Manual for PAJ2PPD (v0.2)

Written by Wouter de Nooy, January 2011.

This manual discusses the data preparation and the operation of PAJ2PPD software. PAJ2PPD transforms longitudinal network data in Pajek project file format (extension .paj) to pair-period data (PPD) format for multilevel (logistic) regression analysis, e.g., for estimating a discrete-time event history model (see de Nooy, 2011). The data format is pair by period – one record for each arc (that could exist) at each time point – which is similar to the person-period data matrix used in discrete event history analysis with persons as units of observation.
PAJ2PPD is a Windows program written in Borland Delphi.

1 Data preparation

PAJ2PPD needs a Pajek project file (extension *.paj) as its input. One longitudinal network representing a series of events or actions is the core of the input file. I refer to this network as the *events network*. Additionally, the project file may contain attributes of the vertices, stored as partitions or vectors, additional networks to be used as network covariates, (small) networks representing fragments (*fragment network*) that can be identified in the events network, and partitions labelling vertices in fragment networks.

1.1 The occurrence and contents of relational events

Simple regression models predict the type or valence of events assuming that events happened. Event history models predict the occurrence of events and possibly their type or valence. PAJ2PPD can create data matrices for both types of analyses if the event is relational, that is, it involves a pair of actors. The goal is to explain or predict the occurrence and/or contents of new arcs in a network. The arc (ordered pairs) is the event. The line value of the arc is the value of the outcome variable (*y*) in the PPD data matrix. For right-censored cases – ordered pairs for which the event has not (yet) happened – the outcome variable is set to zero.

Note that arcs with zero line value in the events network are regarded as events in PAJ2PPD. Cases are constructed for the time point at which these arcs occur (within limits set by the user, see Section 2.3) and they are removed from the risk set (if this option is set by the user, see Section 2.4). In the event history analysis, however, they are erroneously treated as right-censored cases because the outcome variable is zero. Therefore, avoid using zero line values or correct the PPD file afterwards. For complications in detecting local network context arising from zero-valued lines in the events network, see Section 1.6.

1.2 Time indication in the events network

The events network must use Pajek’s time notation for time intervals (not time events). Each vertex and each arc is followed by a time indicator between square brackets containing one or more nonnegative integers representing time moments, separated by comma's or dashes (indicating all time moments in between). A star (*) represents (positive) infinity. See the digital Pajek manual or (de Nooy, Mrvar, & Batagelj, 2005Section 4.5) for more information. An example:

```
*Vertices 12
1 "Vertex 1"    0.1000    0.5000    0.5000 [1-5]
…
*Arcs
1  2  1  [1]
1  4 -1  [2-4]
…
```

Normally, the events network is directed: a particular actor acts towards another actor at a particular moment. Undirected lines (edges) are replaced by double arcs.

The arcs are assumed to represent actions or events, taking place at one moment in time. Therefore, the first time moment in an arc's time indicator only is taken into account. In the example above, the negative arc from vertex 1 to 4 will be considered an event happening only at time 2.

Pajek cannot handle date notation (e.g. mm/dd/yy). If the original data contain dates, they must be transformed into a series of nonnegative integers, e.g., by calculating the number of days (weeks, months, years, hours, minutes, seconds, etc.) since a starting time.

The software assumes that observation (data collection) starts at time one. If there are no arcs at time one, it is assumed that observation (data collection) did not identify arcs at that time. The time of the last arc in the network is assumed to represent the last time for which data have been collected. If this is not correct – later times have been observed but no events (arcs) were found – add an arc to the network with a neutral line value (usually zero) dated at the last observed time point.

1.3 Indication of episodes in the events network

In event history analysis, events are part of an episode. The goal of the analysis is to predict or explain whether an event happens and at which moment it happens within the episode; what is the duration between the start of the episode and the occurrence of the event? Actual events to be predicted or explained by the event history model, must belong to an episode to know from which moment onwards it can be expected to happen. In addition, a multi-episode design may explore systematic differences between episodes with respect to the timing of events.

In the Pajek network file, episodes are identified by line labels: each episode must have a unique name (string of characters), which is stored as a label to each line belonging to the episode. Even if the data set consists of only one episode, an episode name must be stored in the line labels. For instance, the line label `l "episode A"` in the example below:

```
*Arcs :0 "Start moment = 0, episode A or Z"
2        1 1 l "episode A" [1]
```

```
4        2 1 l "episode Z" [3]
3        2 1 [1]
```

The starting time of each episode must be known. This time is stored as the relation number of the lines. In other words, the number of the relation of a line indicates the start time of its episode provided that the line belongs to an episode (that is, it has a line label). Conventionally in event history analysis, the starting time of an episode is the last moment after which events or action can take place. An episode's starting time (relation number in Pajek) *must be lower* than the time of the event occurring first.

Example of Pajek data format above: `*Arcs :0` indicates that the relation number of all arcs that follow until the next arcs statement belong to relation 0, which means that their episodes start at time 0. In this example, two episodes start at time 0: episodes A and Z. For episode A, an arc is observed at time 1, directly after the start of the episode. For episode Z, an arc happens at time 3. The arc from vertex 3 to vertex 2 does not have a line label, so it does not belong to an episode. By default, arcs that do not belong to episodes – they only serve as network context – are listed under relation number zero.

The example shows that not all lines have to belong to episodes. The event history analysis does not predict the occurrence of these lines (their line values will not be shown in the outcome variable *y*) but they are used as part of the network context. In other words, lines outside episodes are not predicted in the event history model but they can help the prediction of other lines because they are part of their local network context.

Because relation numbers are used to identify the start time of episodes, it is not possible to distinguish between different types of relations in the events network within PAJ2PPD. In other words, it is not possible to add a discrete attribute to the lines in the events network (other than the episode's start time). To predict lines for different relations, one must prepare a separate data file for each relation, process them separately in PAJ2PPD and merge the resulting cases (data matrices) in external software, adding a marker for each relation.

