Developments for RSiena – 2022

Tom A.B. Snijders

University of Oxford
University of Groningen

Duisterbelt, 2022

Overview

Maintaining RSiena is something I usually do in contact with
a few others, who make specific contributions.

When I have submitted a new version, this is (normally) noted
in the News page of the website,
and in a message to `Siena@groups.io`

`https://github.com/snlab-nl/rsiena/blob/main/NEWS.md`

But it is nice to share it now with you.

GitHub

After the transition to GitHub (summer 2020),
I continue to rely on GitHub help from James Hollway.

GitHub makes collaboration, and contributions by varied people, much easier, but few others are active in this respect.

The RSiena website says we still are in a transition phase.

Loose ends:

- ⇒ Website (continued)
- ⇒ **`sienaBayes`** now in multiSiena
- ⇒ RSienaTest still in existence on R-Forge

Loose end: Website

Website is duplicated between
`http://www.stats.ox.ac.uk/~snijders/siena/` and
`https://github.com/snlab-nl/rsiena/wiki`,
but I maintain only the former
and nobody seems to do anything with the latter.

Last year's proposal: carry on,
you are welcome to extend/modify the wiki.

how do we carry on?

Loose end: multiSiena

Paper Koskinen-Snijders about **`sienaBayes`** recently accepted in *JRSS-A* (special issue '*Networks in Society*')
There will be a script on the Siena website.

Function **`sienaBayes`** is implemented in package multiSiena , available from the Siena downloads page.

Question: should it be integrated with RSiena ?
This would lead to a very large package.

RSienaTest is still on R-Forge, but not maintained.

(note: extra functionality is **`sienacpp`** and Simmelian effects)

CRAN

Whenever there are enough important changes,
a new version is submitted to CRAN.

The past versions of RSiena at CRAN:

1. 1.3.14 (October, 2022)
2. 1.3.0 (May, 2021)
3. 1.2.23 (January, 2020)

Submitting to CRAN is always something of a hassle, but useful.
For the last submission, Brian Ripley gave important support
to update the `configure` files.

Use of recent options

Two important options, added in the past few years,
are not yet used a lot:

- The GMoM for use of contemporaneous statistics
  (Viviana Amati).
  The hurdle for its use is the necessity to have an adequate fit.
  Work in progress!
- The model for continuous behavioral variables (Nynke Niezink).
  There still is a bug – but where???
  Work in progress!

These situations still are unchanged (as far as RSiena is concerned).

New Effects

Various new effects were implemented. A few highlights:

- Contributions from Roberts Krause and Hellpap, Steffen Triebel.
- `avDeg`, see next page.
- elementary activity effects, see next next page
  `outAct_ego, inAct_ego, reciAct_ego, toAny`
- `gwdspFB` effect added for two-mode networks.
- For effects `to`, `toBack`, `toRecip`, `mixedInXW`, internal effect parameter 3 now specifies truncation (at 1) of the number of twosteps.
- Incoming tie influence weighted by dyadic covariate:
  `avInAltW`, `avWInAlt`, `totInAltW`, `totWInAlt`
- `recipRateInv`, `recipRateLog`
- Several new effects related to primary setting
  (see manual, only names are mentioned...)

New effect `avDeg`

From the manual:

average out-degree (*X*: av. degree) (avDeg)
defined by the out-degree of *i* multiplied by the average outdegree centered by the internal effect parameter *p*

$s_{i3}^{\text{net}}(x) = \sum_j x_{ij} \left(\frac{1}{n} \sum_h x_{h+} - p \right) = x_{i+} \left(\frac{1}{n} \sum_h x_{h+} - p \right)$;

this effect is a network-by-period level effect
and therefore is useful only for multigroup data sets
or for data sets with many periods;

This was used in the Koskinen-Snijders paper about **`sienaBayes`**
(one-mode: friendship; two-mode: delinquency)
for representing feedback of the average level of classroom delinquency.

New effect `avDeg`

From the manual:

average out-degree (X: av. degree) (avDeg)
defined by the out-degree of i multiplied by the average outdegree centered by the internal effect parameter p

$s_{i3}^{\text{net}}(x) = \sum_j x_{ij} \left(\frac{1}{n} \sum_h x_{h+} - p \right) = x_{i+} \left(\frac{1}{n} \sum_h x_{h+} - p \right)$;

this effect is a network-by-period level effect
and therefore is useful only for multigroup data sets
or for data sets with many periods;

This was used in the Koskinen-Snijders paper about **`sienaBayes`**
(one-mode: friendship; two-mode: delinquency)
for representing feedback of the average level of classroom delinquency.

