nwdynam

Dynamic network actor models in Stata
(a package that builds on `nwcommands`)

User manual and reference

Thomas U. Grund
RWTH Aachen University

This manual describes `nwdynam` 1.11.x
(examples produced with `nwdynam` 1.11.0, 21 November 2026)

How to cite. This manual describes the series 1.11 of `nwdynam` (it holds for all versions 1.11.x); the listings were produced with version 1.11.0 (21 November 2026). It is the citable reference for the package. Please cite it as:

Grund, T. U. (2026). *nwdynam: Dynamic network actor models in Stata. Manual.* Zenodo. https://doi.org/10.5281/zenodo.24398099

Please also report the exact version of the program that produced your results, as printed by `nwdynam version` and stored in `e(version)`.

Acknowledgments

`nwdynam` follows the `goldfish` project closely and exists because of it. The dynamic network actor model is due to Christoph Stadtfeld and Per Block [1], the coordination model to Stadtfeld, Hollway and Block [2], and `goldfish` (James Hollway, Christoph Stadtfeld, Marion Hoffman, Alvaro Uzaheta and contributors [3]) is the reference implementation on which the effects, the likelihoods, the defaults and the postestimation tools of `nwdynam` are modeled. The R source code and the documentation of `goldfish` 1.6.12 were the specification against which the Stata code was written and checked. The change point methods of `estat changepoints` are a port of the R package `changepoint` (Rebecca Killick, Kaylea Haynes, Idris Eckley, Paul Fearnhead and Jamie Lee [4]), which `goldfish` calls for the same purpose.

`nwdynam` is not affiliated with or endorsed by the `goldfish` project, and its errors are the author's. It is meant for researchers who work in Stata or prefer it to R. It is not a replacement for `goldfish`, which remains the reference.

Use of an AI coding assistant. The software was developed with Claude Code (Anthropic), an AI coding assistant, working under the author's direction. It helped port algorithms from `goldfish` into Stata, Mata and C and helped write the tests, and it helped draft this manual. The author specified the scope, reviewed the code, and is responsible for the package and for the manual. Its correctness rests on validation, not on the assistant: the results are compared numerically with `goldfish`, and the test suite reruns these comparisons.

License. `nwdynam` is free software under the GNU General Public License, version 3; Thomas U. Grund is its sole author.

Contents

Acknowledgments **2**

1 Introduction **6**
1.1 What `nwdynam` is . . . 6
1.2 Relation to `goldfish` . . . 6
1.3 What is covered . . . 7
1.4 Installation and requirements . . . 7
1.5 How to read this manual . . . 8
1.6 Data sets used in this manual . . . 8

2 The model **9**
2.1 Relational events and the state of the network . . . 9
2.2 The sub-models . . . 9
2.2.1 The choice sub-model . . . 9
2.2.2 The rate sub-model . . . 10
2.2.3 The coordination sub-model . . . 10
2.3 Events with identical times . . . 10
2.4 Estimation . . . 11
2.5 Reading the results . . . 11

3 Getting started **12**
3.1 The version . . . 12
3.2 The events . . . 12
3.3 Declaring the events as a network . . . 13
3.4 The choice and the rate sub-model . . . 14
3.5 Replay and stored results . . . 15

4 Data preparation **17**
4.1 Declaring the events . . . 17
4.2 Actors without events: the risk set . . . 17
4.3 Actor attributes . . . 18
4.4 Event values . . . 18
4.5 Undirected and two-mode events . . . 18
4.6 Composition change, changing covariates, opportunities . . . 19

5 Syntax and options **20**
5.1 Syntax . . . 20
5.2 General options . . . 20
5.3 Effect options and sub-models . . . 21
5.4 Options of the data features . . . 22
5.5 Stored results . . . 22
5.6 Return codes . . . 23

6 Effects **24**
6.1 Structural effects . . . 24
6.2 Closure effects . . . 24
6.3 Covariate effects . . . 25
6.4 Windows . . . 26
6.5 Weighted effects . . . 26
6.6 The coordination sub-model . . . 26
6.7 Two-mode networks . . . 27

7 Effects on other networks **28**
7.1 Which networks . . . 28
7.2 The terms . . . 28
7.3 `tie()` and `ignorerep()` . . . 29
7.4 What is not available . . . 29

8 Changing covariates, composition change and opportunities **30**
8.1 Covariates that change over time . . . 30
8.2 Composition change: actors who join or leave . . . 31
8.3 Opportunities: restricting the candidates . . . 31

9 Postestimation **33**
9.1 Information criteria . . . 33
9.2 The log likelihood of every event . . . 34
9.3 The probabilities of the events . . . 34
9.4 Outliers . . . 34
9.5 Change points . . . 35
9.6 Result plots: who is chosen and who acts . . . 35
9.6.1 estat choiceplot . . . 36
9.6.2 estat rateplot . . . 36
9.7 Graph options of the plots . . . 37
9.8 How the postestimation tools were checked . . . 38

10 Worked examples **40**
10.1 How the output was produced . . . 40
10.2 Effects and covariates . . . 40
10.3 Effects of other networks . . . 44
10.4 Changing covariates and composition change . . . 46
10.5 Opportunities . . . 47
10.6 Event values . . . 49
10.7 Two-mode events . . . 50
10.8 The coordination sub-model . . . 52
10.9 Events with identical times . . . 53
10.10 Postestimation . . . 54
10.11 Stored results . . . 60

11 Validation against goldfish **61**
11.1 How the comparison was done . . . 61
11.2 A comparison run for this manual . . . 61
11.3 Where the two programs differ . . . 62

12 Limits **64**
12.1 What goldfish does that nwdynam does not . . . 64
12.2 Refusals . . . 64
12.3 After the fit . . . 66
12.4 Limits of estimation . . . 67

13 Technical notes **68**
13.1 The native plugin . . . 68
13.2 The Mata part . . . 68
13.3 Optimization . . . 68
13.4 Performance . . . 69

A Translation table: goldfish to nwdynam **71**

Chapter 1

Introduction

1.1 What `nwdynam` is

`nwdynam` fits dynamic network actor models (DyNAMs) in Stata. The data of a DyNAM are *relational events*: a list of acts, each by one actor towards another actor (an email, a phone call, a gift, a citation) with the time at which it happened. Between two events nothing is observed. The model asks, event by event: given everything that has happened before, which actor was going to act (the *rate* model), and whom did that actor choose (the *choice* model)? Both parts weigh effects, such as "an actor writes to someone they wrote to before" (inertia), "an actor answers someone who wrote to them" (reciprocity), or "an actor writes to someone of the same group" (homophily), and `nwdynam` estimates how strong each effect is by maximum likelihood. Chapter 2 states the model; Stadtfeld and Block [1] introduce it, and Stadtfeld, Hollway and Block [2] add the coordination model for undirected events.

`nwdynam` is a Stata package of its own that builds on the `nwcommands` package. The events are declared as a network with `nwset`, the effects are chosen by options of one command, `nwdynam`, and the results are returned in `e()` like those of any Stata estimation command, so that `estimates store`, `lrtest`, `test` and `lincom` work as usual. The likelihood, its gradient and its Hessian are evaluated in a compiled C plugin that is called from Mata. No R installation or other external software is used when you fit a model.

1.2 Relation to `goldfish`

The reference implementation of DyNAMs is the R package `goldfish` [3]. `nwdynam` follows it closely in its model, in the definitions of the effects, in its likelihoods and in the postestimation tools, so that the documentation of `goldfish` and the published literature apply to `nwdynam` as well. It is nevertheless an *independent reimplementation* written in Stata, Mata and C. Its results were compared with `goldfish` 1.6.12 on event logs of several kinds; Chapter 11 documents what was compared and how. `nwdynam` is not affiliated with, endorsed by or supported by the `goldfish` project.

`nwdynam` is meant for researchers who work in Stata, or who prefer it to R. It is *not* a replacement for `goldfish`, which remains the reference. Where the two programs differ in a way that matters to a user, this manual says so (Chapter 12). The most important differences are these.

- Effects are chosen by Stata options (for example `inertia`, `same(div)`), not by a model formula. Without any effect option the command fits a default set of effects of the sub-model (Section 5.3); naming any effect restricts the model to the effects named. In `goldfish` the formula always lists the effects.
- The data are declared with `nwset ..., eventtime()` and the actor attributes are variables of the dataset in memory, not `goldfish` objects (`defineNodes`, `defineNetwork`, `linkEvents`,

`defineDependentEvents`). Appendix A gives the translation.
- `goldfish` has no goodness-of-fit or simulation function and `nwdynam` adds none. The postestimation commands are those that `goldfish` has (Chapter 9).

1.3 What is covered

- **Sub-models.** The *choice* sub-model (directed or two-mode events), the *rate* sub-model without an intercept (the ordinal partial likelihood) and with an intercept (the continuous-time competing-risks hazard), and the *choice_coordination* sub-model (undirected events).
- **Effects.** Structural effects (inertia, reciprocity, degrees, two- and three-path closure, embeddedness in transitive structures), covariate effects (homophily, ego, alter, interaction, the covariate of the in-neighbours), recency windows and weighted versions of the tie-based effects, and effects of other networks (a tie in another network, closures through other networks, mixed closures). Chapters 5–7 define every effect.
- **Data features.** Events with a value, events with identical times, two-mode networks, covariates that change over time, actors who join or leave, and a per-event restriction of the candidates (Chapters 4 and 8).
- **Postestimation.** `estat ic`, `estat eventlik`, `estat eventprob`, `estat outliers`, `estat changepoints`, and two plots of the fit, `estat choiceplot` and `estat rateplot` (extensions: `goldfish` has no such plots).

What `goldfish` has that `nwdynam` does not. Chapter 12 lists it in full: the argument `ignoreRep` in the rate sub-model, `tertiusDiff` with time-varying attributes or in two-mode networks, `tertius`, `indeg`, `outdeg`, `recip` and `nodeTrans` computed through another network, and the effects that `goldfish` itself cannot compute for two-mode networks. `goldfish` also offers the model for interactions (`DyNAMi`) and a relational event model (`REM`); `nwdynam` covers neither. `nwdynam` does not simulate.

How `nwdynam` differs from `nwrem`. The sibling package `nwrem` fits relational event models in the sense of Butts [5]. Both work on timestamped dyadic events; `nwdynam` factors the process into the rate and the choice sub-model, as DyNAMs do.

1.4 Installation and requirements

`nwdynam` needs the `nwcommands` core package (version 3.29.0 or later; the examples of this manual were run with core 3.33.0), which provides the network objects, `nwset` and the data-handling commands, and Stata 17 or later. Install the core first and then `nwdynam`:

```
. net install nwcommands, from(https://nwcommands.org/net/) replace
. nwinstall, all
. nwinstall, modelpkg(nwdynam)
```

`nwinstall, all` installs the help files, dialog boxes and menu of the core, and `nwinstall, modelpkg(nwdynam)` installs `nwdynam` from its repository. To install the core and all four model packages in one step, use `nwinstall, everything`. `nwdynam` can also be installed by itself, once the core is there:

```
. net install nwdynam, from(https://raw.githubusercontent.com/thomas-u-grund/nwdynam/main/)
    replace
```

This installs the state of the main branch. A specific release is installed from its tag, for example

```
. net install nwdynam, from(https://raw.githubusercontent.com/thomas-u-grund/nwdynam/v1.11.0/)
    replace
```

`net describe nwdynam` shows the version and distribution date installed, `nwdynam version` the version of the package, of the core, the plugin in use and the version of `goldfish` the results were compared with (see Chapter 3), and `e(version)` holds the version that produced a set of results. `nwdynam` checks the core when it starts, and also whether the Mata library of the core and that of `nwdynam` fit together, and says how to reinstall if they do not.

Platforms. The package ships a compiled plugin for macOS (universal), Linux and Windows. The examples of this manual were run on macOS with Stata 17. The package is tested on macOS with Stata 17 (Stata/BE is enough) and on Linux (Ubuntu 22.04) and Windows with StataNow 19.5 MP; the run times of Section 13.4 were measured on Linux. Stata 16 and older have not been tested and are not expected to work.

The native plugin. Every fit needs the plugin; there is no slower fallback. If the plugin file of your platform is not installed, `nwdynam` stops with return code 601 and names the missing file; if the installed plugin is older than the `.ado` files, or cannot be loaded, it stops with return code 3495. Reinstall the package with `net install nwdynam, replace` and restart Stata. If the shipped plugin does not work on your computer, `nwinstall, buildplugin(nwdynam)` compiles it there (macOS and Linux; a C compiler and `make` are needed; restart Stata afterwards). The C source is part of the package.

The Mata library. If the Mata library of `nwdynam` does not load in your Stata (for example `r(3499)`, because it was built by a newer Stata than yours), `nwinstall, buildmata(nwdynam)` compiles it with your own Stata and installs it; save your data first, because the build clears the data and the Mata objects in memory, and restart Stata afterwards.

1.5 How to read this manual

Chapter 2 states the model and its likelihoods. Chapter 3 fits a first model. Chapter 4 shows how event data are declared. Chapters 5–8 are the reference for options and effects, Chapter 9 for the postestimation commands. Chapter 10 contains worked examples with output; all output in this manual comes from do-files that are part of the repository (Section 10.1). Chapters 11–13 document the comparison with `goldfish`, the limits and the technical background.

1.6 Data sets used in this manual

Manufacturing company emails. The examples use the metadata of the emails of a manufacturing company: 50,233 timestamped emails among the employees of a mid-sized company in 2010 (sender, receiver and time in days since 1 January 2010; no content), and a second file with the reporting hierarchy (supervisor, level and division) of 154 employees. The data are by Michalski [6] (Harvard Dataverse, doi:10.7910/DVN/6Z3CGX, released under CC0 1.0; see also Nurek and Michalski [7]). They are loaded from the web with `nwwebuse manufacturing` and `nwwebuse manufacturing_nodes`. The core data set is a subset of the original: only emails with exactly one recipient and only employees with a known supervisor were kept. The examples use the 631 emails sent in the first five days of the log (`keep if t < 5`).

Social Evolution. The comparison with `goldfish` (Chapter 11) also uses the data set `Social_Evolution` that ships with `goldfish`: 439 phone calls among students of a dormitory (84 students, 54 of whom call or are called in the log) [8]. The calls are in the file `manual/ex/data/socialevolution_calls.csv` of the repository, exported from `goldfish`.

Chapter 2

The model

This chapter states the dynamic network actor model, the three sub-models of `nwdynam`, the likelihoods that are maximized, and the conventions that govern how the data enter (the order of events, events with identical times, the risk sets). The model and its interpretation are those of Stadtfeld and Block [1] and, for the coordination model, of Stadtfeld, Hollway and Block [2]; the likelihoods are those of `goldfish`.

2.1 Relational events and the state of the network

The data are a list of K events $e_k = (i_k, j_k, t_k, v_k)$, ordered by time: at time t_k actor i_k (the *sender*) acts towards actor j_k (the *receiver*). The number v_k is the *value* of the event ($v_k = 1$ unless a value variable is given, Section 4.4). The n actors of the network are fixed in advance (some actors may have no event; Section 3.3), and $i_k \neq j_k$.

At every time t the model has a *state*: the network of ties that the events before t have created. The value of the tie from i to j is the sum of the values of the earlier events from i to j; the tie *exists* while this sum is above 0. With unit values the tie exists if i has ever acted towards j. With a *window* of length w (Section 6.4) an event is removed again from the tie value at $t_k + w$, so that the tie exists only if an event of the pair happened within the last w time units. Every *effect* is a statistic $s(\cdot)$ computed from this state, from the covariates of the actors and, for some effects, from other networks. The statistic at the time of an event is evaluated from the history *strictly before* that event: there is no look-ahead.

2.2 The sub-models

A DyNAM separates two questions. The *rate* sub-model describes which actor acts next and when, the *choice* sub-model which receiver the acting actor chooses. Given the history, the two are independent, so that each can be estimated alone, and `nwdynam` fits one sub-model per call (option `submodel()`).

2.2.1 The choice sub-model

Given that actor i acts at time t, the probability that i chooses j is a conditional logit over the candidate receivers,

$$P(j \mid i, t) = \frac{\exp\{\beta^\top s(i, j, t)\}}{\sum_{h \in R_i(t)} \exp\{\beta^\top s(i, h, t)\}}, \tag{2.1}$$

where $s(i, j, t)$ is the vector of the effect statistics of the pair (i, j) and $R_i(t)$ is the set of candidates: all other actors who are present at t (Section 8.2), or, with `opportunities()`, the actors who are available for that event (Section 8.3). The log likelihood is the sum of $\log P(j_k \mid i_k, t_k)$ over the events. A coefficient is the change in the log odds of choosing a candidate when the statistic of

that candidate increases by one and the other statistics of the candidates stay the same; $\exp(\beta)$ is an odds ratio of choosing one candidate over another.

2.2.2 The rate sub-model

The rate sub-model asks which actor acts. It has two forms.

Without an intercept (the default). The probability that actor i is the sender of the next event is

$$P(i \mid t) = \frac{\exp\{\beta^\top s(i,t)\}}{\sum_{h \in R(t)} \exp\{\beta^\top s(h,t)\}}, \tag{2.2}$$

where $s(i,t)$ are statistics of the actor alone (its in-degree, its out-degree, its own covariate value; there is no second actor) and $R(t)$ is the set of *all* actors who are present at t (n of them, not $n-1$). This is the ordinal partial likelihood: only the *order* of the events enters it, never the length of the time between them. It is the same kind of likelihood as that of the choice sub-model, taken over a different risk set. A coefficient is the change in the log ratio of the rates of two actors who differ by one unit in the statistic.

With an intercept (`intercept`). In the continuous-time form each actor is an independent Poisson clock with the hazard $\lambda_i(t) = \exp\{\beta^\top s(i,t)\}$, where the first statistic is a constant 1 (the *intercept*). The log likelihood is

$$\ell(\beta) = \sum_{k=1}^{K} \beta^\top s(i_k, t_k) \;-\; \sum_{k=2}^{K} (t_k - t_{k-1}) \sum_{h \in R(t_k)} \exp\{\beta^\top s(h, t_k)\} \tag{2.3}$$

(in the second sum the statistics are those of the period before t_k; when a statistic changes inside the period, because a time-varying covariate changes or a windowed tie expires, the integral is split at that time). The first event contributes only its own term: there is no assumed period of exposure before the first observed event. The model is now sensitive to the *scale* of the time between events, that is, to the unit of the variable given to `eventtime()`: the intercept is the log of the rate of an actor whose other statistics are 0, per time unit.

2.2.3 The coordination sub-model

The sub-model `choice_coordination` is for *undirected* events in which neither actor plays a privileged role (the forming of a treaty, co-authorship). For an event between a and b the model computes the choice probability (2.1) in both directions, $p(a \to b)$ and $p(b \to a)$, multiplies them, and normalizes over all unordered pairs $\{c, d\}$ of actors who are present:

$$P(\{a,b\} \mid t) = \frac{p(a \to b)\, p(b \to a)}{\sum_{\{c,d\}} p(c \to d)\, p(d \to c)}. \tag{2.4}$$

This is a multinomial–multinomial joint model, not a simple conditional logit. The statistics $s(a,b,t)$ are those of the choice sub-model (Chapter 5 says which effects exist); only the aggregation differs. The network must be undirected.

