtaining all statistics. This is because of the slow calculation of tetrachoric correlations when there are many variables and many cases. Execution was also slow, though less slow, in graphs using the MDS layout: 500 events needed more than 1,000 seconds to be represented regardless of the size of the database. In addition, the `coin` command is unable to perform correspondence layouts when there are more than 11,000 observations because of limitation of Stata matrices.

7 Conclusions

Coincidence analysis helps discover patterns of concurrence of a series of events in a given set of scenarios. Its aim is to discover how characteristics are jointly distributed in different units where they may or may not be present.

Three degrees of coincidences can be distinguished: simple, probable, and statistically probable. To decide which degree of coincidence two particular events present, a frequentialist and probabilistic approach is used. Mere frequency and statistical tests such as those of Haberman or Fisher are used to decide which events are coincident in a set of scenarios. The `coin` command permits coincidences to be obtained.

It is important not only to represent the coincidences by an adjacency matrix but also by a good quality graphical representation, which may help the user better understand the distribution of concurrences of a set of multiple events. Using `coin`, coincidences can be represented with four different types of graphs:

- Bar graphs: Their main advantage is simplicity, but their drawback is that they generate a graph for each event, so they are not parsimonious in the study of coincidences.
- Residuals graphs: They are also specific graphics for each event. These graphs are useful for the differentiation of types in coincidences that affect each particular event.
- Dendrograms: These plots join all the coincidences in one graph, but their disadvantage is that the agglomeration modes, as well as the multiple distance measures available, can distort the joint study of pairs of possible coincidences.
- Network graphs: They can represent not only the incidence of the events but also their level of coincidence and even the characteristics of the events under consideration.

Although there are different ways to locate the events in the two-dimensional space of graphical network representations, they all present similar and noncontradictory information through the size of nodes and the links between them. In any case, representation by MDS is recommended because of the simplicity of its geodesic distances and its representation of the events with the highest degree in the center of the graph.

`coin` is also able to display many statistical measures to treat multiresponse in questionnaires easily, to manage bipartite networks, and to discover patterns of co-occurrences among events. These measures are frequencies and percentages, odds ratios and correlations, significance tests, space coordinates and distances between events, and centrality indicators.

8 Acknowledgments

This article has been financed by the Spanish Ministry of Economy and Competitiveness and the European Regional Development Fund (FEDER) (research project number CSO2013-49278-EXP). I thank Susana Borrás, Antonio M. Jaime, Isidoro Maya, Oded Mcdossi, Enrique Pinzón, Félix Requena, Ignacio Sánchez-Cuenca, Alexander Zlotnik, and an anonymous referee for useful comments to this article.

9 References

Borgatti, S. P., M. G. Everett, and L. C. Freeman. 2002. UCINET for Windows: Software for Social Network Analysis. Harvard, MA: Analytic Technologies. http://www.analytictech.com/ucinet/.

Corten, R. 2011. Visualization of social networks in Stata using multidimensional scaling. *Stata Journal* 11: 52–63.

Cox, N. J., and U. Kohler. 2003. Speaking Stata: On structure and shape: The case of multiple responses. *Stata Journal* 3: 81–99.

De Nooy, W., A. Mrvar, and V. Batagelj. 2011. *Exploratory Social Network Analysis with Pajek*. Revised and expanded second ed. New York: Cambridge University Press.

Escobar, M. 2009. Redes semánticas en textos periodísticos: Propuestas técnicas para su representación. *Empiria* 17: 13–39.

Everitt, B. 2003. *The Cambridge Dictionary of Statistics*. Cambridge: Cambridge University Press.

Gabriel, K. R. 1971. The biplot graphic display of matrices with application to principal component analysis. *Biometrika* 58: 453–467.

Galaskiewicz, J., and S. Wasserman. 1989. Mimetic processes within an interorganizational field: An empirical test. *Administrative Science Quarterly* 34: 454–479.

Gower, J. C. 1985. Measures of similarity, dissimilarity, and distance. In Vol. 5 of *Encyclopedia of Statistical Sciences*, ed. S. Kotz, N. L. Johnson, and C. B. Read, 397–405. New York: Wiley.

Grund, T. E. 2015. nwcommands: Software tools for statistical modeling of network data in Stata. http://nwcommands.org.

Haberman, S. J. 1973. The analysis of residuals in cross-classified tables. *Biometrics* 29: 205–220.

Hofmann, T., and J. Puzicha. 1998. Statistical models for co-occurrence data. Memo, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, A.I. Memo No. 1625. ftp://publications.ai.mit.edu/ai-publications/1500-1999/AIM-1625.ps.

Hubálek, Z. 1982. Coefficients of association and similarity, based on binary (presence–absence) data: An evaluation. *Biological Reviews* 57: 669–689.

Jann, B. 2005. Tabulation of multiple responses. *Stata Journal* 5: 92–122.

Leydesdorff, L., and L. Vaughan. 2006. Co-occurrence matrices and their applications in information science: Extending ACA to the web environment. *Journal of the American Society for Information Science and Technology* 57: 1616–1628.

Miura, H. 2012. Stata graph library for network analysis. *Stata Journal* 12: 94–129.

Padgett, J. F., and C. K. Ansell. 1993. Robust action and the rise of the Medici, 1400–1434. *American Journal of Sociology* 98: 1259–1319.

Provalis Research. 2015. WordStat for Stata. http://provalisresearch.com/products/content-analysis-software/wordstat-for-stata/.

StataCorp. 2015. *Stata 14 Multivariate Reference Manual*. College Station, TX: Stata Press.

Tumminello, M., S. Miccichè, F. Lillo, J. Piilo, and R. N. Mantegna. 2011. Statistically validated networks in bipartite complex systems. *PLOS ONE* 6: e17994.

About the author

Modesto Escobar is professor of sociology at the Universidad de Salamanca, where he focuses on self-identity, social research techniques, and statistical analysis of data.