The estimated parameter was negative!
Interpreted as regression to the mean;

'More research is needed to interpret this fully.'

New Effects: `outAct_ego, inAct_ego, reciAct_ego`

I wanted to have more ways of
expressing context-dependence of behavior in RSiena .

First step: express dependence of network choices on ego's position.

First indicators of position: degrees.

However, interactions are not permitted with ego's outdegrees (**`outAct`**) and reciprocal degrees (**`reciAct`**).

The manual says that interaction possibilities are defined by the `interactionType` of the effect.

Interactions – ego

Interactions are defined by multiplying the change statistics.

The explanation of the condition for interactions is given
(not in the manual but) in
`https://www.stats.ox.ac.uk/~snijders/siena/Siena_algorithms.pdf`.

All interactions are allowed if `interactionType="ego"`.

An effect is defined to be an *ego* effect if it can be written as

$$s_{ik}^{X}(x,z) = \sum_{j} x_{ij}\, c_{ki}(x,z)$$

where $c_{ki}(x,z)$ is independent of $(x_{i1},\ldots,x_{in})$ and independent of j.
The change statistic then is

$$\Delta_{kij}^{X}(x,z) = c_{ki}(x,z)\ .$$

New Effect: `outAct_ego`

The `outAct` effect

$$s_i(x) = \left(x_{i+}\right)^2$$

is not an 'ego' effect, and therefore cannot be used for interactions.
The change statistic is $\delta_{ij}(x) = 2\left(x_{i+} - x_{ij}\right) + 1$.

To allow interactions, an ego-type effect was created; it is an elementary effect (see the manual, Section 5.1.3!), described as:

out-degree related activity ego effect (outAct_ego), defined as the elementary effect version of the half-centered squared out-degree of the actor,

$$f_{i3}^{\text{net, el}}(x) = \left(x_{i+} - \bar{x}_{\cdot+}\right) \text{ where } \bar{x}_{\cdot+} = \frac{1}{Mn}\sum_{m=1}^{M}\sum_{i,j} x_{ij} \ ;$$

for internal effect param. $p = 2$, it is defined by $f_{i3}^{\text{net, el}}(x) = \sqrt{x_{i+} - \bar{x}_{\cdot+}}$; since this is an elementary effect, it can be used in interactions and in endowment and creation functions

(but interactions will be elementary effects again).

Elementary effects

From the manual:

An elementary effect is a contribution to the creation or maintenance of a tie,
defined directly, i.e., without expressing it
based on the change in some evaluation function.

Thus, for adding a tie, this is added to the objective function,
and for removing a tie it is subtracted.

This is more general than the evaluation function,
and has lost the interpretation of ‘evaluation’.

It always has `interactionType="ego"`
(whether or not the condition mentioned is satisfied).

New Effects: `inAct_ego, reciAct_ego`

For `inAct` and `reciAct` the same was done;
note that `inAct` is already an ego-type effect,
but the centering may be helpful (e.g., for interactions in **`sienaBayes`**)
and therefore not only `(outAct_ego)` and `reciAct_ego`
but also `inAct_ego` was created.

Other changes to RSiena

- `sigmas` and `meansigmas` added to **`sienaRI`** object.
- Function **`sienaRI`** was made available also for two-mode networks (provided the number of second-mode nodes is smaller than the number of actors = first-mode nodes), and corrected for behavioral dependent variables.
- If `returnThetas` in the call of **`siena07`**, not only theta values but also simulated estimation statistics during Phase 2 (deviations from targets) are returned.
- Function **`includeInteraction`** now also can modify the `initialValue` of an effect; and the order of parameters for this function was changed, bringing it in line with **`setEffect`**.
- And various corrections and minor improvements.

Some kind of effect sizes

Issues treated here for effect sizes:

1. Sometimes parameters are much smaller or larger than usual (e.g., 4-cycles).

   $\Rightarrow$ semi-standardized estimates

2. In a model for network dynamics,
   how large are the separate contributions of each included effect?

   $\Rightarrow$ entropy-based measures.