2.3 Events with identical times

Events that have exactly the same time are *simultaneous*, as in `goldfish`: the statistics of every event of a time point are computed from the network as it was *before* that time point, and the state is updated only after the whole group has been evaluated. The order of the rows of a group therefore does not matter; each event contributes its own term to the log likelihood. In the rate

model with intercept the time between events inside a group is zero, and windowed effects see the group at its common time. Coarsening the time stamps of a log (rounding to days, say) creates ties and changes the estimates; that is intended and it matches `goldfish` (an example is in Section 10.9). A note under the header of the results says how many events share a time point. The option `sequential` processes events with equal times one after another instead, in the order in which the network lists them, each seeing the earlier ones of its group; this is what the sibling package `nwrem` does. Data without tied times are not affected by the choice.

2.4 Estimation

`nwdynam` maximizes the log likelihood of the chosen sub-model by Newton's method on the analytic gradient and Hessian. The log likelihoods of the choice and the rate sub-model are concave in β, so that there is at most one maximum; the log likelihood of the coordination sub-model is not, and can be flat (Section 13.3). The covariance matrix of the coefficients is the inverse of the observed information, minus the inverse Hessian at the maximum, stored in `e(V)`; the table reports z statistics and confidence intervals from it (`level()`).

An effect is flagged as *not identified* when its variance is missing, zero or its standard error exceeds 100 (a separation in the data, too few events for the effect, or effects that are linear functions of each other). The names are stored in `e(unidentified)` and a note under the table says so; such an estimate must not be interpreted.

2.5 Reading the results

- The coefficient of an effect in the choice model is a log odds ratio over candidate receivers (2.1); in the rate model without intercept a log rate ratio of actors; in the rate model with intercept a log rate per time unit (2.3).
- The log likelihood of the choice, the rate and the coordination model are not comparable with each other: they are likelihoods of different events. Models are compared within a sub-model, on the same events and the same risk sets, with `lrtest` (the models must be nested) or with the information criteria of `estat ic`.
- With the default likelihoods only the order of events matters. This is not a weakness of the data but a statement of the model: a log with the same order of events and different times between them gives the same estimates. Only the rate model with intercept uses the elapsed time.

Chapter 3

Getting started

This chapter fits a first model on a log of emails. All output was produced by `manual/ex/ex01_first.do` (Section 10.1).

3.1 The version

`nwdynam version` prints the installed release, the core package and the plugin in use. It stores the release in `r(version)` and `r(date)`, the core version in `r(core)`, the plugin in `r(plugin)`, the version of `goldfish` the results were compared with in `r(reference)`, and the license in `r(license)`. Quote the version when you report an analysis; after estimation it is in `e(version)`.

```
. nwdynam version
nwdynam version 1.11.0, 21nov2026
  core package nwcommands : installed (development copy, no distribution date)
  native plugin           : ~/Library/Application Support/Stata/ado/plus/d/dynam_sim_macos.
    plugin
  compared with           : R goldfish 1.6.12
  license                 : GNU GPL-3
```

3.2 The events

The data are the emails of Section 1.6, loaded with `nwwebuse`. Each row is one email with the sender, the receiver and the time it was sent (days since 1 January 2010). We use the 631 emails of the first five days.

```
. nwwebuse manufacturing, nwclear
(plain Stata dataset loaded - not a saved network file; build a network from
it directly, e.g. with nwfromedge or nwgenerate)

. keep if t < 5
(49,602 observations deleted)

. list in 1/5

     +-----------------------------+
     | sender   receiver          t |
     |-----------------------------|
  1. |    E39        E81   3.235255 |
  2. |    E39        E81   3.235822 |
  3. |   E112       E136    3.25566 |
  4. |   E112        E60    3.25728 |
  5. |   E112        E57   3.257963 |
     +-----------------------------+

. count
```

```
  631

. display "span in days: " %5.2f t[_N] - t[1]
span in days:  1.75

. duplicates report sender receiver

Duplicates in terms of sender receiver

--------------------------------------
   Copies | Observations       Surplus
----------+---------------------------
        1 |          235             0
        2 |          160            80
        3 |           75            50
        4 |           76            57
        5 |           55            44
        6 |            6             5
        8 |           24            21
--------------------------------------
```

The log spans 1.75 days. 235 of the sender–receiver pairs occur once, the others several times: the same person writes to the same person repeatedly, which is the pattern that the inertia effect describes. Three emails share the time stamp of an earlier email.

3.3 Declaring the events as a network

An event list is declared with `nwset` and the option `eventtime()`, which names the time variable. The first two variables are the sender and the receiver. The log mentions only some of the 154 employees of the company; the others neither send nor receive an email in these five days. They still belong to the *risk set*: the rate model asks why *this* actor acted rather than any of the others, and the choice model why this receiver was chosen among all candidates. They are added as isolates with `nwaddnodes`, from the list of employees in the company file.

```
. quietly levelsof sender, local(a) clean

. quietly levelsof receiver, local(b) clean

. local seen : list a | b

. display "employees in the log: " `: word count `seen''
employees in the log: 102

. tempfile mail

. save `mail'
file <temporary file> saved as .dta format

. nwwebuse manufacturing_nodes, nwclear
(plain Stata dataset loaded - not a saved network file; build a network from
it directly, e.g. with nwfromedge or nwgenerate)

. quietly levelsof label, local(everyone) clean

. display "employees in the company file: " `: word count `everyone''
employees in the company file: 154

. local without : list everyone - seen

. display "employees without an email: " `: word count `without''
employees without an email: 52
```

```
. local miss

. foreach e of local without {
  2.     local miss `miss',`e'
  3. }

. local miss = substr("`miss'", 2, .)
```

```
. use `mail', clear
(Manufacturing company emails, 2010 (Nurek and Michalski 2020; CC0))

. nwset sender receiver, directed eventtime(t) name(mail)

. nwaddnodes mail, nodenames(`miss')

. nwset
(1 networks)
--------------------
      mail
```

The network is called `mail`. After `nwset ..., eventtime()` the dataset in memory holds one row per actor; Chapter 4 explains how to attach actor attributes.

3.4 The choice and the rate sub-model

Without options `nwdynam` fits the choice sub-model with its default effects `inertia`, `recip` and `indeg`:

```
. nwdynam mail
------------------------------------------------------------
Dynamic Network Actor Model - choice sub-model (MLE)
Network: mail                           Actors: 154
Events: 631                             Log likelihood: -2.31e+03
Note: 3 events share a time point with an earlier event; events with equal
times are simultaneous (as in goldfish).
------------------------------------------------------------
------------------------------------------------------
        mail | Coefficient  Std. err.      z    P>|z|
-------------+----------------------------------------
     inertia |   1.764847   .1422701    12.40   0.000
       recip |   2.922381   .1401073    20.86   0.000
       indeg |   .1096229   .0083142    13.18   0.000
------------------------------------------------------
```

The header names the sub-model, the network, the number of actors and of events and the log likelihood, and notes how many events share a time point with an earlier event (Section 2.3). The coefficients are log odds ratios: an email goes to someone the sender wrote to before with odds $e^{1.76} \approx 5.8$ times those of someone the sender did not write to, to someone who wrote to the sender before with odds $e^{2.92} \approx 18.6$ times, and every additional distinct person who has written to a candidate raises the odds of the candidate by about 12 per cent ($e^{0.11}$).

The rate sub-model describes who writes next. Its default effects are `indeg` and `outdeg`:

```
. nwdynam mail, submodel(rate)
------------------------------------------------------------
Dynamic Network Actor Model - rate sub-model, no intercept (MLE)
Network: mail                           Actors: 154
Events: 631                             Log likelihood: -2.99e+03
Note: 3 events share a time point with an earlier event; events with equal
times are simultaneous (as in goldfish).
------------------------------------------------------------
```

```
------------------------------------------------
        mail | Coefficient  Std. err.      z    P>|z|
-------------+----------------------------------
       indeg |   .0324628   .0121141     2.68   0.007
      outdeg |   .2981233   .0176336    16.91   0.000
------------------------------------------------
```

Actors who have written to many different people have a higher rate of writing ($e^{0.30}$ per additional distinct recipient); the effect of the in-degree is small.

3.5 Replay and stored results

`nwdynam` without arguments redisplays the last results; only `level()`, `ci` and `noci` may be given. `estimates replay` works as well.

```
. nwdynam, level(90)
------------------------------------------------------------
Dynamic Network Actor Model - rate sub-model, no intercept (MLE)
Network: mail                           Actors: 154
Events: 631                             Log likelihood: -2.99e+03
Note: 3 events share a time point with an earlier event; events with equal
times are simultaneous (as in goldfish).
------------------------------------------------------------
------------------------------------------------------------------------------
        mail | Coefficient  Std. err.      z    P>|z|     [90% conf. interval]
-------------+----------------------------------------------------------------
       indeg |   .0324628   .0121141     2.68   0.007     .0125368    .0523888
      outdeg |   .2981233   .0176336    16.91   0.000     .2691186     .327128
------------------------------------------------------------------------------
```

By default the table has no interval columns at a line size below 100; an explicit `level()` (as above) or `ci` shows them, with the names cut to fit:

```
. nwdynam, ci
------------------------------------------------------------
Dynamic Network Actor Model - rate sub-model, no intercept (MLE)
Network: mail                           Actors: 154
Events: 631                             Log likelihood: -2.99e+03
Note: 3 events share a time point with an earlier event; events with equal
times are simultaneous (as in goldfish).
------------------------------------------------------------
------------------------------------------------------------------------------
        mail | Coefficient  Std. err.      z    P>|z|     [95% conf. interval]
-------------+----------------------------------------------------------------
       indeg |   .0324628   .0121141     2.68   0.007     .0087195     .056206
      outdeg |   .2981233   .0176336    16.91   0.000     .2635621    .3326845
------------------------------------------------------------------------------
```

The results are stored in `e()`; Section 5.5 describes every item.

```
. ereturn list

scalars:
                  e(N) =  631
              e(nodes) =  154
                 e(ll) =  -2986.282302172496
             e(ntimes) =  628

macros:
             e(header) : "Dynamic Network Actor Model - rate sub-model, no i.."
               e(ties) : "simultaneous"
          e(estat_cmd) : "nwdynam_estat"
              e(fitid) : "2110865696000-2"
```

```
            e(effects) : "indeg outdeg"
          e(submodel) : "rate"
            e(depvar) : "mail"
            e(title) : "Dynamic Network Actor Model, rate sub-model, no in.."
          e(version) : "1.11.0"
              e(cmd) : "nwdynam"
        e(properties) : "b V"

matrices:
                 e(b) :  1 x 2
                 e(V) :  2 x 2
```

Chapter 4

Data preparation

`nwdynam` fits the sub-models to a network that was declared as an *event network* with `nwset` and its option `eventtime()`. This chapter describes what such a network must contain, how actor attributes, event values and the special data features are supplied, and what the command refuses.

4.1 Declaring the events

```
. nwset sender receiver, directed eventtime(t) name(mail)
. nwset sender receiver value, directed eventtime(t) name(mail)
. nwset sender receiver, undirected eventtime(t) name(contact)
. nwset sender receiver, twomode eventtime(t) name(tm)
```

The dataset in memory holds one row per event. The first two variables name the sender and the receiver (strings or numbers; they become the actor labels), the optional third variable the value of the event (Section 4.4); `eventtime(`*varname*`)` names the numeric time variable, in any unit and with any origin. The rows need not be sorted; `nwdynam` orders the events by time, and rows with the same time keep the order of the event list (this order matters only with the option `sequential`). The kind of network depends on the sub-model:

- `submodel(choice)` and `submodel(rate)` need a *directed* or a *two-mode* network;
- `submodel(choice_coordination)` needs an *undirected* one-mode network.

Every event must have a time, a sender and a receiver that are actors of the network, and the sender must differ from the receiver. Events without a time, with a missing actor or with a missing value stop the command with error 416 (the message counts them; drop such rows before `nwset`); an event of an actor to itself stops it with error 3498; a log with fewer than two events stops it with error 2001. A network that was not declared with `eventtime()` is refused with error 198.

4.2 Actors without events: the risk set

The actors of the network are the actors of the event list, together with those added by `nwaddnodes`. Because every actor of the network is in the risk set at every time (unless it is absent, Section 8.2), actors who do not appear in the log must be added, otherwise they are silently missing from the denominators of the likelihoods. Section 3.3 shows how, with the list of all employees as the source.

```
. nwaddnodes mail, nodenames(E7,E12,E13)
```

`nwset` (without arguments) lists the networks in memory; `nwdynam` reports the number of actors in the header (`e(nodes)`).

4.3 Actor attributes

The covariate options (`same()`, `diff()`, `sim()`, `ego()`, `alter()`, `tertius()`, `egoalterint()`, `present()`) read numeric variables of the *dataset in memory* at the moment `nwdynam` is called, not the event dataset from which the network was declared. Put the actors into the data with `nwload`, which makes a dataset with one row per actor:

```
. nwload mail, xvars
```

This dataset has the node-label variable `_nwnode`. When it is there, `nwdynam` matches every actor to its row *by label*, whatever the sort order of the data, and every actor must have exactly one row. Without `_nwnode` the dataset must have exactly one row per actor, in the actor order of the network. A missing value of a covariate stops the command with error 416. Attributes from another file are merged on the label; the label must be kept in `_nwnode`, which is why the examples of this manual copy it into a variable `label` and merge on that:

```
. nwload mail, xvars

. gen long row = _n

. gen label = _nwnode

. quietly merge m:1 label using 'actors', keep(master match) nogenerate

. sort row

. list label supervisor level div in 1/4

     +---------------------------------+
     | label   superv~r   level    div |
     |---------------------------------|
  1. |  E102        E36       2    E36 |
  2. |  E103        E85       2    E85 |
  3. |  E104        E86       1   E104 |
  4. |  E105       E104       2   E104 |
     +---------------------------------+
```

Attributes are constant over the log unless they are declared as changing (Section 8.1). A string variable cannot be used; make a numeric one with `encode`.

4.4 Event values

A third variable in `nwset` gives each event a value (`nwset sender receiver value, eventtime(t)`); without it every event has the value 1. As in `goldfish`, where the value is the *increment* of the event, the value of an event is added to the tie from its sender to its receiver (on an undirected network to the tie between the two). A tie exists while its value is above 0: an event with the value 0 creates no tie, and a negative value can remove one. The unweighted effects use whether a tie exists, the weighted effects (Section 6.5) use the value of the tie. As in `goldfish`, the unweighted `indeg` and `outdeg` add the *sign* of each tie value, so a tie whose value is below 0 counts -1 (inside a window this happens when an event with a negative value outlives the event it cancelled). With only positive values the unweighted effects are the same as without values.

4.5 Undirected and two-mode events

Undirected. The coordination sub-model needs `nwset ..., undirected eventtime()`; the pair of an event is unordered.

Two-mode. A two-mode (bipartite) event network has two sets of actors, the senders (mode 1) and the receivers (mode 2), and every event goes from a mode-1 actor to a mode-2 actor, never the reverse; events that violate this are refused with error 198. Declare the network with `twomode`. In the example below the labels of the second mode carry a prefix, so that a division is not mistaken for an employee:

```
. gen str5 division2 = "D" + division
. nwset sender division2, twomode eventtime(t) name(tm)
```

The covariates of a two-mode network are one value per actor of the *combined* actor list, the mode-1 actors first, then the mode-2 actors. Section 6.7 lists the effects available in two-mode networks.

4.6 Composition change, changing covariates, opportunities

Three further features need data of their own, each declared as an event network or as a variable: covariates that change over time (option `change()`), actors who join or leave (options `present()` and `compchange()`) and a per-event restriction of the candidates (options `opportunities()` and `oppframe()`). They are described in Chapter 8.

Chapter 5

Syntax and options

5.1 Syntax

```
nwdynam [netname] [, submodel(choice|rate|choice_coordination) effect options
                     window and weighted options opportunities(varlist) oppframe(name)
                     tie(netname) netterms(terms) ignorerep(effects)
                     change(varname netname [...]) increment
                     present(varname) compchange(netname) intercept
                     seed(#) level(#) ci noci sequential]
nwdynam version
estat ic | eventlik | eventprob | outliers | changepoints | choiceplot | rateplot  [, ...]
```

netname is the event network; without it the current network is used (`nwdynam` alone fits the default model of the current network). Typed without arguments, or with `level()` only, after a fit, `nwdynam` redisplays the results; `estimates replay` works as well. The `estat` commands are described in Chapter 9.

5.2 General options

`submodel(string)`
: The sub-model: `choice` (the default), `rate` or `choice_coordination`. A value other than these three stops the command with error 198.

`intercept`
: With `submodel(rate)`: the continuous-time hazard model of Section 2.2.2, with an intercept. Error 198 with another sub-model.

`sequential`
: Processes events with identical times one after another instead of jointly (Section 2.3).

`level(#)`
: The confidence level of the displayed intervals (default 95). An explicit `level()` also shows the intervals.

`ci`, `noci`
: The coefficient table shows the names of the effects in full (up to 32 characters). The interval columns are left out by default when `linesize` is below 100, where a long name does not fit next to them (the listings of this manual are made with `linesize` 80). `ci` brings the intervals back; the names are then cut to fit. `noci` leaves the intervals out at any line size. `ci` together with `noci` stops with error 198. `nopvalues`, `cformat()`, `pformat()` and `sformat()` are passed to the table.

`seed(#)`
: A random-number seed, an integer between 0 and 2,147,483,647, set before the estimation. The optimizer starts from fixed values and runs without random numbers wherever the Newton iterations succeed; where they do not, further starting values are drawn at random,

and the coordination sub-model always uses several starts (Section 13.3). The seed makes such fits reproducible.

5.3 Effect options and sub-models

Each sub-model has a fixed roster of effects. Giving *no* effect option fits the default effects of the sub-model together: `inertia`, `recip` and `indeg` for `choice`; `indeg` and `outdeg` for `rate`; `inertia` and `indeg` for `choice_coordination`. For a two-mode network the defaults are `inertia` and `indeg` (choice) and `outdeg` (rate). Giving one or more effect options restricts the fit to *exactly* those. An effect option that does not apply to the chosen sub-model is rejected with an error message (return code 198) rather than silently ignored. The coefficients are in the fixed order of Table 5.1, then the terms of `netterms()`, and their names are the effect names (`inertia`, `same`, `tie_hier`, ...); with `intercept` the first coefficient is `Intercept`. `e(effects)` lists the effects actually fitted.

Table 5.1 lists every effect option, the sub-models in which it is available (C choice, R rate, K choice_coordination) and where it is defined. A window or weighted option selects its effect by itself: `inertiawindow(#)` alone fits `inertia` with that window, `weightedinertia` alone fits `inertia` on the tie value (Sections 6.4 and 6.5).

Table 5.1: The effect options and the sub-models in which they are available.