In Pajek, lines can have markers for width, colour, and etcetera. PAJ2PPD can read these markers but it does not use them at present.

1.4 Attributes of vertices in the events network

Characteristics of vertices in the events network can be used to create covariates for the tail or head (activity, attractiveness) of the (potential) line or for dyadic covariates (similarity, dissimilarity, ranking under, ranking over). In Pajek, characteristics of vertices are stored as partitions (discrete attributes) or vectors (continuous attributes). Partitions and vectors that are included in the Pajek project file can be used in PAJ2PPD to create covariates. PAJ2PPD uses a maximum of 4 decimal places for vector values.

Missing (absent) values in a partition or vector are automatically replaced by 9999999. Value 9999998, which is the other Pajek default missing value, is also treated as a missing value. If at least one of the vertices has a missing value, the dyadic covariate receives a missing value (9999998 or 9999999). So it is possible to use attributes of the vertices (variables) that have unknown values but covariates using these attributes will yield missing values for part of the cases.

Partitions and vectors do not have time stamps in Pajek. For time-varying covariates (TVCs) a partition or vector is needed for every subperiod with unique attribute values. For ease of recognition, it is recommended to add the time point(s) to the partition's or vector's name. Example: two partitions covering the entire period (time 1 to 5) with changes between time 3 and time 4 (vertex 4 changes from unknown to 8, vertex 5 changes from 6 to 7):

```
*Partition Attribute 1 (time 1-3)
*Vertices 6
6
6
7
9999998
6
7

*Partition Attribute 1 (time 4-5)
*Vertices 6
6
6
7
8
7
7
```

1.5 Networks as covariates

A network on another relation can function as a dyadic covariate: does the presence of a line in a particular relation help to predict the occurrence of a line in the events network? I will refer to this network as a *covariate network*.

PAJ2PPD will only check whether a line is present at the time an event (may) take(s) place; it won't look back (no lags). As a consequence, relations that are durable are better for predicting the events in the events network than networks of incidental relations.

Directed and undirected (or mixed) covariate networks can be used. Edges are treated as double arcs.

Static and dynamic networks can be used. Static networks do not need time stamps (absence of a time stamp is deemed to mean that a vertex or line is always present) but they can have them. Use the interval notation discussed for the events network (Section 1.1).

In case of multiple lines (at the same time) in the covariate network, the first line encountered is used. Therefore, it is better to make sure that there are no multiple lines (at the same time) in the covariate network.

1.6 Networks as fragments

Effects of network context on the appearance of an arc in the events network include reciprocity, transitivity, balance (in signed networks), and any other network effect that predicts a higher probability of an arc appearing in one structural context rather than in another.

To accommodate for all kinds of local network contexts, the pattern of local network context is not hard-coded in the program. The user can and must create prototypical instances of the local network pattern of interest. In Pajek, these instances are called fragments and they are (usually small) networks themselves. I will call them *fragments* or *fragment networks* here. They should be added to the Pajek project file as networks (see below).

In contrast to fragment searching in Pajek, PAJ2PPD assumes that vertex 1 in the fragment network represents the tail of the (potential) arc to be predicted in the events network and vertex 2 represents its head. PAJ2PPD creates a case, that is, an ordered pair, for each time at which an arc may appear within the pair or actually appears. I will refer to these pairs or cases as *eligible pairs*. It then counts the number of fragments in the network of preceding events to which the eligible pair belongs. The network of preceding events is called the retrospective window, which is a moving window the length and offset of which can be set by the user.

Example: for testing reciprocity (of a positive arc), we need a fragment consisting of two vertices with one (positive) arc pointing from vertex 2 to vertex 1. For each ordered pair that could experience action at a particular time, the count of this fragment will yield the number of (positive) arcs pointing in the opposite direction in the retrospective window of the events network. If arcs are more likely to occur when they reciprocate previous arcs, the data matrix will contain relatively many cases combining presence (coded as 1) of an arc with presence (counted as 1 or a higher integer) of an arc in the opposite direction.

Sometimes, particular arcs are not allowed in the fragment network, for instance, one may hypothesize a tendency to close hitherto intransitive triads. In this example, a fragment network is needed with arcs from vertex 1 to vertex 3 and from vertex 3 to vertex 2 and an indication that there should *not* be a direct arc from vertex 1 to vertex 2. This is accomplished by an arc with line value zero in a fragment network, from vertex 1 to vertex 2 in this example.

A zero-valued arc in the fragment network from vertex *u* to vertex *v* means that local event network context can match the fragment only if there is no arc whatsoever from vertex *x* to vertex *y* in the retrospective window on the events network (vertex *x* matching vertex *u* and vertex *y* matching *v*), regardless of the arc's value in the events network (positive, negative, or zero). The options for ignoring sign, value, and relation are not effective here because any arc is forbidden. If the option for ignoring the direction of the line is set, both an arc from *x* to *y* and an arc from *y* to *x* are forbidden.

There is an important limitation to the use of zero-valued lines in a fragment: each vertex in the fragment must be connected to vertex 1 or 2 by a semipath without zero-valued lines. Fragments may contain lines with value zero, which indicate that in the network a line is not allowed, but they may only occur between vertices that are also connected by nonzero semipaths to vertex 1 and/or vertex 2. This restriction is needed for the fragment detection algorithm. PAJ2PPD issues a warning when it encounters a fragment that does not satisfy this criterion.

Note that the events network may also contain lines with line value zero, e.g., representing acts that are neither positive nor negative. A zero-valued line in an events network therefore means something completely different from a zero-valued line in a fragment network. A zero-valued line in the events network represents an event that has no sign and no value. Nevertheless, it represents an event. As a consequence, a zero-valued line in the events network is matching a line in the fragment network if and only if fragment search ignores the sign and value of the lines and the line in the fragment is not zero-valued.