Semi-standardized parameter estimates

To compare parameters β_k within an estimated model for network x, consider the formulae

$$\text{probability} \quad \pi_{ij}(\beta, x) = \frac{\exp\left(f_i(\beta, x^{(\pm ij)}) - f_i(\beta, x)\right)}{1 + \sum_{h \neq i} \exp\left(f_i(\beta, x^{(\pm ih)}) - f_i(\beta, x)\right)} ,$$

$$\text{change statistic} \quad f_i(\beta, x^{(\pm ij)}) - f_i(\beta, x) = \sum_k \beta_k \, (1 - 2\, x_{ij}) \, \delta_{ij,\,k}(x) .$$

This shows that β_k is multiplied by the 'variable'

$$(1 - 2\, x_{ij}) \, \delta_{ij,\,k}(x) .$$

Define

$$\delta_{ii,\,k}(x) = 0 .$$

In analogy to linear regression,
we can make the values β_k comparable by considering

$$\begin{aligned}\sigma^2_{ik}(x) &= \operatorname{var}\Big\{ \delta_{ij,\,k}(x)\,(1 - 2\,x_{ij}) \;\Big|\; i \text{ fixed} \Big\} \\ &= \frac{1}{n}\sum_{j=1}^{n} \big(\delta_{ij,\,k}(x)\,(1 - 2\,x_{ij})\big)^2 \;-\; \left(\frac{1}{n}\sum_{j=1}^{n} \delta_{ij,\,k}(x)\,(1 - 2\,x_{ij})\right)^2 ,\end{aligned}$$

the within-actor variance of this ‘variable’; and their average,

$$\sigma^2_k(x) = \frac{1}{n}\sum_i \sigma^2_{ik}(x) \; .$$

The semi-standardized parameter estimates is defined as the product

$$\sigma_k(x)\,\beta_k \; .$$

This expresses the parameters β_k, for different k and a given x,
on a common scale (for a given model).
The standard deviation is used here as a somewhat arbitrary,
but well-known measure of scale.

Example: activities in Glasgow

Two-mode network for Glasgow data for 14 activities:

		daily	weekly	monthly	less
1	I listen to tapes or CDs	**388**	23	5	16
2	I look around in the shops	**65**	290	48	30
3	I read comics, mags or books	**186**	121	65	60
4	I go to sport matches	**30**	**113**	90	200
5	I take part in sports	**218**	117	30	68
6	I hang round in the streets	**216**	64	26	125
7	I play computer games	**157**	109	45	122
8	I spend time on my hobby (e.g. art, an instrument)	**114**	113	36	170
9	I go to something like B.B., Guides or Scouts	**36**	**81**	1	314
10	I go to cinema	**11**	**81**	269	71
11	I go to pop concerts, gigs	**7**	**6**	**92**	326
12	I go to church, mosque or temple	**2**	**52**	**10**	368
13	I look after a pet animal	**197**	25	6	203
14	I go to dance clubs or raves	**15**	**44**	**104**	266
15	I do nothing much (am bored)	37	39	24	331

Number of students participating in each of a list of activities, summed over three waves, for Glasgow data. Bold-faced are frequencies counted as a tie.

Semi-standardized parameter estimates

Effect	$\hat{\beta}_k$	(s.e.)	σ_k	$\sigma_k \hat{\beta}_k$
rate period 1	4.227	(0.268)		
rate period 2	4.047	(0.278)		
outdegree	–5.891	(0.660)	0.79	–4.67
outdegree - activity	0.637	(0.088)	5.48	4.33
indegree - popularity ($\sqrt{}$)	0.790	(0.100)	6.74	4.29
out-in degree assortativity	–0.0184	(0.0025)	331.54	–6.11
4-cycles	0.0389	(0.0057)	55.08	2.14

$\hat{\beta}_k$: estimates; s.e.: standard errors;
σ_k: mean within-ego standard deviations of change statistics;
$\sigma_k \hat{\beta}_k$: their product.

The order of magnitude of the semi-standardized parameter estimates is similar.

Entropy-based measures

The uncertainty / variability
in the outcomes of categorical random variables
can be expressed by the *entropy* (Shannon, 1948).

For a probability vector $p = (p_1, \ldots, p_K)$, entropy is defined as

$$H(p) = -\sum_{k=1}^{K} p_k \, {}^2\log(p_k) \, . \tag{1}$$

Minimum 0 (if one category has $p_k = 1$, outcome is certain);
maximum ${}^2\log(K)$ (if $p_k = 1/K$ for all k, totally random).