Option	C	R	K	Effect (Chapter 6)
Structural effects				
`inertia`	x		x	the sender has sent to the candidate before
`recip`	x			the candidate has sent to the sender before
`indeg`	x	x	x	in-degree of the candidate (the actor in the rate model)
`outdeg`	x	x		out-degree of the candidate (the actor in the rate model)
`trans`	x		x	two-paths sender $\rightarrow$ third $\rightarrow$ candidate
`cycle`	x			two-paths candidate $\rightarrow$ third $\rightarrow$ sender
`commonsender`	x			a common third actor sent to both
`commonreceiver`	x			both sent to a common third actor
`four`	x		x	three-paths sender $\rightarrow k \leftarrow l \rightarrow$ candidate
`nodetrans`	x	x	x	embeddedness of the candidate in transitive structures
Covariate effects (a numeric variable of the dataset)				
`same(var)`	x		x	sender and candidate have the same value
`diff(var)`	x		x	absolute difference of the values
`sim(var)`	x		x	negative absolute difference
`ego(var)`		x		the value of the actor
`alter(var)`	x		x	the value of the candidate
`tertius(var)`	x	x	x	mean value of the in-neighbours of the candidate
`egoalterint(v1 v2)`	x		x	value v_1 of the sender times value v_2 of the candidate

Option	C	R	K	Effect
Windows (self-activating)				
`inertiawindow(#)`	x		x	`inertia` within the last # time units
`recipwindow(#)`	x			`recip` within a window
`indegwindow(#)`		x	x	`indeg` within a window
`outdegwindow(#)`		x		`outdeg` within a window
`tertiuswindow(#)`	x	x	x	`tertius()` on the ties of the last # time units
Weighted effects (self-activating)				
`weightedinertia`	x		x	`inertia` on the tie value
`weightedrecip`	x			`recip` on the tie value
`weightedindeg`	x	x	x	`indeg` on the tie values
`weightedoutdeg`	x	x		`outdeg` on the tie values
Other networks				
`tie(net)`	x		x	the tie exists in another network
`netterms(terms)`	x		x	effects through other networks, with their own window (Chapter 7)
`ignorerep(effects)`	x		x	the named effects are 0 for pairs that already have a tie

5.4 Options of the data features

`opportunities(evvar actvar), oppframe(name)`
Choice sub-model only. Per-event restriction of the candidate set (Section 8.3).

`change(var net [var net ...]), increment`
Covariates that change over time (Section 8.1).

`present(var), compchange(net)`
Actors who join or leave (Section 8.2).

5.5 Stored results

`nwdynam` stores in `e()`:

Scalars
`e(N)` the number of events; `e(nodes)` the number of actors; `e(ntimes)` the number of distinct time points; `e(ll)` the log likelihood at the maximum; `e(ncompchange)` the number of join and leave events (only with `present()` or `compchange()`).

Macros
`e(cmd)` (`nwdynam`); `e(version)` the release that produced the results; `e(title)` and `e(header)`; `e(depvar)` the name of the event network; `e(submodel)`; `e(effects)` the effects fitted, in their fixed order; `e(ties) none` (all times distinct), `simultaneous` or `sequential`; `e(unidentified)` the effects without a usable standard error (empty when all are identified); `e(present)` and `e(compchange)` (with composition change); `e(fitid)` and `e(estat_cmd)`, which tie the `estat` commands to the fit in memory.

Matrices
`e(b)` the coefficient vector (the columns are named after `e(effects)`); `e(V)` the variance–covariance matrix (the inverse of the observed information).

Section 10.11 shows the output of `ereturn list`. The results of the `estat` commands are in `r()` (Chapter 9).

5.6 Return codes

Besides the error codes of Stata itself, `nwdynam` stops with these return codes (the full list of refusals is in Chapter 12): 198 an option that does not apply or has a bad value; 416 a missing value in the events or in a covariate; 3498 an event with the same sender and receiver; 2001 fewer than two events; 601 the plugin of the platform is missing; 3495 the plugin is older than the `.ado` files or cannot be loaded; 498 an `estat` command after results that do not belong to the fit in memory.

Chapter 6

Effects

An effect is a statistic of the state of the network and of the covariates that enters the log linear predictor $\beta^\top s$ of a sub-model. This chapter defines the effects that are chosen by their own option; the effects of other networks (`tie()` and `netterms()`) are in Chapter 7. Table 5.1 says in which sub-models each effect exists.

Notation. In the choice and the coordination sub-model the statistic belongs to a pair: i is the sender (the actor who acts) and j the *candidate* receiver. In the rate sub-model it belongs to a single actor i, the candidate sender. Let $A_{ij} = 1$ if the tie from i to j exists at the time of the event (Chapter 2) and 0 otherwise; x_i is the value of a covariate for actor i. In an undirected network A is symmetric. All sums run over other actors, k and l are third actors. The unweighted effects use A; the weighted effects (Section 6.5) use the value of the tie instead.

6.1 Structural effects

`inertia`
: $s(i, j) = A_{ij}$: whether the sender has ever sent to the candidate (within a window, if `inertiawindow()` is given). Choice and coordination.

`recip`
: $s(i, j) = A_{ji}$: whether the candidate has sent to the sender before. Choice only: an undirected tie has no direction to reciprocate.

`indeg`
: $s(j) = \sum_h A_{hj}$: the in-degree of the candidate, the number of distinct actors who have sent to it. In the rate sub-model it is the in-degree of the actor itself (the "ego" type), so that the same option has a different meaning in the choice sub-model (which receivers attract ties) and in the rate sub-model (which actors tend to act). Available in all sub-models. (As noted in Section 4.4, a tie with a negative value counts -1.)

`outdeg`
: $s(j) = \sum_h A_{jh}$: the out-degree, the number of distinct actors the candidate (the actor, in the rate sub-model) has sent to. Choice and rate. Not available in the coordination sub-model, where in-degree and out-degree coincide.

6.2 Closure effects

Each counts paths *through the dependent network* that link the sender i, the candidate j and third actors. The same effects through another network, and the mixed closures with two networks, are terms of `netterms()` (Chapter 7).

`trans`
: $s(i, j) = \sum_k A_{ik}A_{kj}$: two-paths $i \to k \to j$ (transitive closure). Choice and coordination.

`cycle`
: $s(i,j) = \sum_k A_{jk}A_{ki}$: two-paths $j \to k \to i$ (cyclical closure). Choice only.

`commonsender`
: $s(i,j) = \sum_k A_{ki}A_{kj}$: a common third actor sent to both. Choice only.

`commonreceiver`
: $s(i,j) = \sum_k A_{ik}A_{jk}$: both sent to a common third actor. Choice only.

`four`
: $s(i,j) = \sum_{k \neq l} A_{ik}A_{lk}A_{lj}$: three-paths $i \to k \leftarrow l \to j$ through two distinct third actors. Choice and coordination.

`nodetrans`
: $s(j) = \sum_h A_{jh} \sum_k A_{jk}A_{kh}$: the embeddedness of the candidate in transitive structures, the number of its ties that are closed by a two-path from the same actor. A property of the candidate alone, not a comparison with the sender, so that it is available, like `indeg` and `outdeg`, in all sub-models.

The statistics `inertia`, `recip`, `trans`, `cycle`, `commonsender` and `commonreceiver` compare the realized sender with a specific candidate; the rate sub-model has no candidate-receiver role (it is a hazard over which actor acts next), so that they exist only in the choice sub-model. `indeg`, `outdeg`, `nodetrans` are properties of a single actor and are defined in both. `four` needs two distinct third actors and is a statement about a sender–candidate pair.

6.3 Covariate effects

The covariates are numeric variables of the dataset in memory, one value per actor (Section 4.3); each option takes its own variable, and the variables need not be the same for different effects. They are constant over the log unless `change()` is used (Section 8.1).

`same(var)`
: $s(i,j) = \mathbb{1}\{x_i = x_j\}$: homophily; whether sender and candidate have the same value.

`diff(var)`
: $s(i,j) = |x_i - x_j|$: the absolute difference of the values (heterophily).

`sim(var)`
: $s(i,j) = -|x_i - x_j|$: homophily by similarity, the negative absolute difference. It is exactly $-$`diff()` on the same variable, so that `diff()` and `sim()` on the same variable cannot be combined (they are not identified).

`ego(var)`
: $s(i) = x_i$: the value of the actor, with no comparison. Rate sub-model only.

`alter(var)`
: $s(j) = x_j$: the value of the candidate receiver, with no comparison to the sender. Choice and coordination. (`ego()` and `alter()` are the same statistic, named after the role in the sub-model: the potential next sender in the rate model, the potential receiver in the choice model.)

`egoalterint(v1 v2)`
: $s(i,j) = x_i^{(1)}\, x_j^{(2)}$: the value of the first variable of the sender times the value of the second variable of the candidate; an interaction, not a comparison. It needs exactly two numeric variables.

`tertius(var)`
: $s(j) = \dfrac{1}{|N_j|} \sum_{h \in N_j} x_h$ with $N_j = \{h : A_{hj} = 1\}$: the *mean* covariate value of the in-neighbours of the candidate on the dependent network (everyone who has sent to it). An actor with no in-neighbours has the value 0. It is available in all sub-models ("alter type" in the choice model, "ego type" in the rate model) and has a window of its own, `tertiuswindow( )`. The version for another network and the absolute-difference version are in Chapter 7.

`same()`, `diff()`, `sim()`, `ego()`, `alter()` and `egoalterint()` have no window or weighted version:

a window is a recency filter on *ties* (has this pair had a recent event?), and these effects read a covariate value, which has no most recent occurrence.

6.4 Windows

By default `inertia`, `recip`, `indeg` and `outdeg` look back over the entire event history. A *window* restricts one of them to the events of the last w time units: `inertiawindow(#)`, `recipwindow(#)`, `indegwindow(#)`, `outdegwindow(#)`. The number w is positive and in the time units of `eventtime()`. An event is removed from the tie value again at its time plus w, and a tie exists if the summed value of the events within the window is above 0 (with positive values: if an event of the pair happened within the window). An event exactly w time units before the current event still counts; the window is a hard cutoff, not a decaying weight, and a tie can expire once enough time has passed. A window option selects its own effect and sets its window; no separate `inertia` is needed. The windows are independent of each other, and `tertiuswindow(#)` limits `tertius()` to the ties of the last w time units (it needs `tertius()`).

Where the windows exist: `inertiawindow()` in the choice and the coordination sub-model; `recipwindow()` in the choice sub-model; `indegwindow()` in the rate and the coordination sub-model; `outdegwindow()` in the rate sub-model; `tertiuswindow()` in all three. (`indeg` means something different in the choice and rate sub-models, so its window option is specific to the sub-model that has the actor as the candidate.) `indegwindow()`, `outdegwindow()` and `tertiuswindow()` combine with `intercept`; the hazard integral is then split at the expiry times. Only a `change()` of the `tertius()` variable together with `tertiuswindow()` and `intercept` is not available (Chapter 12).

```
. nwdynam mail, inertiawindow(0.5) recip indeg
. nwdynam mail, submodel(rate) indegwindow(3) outdegwindow(5)
```

6.5 Weighted effects

By default the tie-based effects use tie *presence*. `weightedinertia`, `weightedrecip`, `weightedindeg` and `weightedoutdeg` use the *value* of the tie instead: the sum of the values of all earlier events of the pair (`weightedindeg`: of all ties to the actor; `weightedoutdeg`: of all ties from the actor). Without event values every event has the value 1, so that the weighted effects count the earlier events. Like the windows they select their effect by themselves: `weightedinertia` alone fits `inertia` on the tie value. A weighted effect combined with that effect's window uses the summed value of the events within the window. `weightedinertia` exists in the choice and the coordination sub-model, `weightedrecip` in the choice sub-model, `weightedindeg` in all three, `weightedoutdeg` in the choice and the rate sub-model.

```
. nwdynam mail, weightedinertia recip indeg
. nwdynam mail, weightedinertia inertiawindow(0.5)
```

6.6 The coordination sub-model

The statistics of the coordination sub-model are those of the choice sub-model (the model differs only in how the statistics are aggregated, Section 2.2.3). The effects available are `inertia`, `indeg`, `trans`, `four`, `nodetrans`, `same()`, `diff()`, `sim()`, `alter()`, `tertius()`, `egoalterint()`, `tie()` and the terms `tie`, `trans`, `four`, `mixedtrans` and `tertiusdiff` of `netterms()`; also `inertiawindow()`, `indegwindow()`, `tertiuswindow()`, `weightedinertia` and `weightedindeg`. Rejected are `recip` and `outdeg` (an undirected tie has no direction), `cycle`, `commonsender` and `commonreceiver` (`goldfish` registers no coordination version of them), `ego()` (a rate effect), `opportunities(`

`)`, `recipwindow()`, `outdegwindow()`, `weightedrecip`, `weightedoutdeg` and two-mode networks. Without effect options the default is `inertia` and `indeg`.

6.7 Two-mode networks

In a two-mode network every event goes from a mode-1 actor to a mode-2 actor (Section 4.1). The effects available are `inertia`, `indeg`, `alter()`, `four`, `egoalterint()` and `tertius()` in the choice sub-model, and `outdeg` and `ego()` in the rate sub-model. The covariates are one value per actor of the combined actor list (mode 1 first, then mode 2): `alter()` reads the value of the receiving actor (mode 2), `ego()` that of the sending actor (mode 1), `egoalterint()` multiplies the sender's value of the first variable by the receiver's value of the second, and `tertius()` is the mean of the values of the mode-1 actors who have a tie to the candidate. In the rate sub-model the senders (mode 1) matter, in the choice sub-model the receivers (mode 2). `change()` works with the two-mode effects above except `egoalterint()`. `recip`, `outdeg` (choice), `indeg` (rate), `tertius()` and `nodetrans` (rate), `trans`, `cycle`, `commonreceiver`, `commonsender`, `nodetrans`, `same()`, `diff()`, `sim()`, `tie()`, `netterms()`, `opportunities()` and every windowed or weighted effect are rejected for two-mode networks: `goldfish` itself either refuses them or returns statistics of the wrong dimension. Section 10.7 fits both sub-models on a two-mode network, and Chapter 12 shows the messages.

Chapter 7

Effects on other networks

The effects of Chapter 6 read the dependent network, the network of the events that are modeled. Often the question is whether ties in *another* network explain the events: a reporting hierarchy, a friendship network, an earlier sequence of events. `tie()`, `netterms()` and `ignorerep()` add such effects to the choice and the coordination sub-model. They are the effects that `goldfish` offers through `tie(network)`, the closure effects with a named network, `mixedTrans` and its relatives, `tertiusDiff` and the argument `ignoreRep`.

7.1 Which networks

A network used by one of these terms is the name of a network that was declared with `nwset`, in one of three forms:

- the *fitted network itself*: the same effect with its own window, weight or `ignorerep`;
- an *ordinary* network (not declared with `eventtime()`) with exactly the same number of actors as the fitted network, in the same order. The actors are matched by row and index, not by label, as the covariate options of Section 4.3 do when the dataset has no label variable;
- a network declared with `eventtime()` that *changes over time* with its own events. It starts empty and its actors are matched to those of the fitted network by label; an actor of this network that is not an actor of the fitted network stops the command with error 198.

Events of another network at exactly the time of an event of the dependent network are seen by the following events only, as for covariate changes (Section 8.1). With a window the statistic starts at 0 whatever the other network holds at the start of the log, as in `goldfish`.

7.2 The terms

The terms of `netterms()` are written as in `goldfish`, separated by spaces, each one *effect*(*networks* [, *options*]). The coefficients follow those of the other effects, in the order of the terms, and are named after them (`trans_friends`, `tie_adv_wgt`, `mixedtrans_net_adv`, ...; long names are cut in the table, the full name is in `e(effects)`). With $A^{(\text{net})}$ the tie matrix of a network at the time of the event:

Term	Statistic $s(i,j)$	Sub-models
`tie(net)`	$A_{ij}^{(\text{net})}$ (with `weighted`: the value of the tie)	C, K
`trans(net)`	$\sum_k A_{ik}A_{kj}$ in net	C, K
`cycle(net)`	$\sum_k A_{jk}A_{ki}$ in net	C
`commonsender(net)`	$\sum_k A_{ki}A_{kj}$ in net	C
`commonreceiver(net)`	$\sum_k A_{ik}A_{jk}$ in net	C

Term	Statistic $s(i,j)$	Sub-models
`four(net)`	$\sum_{k \neq l} A_{ik} A_{lk} A_{lj}$ in net	C, K
`mixedtrans(net1 net2)`	$\sum_k A^{(1)}_{ik} A^{(2)}_{kj}$	C, K
`mixedcycle(net1 net2)`	$\sum_k A^{(1)}_{ki} A^{(2)}_{jk}$	C
`mixedcommonsender(net1 net2)`	$\sum_k A^{(1)}_{ki} A^{(2)}_{kj}$, counted in both directions of the pair	C
`mixedcommonreceiver(net1 net2)`	$\sum_k A^{(1)}_{ik} A^{(2)}_{jk}$, counted in both directions of the pair	C
`tertiusdiff(var [net])`	$\lvert x_i - \overline{x}_{N_j} \rvert$, N_j the in-neighbours of j in net (default: the fitted network)	C, K

`tertiusdiff` is the absolute difference between the covariate of the sender and the mean covariate of the in-neighbours of the candidate; a candidate without in-neighbour gets the mean of the statistics of the others. A mixed term needs two *different* networks; either of them may be the fitted network, or both may be other networks. In the coordination sub-model only `tie`, `trans`, `four`, `mixedtrans` and `tertiusdiff` exist, as in `goldfish`.

Options of a term. After a comma: `window(#)` counts only the events of the last # time units (as `inertiawindow()`); `weighted` (only `tie`) uses the value of the tie instead of its presence; `ignorerep` sets the statistic to 0 for the pairs that already have a tie in the dependent network (`goldfish`: `ignoreRep = TRUE`).

```
. nwdynam chat, inertia netterms(tie(advice) trans(advice))
. nwdynam chat, inertia netterms(tie(advice, weighted window(30)) mixedtrans(chat advice))
. nwdynam chat, inertia netterms(tertiusdiff(dept) four(advice, ignorerep))
```

7.3 `tie()` and `ignorerep()`

`tie(netname)` is the term `tie(netname)` without options: whether the tie $i \to j$ exists in another network (present when its value is above 0; negative values are not ties), identical in shape to `inertia` but reading another network instead of the event history of the dependent network. It exists in the choice and the coordination sub-model. The value of the tie, a window and `ignorerep` are options of the term in `netterms()`.

`ignorerep(effects)` sets the named effects of the dependent network to 0 for the pairs that already have a tie in the dependent network, for the effects given by the options `trans`, `four`, ...; `inertia` and the covariate effects have no such argument in `goldfish`. `ignorerep` is not available in the rate sub-model: `goldfish` accepts it there but the statistic it then computes depends on an internal array layout.

```
. nwdynam chat, indeg trans ignorerep(trans)
```

7.4 What is not available

The terms of `netterms()` are not available for two-mode networks, with `submodel(rate)`, or for a covariate that is changed with `change()`. `tertius`, `indeg`, `outdeg`, `recip` and `nodetrans` through another network are not available (the closure effects, `tie` and the mixed closures are). Section 10.3 fits several of the terms on the emails with the reporting hierarchy as the other network.

Chapter 8

Changing covariates, composition change and opportunities

Three features need data of their own. Each follows `goldfish` and has its own option: covariates that change over time, actors who join or leave the network, and a restriction of the candidates for every event.