Using zero-valued lines in the fragment to indicate forbidden lines in the events network, we cannot include zero-valued lines in the fragment that indicate the occurrence of an event that has no determinate meaning (sign or value).

The arcs in the fragment network may have particular attributes: a sign, a line value, and even a particular relation number. Detection of fragments in the local events network context can be constrained to arcs with exactly the same attributes as the arcs in the fragment. It is, for instance, necessary to take into account the sign of arcs when detecting balance in signed networks. In addition, the user can specify whether or not the lines in the event network should have the same direction as the fragment. Relation numbers can be used to constrain fragment matching only in very special cases because the relation numbers in the events network represent the starting time of the episode.

The option to ignore (not to check) line sign, direction, or value can be very helpful to reduce the number of fragments needed to cover a particular effect of local structure. For instance, finding all semipaths of a particular length between two vertices requires just one fragment, e.g., with a sequence of positive lines starting at vertex1 and ending at vertex2, provided that the direction, sign, and value (and relation number) of the lines are not checked when constructing the local network context covariate (see Section 2.8)

Many structural effects require more than one fragment because they include several prototypical instances. For example, the popularity effect, that is, the indegree of the head of the eligible pair in the network context, requires a fragment with a direct arc from vertex 1 to vertex 2 and a fragment containing an arc from a third vertex to vertex 2. The first fragment determines whether there is an arc incoming from the tail vertex itself whereas the second fragment counts all incoming arcs from other vertices (excluding loops from vertex 2 itself). The total indegree of the eligible pair’s head is the sum of the two fragment counts. Both fragments must be included in the Pajek project file. Their counts are automatically summed in PAJ2PPD if they are linked to the same covariate name (see Section 2.8).

Sets of fragments measuring particular structural tendencies can be stored in a Pajek project files for easy reuse, which can be attached to the events network data by simply

reading the project file after the events network data have been opened in Pajek. Finally, save all data as a new Pajek project file and open this file in PAJ2PPD.
Examples:

- Fragments.activity.paj contains the two fragments required to count the outdegree of the tail vertex (sender) in the retrospective window.
- Fragments.popularity.paj contains the two fragments required to count the indegree of the head vertex (receiver) in the retrospective window.
- Fragments.balance.positive.paj contains the fragments counting the number of balanced semicycles (up to length four) in the retrospective window produced by a positive arc from tail to head vertex. Note: do not check the direction, line value, and relation number, but check the sign of the lines while detecting fragments.
- Fragments.balance.negative.paj contains the fragments counting the number of balanced semicycles (up to length four) in the retrospective window produced by a positive arc from tail to head vertex. Note: do not check the direction, line value, and relation number, but check the sign of the lines while detecting fragments.

Note that one ordinarily subtracts the number of balanced fragments arising with a negative line between sender and receiver from the number of balanced fragments created by a positive line to obtain the prevalence of balance with a positive line over a negative line. The prevalence variable requires different sets of fragments, e.g., one set of fragments favouring a positive action from vertex 1 to 2 and one set favouring a negative action. The preponderance count must be calculated in a statistical package before applying the discrete-time events history model.

Splitting up a local network effect into two or more fragments, one must be careful not to use homomorphic fragments, that is, fragments that identify the same local network structure. Homomorphic fragments yield double counts. The risk is especially high if fragments are counted with the option not to check line sign or direction. For example, a fragment containing a positive arc from vertex 1 to vertex 3 and a positive arc from vertex 3 to vertex 2 (predicting a positive arc from vertex 1 to vertex 2 under balance), counts exactly the same local network configurations as a fragment with the arcs in the opposite direction if the direction of lines is not checked.

A complication arises with multiple arcs within the same ordered pair in the fragment network. PAJ2PPD can detect fragments correctly if all arcs within the same ordered pair have different line values in the fragment network. One instance of the fragment is counted if each of the arcs occurs exactly once in the events network provided that the option ‘Check value of lines’ is selected and the ‘Just use last arc’ option is not selected (multiple lines in the fragment require considering multiple lines in the events network). If the events network contains multiple arcs for any of the arcs in the fragment, a fragment is counted for every combination of arcs found. The order in which multiple arcs appear in the events network is not taken into consideration.

Do **not** use fragments with multiple arcs that have the same value unless they differ with respect to their relation number. Note that relation numbers express episodes’ starting times, so it is usually meaningless to include relation number in the fragment detection process.

1.7 Networks and partitions as labelled fragments

It is possible to take into account characteristics of the event network's vertices when detecting fragments. Think, for example, of the brokerage roles defined by Gould and Fernandez (Gould & Fernandez, 1989) as two-paths with different patterns of group membership of the three vertices involved. One may want to test whether two members of the same group are more likely to start interacting directly if they are linked by a peer from the same group (a coordinator role) than by someone from another group (an itinerant broker).

Counting the number of coordinators and the number of itinerant brokers between two actors in the retrospective window on the events network requires both a partition on the events network identifying the empirical group affiliations of the actors and a partition on the fragment network identifying the pattern of group membership associated with a particular role. The coordinator fragment, for example, would need a partition assigning all three vertices to the same class whereas the itinerant broker fragment requires a partition with vertices 1 and 2 belonging to one group and vertex 3 to another group.

Note that labelled fragments are needed if more than two vertices are involved. If the attributes are relevant only of the first two vertices, that is, the sender and receiver of the arc that is to be explained, the attribute covariates discussed in Section 1.4 can be used.

All restrictions and caveats on specifying fragments (Section 1.6) apply.

In the current implementation, fragment partition clusters numbered 1 and up identify different groups. Note that these numbers do not indicate particular groups, e.g., fragment partition cluster number 1 does not require that a vertex is a member of the group coded 1 in the events network partition. It merely indicates that if there is another vertex in the fragment labelled 1, it must also be in the same events network cluster and any vertex assigned to another cluster in the fragment must be in a different cluster in the events network partition. In other words, it is possible to code which vertices must belong to the same or different clusters but it is not possible to code to which particular events network cluster an actor must belong.

Fragment partition cluster 0 has a special meaning. It indicates that the events network cluster does not matter for this vertex ('never mind'). Note that the events network partition cluster coded 0 has no special meaning, it is supposed to indicate a cluster just like any other code (except missing values, see below).

Missing values (9999998 and 9999999) are not allowed in the fragment partition. They are allowed in the events network partition. A labelled fragment is not identified if any of the vertices has a missing value on the events network partition unless it is assigned to cluster 0 in the fragment partition.

Time-varying attributes (TVCs) are allowed for the events network, not for the fragment. As described in Section 1.4, two or more partitions can be selected for the events network that cover different subperiods. PAJ2PPD uses the events network partition that is valid at the time for which the arc must be predicted. In other words, actors are assumed to take into account the current group membership of vertices in the local network context, even if they belonged to a different group at the time they acted.

Note that the first partition is used if no other partition contains the time of the predicted arc.

1.8 Restrictions on eligible pairs

Not all vertices in the events network may be able to act or be acted upon at all times. Temporary presence or absence of vertices is coded by the time indicators in the list of vertices of the events network.

Vertices that are present but that cannot act and/or cannot be acted upon, can be identified by partitions provided that this classification is valid for the entire period. In an affiliation network, for instance, persons may be the only ones that act and the organizations the only ones that are acted upon.

One class in a partition identifies the vertices that can act and one class in another partition identifies the vertices that can be acted upon. The two sets may overlap: some vertices can only act, some can only be acted upon, some can be actors and acted upon, and some cannot act or be acted upon. One or two partitions can be used.

In addition, empirical situations may occur where just one vertex can act or can be acted upon (or both) within an episode. In the case of birthday invitations, for instance, each birthday defines an episode and each episode only has one person sending invitations. To accommodate for this type of restriction, no additional data are needed in the Pajek project file because this restriction can be set in the user interface (see Section 2.5).

2 Operating PAJ2PPD

PAJ2PPD is started by double-clicking PAJ2PPD.exe or by selecting this file with the Windows Run command. In the program's main screen (Figure 1), several choices must be made before the PPD dataset can be produced. Each step (numbered 1 – 7) is discussed in a Section (2.2 – 2.8). First, however, a network file must be opened (Section 2.1).

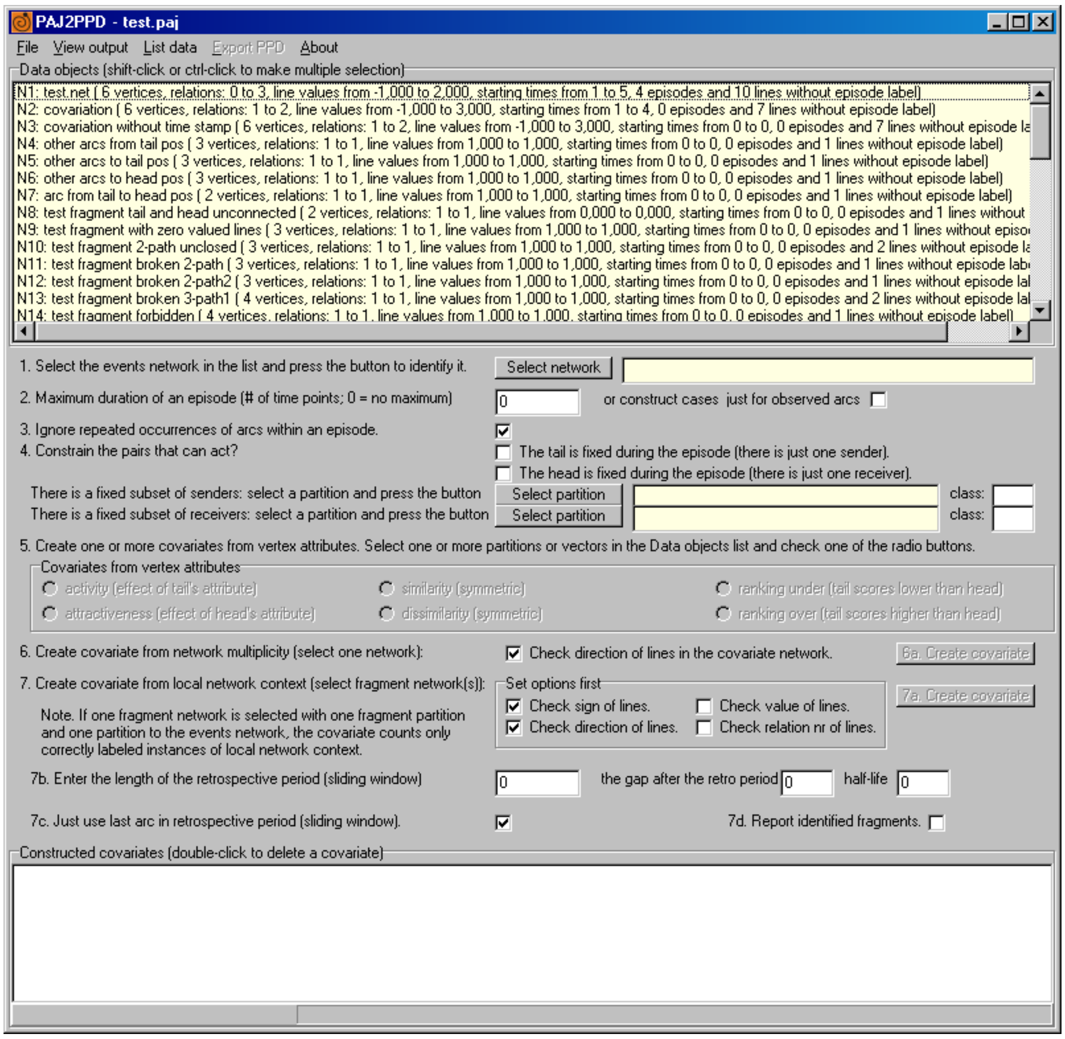
2.1 Open Pajek project file

First, open the Pajek project file containing the data.