8.1 Covariates that change over time

A covariate may change while the events happen, as in `goldfish` (attribute change events linked to the nodes with `linkEvents()`). The values in the dataset are the values *at the start* of the log; the changes are an event list of their own, declared with `nwset` as an event network whose events name one actor (sender and receiver are the same actor, `selfloop`) and carry the new value of the covariate as the event value:

```
. nwset label label2 newlevel, eventtime(tchg) selfloop name(levelchg)
. nwdynam mail, inertia alter(level) change(level levelchg)
```

(`label2` is a copy of the actor variable.) `change()` takes pairs of a covariate variable of the dataset and the name of its change network; several covariates change with `change(floor floorchg grade gradechg)`. The variable must be used by `same()`, `diff()`, `sim()`, `ego()`, `alter()`, `tertius()` or `egoalterint()`; every effect that reads it then sees the value that holds at the time of each event, in all sub-models (also with `intercept`).

- Actors are matched by label; an actor of the change network that is not an actor of the fitted network stops the command with error 198. The change events need not be sorted, and a covariate is changed by one network.
- The value of a change event is the new value; with `increment` it is added to the value the actor has at that time (in the order of time, and of the rows of the change network for equal times), as the `increment` column of `goldfish`. The change network needs a value variable; without it every change sets the value 1.
- A change takes effect for the events *after* it: an event at exactly the time of a change still sees the old value, because `goldfish` evaluates the events of a time point before the other event lists. Changes before the first and after the last event are allowed (they matter for `tertius()` and for the estimated rates).
- With `intercept` the observation period reaches from the first to the last event of either list, as in `goldfish`, and a change splits the time between two events (the rates of the actors change at that moment).
- Missing starting values stay an error (416). `change()` works in two-mode networks (except with `egoalterint()`) and together with `opportunities()` (with `oppframe()`); it does not apply to the covariates of `netterms()`.

Section 10.4 gives an example (with artificial changes).

8.2 Composition change: actors who join or leave

As in `goldfish` (the node attribute `present`), actors may join or leave the network while the events happen. An actor who is *absent* is not in the risk set: it cannot be a sender (in the rate sub-model it is left out of the hazard and of the normalisation), nor a receiver (in the choice sub-model it is not among the candidates), nor a member of a pair of the coordination sub-model. Its statistics are still computed, and its earlier ties still count for the degrees of the others.

`present(var)`
: The presence at the start of the log: a variable of the dataset with the values 1 and 0, one row per actor, matched like the covariates. The default is that everybody is present.

`compchange(net)`
: An event network declared with `nwset` that holds the joins and the leaves, one actor per event (sender and receiver are the same actor), with the new presence as the event value: 1 the actor joins, 0 the actor leaves.

```
. nwset label label2 joined, eventtime(tjoin) selfloop name(membership)
. nwdynam mail, inertia indeg present(member0) compchange(membership)
```

- A change takes effect for the events at its time *and after it*: an actor who joins at exactly the time of an event can take part in it, an actor who leaves at exactly the time of an event cannot (unlike the covariate changes). Of several changes of one actor at one time the later row wins. Changes before the first and after the last event are allowed; an actor who is absent during the whole log is simply not in any risk set.
- An event whose sender or receiver is absent at its time is an error (198), as in `goldfish`; so is a change event of a covariate for an actor who is absent at that time.
- Actors are matched by label (an actor of the change network that is not an actor of the fitted network stops the command with error 198). An actor with no event of its own at all must be added to the network first (`nwaddnodes`) when it is to be in the risk sets. Times that must coincide exactly must be stored in the same way (the same variable type) in both event lists.
- With `submodel(rate)` and `intercept` the whole waiting time that ends at an event is exposed to the actors who are present at the *end* of it: the leave of an actor at the time of the event, or earlier in the waiting time, removes it from the whole waiting time; a join at the end adds it to the whole, as in `goldfish`.
- In a two-mode network the senders (mode 1) matter for the rate and the receivers (mode 2) for the choice sub-model. With `opportunities()` the risk set is the intersection of the opportunities and the present actors.
- `present()` cannot be combined with `opportunities()` (both need the data in memory in a different layout); an actor who is absent at the start can then be given a leave event before the first event of the log. Missing values and values other than 0 and 1 in `present()` or in the change network stop the command with error 198. `compchange()` and `present()` work together with `change()`.

The output header then says how many join and leave events there are and which variable gives the presence at the start. Section 10.4 gives an example.

8.3 Opportunities: restricting the candidates

`opportunities(evvar actvar)` restricts the candidate set of the choice sub-model for *each event* to a list of available actors you supply (`goldfish`: `opportunitiesList` of `estimationInit`, documented there for choice models only; `nwdynam` rejects it in the other sub-models). It takes

two numeric variables with *one row per (event, available actor) pair*, not one row per actor: *evvar* is the sequence number of the event (1 to the number of events, in the chronological order of the network) and *actvar* the number of an actor (1 to the number of actors, in the actor order of the network) who was available as a candidate receiver for that event. An actor not listed for an event is excluded from the risk set of that event only; the sender is always excluded, whether listed or not. As in `goldfish`, the realized receiver of every event must be listed as available for it (error 198 otherwise); rows with a missing value are refused (error 416).

Because `opportunities()` needs this event-by-actor shape, while the covariate effects need one row per actor, the opportunities data must be in a different frame when covariates are used, named by `oppframe(framename)`: the covariates stay in the dataset in memory, the two variables are read from the frame. Without `oppframe()` the combination stops with error 198; `change()` together with `opportunities()` also needs `oppframe()`.

```
. frame create opps
. frame opps: use opportunities, clear
. nwdynam mail, inertia alter(level) opportunities(evseq candid) oppframe(opps)
. nwdynam mail, inertia opportunities(evseq candid)
```

Section 10.5 builds the opportunities of the emails (the employees of the division of the sender, plus the realized receiver) and fits both forms.

Chapter 9

Postestimation

After `nwdynam`, five `estat` commands work for every sub-model, and two plots work for the sub-models they are made for. They offer what `goldfish` offers after estimation, with the same definitions and defaults: the log likelihood of every event (`returnIntervalLogL`), the model probabilities of the events (`returnEventProbabilities`), `logLik`, `AIC` and `BIC`, `examineOutliers` and `examineChangepoints`. `goldfish` has no goodness-of-fit or simulation function, and `nwdynam` adds none; the two plots `estat choiceplot` and `estat rateplot` (Section 9.6) are extensions that show how well the fitted model describes whom the actors choose and who acts. The commands are evaluated at the coefficients in `e(b)` by the same native code as the fit.

The data of the fit is kept in memory after `nwdynam`. It is replaced by the next fit and gone after `clear all`, and the commands stop with return code 498 when the estimation results on screen are not those of the fit in memory (for example after `estimates restore` of an earlier fit; the message is shown in Section 12.3). Standard Stata postestimation (`estimates`, `lrtest`, `test`, `lincom`) works on the usual `e(b)` and `e(V)`.

```
estat ic
estat eventlik   [, frame(name) replace]
estat eventprob  , frame(name) [replace maxrows(#)]
estat outliers   [, method(hampel|iqr|top) parameter(#) window(#) frame(name) replace
                   plot name(name) saving(file) export(file[, opts]) twoway_options]
estat changepoints [, moment(mean|variance) method(pelt|amoc|binseg) window(#) frame(name)
    replace
                   plot name(name) saving(file) export(file[, opts]) twoway_options]
estat choiceplot [, topk(numlist) maxrank(#) frame(name) replace noplot name(name)
                   saving(file) export(file[, opts]) maxrows(#) twoway_options]
estat rateplot   [, over(actor|time) periods(#) topactors(#) label(#) frame(name) replace noplot
                   name(name) saving(file) export(file[, opts]) maxrows(#) twoway_options]
```

The option `frame(name)` stores the table in a new frame and `replace` overwrites an existing frame. The graph options of the four plots are described together in Section 9.7. All examples of this chapter are in Section 10.10.

9.1 Information criteria

`estat ic` reports the log likelihood, the log likelihood per event, AIC and BIC. It stores in `r()`: `r(ll)`, `r(ll_per_event)`, `r(N)` (the records of the log likelihood, as `goldfish` counts them), `r(N_events)` (the events), `r(df)`, `r(aic)`, `r(bic)` and the matrix `r(S)`. The BIC uses the number of events as the sample size. In the rate model with intercept the number of records is larger than the number of events (below), which `goldfish` counts in the same way.

9.2 The log likelihood of every event

`estat eventlik` gives the log likelihood of every event: for the choice sub-model the log of the probability of the observed receiver; for the rate sub-model without intercept the log of the probability that the observed sender acts; for the coordination sub-model the log of the probability of the observed pair. It prints the number of events, the sum, the mean, the minimum and the maximum, and stores them as `r(N)`, `r(ll_events)` (the sum over the events), `r(ll)`, `r(ll_mean)`, `r(ll_min)` and `r(ll_max)`. With `frame()` the table has the variables `event`, `origevent` (the row of the event in the data), `time`, `sender`, `receiver` and `ll`.

In the rate model *with* intercept the log likelihood is a sum over the periods between events: besides one record per event, `goldfish` includes the right-censored periods that a covariate change or the expiry of a window ends (they are part of `e(ll)`). The table then has one row per event and puts the censored periods before an event in the variable `llcens`; `r(n_cens)` counts them and the part after the last event is in `r(ll_trailing)`.

9.3 The probabilities of the events

`estat eventprob` gives the model probabilities of every event; because the table has one row per event and candidate it always needs `frame()`. The variables are `event`, `time`, `actor` and `actorid` (`actor1`, `actor2` and their ids for the coordination sub-model), `observed` and `p`. The rows:

- choice: one row per event and candidate receiver, p the probability that the actor is the receiver; the probabilities of an event sum to 1 over its candidates (with composition change or `opportunities()` the candidates are those present or available);
- rate without intercept: one row per event and present actor, the probability that the actor is the sender;
- rate with intercept: for each actor `goldfish` returns the linear predictor minus the exposure of the period that ends at the event; this is not a probability and is stored as `lnrate`;
- coordination: one row per event and pair of actors (actor1 < actor2); `p` is `goldfish`'s matrix entry, twice the probability of the pair, so that the entries of an event sum to 1.

`maxrows(#)` limits the number of rows (events times candidates); the command stops with an error before it builds a larger table (default 200,000,000). `r(N_rows)` and `r(frame)` are stored.

9.4 Outliers

`estat outliers` flags events whose log likelihood is unusually low, with the rules of `goldfish`'s `examineOutliers`; `method()`, `parameter()` and `window()` choose the rule.

`method(hampel)`
: (the default) flags an event whose log likelihood differs from the median of the `window()` events on each side by more than `parameter()` (default 3) scaled median absolute deviations. The default window is half the number of events minus 1, as in `goldfish`, which tests only the middle event(s) of the log; give `window()` for a rolling rule.

`method(iqr)`
: flags log likelihoods below the median minus `parameter()`/2 interquartile ranges.

`method(top)`
: flags the `parameter()` events with the lowest log likelihood. Events whose log likelihoods agree to the last digits are ranked in an arbitrary order, so that `goldfish` may name another of them.

`plot` draws the log likelihood against time with the flagged events marked (the graph options are in Section 9.7). The command stores `r(n_outliers)`, `r(N)`, `r(parameter)`, `r(window)`, `r(method)` and the row vector `r(outliers)` of the event numbers in increasing order; with `frame()` the table has an indicator `outlier`. Figure 9.1 shows the plot of the example.

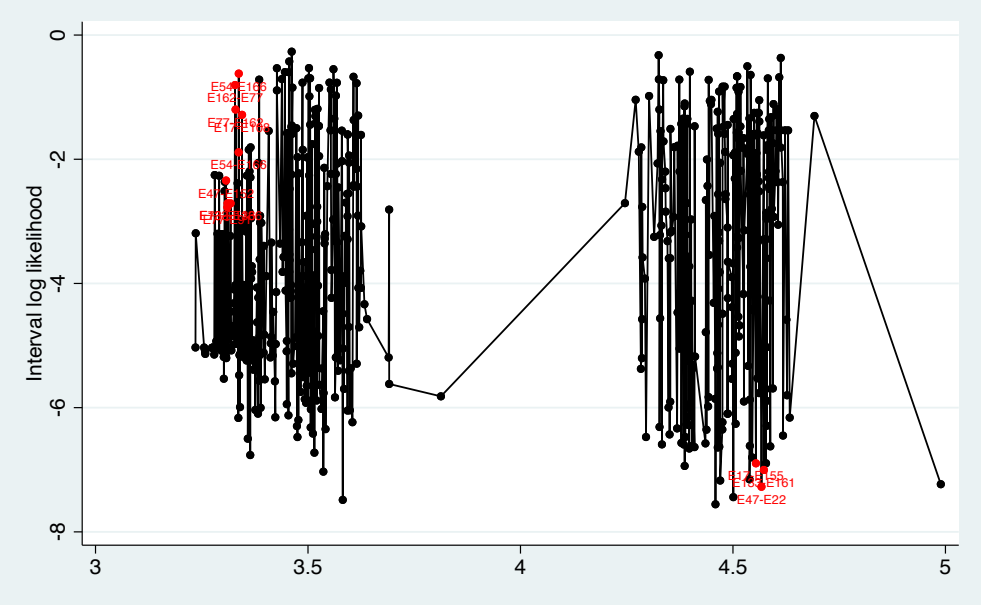


Figure 9.1: `estat outliers, window(50) plot` after the choice model of Section 10.10: the log likelihood of every event (the manufacturing emails), with the flagged events labelled sender–receiver.

9.5 Change points

`estat changepoints` looks for regime changes in the log likelihood over the event sequence, with the methods of `goldfish`'s `examineChangepoints`, which calls the R package `changepoint`. The methods are `method(pelt)` (the default), `method(amoc)` (at most one change) and `method(binseg)` (binary segmentation, up to 5 changes), for the `moment(mean)` (default) or the `moment(variance)` of the log likelihood. `window()` is the minimum segment length in events; the default is the largest number of events that share a time point, as in `goldfish`. The penalty is the modified Bayesian information criterion and the test statistic is the normal one.

The change points are the events that end a regime: the log likelihood of the following events belongs to the next regime. The command stores `r(n_cpts)`, `r(N)`, `r(window)`, `r(method)`, `r(moment)` and the row vector `r(cpts)` with the last event of every regime; the last entry is always the number of events, as in the R package. With `frame()` the table has an indicator `cpt`.

Credit. The change point methods (PELT, AMOC and binary segmentation, for the mean and the variance of normal data, with a minimum segment length and the modified BIC penalty) are a port of the C code and of the penalty and cost functions of the R package `changepoint` (Rebecca Killick, Kaylea Haynes, Idris Eckley, Paul Fearnhead and Jamie Lee), so that `estat changepoints` gives the same change points as `goldfish`.

9.6 Result plots: who is chosen and who acts

The two commands of this section are extensions: `goldfish` has no such plots. They use the model probabilities of `estat eventprob`, evaluated at `e(b)` by the same native code as the fit, and need the sub-model they are made for (return code 198 otherwise) and the fit in memory (return code 498, as for the other commands). They draw by default; `noplot` gives the numbers without the graph, which is useful with `frame()`.

9.6.1 estat choiceplot

`estat choiceplot` (sub-models `choice` and `choice_coordination`) asks how high the model ranks the receiver (for the coordination model, the pair) that was actually chosen. In every event it ranks the members of the risk set by the fitted probability; rank 1 is the most probable one, and members with an equal probability (the realised one included) form a tie group. The rank of the realised member, and so the mean and the median rank, is the mid rank of its tie group, the mean of the places the group occupies. A top-k share counts, for an event whose realised member ties with $m-1$ others for the places $g+1$ to $g+m$, the fraction $\min(1, \max(0, (k-g)/m))$ of the places up to k. This is the exact expectation of the share under a random order within the tie group, so the result is the same in every run and uses no random numbers (the mid rank would count the event as a hit only when the middle of the group lies within k). The model curve does not depend on ties. The graph shows, for $k = 1, 2, \ldots$, the share of events whose realised receiver is among the k most probable candidates, together with two references: what a model without information gives (every member of the risk set equally probable) and the expectation under the fitted model (the mean over the events of the sum of the k largest probabilities). A fitted model whose observed curve lies close to the model curve and well above the uniform one describes well who is chosen; an observed curve below the model curve means that the model is too confident. Figure 9.2 shows the plot of the example.

`topk(numlist)`
: Lists, for every k of the list, the share of events with rank at most k (default 1 3 5 10), against the uniform and the model expectation.

`maxrank(#)`
: The largest k of the graph (default 20; the curve ends where every risk set is covered).

`frame(name), replace`
: The per-event table with the variables `event`, `time`, `m` (the size of the risk set), `rank` (the mid rank), `pobs` (the fitted probability of the realised receiver) and `nties` (the size of the tie group of the realised member).

`maxrows(#)`
: As for `estat eventprob`.

The output also gives the mean and the median rank; when ties occurred, a line under the table (`Ties:`) gives the number of events whose realised member ties with others. The command stores `r(N)`, `r(N_ties)` (that number), `r(mean_rank)`, `r(median_rank)`, `r(mean_rank_uniform)`, `r(mean_rank_relative)` (the mean of the rank divided by the size of the risk set), `r(mean_riskset)`, the matrices `r(topk)` (columns k, observed, uniform, model) and `r(curve)` (the same for $k = 1, 2, \ldots$) and `r(frame)`.

9.6.2 estat rateplot

`estat rateplot` (sub-model `rate`, with or without `intercept`) adds up, over the events, the probability that each actor is the sender; this is the expected number of events of the actor as sender. For the rate model with `intercept` the probability is the share of the actor in the total hazard, exp(lnrate) divided by the sum over the actors, so that the expected counts always add up to the observed number of events: the plot checks who acts, not how many events occur.

`over(actor)`
: (the default) plots the observed against the expected number of events of every actor as sender, with the 45 degree line (Figure 9.3), and labels the `label(#)` (default 5) actors that deviate most.

`over(time)`
: cuts the log into `periods(#)` (default 10) periods of equal length in time and plots the observed and the expected number of events per period of the `topactors(#)` (default 5) actors with the largest expected count (Figure 9.4).

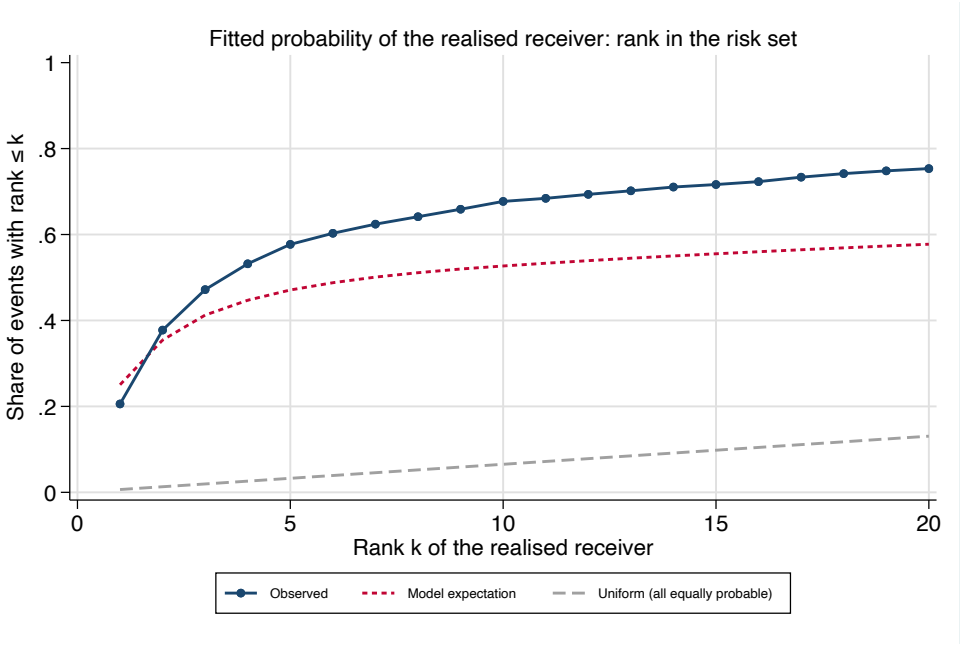


Figure 9.2: `estat choiceplot` after the choice model of Section 10.10: the share of the 631 emails whose receiver is among the k most probable candidates of the model, against the expectation under the fitted model and against a uniform choice among the candidates.