File>Open Pajek project file

Use the *File>Open Pajek project file* command and select the prepared Pajek project file, which must have the filename extension .paj.

If the program encounters missing (absent) values in the partitions or vectors, it issues a warning. Mistakes in the networks, partitions, or vectors may cause the program to issue warnings or error messages and stop reading the project file. If the project file was saved from Pajek, errors should not occur.



Data objects (shift-click or ctrl-click to make multiple selection)

```
N1: test.net ( 6 vertices, relations: 0 to 3, line values from -1,000 to 2,000, starting times from 1 to 5, 4 episodes and 10 lines without episode label)
N2: covariation ( 6 vertices, relations: 1 to 2, line values from -1,000 to 3,000, starting times from 1 to 4, 0 episodes and 7 lines without episode label)
N3: covariation without time stamp ( 6 vertices, relations: 1 to 2, line values from -1,000 to 3,000, starting times from 0 to 0, 0 episodes and 7 lines without episode la
N4: other arcs from tail pos ( 3 vertices, relations: 1 to 1, line values from 1,000 to 1,000, starting times from 0 to 0, 0 episodes and 1 lines without episode label)
N5: other arcs to tail pos ( 3 vertices, relations: 1 to 1, line values from 1,000 to 1,000, starting times from 0 to 0, 0 episodes and 1 lines without episode label)
N6: other arcs to head pos ( 3 vertices, relations: 1 to 1, line values from 1,000 to 1,000, starting times from 0 to 0, 0 episodes and 1 lines without episode label)
N7: arc from tail to head pos ( 2 vertices, relations: 1 to 1, line values from 1,000 to 1,000, starting times from 0 to 0, 0 episodes and 1 lines without episode label)
N8: test fragment tail and head unconnected ( 2 vertices, relations: 1 to 1, line values from 0,000 to 0,000, starting times from 0 to 0, 0 episodes and 1 lines without
N9: test fragment with zero valued lines ( 3 vertices, relations: 1 to 1, line values from 1,000 to 1,000, starting times from 0 to 0, 0 episodes and 1 lines without episo
N10: test fragment 2-path unclosed ( 3 vertices, relations: 1 to 1, line values from 1,000 to 1,000, starting times from 0 to 0, 0 episodes and 2 lines without episode la
N11: test fragment broken 2-path ( 3 vertices, relations: 1 to 1, line values from 1,000 to 1,000, starting times from 0 to 0, 0 episodes and 1 lines without episode lab
N12: test fragment broken 2-path2 ( 3 vertices, relations: 1 to 1, line values from 1,000 to 1,000, starting times from 0 to 0, 0 episodes and 1 lines without episode lal
N13: test fragment broken 3-path1 ( 4 vertices, relations: 1 to 1, line values from 1,000 to 1,000, starting times from 0 to 0, 0 episodes and 2 lines without episode lal
N14: test fragment forbidden ( 4 vertices, relations: 1 to 1, line values from 1,000 to 1,000, starting times from 0 to 0, 0 episodes and 1 lines without episode label)
```

1. Select the events network in the list and press the button to identify it.

2. Maximum duration of an episode (# of time points; 0 = no maximum) 0 — or construct cases just for observed arcs ☐

3. Ignore repeated occurrences of arcs within an episode. ☑

4. Constrain the pairs that can act?
   - ☐ The tail is fixed during the episode (there is just one sender).
   - ☐ The head is fixed during the episode (there is just one receiver).

   There is a fixed subset of senders: select a partition and press the button — class:
   There is a fixed subset of receivers: select a partition and press the button — class:

5. Create one or more covariates from vertex attributes. Select one or more partitions or vectors in the Data objects list and check one of the radio buttons.

   Covariates from vertex attributes
   - ○ activity (effect of tail's attribute)
   - ○ attractiveness (effect of head's attribute)
   - ○ similarity (symmetric)
   - ○ dissimilarity (symmetric)
   - ○ ranking under (tail scores lower than head)
   - ○ ranking over (tail scores higher than head)

6. Create covariate from network multiplicity (select one network): ☑ Check direction of lines in the covariate network.

7. Create covariate from local network context (select fragment network(s)):

   Set options first
   - ☑ Check sign of lines.
   - ☐ Check value of lines.
   - ☑ Check direction of lines.
   - ☐ Check relation nr of lines.

   Note. If one fragment network is selected with one fragment partition and one partition to the events network, the covariate counts only correctly labeled instances of local network context.