`frame(name), replace`
: The per-actor table with the variables `actorid`, `actor`, `observed` and `expected`; with `over(time)` the table has one row for every actor and period and the variables `period`, `tlo` and `thi` in addition.

`maxrows(#)`
: As for `estat eventprob`.

The command stores `r(N)` (the events), `r(N_actors)`, `r(corr)` (the correlation of observed and expected counts), `r(pearson)` (the sum of (observed − expected)^2/expected over the actors, descriptive only, not a test), `r(over)` and `r(frame)`.

9.7 Graph options of the plots

`estat outliers` and `estat changepoints` draw with `plot`, `estat choiceplot` and `estat rateplot` unless `noplot` is given. The graph options are the same for all four. They need the graph helper of the `nwcommands` core, version 3.29.0 or later.

`name(name)`
: Names the graph in memory; a trailing `, replace` is accepted (an existing graph of the name is always replaced). The default names of `estat choiceplot` and `estat rateplot` are `choiceplot` and `rateplot`, so that `graph display choiceplot` works without `name()`; `estat outliers` and `estat changepoints` draw the unnamed graph (Stata's `Graph`) when no name is given.

`saving(filename)`
: Saves the graph as a `.gph` file (the extension is added; an existing file is replaced).

`export(filename [, options])`
: Exports the graph with `graph export`; the extension sets the format (for example `.pdf` or `.png`), an existing file is replaced, and the options of `graph export` such as `width()` may follow a comma.

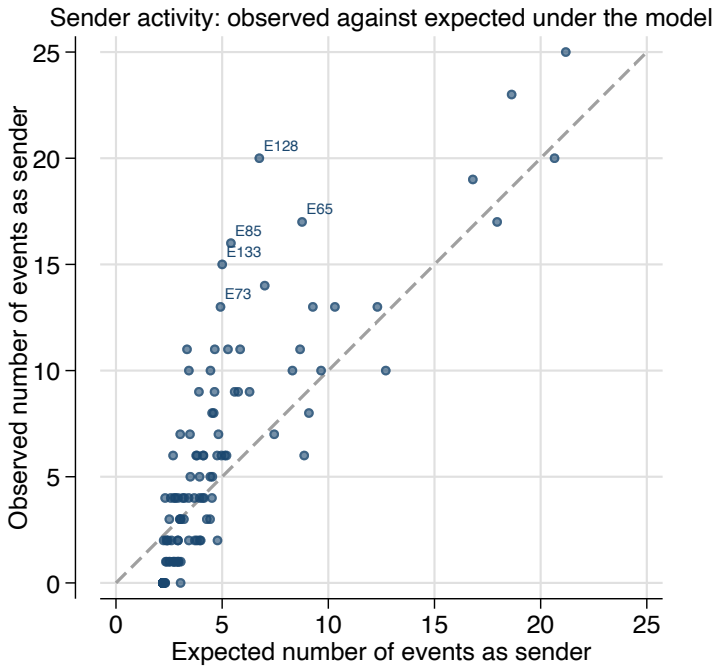


Figure 9.3: `estat rateplot` after the rate model without intercept of Section 10.10: observed against expected number of emails sent by every actor; the actors that deviate most are labelled.

twoway_options

Any option of `twoway` (`title()`, `xlabel()`, `ylabel()`, `scheme()`, `xsize()`, `ysize()`, ...); they come last and override the defaults of the graph. The default titles are black; `title("")` removes the default title. Axes that count events have integer labels.

With `plot` missing in `estat outliers` and `estat changepoints`, or `noplot` given in the two plots, the options `name()`, `saving()`, `export()` and the `twoway` options stop the command with return code 198 (only `name()` alone is accepted without `plot`). A mistyped graph name, or a file or directory that cannot be written, stops the command with a message before anything is computed. For example, the plot of Figure 9.1 was exported to a PDF file in the same call:

```
estat outliers, window(50) plot name(ol) export(fig/outliers.pdf)
estat choiceplot, topk(1 3 5 10) title("") saving(choice) export(choice.png, width(900))
graph display choiceplot
```

9.8 How the postestimation tools were checked

The two result plots are not part of this comparison, because `goldfish` has none. Per-event log likelihoods and probabilities agree with `goldfish` to 1e-13 on 90 random event logs (choice, rate, rate with intercept, coordination; with composition change, time-varying covariates, windows, two-mode networks, opportunities and effects of other networks); the outlier indices and the change points agree exactly (Chapter 11).

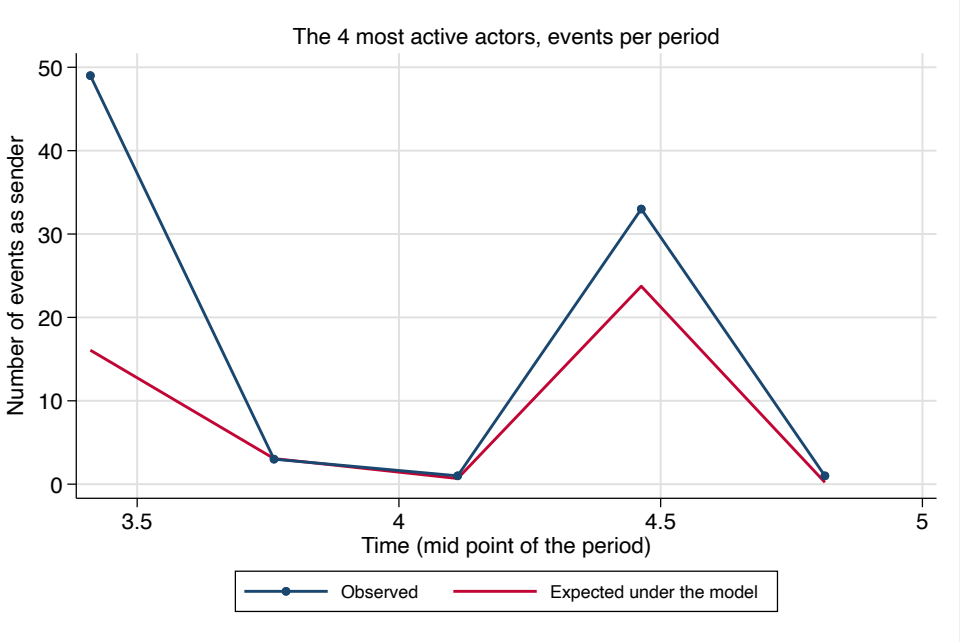


Figure 9.4: `estat rateplot, over(time) periods(5) topactors(4)` after the rate model with intercept: observed and expected emails per period of the four actors with the largest expected count.

Chapter 10

Worked examples

10.1 How the output was produced

All output in this chapter was produced by the do-files in the directory `manual/ex` of the repository, run with `nwdynam` 1.11.0 and the `nwcommands` core 3.33.0 in Stata/BE 17 on macOS (Apple silicon), with the native plugin. Each do-file marks the pieces of its log with a line `*# name`; the program `ex/mklistings.py` cuts the Stata logs at these markers into the listings that this manual includes. The data are the emails of a manufacturing company of Section 1.6, loaded from the web with `nwwebuse`, and the data of the comparison with `goldfish`. Fits without a `seed()` are deterministic here because the optimizer starts from fixed values. Nothing in this chapter is a substantive analysis; the models are chosen to show the commands, and the artificial data that some examples need (changes of a covariate, memberships, event values) are marked as such. The first example, with the declaration of the events, is in Chapter 3.

10.2 Effects and covariates

The file `ex02_effects.do` puts the level in the reporting hierarchy and the division of each employee into the data (Section 4.3) and fits the choice model with and without homophily on the division:

```
. quietly nwwebuse manufacturing_nodes, nwclear

. encode division, generate(div)

. tempfile actors

. quietly save `actors'

. mkmail 5

. nwload mail, xvars

. gen long row = _n

. gen label = _nwnode

. merge m:1 label using `actors', keep(master match) nogenerate

    Result                      Number of obs
    -----------------------------------------
    Not matched                             0
    Matched                               154
    -----------------------------------------

. sort row
```

```
. list label level div in 1/4

     +----------------------+
     | label   level    div |
     |----------------------|
  1. |  E102       2    E36 |
  2. |  E103       2    E85 |
  3. |  E104       1   E104 |
  4. |  E105       2   E104 |
     +----------------------+

. tabulate level

   Steps to |
 the CEO in |
       the |
  reporting |
  hierarchy |
  (0 = CEO) |      Freq.     Percent        Cum.
------------+-----------------------------------
          0 |          1        0.65        0.65
          1 |         11        7.14        7.79
          2 |         64       41.56       49.35
          3 |         33       21.43       70.78
          4 |         44       28.57       99.35
          5 |          1        0.65      100.00
------------+-----------------------------------
      Total |        154      100.00
```

```
. nwdynam mail, inertia recip indeg same(div)
------------------------------------------------------------
Dynamic Network Actor Model - choice sub-model (MLE)
Network: mail                          Actors: 154
Events: 631                            Log likelihood: -2.29e+03
Note: 3 events share a time point with an earlier event; events with equal
times are simultaneous (as in goldfish).
------------------------------------------------------------
------------------------------------------------------
        mail | Coefficient  Std. err.      z    P>|z|
-------------+----------------------------------------
     inertia |   1.712647    .140104    12.22   0.000
       recip |   2.870691   .1379323    20.81   0.000
       indeg |   .1142464   .0084152    13.58   0.000
        same |   .5954169   .1025453     5.81   0.000
------------------------------------------------------

. estimates store samediv

. estimates table base samediv, b(%9.3f) stats(ll N)

---------------------------------------
    Variable |   base       samediv
-------------+-------------------------
     inertia |     1.765       1.713
       recip |     2.922       2.871
       indeg |     0.110       0.114
        same |                 0.595
-------------+-------------------------
          ll | -2307.397   -2290.415
           N |       631         631
---------------------------------------

. lrtest samediv base
```

```
Likelihood-ratio test
Assumption: base nested within samediv

 LR chi2(1) =  33.96
Prob > chi2 = 0.0000
```

Employees write more to someone of their own division (odds ratio $e^{0.60} \approx 1.8$) and the likelihood-ratio test rejects the model without the effect ($\chi^2(1) = 33.96$). The coefficients of the other effects change little.

Closure effects: two-path closure (`trans`) has a clearly positive coefficient, embeddedness in transitive structures (`nodetrans`) a small one that is not significant at the 5 per cent level.

```
. nwdynam mail, inertia recip indeg trans
------------------------------------------------------------
Dynamic Network Actor Model - choice sub-model (MLE)
Network: mail                           Actors: 154
Events: 631                             Log likelihood: -2.30e+03
Note: 3 events share a time point with an earlier event; events with equal
times are simultaneous (as in goldfish).
------------------------------------------------------------
------------------------------------------------------
        mail | Coefficient  Std. err.      z    P>|z|
-------------+----------------------------------------
     inertia |   1.732437   .1417931    12.22   0.000
       recip |   2.978432   .1384791    21.51   0.000
       indeg |    .094776   .0091763    10.33   0.000
       trans |   .5027673   .1032202     4.87   0.000
------------------------------------------------------

. nwdynam mail, inertia recip indeg nodetrans
------------------------------------------------------------
Dynamic Network Actor Model - choice sub-model (MLE)
Network: mail                           Actors: 154
Events: 631                             Log likelihood: -2.31e+03
Note: 3 events share a time point with an earlier event; events with equal
times are simultaneous (as in goldfish).
------------------------------------------------------------
------------------------------------------------------
        mail | Coefficient  Std. err.      z    P>|z|
-------------+----------------------------------------
     inertia |   1.773269   .1423466    12.46   0.000
       recip |   2.868626   .1431842    20.03   0.000
       indeg |    .107881   .0084966    12.70   0.000
   nodetrans |    .042856   .0230401     1.86   0.063
------------------------------------------------------
```

Recency: `inertiawindow(0.5)` lets an email count as a previous contact for half a day only (the time unit is the day); `weightedinertia` replaces the presence of a tie by the number of earlier emails of the pair.

```
. nwdynam mail, inertiawindow(0.5) recip indeg
------------------------------------------------------------
Dynamic Network Actor Model - choice sub-model (MLE)
Network: mail                           Actors: 154
Events: 631                             Log likelihood: -2.31e+03
Note: 3 events share a time point with an earlier event; events with equal
times are simultaneous (as in goldfish).
------------------------------------------------------------
------------------------------------------------------
        mail | Coefficient  Std. err.      z    P>|z|
-------------+----------------------------------------
     inertia |   1.651976   .1348707    12.25   0.000
```

```
       recip |   3.225304    .126903    25.42   0.000
       indeg |   .1241489   .0077912    15.93   0.000
-------------------------------------------------------

. nwdynam mail, weightedinertia recip indeg
-------------------------------------------------------------
Dynamic Network Actor Model - choice sub-model (MLE)
Network: mail                           Actors: 154
Events: 631                             Log likelihood: -2.35e+03
Note: 3 events share a time point with an earlier event; events with equal
times are simultaneous (as in goldfish).
-------------------------------------------------------------
-------------------------------------------------------
        mail | Coefficient  Std. err.      z    P>|z|
-------------+-----------------------------------------
     inertia |   .4343532   .0509664     8.52   0.000
       recip |   3.499102   .1203137    29.08   0.000
       indeg |   .1305183   .0076684    17.02   0.000
-------------------------------------------------------
```

The log likelihood falls from −2307.4 (inertia without window) to −2312.8 with the window (a `goldfish` fit of the same model has the same value, Table 11.2), and to about −2350 with the weighted version; the models are not nested, so they are compared by log likelihood or information criteria, not by `lrtest`.

The rate model with a covariate: do employees who are higher up in the hierarchy (smaller `level`) write more often?

```
. nwdynam mail, submodel(rate) indeg outdeg
-------------------------------------------------------------
Dynamic Network Actor Model - rate sub-model, no intercept (MLE)
Network: mail                           Actors: 154
Events: 631                             Log likelihood: -2.99e+03
Note: 3 events share a time point with an earlier event; events with equal
times are simultaneous (as in goldfish).
-------------------------------------------------------------
-------------------------------------------------------
        mail | Coefficient  Std. err.      z    P>|z|
-------------+-----------------------------------------
       indeg |   .0324628   .0121141     2.68   0.007
      outdeg |   .2981233   .0176336    16.91   0.000
-------------------------------------------------------

. estimates store rate0

. nwdynam mail, submodel(rate) indeg outdeg ego(level)
-------------------------------------------------------------
Dynamic Network Actor Model - rate sub-model, no intercept (MLE)
Network: mail                           Actors: 154
Events: 631                             Log likelihood: -2.98e+03
Note: 3 events share a time point with an earlier event; events with equal
times are simultaneous (as in goldfish).
-------------------------------------------------------------
-------------------------------------------------------
        mail | Coefficient  Std. err.      z    P>|z|
-------------+-----------------------------------------
       indeg |   .0317156   .0119843     2.65   0.008
      outdeg |   .2891426   .0177914    16.25   0.000
         ego |  -.1205671   .0423816    -2.84   0.004
-------------------------------------------------------

. estimates store rate1

. lrtest rate1 rate0
```

```
Likelihood-ratio test
Assumption: rate0 nested within rate1

 LR chi2(1) =   8.12
Prob > chi2 = 0.0044
```

`estat ic` after the last fit:

```
. estat ic

Akaike's information criterion and Bayesian information criterion

------------------------------------------------------------------------------
       Model |        N           ll       ll/N    df        AIC        BIC
-------------+----------------------------------------------------------------
        mail |      631     -2982.22   -4.72618     3    5970.44   5983.782
------------------------------------------------------------------------------
Note: BIC uses N = number of events.
```

10.3 Effects of other networks

The file `ex03_special.do` builds the reporting hierarchy as an ordinary network in the actor order of `mail` (the tie $i \to j$ is present when j is the supervisor of i) and uses it in `netterms()`:

```
. * the reporting hierarchy as an ordinary (static) network in the actor order o
> f mail: the tie i -> j is present when j is the supervisor of i
. mata:
------------------------------------------------- mata (type end to exit) ------
: lab = st_sdata(., "label"); sup = st_sdata(., "supervisor"); n = rows(lab); M
> = J(n, n, 0)

: for (i=1; i<=n; i++) {
>     j = selectindex(lab :== sup[i])
>     if (length(j)) M[i, j[1]] = 1
> }

: st_matrix("HM", M)

: end
------------------------------------------------------------------------------

. nwset, mat(HM) name(hier) directed

. nwset
(2 networks)
--------------------
      mail
      hier
```

The odds of choosing a candidate are about 2.6 times higher ($e^{0.97}$) if the candidate is the sender's supervisor. The mixed closure `mixedtrans(hier mail)` counts the two-paths *sender* $\to$ *supervisor* $\to$ *candidate* in which the second step is an email, that is, the supervisor of the sender wrote to the candidate; every such path multiplies the odds by 3.4 ($e^{1.23}$).

```
. nwdynam mail, inertia recip indeg netterms(tie(hier))
------------------------------------------------------------
Dynamic Network Actor Model - choice sub-model (MLE)
Network: mail                          Actors: 154
Events: 631                            Log likelihood: -2.30e+03
Note: 3 events share a time point with an earlier event; events with equal
times are simultaneous (as in goldfish).
```

```
------------------------------------------------------------
------------------------------------------------------
        mail | Coefficient  Std. err.      z    P>|z|
-------------+----------------------------------------
     inertia |   1.715472   .1399473    12.26   0.000
       recip |   2.893127   .1385704    20.88   0.000
       indeg |   .1108786   .0082819    13.39   0.000
    tie_hier |    .971987   .1954003     4.97   0.000
------------------------------------------------------
```

```
. nwdynam mail, inertia recip indeg netterms(tie(hier) mixedtrans(hier mail))
------------------------------------------------------------
Dynamic Network Actor Model - choice sub-model (MLE)
Network: mail                          Actors: 154
Events: 631                            Log likelihood: -2.27e+03
Note: 3 events share a time point with an earlier event; events with equal
times are simultaneous (as in goldfish).
------------------------------------------------------------
------------------------------------------------------------
                mail | Coefficient  Std. err.      z    P>|z|
---------------------+--------------------------------------
             inertia |    1.65799   .1368726    12.11   0.000
               recip |   2.947109    .136496    21.59   0.000
               indeg |   .0969333   .0087325    11.10   0.000
            tie_hier |   1.150251   .1951143     5.90   0.000
mixedtrans_hier_mail |   1.234138   .1638582     7.53   0.000
------------------------------------------------------------
```

`tertiusdiff(level)`: the absolute difference between the level of the sender and the mean level of the people who wrote to the candidate; and `ignorerep`:

```
. nwdynam mail, inertia recip indeg netterms(tertiusdiff(level))
------------------------------------------------------------
Dynamic Network Actor Model - choice sub-model (MLE)
Network: mail                          Actors: 154
Events: 631                            Log likelihood: -2.30e+03
Note: 3 events share a time point with an earlier event; events with equal
times are simultaneous (as in goldfish).
------------------------------------------------------------
------------------------------------------------------------
             mail | Coefficient  Std. err.      z    P>|z|
------------------+-----------------------------------------
          inertia |   1.598431   .1432746    11.16   0.000
            recip |   2.944621   .1391022    21.17   0.000
            indeg |   .1170648   .0082195    14.24   0.000
tertiusdiff_level |  -.4821283   .1005803    -4.79   0.000
------------------------------------------------------------
```

```
. nwdynam mail, indeg trans ignorerep(trans)
------------------------------------------------------------
Dynamic Network Actor Model - choice sub-model (MLE)
Network: mail                          Actors: 154
Events: 631                            Log likelihood: -2.98e+03
Note: 3 events share a time point with an earlier event; events with equal
times are simultaneous (as in goldfish).
------------------------------------------------------------
------------------------------------------------------
        mail | Coefficient  Std. err.      z    P>|z|
-------------+----------------------------------------
       indeg |   .1459031   .0059559    24.50   0.000
       trans |  -.1497738   .1165686    -1.28   0.199
------------------------------------------------------
```

10.4 Changing covariates and composition change

The data for these two examples are *artificial* and serve to show the syntax. Ten employees, chosen at random, move one level up on day 4.2, which is declared as a change network; `alter(level)` then reads the level that holds at the time of each event:

```
. * artificial change events, for illustration: ten employees, chosen at random,
>  move one level up on day 4.2
. set seed 2026

. gen double u = runiform()

. sort u

. gen byte moved = _n <= 10

. sort row

. preserve

. keep if moved
(144 observations deleted)

. gen newlevel = level - 1

. gen label2 = label

. gen double tchg = 4.2

. nwset label label2 newlevel, eventtime(tchg) selfloop name(levelchg)

. restore

. nwset
(3 networks)
--------------------
      mail
      hier
      levelchg
```

```
. nwdynam mail, inertia recip indeg alter(level) change(level levelchg)
------------------------------------------------------------
Dynamic Network Actor Model - choice sub-model (MLE)
Network: mail                          Actors: 154
Events: 631                            Log likelihood: -2.31e+03
Note: 3 events share a time point with an earlier event; events with equal
times are simultaneous (as in goldfish).
------------------------------------------------------------
-----------------------------------------------------
        mail | Coefficient  Std. err.      z    P>|z|
-------------+---------------------------------------
     inertia |   1.761572   .1422571    12.38   0.000
       recip |   2.923674   .1402963    20.84   0.000
       indeg |   .1079362    .008374    12.89   0.000
       alter |  -.0643641   .0445449    -1.44   0.148
-----------------------------------------------------
```

For composition change, the employees who neither send nor receive an email before day 4.5 are declared absent at the start and join at day 4.5 (a defensible use would be a real membership list). The 54 absent employees are not in the risk sets before day 4.5, and the header says so:

```
. * artificial membership, for illustration of present() and compchange() (not a
>  recommended analysis):
```

```
. * the employees who neither send nor receive an email before day 4.5 are absen
> t at the start and join at day 4.5
. quietly merge m:1 label using 'early', keep(master match) nogenerate

. gen byte member0 = seen == 1

. tabulate member0

    member0 |      Freq.     Percent        Cum.
------------+-----------------------------------
          0 |         54       35.06       35.06
          1 |        100       64.94      100.00
------------+-----------------------------------
      Total |        154      100.00

. sort row

. preserve

. keep if member0 == 0
(100 observations deleted)

. gen label2 = label

. gen double tjoin = 4.5

. gen byte joined = 1

. nwset label label2 joined, eventtime(tjoin) selfloop name(membership)

. restore
```

```
. nwdynam mail, inertia recip indeg present(member0) compchange(membership)
------------------------------------------------------------
Dynamic Network Actor Model - choice sub-model (MLE)
Network: mail                          Actors: 154
Events: 631                            Log likelihood: -2.19e+03
Note: 3 events share a time point with an earlier event; events with equal
times are simultaneous (as in goldfish).
Composition change: 54 join/leave events (membership); presence at the start:
member0.
------------------------------------------------------------
------------------------------------------------------
        mail | Coefficient  Std. err.      z    P>|z|
-------------+----------------------------------------
     inertia |   1.721412   .1358423    12.67   0.000
       recip |   2.709818   .1355345    19.99   0.000
       indeg |   .0911802   .0084634    10.77   0.000
------------------------------------------------------
```

10.5 Opportunities

A per-event restriction of the candidates needs one row per event and available actor (Section 8.3). As an illustration, the candidates of every event are the employees of the sender's division, plus the realized receiver (which must be available). The file `ex06_opp.do` builds this data:

```
. nwload mail, xvars

. gen long actid = _n

. gen label = _nwnode
```

```
. quietly merge m:1 label using ‘actors’, keep(master match) nogenerate

. tempfile ids

. quietly save ‘ids’
```

```
. * the candidates of every event: the employees of the division of the sender,
> and the realised receiver
. quietly use ‘ev’, clear

. rename sender label

. quietly merge m:1 label using ‘ids’, keep(match) nogenerate keepusing(div)

. rename (label div) (sender sdiv)

. rename sender label

. tempfile evdiv

. quietly save ‘evdiv’

. quietly use ‘ids’, clear

. rename (label actid) (cand candid)

. keep cand candid div

. rename div sdiv

. joinby sdiv using ‘evdiv’

. keep evseq candid

. tempfile opp1

. quietly save ‘opp1’

. quietly use ‘ev’, clear

. rename receiver label

. quietly merge m:1 label using ‘ids’, keep(match) nogenerate keepusing(actid)

. rename actid candid

. keep evseq candid

. append using ‘opp1’

. quietly duplicates drop

. sort evseq candid

. count
  29,405

. bysort evseq: gen n = _N

. summarize n

    Variable |        Obs        Mean    Std. dev.       Min        Max
-------------+---------------------------------------------------------
```

```
           n |     29,405    75.78048    22.19273          2         84

. keep evseq candid

. tempfile opp

. quietly save `opp'
```

The risk set of an event now has 76 candidates on average, instead of 153. The choice model with the restriction, with the opportunities in the dataset in memory and with the covariates in memory and the opportunities in a frame:

```
. quietly use `opp', clear

. nwdynam mail, inertia recip indeg opportunities(evseq candid)
------------------------------------------------------------
Dynamic Network Actor Model - choice sub-model (MLE)
Network: mail                          Actors: 154
Events: 631                            Log likelihood: -1.45e+03
Note: 3 events share a time point with an earlier event; events with equal
times are simultaneous (as in goldfish).
------------------------------------------------------------
-----------------------------------------------------
        mail | Coefficient  Std. err.      z    P>|z|
-------------+---------------------------------------
     inertia |    1.43918   .1389894    10.35   0.000
       recip |   2.423174   .1438453    16.85   0.000
       indeg |   .1647833   .0109515    15.05   0.000
-----------------------------------------------------
```

```
. quietly use `ids', clear

. frame create opps

. frame opps: use `opp', clear
(Manufacturing company emails, 2010 (Nurek and Michalski 2020; CC0))

. nwdynam mail, inertia recip indeg alter(level) opportunities(evseq candid) opp
> frame(opps)
------------------------------------------------------------
Dynamic Network Actor Model - choice sub-model (MLE)
Network: mail                          Actors: 154
Events: 631                            Log likelihood: -1.45e+03
Note: 3 events share a time point with an earlier event; events with equal
times are simultaneous (as in goldfish).
------------------------------------------------------------
-----------------------------------------------------
        mail | Coefficient  Std. err.      z    P>|z|
-------------+---------------------------------------
     inertia |   1.431847    .138371    10.35   0.000
       recip |   2.419106    .143249    16.89   0.000
       indeg |   .1570361   .0114112    13.76   0.000
       alter |  -.1316713   .0617853    -2.13   0.033
-----------------------------------------------------
```

(The covariate `alter(level)` could not be combined with the opportunities in the same dataset.)

10.6 Event values

With a value for every event (artificial here: 1 or 2 at random) the weighted effects use the sum of the values of the earlier events of the pair:

```
. * event values (artificial, to show the syntax): every email gets the value 1
> or 2 at random
. quietly nwwebuse manufacturing, nwclear

. keep if t < 5
(49,602 observations deleted)

. set seed 7

. gen value = 1 + (runiform() < 0.3)

. tabulate value

      value |      Freq.     Percent        Cum.
------------+-----------------------------------
          1 |        437       69.26       69.26
          2 |        194       30.74      100.00
------------+-----------------------------------
      Total |        631      100.00

. nwset sender receiver value, directed eventtime(t) name(valued)

. nwdynam valued, inertia recip indeg
------------------------------------------------------------
Dynamic Network Actor Model - choice sub-model (MLE)
Network: valued                         Actors: 102
Events: 631                             Log likelihood: -2.19e+03
Note: 3 events share a time point with an earlier event; events with equal
times are simultaneous (as in goldfish).
------------------------------------------------------------
------------------------------------------------
      valued | Coefficient  Std. err.      z    P>|z|
-------------+----------------------------------
     inertia |   1.712204   .1357223    12.62   0.000
       recip |   2.658244   .1357849    19.58   0.000
       indeg |   .0869861   .0085657    10.16   0.000
------------------------------------------------

. nwdynam valued, weightedinertia recip indeg
------------------------------------------------------------
Dynamic Network Actor Model - choice sub-model (MLE)
Network: valued                         Actors: 102
Events: 631                             Log likelihood: -2.23e+03
Note: 3 events share a time point with an earlier event; events with equal
times are simultaneous (as in goldfish).
------------------------------------------------------------
------------------------------------------------
      valued | Coefficient  Std. err.      z    P>|z|
-------------+----------------------------------
     inertia |   .3096642   .0346541     8.94   0.000
       recip |   3.249779   .1156365    28.10   0.000
       indeg |   .1078956   .0079517    13.57   0.000
------------------------------------------------
```

10.7 Two-mode events

An employee (mode 1) writes to a division (mode 2: the division of the receiver, labelled `D` plus the identifier of the head of the division). With a two-mode network the default effects of the choice model are `inertia` and `indeg`, and of the rate model `outdeg`:

```
. * two-mode events: an employee (mode 1) writes to a division (mode 2: the divi
> sion of the receiver, labelled D plus the
```

```
. * identifier of the head of the division, so that a division is not mistaken f
> or an employee)
. quietly nwwebuse manufacturing_nodes, nwclear

. rename label receiver

. gen str5 recdiv = "D" + division

. keep receiver recdiv

. tempfile rdiv

. quietly save ‘rdiv’

. quietly nwwebuse manufacturing, nwclear

. keep if t < 5
(49,602 observations deleted)

. quietly merge m:1 receiver using ‘rdiv’, keep(match) nogenerate

. keep sender recdiv t

. rename recdiv division

. nwset sender division, twomode eventtime(t) name(tm)
file <temporary file> saved as .dta format

. nwset
(1 networks)
--------------------
      tm
```

```
. nwdynam tm
------------------------------------------------------------
Dynamic Network Actor Model - choice sub-model (MLE)
Network: tm                           Actors: 106
Events: 631                           Log likelihood: -982.6505
Note: 3 events share a time point with an earlier event; events with equal
times are simultaneous (as in goldfish).
------------------------------------------------------------
------------------------------------------------------
          tm | Coefficient  Std. err.      z    P>|z|
-------------+----------------------------------------
     inertia |   2.069237   .1191001    17.37   0.000
       indeg |   .0314054   .0021491    14.61   0.000
------------------------------------------------------

. nwdynam tm, submodel(rate) outdeg
------------------------------------------------------------
Dynamic Network Actor Model - rate sub-model, no intercept (MLE)
Network: tm                           Actors: 106
Events: 631                           Log likelihood: -2.83e+03
Note: 3 events share a time point with an earlier event; events with equal
times are simultaneous (as in goldfish).
------------------------------------------------------------
------------------------------------------------------
          tm | Coefficient  Std. err.      z    P>|z|
-------------+----------------------------------------
      outdeg |   .3439103   .0350013     9.83   0.000
------------------------------------------------------
```

10.8 The coordination sub-model

The same emails, taken as undirected contacts (a convenient illustration; emails are not coordination ties):

```
. quietly nwwebuse manufacturing, nwclear

. keep if t < 5
(49,602 observations deleted)

. nwset sender receiver, undirected eventtime(t) name(contact)

. nwset
(1 networks)
--------------------
      contact
```

```
. nwdynam contact, submodel(choice_coordination) inertia
------------------------------------------------------------
Dynamic Network Actor Model - choice_coordination sub-model (MLE)
Network: contact                        Actors: 102
Events: 631                             Log likelihood: -4.62e+03
Note: 3 events share a time point with an earlier event; events with equal
times are simultaneous (as in goldfish).
------------------------------------------------------------
-----------------------------------------------------
     contact | Coefficient  Std. err.      z    P>|z|
-------------+---------------------------------------
     inertia |   1.968822   .0460709    42.73   0.000
-----------------------------------------------------
```

```
. nwdynam contact, submodel(choice_coordination) inertia trans
------------------------------------------------------------
Dynamic Network Actor Model - choice_coordination sub-model (MLE)
Network: contact                        Actors: 102
Events: 631                             Log likelihood: -4.60e+03
Note: 3 events share a time point with an earlier event; events with equal
times are simultaneous (as in goldfish).
------------------------------------------------------------
-----------------------------------------------------
     contact | Coefficient  Std. err.      z    P>|z|
-------------+---------------------------------------
     inertia |   1.936148    .047259    40.97   0.000
       trans |   .1734283   .0275808     6.29   0.000
-----------------------------------------------------
```

With `indeg`, whose values are large, the fit starts from a quasi-Newton search that takes damped steps before the final Newton iterations (Section 13.3):

```
. nwdynam contact, submodel(choice_coordination) inertia indeg
------------------------------------------------------------
Dynamic Network Actor Model - choice_coordination sub-model (MLE)
Network: contact                        Actors: 102
Events: 631                             Log likelihood: -4.60e+03
Note: 3 events share a time point with an earlier event; events with equal
times are simultaneous (as in goldfish).
------------------------------------------------------------
-----------------------------------------------------
     contact | Coefficient  Std. err.      z    P>|z|
-------------+---------------------------------------
     inertia |   1.876263   .0495042    37.90   0.000
       indeg |   .0410616   .0062669     6.55   0.000
-----------------------------------------------------
```

10.9 Events with identical times

Coarsening the time stamps creates ties between times. Rounding the emails down to a quarter of a day gives 623 events that share a time point with an earlier event; the estimates differ from those of the exact times, and the option `sequential` gives yet another result (each event then sees the earlier events of its group):

```
. mkmail 5

. nwdynam mail, inertia recip indeg
------------------------------------------------------------
Dynamic Network Actor Model - choice sub-model (MLE)
Network: mail                          Actors: 154
Events: 631                            Log likelihood: -2.31e+03
Note: 3 events share a time point with an earlier event; events with equal
times are simultaneous (as in goldfish).
------------------------------------------------------------
------------------------------------------------------
        mail | Coefficient  Std. err.      z    P>|z|
-------------+----------------------------------------
     inertia |   1.764847   .1422701    12.40   0.000
       recip |   2.922381   .1401073    20.86   0.000
       indeg |   .1096229   .0083142    13.18   0.000
------------------------------------------------------

. quietly nwwebuse manufacturing, nwclear

. keep if t < 5
(49,602 observations deleted)

. replace t = floor(t * 4) / 4
(631 real changes made)

. nwset sender receiver, directed eventtime(t) name(coarse)

. nwdynam coarse, inertia recip indeg
------------------------------------------------------------
Dynamic Network Actor Model - choice sub-model (MLE)
Network: coarse                        Actors: 102
Events: 631                            Log likelihood: -2.58e+03
Note: 623 events share a time point with an earlier event; events with equal
times are simultaneous (as in goldfish).
------------------------------------------------------------
------------------------------------------------------
      coarse | Coefficient  Std. err.      z    P>|z|
-------------+----------------------------------------
     inertia |   1.460311   .1767983     8.26   0.000
       recip |   2.318483   .1698931    13.65   0.000
       indeg |   .0556754   .0106203     5.24   0.000
------------------------------------------------------

. nwdynam coarse, inertia recip indeg sequential
------------------------------------------------------------
Dynamic Network Actor Model - choice sub-model (MLE)
Network: coarse                        Actors: 102
Events: 631                            Log likelihood: -2.18e+03
Note: 623 events share a time point; processed one after another (option
sequential).
------------------------------------------------------------
------------------------------------------------------
      coarse | Coefficient  Std. err.      z    P>|z|
-------------+----------------------------------------
     inertia |   1.701936   .1356924    12.54   0.000
```

```
       recip |   2.674651    .135617    19.72   0.000
       indeg |   .0872648   .0085683    10.18   0.000
------------------------------------------------------
```

10.10 Postestimation

The choice model with the default effects, then the information criteria and the log likelihood of every event, in a frame:

```
. estat ic

Akaike's information criterion and Bayesian information criterion

------------------------------------------------------------------------------
       Model |        N           ll       ll/N    df        AIC        BIC
-------------+----------------------------------------------------------------
        mail |      631    -2307.397   -3.65673     3   4620.794   4634.136
------------------------------------------------------------------------------
Note: BIC uses N = number of events.

. return list

scalars:
                r(bic) =  4634.136085253767
                r(aic) =  4620.794167666144
                 r(df) =  3
           r(N_events) =  631
                  r(N) =  631
       r(ll_per_event) =  -3.656730719228323
                 r(ll) =  -2307.397083833072

matrices:
                  r(S) :  1 x 6
```

```
. estat eventlik, frame(evl)

Log likelihood of every event (goldfish: returnIntervalLogL), submodel choice
  Events: 631   sum: -2307.397084   mean:  -3.65673   min:  -7.55582   max:  -0.
> 26741
Table of the events in frame evl: event, origevent, time, sender, receiver, ll.

. frame evl: list in 1/5

     +-------------------------------------------------------------+
     | event   origev~t        time   sender   receiver          ll |
     |-------------------------------------------------------------|
  1. |     1          1    3.235255      E39        E81   -5.0304379 |
  2. |     2          2    3.235822      E39        E81    -3.191394 |
  3. |     3          3   3.2556601     E112       E136   -5.0311949 |
  4. |     4          4   3.2572801     E112        E60   -5.0665948 |
  5. |     5          5   3.2579629     E112        E57   -5.1007842 |
     +-------------------------------------------------------------+

. frame evl: summarize ll

    Variable |        Obs        Mean    Std. dev.       Min        Max
-------------+---------------------------------------------------------
          ll |        631   -3.656731    1.784158  -7.555819  -.2674075
```