   7b. Enter the length of the retrospective period (sliding window) 0 — the gap after the retro period 0 — half-life 0

   7c. Just use last arc in retrospective period (sliding window). ☑ — 7d. Report identified fragments. ☐

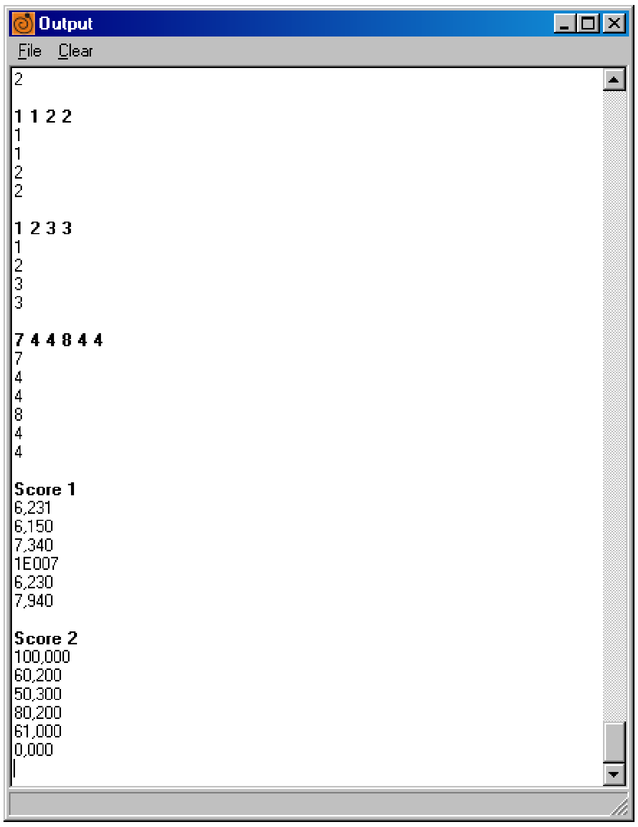
Constructed covariates (double-click to delete a covariate)

Figure 1 - PAJ2PPD main screen with opened Pajek project file.

After reading the data, a summary of the contents of the Pajek project file is displayed in the light-yellow window of the main screen (Figure 1). Each data object is concisely described by the number of vertices, the range of line values, cluster numbers, or vector values.

Main: List data
Report:File>Save
Report:File>Exit
Report:Clear
Main: View output

To inspect the data objects, use the *List data* command in the main menu. It will list the contents of all data objects in the report screen (Figure 2). In general, the report screen reports most of the choices and commands used. The report can be saved in rich text format (RTF) from the window's *File* menu. If it is closed (*File>exit*), it's contents are not lost. The contents can be removed with the *Clear* command. The window can be reopened with the *View output* command in the main screen.



```
2

1 1 2 2
1
1
2
2

1 2 3 3
1
2
3
3

7 4 4 8 4 4
7
4
4
8
4
4

Score 1
6,231
6,150
7,340
1E007
6,230
7,940

Score 2
100,000
60,200
50,300
80,200
61,000
0,000
```

Figure 2 - Report screen, showing the end of the output of the View command.

2.2 Step 1: Identify the events network.

Now, open the events network. Select the network in the list of data objects in the yellow window by clicking on it. This will turn the events network's background to dark blue.

'Select network' button

Then press the 'Select network' button. The events network's name is now displayed in the light-yellow box to the right of this button.

2.3 Step 2: Specify the maximum number of time points per episode

'or construct cases just for observed arcs' checkbox

Do you want to produce a dataset for discrete-time event history analysis, predicting timing and possibly the nature of events, or a dataset to predict just the latter? For the first option, cases must be created for all time points after the start of each episode. In contrast, the second option produces cases only for the events within episodes that have been observed. If the checkbox at step 2 (behind 'or construct cases just for observed arcs') is ticked, cases will only be produced for observed arcs with episode labels and the options at Step 4 are not applicable. Leave the checkbox empty to produce a full dataset for discrete-time event history analysis.

'Maximum duration of an episode'

If the dataset contains several episodes starting at different times, it may not always be sensible to assume that every episode extends to or beyond the time point for which the last observations are made. On substantive grounds, one may decide that no new events are to be expected within an episode after a certain period of time since its start. If this is the case, the number of time points per episode for which cases are created can be set to a maximum, which must be a nonnegative integer. If it is set to zero, no upper limit is set and cases are created for all time points up to and including the time of the last observation.

2.4 Step 3: Decide on repeating arcs within an episode

In event history models, events are usually assumed to happen only once within an episode. As a consequence, once an entity has experienced the event, it is no longer available as a candidate for the event. Because arcs represent the events in our case, we generally expect arcs to appear only once within an episode. If they do appear more than once, we may regard them as an extraordinary event, which cannot be predicted, e.g., if someone sends an invitation for a birthday twice by mistake.

In contrast, if we expect events to reappear, any pair is always a candidate for an event (arc). This assumption produces a different type of data set in which episodes may not have much meaning.

'Ignore repeated occurrences of arcs within an episode' checkbox

The 'Ignore repeated occurrences of arcs within an episode' checkbox determines the way repeated arcs are treated in the next run of the program. The default choice is here to ignore repeated arcs within an episode.

2.5 Step 4: Identify constraints on eligible tails and heads

If there are constraints on the pairs that can interact, they must be specified now. Note that these options are irrelevant (and inaccessible) if the checkbox is ticked to produce cases only for observed arcs at Step 2.

'The tail is fixed during the episode'
'The head is fixed during the episode'

The first option is to fix the sender and/or receiver of the arc within each episode. See the explanation at Section 0. Just check the appropriate boxes next to 'The tail is fixed during the episode (there is just one sender)' and/or 'The head is fixed during the episode (there is just one receiver)'.

'There is a fixed subset of senders'
'There is a fixed subset of receivers'

The second option is to use a partition to identify a subset of vertices that can act or can be acted upon during the entire period. Select a partition in the list of data objects (light-yellow window), press the 'Select partition' button next to 'There is a fixed subset of senders' or 'There is a fixed subset of receivers.' Finally, enter the number of the class identifying the vertices that can act or can be acted upon in the box(es) behind 'class:'. The same partition may be used to fix tails and heads.