The probabilities of every event, one row per event and candidate (96,543 rows: 631 events, 153 candidates each; the probability of every candidate of the first event is 1/153 because nothing has

happened yet):

```
. estat eventprob, frame(evp)

Model probabilities of every event (goldfish: returnEventProbabilities), submode
> l choice: 96543 rows in frame evp
  One row per event and candidate receiver: p = probability that the actor is
  the receiver; observed = the realised receiver.

. frame evp: list in 1/6

     +-------------------------------------------------------------+
     | event       time   actor   actorid   observed           p |
     |-------------------------------------------------------------|
  1. |     1   3.235255    E102         1          0   .00653595 |
  2. |     1   3.235255    E103         2          0   .00653595 |
  3. |     1   3.235255    E104         3          0   .00653595 |
  4. |     1   3.235255    E105         4          0   .00653595 |
  5. |     1   3.235255    E107         5          0   .00653595 |
     |-------------------------------------------------------------|
  6. |     1   3.235255    E108         6          0   .00653595 |
     +-------------------------------------------------------------+

. frame evp: summarize p

    Variable |        Obs        Mean    Std. dev.       Min        Max
-------------+---------------------------------------------------------
           p |     96,543    .0065359     .027742   .0003688   .7653612
```

The five events with the lowest log likelihood (`method(top)`). The rolling Hampel rule with a window of 50 events (`estat outliers, window(50)`) flags 12 events; its plot is Figure 9.1.

```
. estat outliers, method(top) parameter(5) frame(out)

Outliers of the log likelihood of the events, method top, parameter 5
(626 missing values generated)

  +-----------------------------------------------------+
  | event        time   sender   receiver           ll |
  |-----------------------------------------------------|
  |   324   3.5818291     E103       E141   -7.4849733 |
  |   471   4.4588542     E103       E107   -7.5558189 |
  |   512   4.5008559      E62        E88   -7.4409287 |
  |   579   4.5671058      E47        E22   -7.2737644 |
  |   631     4.98912      E17       E130   -7.2323627 |
  +-----------------------------------------------------+
Table of the events in frame out (variable outlier = 1 for the flagged events).

. return list

scalars:
          r(parameter) =  5
                  r(N) =  631
         r(n_outliers) =  5

macros:
              r(frame) : "out"
             r(method) : "top"

matrices:
           r(outliers) :  1 x 5
```

Change points in the mean of the log likelihood (PELT, minimum segment length 20 events), and in the variance by binary segmentation:

```
. estat changepoints, window(20)

Change points of the log likelihood of the events, pelt method, change in the me
> an, minimum segment length (window) 20
(1 real change made)
(1 real change made)
(1 real change made)

  +-------------------------------------------------------+
  | event        time   sender   receiver           ll |
  |-------------------------------------------------------|
  |   180    3.425231     E166       E128   -4.1383163 |
  |   435   4.3926392     E117        E47   -1.3500617 |
  |   492   4.4755092     E152        E62   -4.4874309 |
  +-------------------------------------------------------+
  The change points are the events that end a regime: the log likelihood of
  the following events belongs to the next regime.

. return list

scalars:
             r(window) =  20
                  r(N) =  631
             r(n_cpts) =  3

macros:
             r(moment) : "mean"
             r(method) : "pelt"

matrices:
               r(cpts) :  1 x 4
```

```
. estat changepoints, moment(variance) method(binseg) window(20)

Change points of the log likelihood of the events, binseg method, change in the
> variance, minimum segment length (window) 20
(1 real change made)

  +-------------------------------------------------------+
  | event      time   sender   receiver           ll |
  |-------------------------------------------------------|
  |   176   3.41816     E128       E166   -4.4574984 |
  +-------------------------------------------------------+
  The change points are the events that end a regime: the log likelihood of
  the following events belongs to the next regime.

. matrix list r(cpts)

r(cpts)[1,2]
     c1   c2
r1  176  631
```

The plot of the outliers (Figure 9.1), exported to a PDF file in the same call, and the rank plot of the choice model (Figure 9.2; the table of the rank of every event is stored in a frame):

```
. estat outliers, window(50) plot name(ol) export(../fig/outliers.pdf) xsize(5.5
> ) ysize(3.4)

Outliers of the log likelihood of the events, method hampel, parameter 3, window
>  50
(619 missing values generated)

  +-------------------------------------------------------+
```

```
  | event        time   sender   receiver          ll |
  |-------------------------------------------------|
  |    52   3.3072109      E47       E152    -2.342193 |
  |    60   3.3102779      E77        E91   -2.7719107 |
  |    62   3.3108101      E73       E136   -2.7097845 |
  |    65    3.317894     E133       E136   -2.7106356 |
  |    73    3.328438     E162        E77   -.80453111 |
  |    77    3.329479      E77       E162   -1.1997626 |
  |    84   3.3368061      E54       E166   -1.8882505 |
  |    85   3.3370249      E54       E166   -.62081277 |
  |   102   3.3448031      E17       E108   -1.2848734 |
  |   565   4.5540051      E17       E155   -6.8962552 |
  |   579   4.5671058      E47        E22   -7.2737644 |
  |   581   4.5727429     E133       E161     -7.00351 |
  +-------------------------------------------------+
```

```
. estat choiceplot, topk(1 3 5 10) name(cp) export(../fig/choiceplot.pdf) xsize(
> 5.5) ysize(3.6) frame(cpf)

Rank of the realised receiver among the risk set by the fitted probability (rank
>  1 = most probable), 631 events
  Mean rank 19.349 median rank 4.0 expected mean rank if all risk-set members
  were equally probable 77.000 mean risk-set size 153.00.

------------------------------------------------------------
       Top k |       observed        uniform          model
-------------+----------------------------------------------
           1 |         0.2058         0.0065         0.2504
           3 |         0.4719         0.0196         0.4125
           5 |         0.5771         0.0327         0.4710
          10 |         0.6771         0.0654         0.5268
------------------------------------------------------------
Ties: in 288 of the 631 events the realised receiver shares its fitted
probability with other members of the risk set. The rank is then the mid rank
of the tie group; a top-k share counts the fraction of the tie group's places
up to k (the exact expectation under a random order within the group; no
random numbers are used).
Table of the events in frame cpf: event, time, m (size of the risk set), rank (m
> id rank for ties), pobs (fitted probability of the realised receiver), nties (
> number of risk-set members with that probability, 1 = no tie).

. return list

scalars:
       r(mean_riskset) =  153
  r(mean_rank_relativ
    e)                 =  .1264669629077199
  r(mean_rank_uniform) =  77
        r(median_rank) =  4
          r(mean_rank) =  19.34944532488114
             r(N_ties) =  288
                  r(N) =  631

macros:
              r(frame) : "cpf"

matrices:
              r(curve) :  20 x 4
               r(topk) :  4 x 4
```

The log likelihood of every event after a rate model with intercept, whose log likelihood of -1.4359 is the sum of the records of Section 2.2.2:

```
. nwdynam mail, submodel(rate) intercept indeg outdeg
```

```
------------------------------------------------------------
Dynamic Network Actor Model - rate sub-model, with intercept (MLE)
Network: mail                           Actors: 154
Events: 631                             Log likelihood:   -1.4359
Note: 3 events share a time point with an earlier event; events with equal
times are simultaneous (as in goldfish).
------------------------------------------------------------
------------------------------------------------
        mail | Coefficient  Std. err.      z    P>|z|
-------------+----------------------------------
   Intercept |   .3928353   .0566414     6.94   0.000
       indeg |   .0137263   .0119912     1.14   0.252
      outdeg |   .1857424   .0158219    11.74   0.000
------------------------------------------------

. estat eventlik, frame(evr)

Log likelihood of every event (goldfish: returnIntervalLogL), submodel rate
  Events: 631   sum:    -1.435943   mean:  -0.00228   min: -1.48e+02   max:   2.
> 54735
Table of the events in frame evr: event, origevent, time, sender, receiver, ll.

. return list

scalars:
        r(ll_trailing) =  0
             r(n_cens) =  0
             r(ll_max) =  2.547346975547184
             r(ll_min) =  -148.0840096961948
            r(ll_mean) =  -.002275662733242
                 r(ll) =  -1.435943184675761
          r(ll_events) =  -1.435943184675706
                   r(N) =  631

macros:
              r(frame) : "evr"
```

The plots of the rate models: observed against expected number of emails sent by every actor after the rate model without intercept (Figure 9.3), and observed and expected emails per period after the rate model with intercept (Figure 9.4):

```
. nwdynam mail, submodel(rate)
------------------------------------------------------------
Dynamic Network Actor Model - rate sub-model, no intercept (MLE)
Network: mail                           Actors: 154
Events: 631                             Log likelihood: -2.99e+03
Note: 3 events share a time point with an earlier event; events with equal
times are simultaneous (as in goldfish).
------------------------------------------------------------
------------------------------------------------
        mail | Coefficient  Std. err.      z    P>|z|
-------------+----------------------------------
       indeg |   .0324628   .0121141     2.68   0.007
      outdeg |   .2981233   .0176336    16.91   0.000
------------------------------------------------

. estat rateplot, name(ra) export(../fig/rateplot_actor.pdf) xsize(4.5) ysize(4.
> 4) frame(raf)

Observed and expected number of events per actor as sender, 154 actors, 631 even
> ts
  Correlation of observed and expected counts 0.8444 Pearson statistic sum
  (obs-exp)^2/exp 355.39 over 154 actors (descriptive only, not a test).
```

```
  +----------------------------+
  | actor   observed   expected |
  |----------------------------|
  |   E17         25      21.18 |
  |   E47         20      20.65 |
  |   E90         23      18.63 |
  |  E136         17      17.95 |
  |   E36         19      16.81 |
  |  E152         10      12.71 |
  |   E33         13      12.31 |
  |  E126         13      10.31 |
  |  E117         10       9.66 |
  |  E166         13       9.27 |
  +----------------------------+
Table in frame raf: actorid, actor, observed, expected.

. return list

scalars:
           r(pearson) =  355.3919902353261
              r(corr) =  .8443716133643435
           r(N_actors) =  154
                 r(N) =  631

macros:
             r(frame) : "raf"
              r(over) : "actor"

. nwdynam mail, submodel(rate) intercept indeg outdeg
------------------------------------------------------------
Dynamic Network Actor Model - rate sub-model, with intercept (MLE)
Network: mail                          Actors: 154
Events: 631                            Log likelihood:   -1.4359
Note: 3 events share a time point with an earlier event; events with equal
times are simultaneous (as in goldfish).
------------------------------------------------------------
-----------------------------------------------------
        mail | Coefficient  Std. err.      z    P>|z|
-------------+---------------------------------------
   Intercept |   .3928353   .0566414     6.94   0.000
       indeg |   .0137263   .0119912     1.14   0.252
      outdeg |   .1857424   .0158219    11.74   0.000
-----------------------------------------------------

. estat rateplot, over(time) periods(5) topactors(4) name(rt) export(../fig/rate
> plot_time.pdf) xsize(5.5) ysize(3.6)

Observed and expected number of events per actor as sender, 154 actors, 631 even
> ts
  Correlation of observed and expected counts 0.8852 Pearson statistic sum
  (obs-exp)^2/exp 471.19 over 154 actors (descriptive only, not a test).
  Rate model with intercept: the expected count of an actor is the sum over
  the events of its share of the total hazard, exp(lnrate) divided by the sum
  over the actors; the expected counts add up to the observed number of
  events, so the plot checks who acts, not how many events occur.

  +-----------------------------------------------+
  | period     tlo     thi   observed   expected |
  |-----------------------------------------------|
  |      1   3.235   3.586         49      16.07 |
  |      2   3.586   3.937          3       3.07 |
  |      3   3.937   4.288          1       0.68 |
  |      4   4.288   4.638         33      23.74 |
  |      5   4.638   4.989          1       0.22 |
```

```
+-----------------------------------------------+
```

10.11 Stored results

`ereturn list` after the rate model of Chapter 3 is shown there.

Chapter 11

Validation against goldfish

`nwdynam` is a reimplementation, and the only reason to trust its numbers is that they agree with the program that it reimplements. This chapter describes how the comparison with `goldfish` 1.6.12 (R 4.6) was done, shows a comparison that was run for this manual, and lists the places where the two programs differ. Appendix A gives the translation of the commands and Chapter 12 lists what `nwdynam` does not do.

11.1 How the comparison was done

The comparisons are of three kinds. The package's tests (the do-files in the directory `cscripts`) rerun the first two without R; the scripts that produced the reference values from `goldfish` are in the directory `dev`.

1. **The compiled plugin against reference evaluators.** Every likelihood is also written in Mata, as plain reference code that does not belong to the package's estimation path and was verified against `goldfish` while it was developed. The plugin is compared with these evaluators on random small event sequences: with simultaneous events, event values (also zero and negative), the option `sequential`, windows, weighted effects, two-mode networks, `opportunities()`, `tie()` and every subset of the effects of a sub-model. The log likelihood and the gradient must agree to a relative 10^{-9} and the Hessian with the numerical derivative of the reference gradient to 10^{-5}.
2. **Random event logs against goldfish.** Event logs (random or hand-built), including attribute changes, memberships, windows and networks, were fitted by `goldfish`; its estimates, standard errors and log likelihoods are stored with the package, and the tests compare `nwdynam` with them. Table 11.1 lists the sets.
3. **The Social Evolution data and the emails.** The data set that ships with `goldfish` (439 calls among 84 students) and the first 631 emails of the manufacturing data were fitted by both programs. Section 11.2 shows this comparison, rerun for this manual.

For the coordination sub-model the standard errors are numerical (from a numerical derivative of the likelihood, which is flat and not concave) in both programs and agree to 10 per cent. Effects that `goldfish` cannot compute, or computes with an ordering of events that `nwdynam` does not reproduce, are refused with a message (Chapter 12), not silently approximated. The tolerances are those of the tests; a relative tolerance of 10^{-6} is not a claim of agreement to the last digit of every number, and a fit that ends in a different local maximum of the coordination likelihood would show up as a larger difference, not as a smaller one.

11.2 A comparison run for this manual

To show what these comparisons look like, eight models were fitted by both programs on two data sets: the Social Evolution calls (439 events among 84 actors, of whom 54

Features	Logs	Tolerance
composition change (`present()`, `compchange()`), together with time-varying covariates, in all sub-models, one- and two-mode	67 random and fixed	estimates, standard errors, log likelihood 10^{-6}
time-varying covariates (`change()`), every attribute effect, replace and increment, changes at the time of an event, a window, several variables	42 random	the same; coordination: estimates 10^{-5}, standard errors 10%
windowed rate effects with `intercept`, `tertiuswindow()`, windows and weights in the coordination sub-model, effects in two-mode networks, `opportunities()` with covariates, `change()` in two-mode networks and with `opportunities()`	306 random	the same; coordination: estimates $5 \cdot 10^{-4}$, standard errors 10%
effects of other networks (`tie()`, closures, mixed closures, `tertiusdiff`, `ignorerep`, with windows and weights, static and changing networks)	about 300 random	the same; coordination: estimates $5 \cdot 10^{-4}$, standard errors 10%
postestimation (`estat`): per-event log likelihoods and probabilities, `logLik`, AIC, BIC, outliers, change points	90 random	10^{-10} (values agree to 10^{-13}); outlier indices and change points exact

Table 11.1: The comparisons with `goldfish` 1.6.12 on random and hand-built event logs.

are active) and the first 631 emails among 154 employees. The Stata side is the do-file `manual/ex/ex91_compare_stata.do`, the R side `manual/ex/r/compare.R`; both read the same event files, exported by `ex90_export_for_r.do` (the time stamps in full precision). The models are the choice model with `inertia`, `recip`, `indeg` (with `trans` and with a window of half a day on `inertia` for the emails), and the rate model with `indeg` and `outdeg` without and with an intercept. Table 11.2 shows the estimates, the relative differences and the standard errors.

The largest relative difference of an estimate is 1.3e-06 (Social Evolution, choice, `indeg`), that of a standard error 5.3e-07 and that of a log likelihood 1.6e-13. The agreement does not depend on the kind of model: all differences are of the same small size in the choice and in the rate models, with and without an intercept, with a window and with a closure effect.

11.3 Where the two programs differ

- **Specification.** Effects are options in `nwdynam`, a formula in `goldfish`. Without effect options `nwdynam` fits the default effects of the sub-model (Section 5.3).
- **Starting values and optimizer.** Both maximize the same likelihood, so that the estimates agree to the tolerance of their convergence criteria; the optimizers are different (Section 13.3).
- **The `tertius` effect for an actor with no in-neighbours.** The documentation of `goldfish` describes an imputation of the average over the actors that have in-neighbours; `goldfish` 1.6.12 itself computes 0 at estimation time, which is what `nwdynam` does.
- **Simultaneous events.** `nwdynam` follows `goldfish` (Section 2.3); the option `sequential` is an addition.
- **Defaults.** `nwdynam` has no formula that lists all effects; compare the translation table in Appendix A.
- **Postestimation.** Nothing that `goldfish` does not have: no simulation and no goodness of fit.

Before using `nwdynam` for an important result, run the same model in `goldfish`.

Effect	nwdynam	goldfish	rel. diff.	s.e. nwdynam	s.e. goldfish
Social Evolution, choice: inertia, recip, indeg; log likelihood -695.4608 (nwdynam), -695.4608 (goldfish)					
`inertia`	5.36556	5.36556	4.1e-07	0.19375	0.19375
`recip`	1.38944	1.38944	4.9e-07	0.16729	0.16729
`indeg`	-0.17353	-0.17353	1.3e-06	0.09097	0.09097
Social Evolution, rate: indeg, outdeg; log likelihood -1750.9123 (nwdynam), -1750.9123 (goldfish)					
`indeg`	0.55144	0.55144	3.4e-08	0.06634	0.06634
`outdeg`	0.26378	0.26378	9.6e-09	0.02839	0.02839
Social Evolution, rate with intercept: indeg, outdeg; log likelihood -6008.9834 (nwdynam), -6008.9834 (goldfish)					
`Intercept`	-14.15642	-14.15642	4.3e-08	0.06749	0.06749
`indeg`	0.76226	0.76226	3.2e-07	0.06018	0.06018
`outdeg`	0.25684	0.25684	7.8e-08	0.02909	0.02909
Emails, choice: inertia, recip, indeg; log likelihood -2307.3971 (nwdynam), -2307.3971 (goldfish)					
`inertia`	1.76485	1.76485	6.5e-09	0.14227	0.14227
`recip`	2.92238	2.92238	9.9e-08	0.14011	0.14011
`indeg`	0.10962	0.10962	4.2e-08	0.00831	0.00831
Emails, choice: inertia, recip, indeg, trans; log likelihood -2296.4872 (nwdynam), -2296.4872 (goldfish)					
`inertia`	1.73244	1.73244	9.3e-08	0.14179	0.14179
`recip`	2.97843	2.97843	3.2e-07	0.13848	0.13848
`indeg`	0.09478	0.09478	1.9e-07	0.00918	0.00918
`trans`	0.50277	0.50277	5.6e-07	0.10322	0.10322
Emails, choice: inertia (window 0.5), recip, indeg; log likelihood -2312.8474 (nwdynam), -2312.8474 (goldfish)					
`inertia`	1.65198	1.65198	5.3e-08	0.13487	0.13487
`recip`	3.22530	3.22530	5.1e-08	0.12690	0.12690
`indeg`	0.12415	0.12415	4.3e-08	0.00779	0.00779
Emails, rate: indeg, outdeg; log likelihood -2986.2823 (nwdynam), -2986.2823 (goldfish)					
`indeg`	0.03246	0.03246	2.7e-08	0.01211	0.01211
`outdeg`	0.29812	0.29812	1.9e-09	0.01763	0.01763
Emails, rate with intercept: indeg, outdeg; log likelihood -1.4359 (nwdynam), -1.4359 (goldfish)					
`Intercept`	0.39284	0.39284	4.0e-10	0.05664	0.05664
`indeg`	0.01373	0.01373	2.1e-10	0.01199	0.01199
`outdeg`	0.18574	0.18574	1.2e-10	0.01582	0.01582

Table 11.2: Estimates and standard errors of `nwdynam` (Stata 17, macOS) and `goldfish` 1.6.12 (R 4.6.0) on the same event logs.