2.6 Step 5: Create covariates from partitions and vectors

Partitions and vectors specify attributes of vertices. They can be used to create attribute-based covariates representing effects of characteristics of single vertices (activity and attractiveness), similarity of the sender's and receiver's attribute (similarity or homophily), and dissimilarity of sender and receiver on an attribute, including ranking.

Select partitions or vectors in the data objects list

Covariates from vertex attributes

Attributes of vertices may be constant during the period under investigation or they may change (Time-varying covariate, TVC). For time-constant attributes, select one partition or vector in the list of data objects. Then select one out of the six options for covariates from vertex attributes. On selection, a dialog box (Figure 3) appears asking for the name and optionally a description (label) for the covariate to be created. Note that the covariate name (and label) will be used in the exported data, so it must comply with the naming conventions of the statistical package you are going to use for analyzing the data.

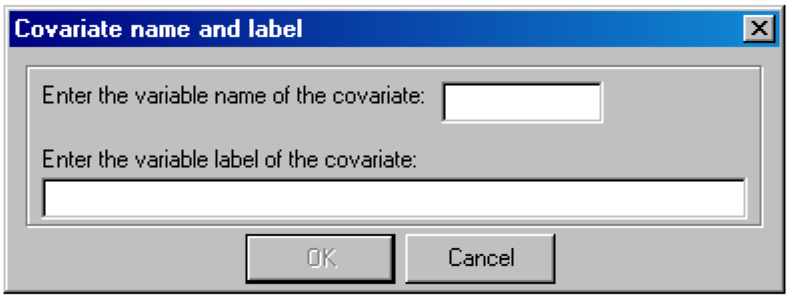
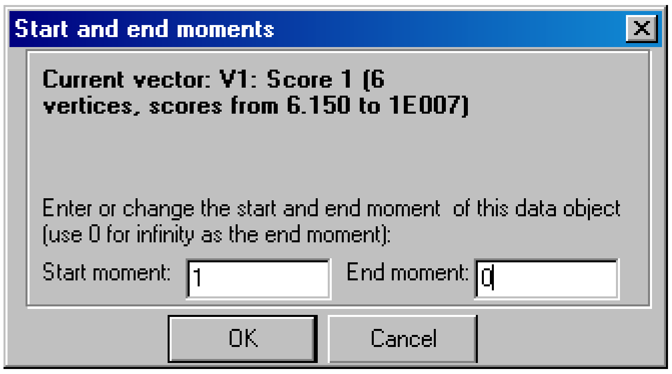

Covariate name and label
Enter the variable name of the covariate:
Enter the variable label of the covariate:
OK
Cancel

Figure 3 - Dialog box for naming and labelling a covariate.

Then, a new dialog box (Figure 4) appears, asking for the first and last moment at which this attribute is valid. Moments must be specified as integers that are consistent with the time indicators on the vertices and arcs in the events network. A time-constant attribute is valid for the entire period, so it is recommended to enter 1 as the start moment and 0 (meaning positive infinity) as the end moment. If start and end moments are chosen in a way that do not cover the entire period, no warning is issued. It is assumed that the attribute is only valid for the chosen moments, so it is set to missing (9999999) with other moments.



Current vector: V1: Score 1 (6 vertices, scores from 6.150 to 1E007)

Enter or change the start and end moment of this data object (use 0 for infinity as the end moment):

Start moment: 1 End moment: 0

Figure 4 - Dialog box for assigning the dates to a partition or vector.

After completion of this dialog box, the covariate is created (initialized) and added to the list of constructed covariates at the bottom of the main screen (Figure 1). Note that (TCC) stands for time-constant covariate.

With a time-varying covariate (TVC) the procedure is similar except that one has to select two or more partitions or vectors in the list of data objects before the type of covariate is selected in the list of radio buttons. Multiple data objects can be selected either by dragging the mouse over a series of data objects in the list while pressing the left mouse button, or by selecting one data object at a time, holding the Ctrl-key while left-clicking a data object.

The covariate naming dialog box appears (Figure 3) and afterwards the dialog box for assigning time appears for each selected data object. Note that this dialog box does not appear if a partition or vector was assigned a start and end moment before. The program assumes that vertex attributes are valid during the same period for each covariate created from them. Time assignment to data objects can not be undone within the program. One must reopen the original Pajek project file to reset everything.

Finally, be careful not to have overlapping periods for partitions or vectors with a TVC. The program will use the attribute value for the later period at moments in which periods overlap.

2.7 Step 6: Create covariates from other networks

Select a network in the data objects list

'6. Create covariate from network multiplicity' button

Lines in the events network may respond to lines on another relation. A covariate representing the presence, absence, or value of a line on another relation can be created by selecting a network with the same (number of) vertices in the data objects list and pressing the button '6. Create covariate.' The covariate naming dialog box (Figure 3) appears and afterwards the network multiplicity covariate is created and added to the list of covariates.

'Check direction of lines in the covariate network' checkbox

Checking the box 'Check direction of lines in the covariate network' before pressing the 'Create covariate network' button, ensures that lines in the opposite direction are not taken into account.

The covariate network may have time indicators on the vertices and lines. If so, the network multiplicity covariate contains the value of the matching line in this network only at times at which this line is present. Then, the covariate is time-varying. In contrast, if the covariate network does not contain time indicators (or indicators specifying the entire period for each vertex and line), a time-constant covariate is created assuming that each line is present during the entire period. We do not need to select more than one network to create a time-varying covariate – in contrast to TVCs based on partitions or vectors – because the covariate network may contain time indicators (whereas partitions and vectors do not).

2.8 Step 7: Create covariates for local network structure

Select fragments in the data objects list

Local network structure refers to the lines (events, actions) that occurred in a period preceding the current event in the events network. As explained in Section 1.6, the local structure of interest is captured by one or more fragments, that is, small networks with vertex 1 representing the tail of the current line under analysis and vertex 2 representing the head. Therefore, the first step consists of selecting one or more networks in the data objects list (see creating covariates from vertex attributes, Section 2.6). Usually, these networks are rather small, containing 2 to 4 vertices, because we are mainly interested in the local network context, viz., the two vertices involved and their (neighbours') neighbours. See Section 1.6 for restrictions on the networks that can be used as fragments.

If a labelled fragment is needed, at least two partitions must be selected in addition to the fragment network: exactly one partition matching the fragment network and one or more partitions matching the events network. If the selection does not meet these requirements or if the partition to the fragments network contains missing values (9999998 or 9999999), the program will issue an error later on and it will not create the covariate. Multiple data objects can be selected by selecting one data object at a time, holding the Ctrl-key while left-clicking a data object.

Set options first

Next, set the options for this covariate: should the sign, direction, value, or relation number of lines be checked while matching fragments to the events network? If a property is checked, the line in the events network must have the same property as the line in the fragments network for the fragment to be identified and counted.

Except for very special cases, the relation number should not be checked because this number represents the start moment of the episode to which the line belongs. In many situations, the exact line value should also not be checked but it is easy to think of situations in which you are interested specifically in lines with values that are either high (strong ties) or low (weak ties). Direction should not be checked if one is interested in semipaths or semicycles rather than paths and cycles. Finally, the sign of lines can be checked. Note that zero-valued lines in the events network are considered to have no sign, so they are taken into account if and only if fragment search does not check the sign and value of the lines (see Section 1.6).

'7a. Create covariate' button

Then press the '7a. Create covariate' button. The covariate naming dialog box (Figure 3) appears and the local network context covariate is added to the list of covariates. The type of covariate, variable name (varname), original fragment network names, and the covariate settings (T = true, F = False for each ignore option) are summarized in the list of covariates.

'7b. Enter the length of and gap after the retrospective window'

The length of the period preceding the event that is used as the retrospective window has not been set yet. This can be set in the two fields specified as step 7b in the main screen of the program. The first field requires a non-negative integer specifying the length of the retrospective window as a number of time steps. The time unit should match the time unit used for timing the vertices and lines in the events network. The second field specifies the length (a non-negative number of moments) of the gap between the end of the retrospective window and the current event. If it is impossible or unlikely that an actor reacts to events happening right before s/he acts, e.g., because s/he cannot yet perceive the event, one should introduce a gap.

The length of the retrospective window is primarily relevant to the calculation of local network context covariates. However, it also affects the output in a general way, viz., that it does not produce cases for the length of the retrospective window at the start of the period. In other words, cases are made only for moments which have a complete retrospective period within the collected dataset. If no local network context covariates are used, one should set the length and gap after the retrospective window to 0 to create data cases for all events.

'7b. half-life'

The third field specifies the half-life time of lines in the retrospective window. The value 0 indicates that a fragment's lines are not weighed according to the time span between them and the event for which they provide the local network context. Enter a positive real number (decimal separator is '.') to use a decay function which halves the weight of each fragment for the specified number of time units. Note that the oldest line in a fragment determines the timing of the fragment. For example, a half-time of '10' will halve the weight of a fragment for each period of ten time units. The weight (w) is calculated as: $w = .5^{\text{(current time – minimum time of the fragment's lines) / half-time}}$.

'7c. Just use last arc in retrospective window' checkbox

The next option specifies what to do with repeated lines in the retrospective window. Should the program treat each act or event within the same pair in the retrospective window separately, creating a fragment for each instance, or should it just consider the last line within a pair? If the '7c Just use last arc in retrospective window' checkbox is checked, the latter option is used. If it is not checked, the former option is in effect. Note that multiple arcs with the same value, relation number, and start moment are considered to represent the same arc in the fragment count.

'7d. Report identified fragments' checkbox

Finally, the user may request to list all identified fragments in the Report screen. If this checkbox is checked, one line is printed for each fragment indicating the covariate name, the time and arc (ordered pair) predicted as well as a list of the fragment's lines specifying its sender, receiver, and time of appearance. The fragments are sorted by time and covariate. Note that this option may produce long lists with potential memory overflow and significantly longer execution time if many fragments are found.

The settings of the length and gap of the retrospective window (7b) and whether all arcs in the retrospective window should be used or only the last (7c), determine the calculation of all local network covariates at the time the Export command is selected. It is impossible to specify different settings for different covariates in one run of the program because that would not make sense. As a consequence, options 7b and 7c may be (re)set before or after creating the local network covariates with button 7a.

2.9 Deleting covariates

Double-click a covariate in the list

Double-click a covariate in the list of constructed covariates if you want to remove it. It is not possible to change the covariate's definition.

Note that all covariates that are created must have different names. Covariates that mistakenly receive the same name yield only one covariate in the output. The last covariate with the name is actually created.

2.10 Create and export the data matrix

Export PPD

Select the Export PPD option in the menu of the main screen to create the pair-period data for analysis with statistical software. A dialog screen appears asking for the type and name of the file that should be saved. Currently, it is possible to save the data as an SPSS command file (*.sps) or as a tab-delimited text file (*.tab) with values or value labels. The first row of the tab-delimited text file contains the variable names; the variable labels are stored in the second line.

References

de Nooy, W. (2011). Networks of action and events over time. A multilevel discrete-time event history model for longitudinal network data. *Social Networks,* 33/1, 31-40. doi:DOI: 10.1016/j.socnet.2010.09.003

de Nooy, W., Mrvar, A., & Batagelj, V. (2005). *Exploratory Social Network Analysis with Pajek* (1st ed.). Cambridge: Cambridge University Press.

Gould, R. V., & Fernandez, R. M. (1989). Structures of mediation: A formal approach to brokerage in transaction networks. In *Sociological methodology 1990* (pp. 89-126). San Francisco: Jossey-Bass.