Chapter 12

Limits

This chapter lists what `goldfish` does and `nwdynam` does not, the combinations of options that are refused (with the messages as `nwdynam` prints them), and the limits of estimation. All messages in this chapter are the output of `manual/ex/ex07_limits.do`.

12.1 What goldfish does that nwdynam does not

`nwdynam` supports what `goldfish` 1.6.12 offers for DyNAMs with the exceptions below.

- `ignoreRep` in the rate sub-model (`goldfish` accepts it; its likelihood could not be reproduced).
- `tertiusDiff` with time-varying attributes or in two-mode networks.
- `tertius`, `indeg`, `outdeg`, `recip` and `nodeTrans` computed through another network (the closure effects, `tie` and the mixed closures are supported).
- Effects of two-mode networks that `goldfish` itself cannot compute: `nwdynam` refuses them with a message.
- With `intercept` in the rate sub-model: a `change()` of the `tertius()` variable together with `tertiuswindow()`, because `goldfish` orders such a change and the expiry of the window in a way that `nwdynam` does not reproduce.
- Data preparation is different (instead of `goldfish`'s objects, an event list declared with `nwset ..., eventtime()`); Appendix A.

`goldfish` has no goodness-of-fit or simulation function, and `nwdynam` adds none.

12.2 Refusals

An effect option that does not apply to the chosen sub-model is rejected with a message and return code 198, never silently ignored. The following listings show the messages of the common refusals.

Effects of the wrong sub-model. `recip` and the other dyadic effects belong to the choice sub-model; `ego()` to the rate sub-model; `intercept` to the rate sub-model; the terms of `netterms()` and `opportunities()` do not exist in the rate sub-model.

```
. capture noisily nwdynam mail, submodel(rate) recip
inertia/recip/same()/diff()/sim()/alter()/trans/cycle/commonsender/commonrecei
> ver/four/egoalterint()/tie() are choice sub-model effects
(submodel(choice)), not available with submodel(rate) - see nwdynam.
invalid syntax

. display "return code: " _rc
return code: 198
```

```
. capture noisily nwdynam mail, ego(x)
ego() is a rate sub-model effect (submodel(rate)), not available with
submodel(choice) - see nwdynam.
invalid syntax

. display "return code: " _rc
return code: 198
```

```
. capture noisily nwdynam mail, intercept
intercept only applies to submodel(rate) - see nwdynam.
invalid syntax

. display "return code: " _rc
return code: 198
```

```
. capture noisily nwdynam mail, submodel(rate) netterms(trans(mail))
netterms() is available for submodel(choice) and submodel(choice_coordination)
only: goldfish's rate sub-model has no effect on another network in this
release.

. display "return code: " _rc
return code: 198
```

```
. capture noisily nwdynam mail, submodel(rate) opportunities(x x)
opportunities() only applies to the choice sub-model (goldfish's own
opportunitiesList is choice-only) - see nwdynam.
invalid syntax

. display "return code: " _rc
return code: 198
```

Bad values. A window must be a positive number; the sub-model must be one of the three.

```
. capture noisily nwdynam mail, inertiawindow(0)
inertiawindow() must be a positive number (in the same time units as the
network's own eventtime()).
invalid syntax

. display "return code: " _rc
return code: 198
```

```
. capture noisily nwdynam mail, submodel(frequency)
submodel() must be choice, rate, or choice_coordination - see nwdynam.
invalid syntax

. display "return code: " _rc
return code: 198
```

The wrong kind of network. The coordination sub-model needs an undirected network; every sub-model needs a network declared with `eventtime()`.

```
. capture noisily nwdynam mail, submodel(choice_coordination)
submodel(choice_coordination) requires an UNDIRECTED network (goldfish's own
coordination model is for undirected tie-formation events) - declare mail via
nwset ..., undirected eventtime() - see nwdynam.
invalid syntax

. display "return code: " _rc
return code: 198
```

```
. nwset, mat((0,1\1,0)) name(plain) directed

. capture noisily nwdynam plain
nwdynam requires a network declared via nwset's eventtime() option (an
event-type temporal network) - see nwset.
invalid syntax

. display "return code: " _rc
return code: 198
```

Two-mode networks. Many effects are refused in two-mode networks (Section 6.7):

```
nwdynam.
invalid syntax

. display "return code: " _rc
return code: 198
```

Events that cannot be modeled. An event of an actor to itself is refused when the model is fitted (the network itself is accepted by `nwset`):

```
. clear

. input sender receiver t

        sender   receiver          t
  1. 1 2 1
  2. 2 2 2
  3. 2 1 3
  4. end

. nwset sender receiver, directed eventtime(t) name(loops)

. capture noisily nwdynam loops
nwdynam: event 2 has sender == receiver (self-loop); DyNAM's choice sub-model
is defined over i != j dyads only.

. display "return code: " _rc
return code: 3498
```

Other combinations that are refused (all with return code 198; the message names the combination): `present()` together with `opportunities()`; `opportunities()` with a covariate option without `oppframe()`; `change()` with `opportunities()` without `oppframe()`; `change()` for a variable that no covariate option uses; the terms of `netterms()` for two-mode networks, with `submodel(rate)`, or with `change()`; an event of an absent actor; a change event of a covariate for an absent actor; `increment` without `change()`.

12.3 After the fit

The `estat` commands read the data of the fit in memory (Chapter 9). After `estimates restore` of an earlier fit, or after `clear all`, they stop with return code 498:

```
. mkmail 5

. quietly nwdynam mail, inertia recip indeg

. estimates store m1
```

```
. quietly nwdynam mail, inertia

. estimates restore m1
(results m1 are active now)

. capture noisily estat ic
no fitted nwdynam model available for this command (the data of the fit is
kept in memory after nwdynam and is gone after clear all, a later fit, or when
the estimates come from estimates restore of another fit). Run nwdynam again.

. display "return code: " _rc
return code: 498
```

12.4 Limits of estimation

The coordination sub-model. The log likelihood of the coordination sub-model is not concave and can be flat. `nwdynam` first runs a damped quasi-Newton search, then several starting values, keeps the best converged result and, as a last resort, a simplex search (Section 13.3); when none converges it stops with the message of Mata's optimizer and return code 111. When this happens, drop or recode an effect, or use a different set of effects.

Effects that are not identified. An effect is flagged when its standard error is missing, zero or above 100: `e(unidentified)` lists them and a note under the table says so. Such an estimate must not be interpreted. Typical reasons are too few events for the effect, a perfectly predicted pattern in the data (for instance a covariate that is constant within the candidate sets of every event, as `same()` when the opportunities are restricted to the sender's own division), and effects that are linear functions of each other (`diff()` and `sim()` of the same variable).

Small logs. A handful of events on very few actors has too little information to pin down the coefficients; wide confidence intervals are to be expected. The precision of a coefficient grows with the number of events.

Size. The number of rows of the table of `estat eventprob` is the number of events times the number of candidates and is limited by `maxrows()`. The fits themselves are fast (Section 13.4); the largest fit that was timed has 20,000 events among 100 actors.

Platforms. The plugins for Linux and Windows are built by the build workflow of the repository; the package is tested on Linux and Windows with StataNow 19.5 MP, and the benchmark of Section 13.4 ran on Linux.

Chapter 13

Technical notes

13.1 The native plugin

Every fit of `nwdynam` runs in a compiled plugin (`dynam_sim`, written in C): the log likelihood, the gradient and the exact Hessian of all sub-models, effects, windows, weighted effects, two-mode networks, `opportunities()`, `tie()` and the terms of `netterms()` are evaluated natively, and Mata's optimizer calls the plugin for each evaluation. The postestimation commands run in the same plugin. There is *no* slower fallback: Mata code that evaluates the likelihoods exists only as reference code for the tests of the package (Section 11.1). If the plugin file of the platform is missing, `nwdynam` stops with return code 601 and says which file is missing; if the installed plugin is older than the `.ado` files or cannot be loaded, with return code 3495.

The package installs plugins for macOS (a universal binary), Linux and Windows. The C source is part of the package: `net get nwdynam` (with the same `from()` as the installation) copies it into the current directory, together with a file that explains how to build the plugin; it is also in the folder `native` of the repository, https://github.com/thomas-u-grund/nwdynam. `net install` installs the compiled plugins only. On macOS the plugin can be rebuilt with `make -C native macos`; `nwinstall, buildplugin(nwdynam)` fetches the source, compiles it on your computer (macOS and Linux) and installs it. Windows is not tested yet.

13.2 The Mata part

The Mata library of the package, `lnwdynam.mlib`, prepares the event stream (sorting by time, grouping events with identical times, the actors, the covariate and change tables, the risk sets), writes the model into the data of the plugin, runs the optimizer and returns the estimates to the `.ado` file. The library is built with Stata 17, the oldest tested version, because a library built by a newer Stata is ignored by an older one (`r(3499)`); `nwinstall, buildmata(nwdynam)` rebuilds it with your own Stata. The library also depends on the Mata library of the core, and `nwdynam` says so if the two were not built together.

13.3 Optimization

Newton's method. The log likelihoods of the choice and the rate sub-model are concave. They are maximized by damped Newton steps on the analytic gradient and Hessian, starting from $\beta = 0$: a full step, halved until the log likelihood does not fall. The iteration stops when the Newton decrement $g^\top(-H)^{-1}g$ is below 10^{-12}, or below 10^{-13} times the absolute value of the log likelihood where round-off does not allow less, within at most 100 iterations. The covariance matrix is $(-H)^{-1}$ at the maximum. For the rate sub-model with intercept the start is a vector computed from the data.

Fallbacks. If the Newton iterations do not converge (a Hessian that is not negative definite, effects that are not identified), up to eight starts of BFGS on the analytic gradient follow, the first start fixed and the others drawn at random in an increasing range (with `seed()` they are reproducible); the first converged start wins and Newton polishes it. A Nelder–Mead search from a fixed start is the last resort.

The coordination sub-model. Its log likelihood is not concave. All starting values are run, with BFGS on the gradient (the Hessian is numerical), and the best converged result is kept. Its standard errors come from a numerical derivative.

Reproducibility. Fits that Newton's method solves from zero do not use random numbers and need no seed. Fits that fall back on random starting values, and every fit of the coordination sub-model, depend on the seed; give `seed()` to make them reproducible. Results are reproducible on the same platform; across platforms they agree to the accuracy of the optimizer, not to the last digit.

13.4 Performance

The run times of `nwdynam` 1.8.0 were compared with `goldfish` on simulated event logs of 200 to 20,000 events among 20 to 100 actors. Four effect sets were used: C1, the default choice effects (`inertia`, `recip`, `indeg`); C2, adding `same()`; C3, `inertia` and `trans`; R1i, the rate sub-model with `intercept` and the default effects. `goldfish` has three engines; `default_c` and `gather_compute` took the same time, and `default` was 1.8 to 15 times slower. Table 13.1 gives a selection of the cells (seconds per fit, medians). `nwdynam` was faster than `goldfish` in every

Cell	nwdynam 1.8.0	goldfish `default_c`	goldfish `default`
C1, 200 events, 20 actors	0.038	0.256	0.463
C1, 1000 events, 30 actors	0.043	1.254	2.343
C1, 5000 events, 50 actors	0.063	6.332	14.0
C1, 20000 events, 100 actors	0.236	26.2	101.4
C2, 5000 events, 100 actors	0.089	6.565	26.2
C3, 20000 events, 100 actors	0.166	19.8	71.0
R1i, 5000 events, 100 actors	0.089	5.405	35.0
R1i, 20000 events, 100 actors	0.217	24.3	355.3

Table 13.1: Seconds per fit on Linux (Intel Core i7-12700K, Ubuntu 22.04.4, StataNow 19.5 MP limited to 8 processors, R 4.6.0), 16 November 2026.

one of the 16 cells measured: 7 to 119 times against `default_c` and 12 to 1,637 times against the default engine. The gain grows with the number of events, because the time of `goldfish` is dominated by a preprocessing step that computes the statistics of each event for each possible receiver, which `nwdynam` does not have as a separate step. Estimates agree with `goldfish` to a relative difference of at most $6 \cdot 10^{-7}$ in every cell, the log likelihood to round-off.

How it was measured. One fit per process, after an untimed warm-up fit (so that the first call, which loads the libraries, is not in the time); the timer covers the estimation command (for `goldfish`, the estimation call including its preprocessing), not the loading of the data. Up to seven repeats with a different seed for each repeat (five, three or one in the largest cells of `goldfish`); the table shows medians. Both programs get the same data and the same specification with their default control settings. Each fit has a limit of 600 seconds. The times of `goldfish` were measured once, with an earlier version of `nwdynam`; they were not repeated for 1.8.0, so that the two programs were not timed in turn for this version.

What was not measured. The effects that were added in version 1.8.0 (effects through other networks, windows, composition change, time-varying covariates, two-mode networks) and the `estat` commands were not benchmarked; fits that do not use them are as fast as in earlier versions (within noise in all 16 cells). The coordination sub-model was not compared with `goldfish` in this benchmark.

Appendix A

Translation table: goldfish to nwdynam

`goldfish` specifies a model by a formula, `nwdynam` by options. The first rows of the table translate the data preparation; the others the effects, the sub-models and the postestimation. Without any effect option `nwdynam` fits `inertia`, `recip` and `indeg` for the choice model and `indeg` and `outdeg` for the rate model; naming any effect restricts the model to the effects named.

goldfish	nwdynam
`defineNodes()`, `defineNetwork()`, `linkEvents()`, `defineDependentEvents()`	`nwset sender receiver, directed eventtime(t)` `name(net)`; isolates with `nwaddnodes`; attributes with `nwload net, xvars`
`estimate(f, model = "DyNAM", subModel = "choice")`	`nwdynam net` (the default sub-model)
`subModel = "rate"`	`nwdynam net, submodel(rate)`
`subModel = "choice_coordination"`	`nwdynam net, submodel(choice_coordination)` on an undirected network
`inertia`, `recip`	`inertia`, `recip`
`indeg`, `outdeg`	`indeg`, `outdeg` (both sub-models)
`trans`, `cycle`, `commonSender`, `commonReceiver`, `four`	`trans`, `cycle`, `commonsender`, `commonreceiver`, `four`
`nodeTrans`	`nodetrans`
`same(attr)`, `diff(attr)`, `sim(attr)`	`same(var)`, `diff(var)`, `sim(var)`
`ego(attr)` (rate), `alter(attr)` (choice)	`ego(var)`, `alter(var)`
`egoAlterInt(attr)`	`egoalterint(var1 var2)`
`tertius(attr)`	`tertius(var)`
`tie(network)` (also a network that changes over time; `weighted`, `window`, `ignoreRep`)	`tie(netname)`; through another network `netterms(tie(netname) ...)` with `window(#)`, `weighted`, `ignorerep`
`trans`, `cycle`, `commonSender`, `commonReceiver`, `four` through another network	`netterms(trans(netname) cycle(netname) ...)`
`mixedTrans`, `mixedCycle`, `mixedCommonSender`, `mixedCommonReceiver`	`netterms(mixedtrans(net1 net2) ...)` and the other mixed terms
`tertiusDiff(attr)`	`netterms(tertiusdiff(var [net]))`
`ignoreRep = TRUE` on a built-in effect	`ignorerep(effects)`
a window on `tertius`	`tertiuswindow(#)`
`window = w` on an effect	`inertiawindow(#)`, `recipwindow(#)`, `indegwindow(#)`, `outdegwindow(#)`
`weighted = TRUE` on an effect	`weightedinertia`, `weightedrecip`, `weightedindeg`, `weightedoutdeg`

goldfish	nwdynam
`weighted = TRUE, window = w`	a weighted effect with its own window, for example `weightedinertia inertiawindow(#)`
`opportunitiesList` in `estimationInit` (also with attribute effects)	`opportunities(evvar actvar)`, with `oppframe(framename)` for covariate effects
`linkEvents(nodes, changeEvents, attribute = "x")` with `replace` or `increment` columns (time-varying attributes)	the changes of `x` as an event list in which sender and receiver are the same actor and the value is the new value: `nwset id id2 value, eventtime(t) selfloop name(xchg)` (`id2` a copy of `id`), then `change(x xchg)`; `increment` makes the values increments
`defineNodes(..., present = TRUE/FALSE)` (composition change)	`present(var)`, a 0/1 variable with one row per actor (default: everybody present)
`linkEvents(nodes, changeEvent = comp, attribute = "present")`	`compchange(net)`, an event network made with `nwset id id2 value, eventtime(t) selfloop name(net)`; value 1 the actor joins, 0 it leaves
`increment` of an event (`linkEvents()`)	a value variable: `nwset sender receiver value, directed eventtime(t)`; without it every event has the value 1
a leading `1 +` in the formula (intercept, rate model)	`intercept` with `submodel(rate)`
`logLik()`, `AIC()`, `BIC()`	`estat ic`
`returnIntervalLogL = TRUE`	`estat eventlik`
`returnEventProbabilities = TRUE`	`estat eventprob`
`examineOutliers()` (Hampel, IQR, Top)	`estat outliers`
`examineChangepoints()` (PELT, AMOC, BinSeg)	`estat changepoints`
`summary()` of the estimate	the table printed by `nwdynam`; `e(b)`, `e(V)`, `e(ll)`, `e(N)`, `e(nodes)`

Bibliography

[1] Stadtfeld, C., Block, P. (2017). Interactions, actors, and time: Dynamic network actor models for relational events. *Sociological Science* 4, 318–352. https://doi.org/10.15195/v4.a14

[2] Stadtfeld, C., Hollway, J., Block, P. (2017). Dynamic network actor models: Investigating coordination ties through time. *Sociological Methodology* 47(1), 1–40.

[3] Hollway, J., Stadtfeld, C., Hoffman, M., Uzaheta, A. and contributors. *goldfish: Statistical Network Models for Dynamic Network Data*. R package. https://cran.r-project.org/package=goldfish (compared with version 1.6.12).

[4] Killick, R., Eckley, I.A. (2014). changepoint: An R package for changepoint analysis. *Journal of Statistical Software* 58(3), 1–19.

[5] Butts, C.T. (2008). A relational event framework for social action. *Sociological Methodology* 38, 155–200.

[6] Michalski, R. (2020). Manufacturing company email metadata and corporate hierarchy. Harvard Dataverse, V2. https://doi.org/10.7910/DVN/6Z3CGX. Released under CC0 1.0.

[7] Nurek, M., Michalski, R. (2020). Combining machine learning and social network analysis to reveal the organizational structures. *Applied Sciences* 10(5), 1699.

[8] Madan, A., Cebrian, M., Moturu, S., Farrahi, K., Pentland, A. (2012). Sensing the "health state" of a community. *IEEE Pervasive Computing* 11(4), 36–45.