This can be transformed to the range $[0, 1]$ by

$$1 - \frac{H(p)}{{}^2\log(K)}$$

so that 0 means total uncertainty and 1 means total certainty.

This can be applied to the probability vector

$$\pi_i(\beta, x) = \Big(\pi_{i1}(\beta, x), \ldots, \pi_{in}(\beta, x)\Big)$$

of choices in the ministep for current network state x.

The degree of determination (certainty), or amount of information,
in the outcome of the ministep for actor i,
for current network state x, can be expressed by

$$R_H(i, \beta, x) = 1 - \frac{H(\pi_i(\beta, x))}{^2\log(K)} \; . \tag{2}$$

For models with constant rate function, this can be averaged:

$$R_H(\beta, x) = \frac{1}{n} \sum_i R_H(i, \beta, x) \; . \tag{3}$$

For network panel data we may average over waves:

$$R_H(\beta) = \frac{1}{M} \sum_m R_H\big(\beta, x(t_m)\big) \ . \tag{4}$$

Note that this refers to the ministeps, by an arbitrary actor, taken when the network state is as observed.

This is a measure between 0 and 1:
1 if the outcome of the ministep for a given actor is certain;
0 if all actors choose a random change (all probabilities $1/n$).

This measure was proposed in Snijders (*Mathématiques et Sciences Humaines*, 2004).

Values generally will be low!

Contributions to R_H

For the contribution of an effect, or set of effects,
to the information about the outcome:
estimate the model twice,
giving parameter estimates $\hat{\beta}_1$ for full model
and $\hat{\beta}_2$ for restricted model, and consider the difference

$$R_H(\hat{\beta}_1) - R_H(\hat{\beta}_2) \,. \tag{5}$$

Note that this is not necessarily positive,
because the estimation method does not maximize $R_H(\hat{\beta})$.

Example: van de Bunt data

As an example we consider the friendly relation for Van de Bunt's data set, waves $t_1 - t_4$.

Effect	par.	(s.e.)
Rate 1	3.94	(0.64)
Rate 2	4.71	(0.81)
Rate 3	3.56	(0.59)
outdegree (density)	–1.545***	(0.208)
reciprocity	2.111***	(0.272)
outdegree-popularity	–0.192***	(0.039)
transitive triplets	0.456***	(0.049)
sex alter	0.384*	(0.191)
sex ego	–0.535*	(0.209)
same sex	0.156	(0.181)
program similarity	0.727***	(0.207)

* $p < 0.05$; ** $p < 0.01$; *** $p < 0.001$;
Overall maximum convergence ratio 0.07.

The following table gives estimates and semi-standardized parameter estimates.

Effect	$\hat{\beta}_k$	(s.e.$_k$)	σ_k	$\sigma_k\,\hat{\beta}_k$	$t_k = \hat{\beta}_k/\text{s.e.}_k$
outdegree (density)	–1.545***	(0.208)	0.58	–0.90	–7.44
reciprocity	2.111***	(0.272)	0.32	0.68	7.76
outdegree-popularity	–0.192***	(0.039)	4.41	–0.84	–4.86
transitive triplets	0.456***	(0.049)	1.55	0.70	9.21
sex alter	0.384*	(0.191)	0.42	0.16	2.01
sex ego	–0.535*	(0.209)	0.22	–0.12	–2.56
same sex	0.156	(0.181)	0.59	0.09	0.86
program similarity	0.727***	(0.207)	0.34	0.24	3.52

* $p < 0.05$; ** $p < 0.01$; *** $p < 0.001$;

Effect sizes for friendly relation between van de Bunt's students, waves $t_1 - t_4$.
$\hat{\beta}_k$: parameter estimate; s.e.: standard error;
σ_k: within-ego standard-deviation of change statistic (averaged over actors and waves)
$\sigma_k\,\hat{\beta}_k$: semi-standardized beta.

The following table shows the degrees of determination (certainty), by wave, for four models:

1. the empty model, with only the outdegree effect;
2. the purely structural model, with the effects of reciprocity, outdegree popularity, and transitive triplets;
3. the model with only covariates, with the three effects of sex and program similarity;
4. the full model.

Model	t_1	t_2	t_3	t_4
Empty model	0.02	0.02	0.02	0.02
Structural model	0.19	0.18	0.17	0.15
Covariate model	0.05	0.05	0.05	0.04
Full model	0.21	0.19	0.20	0.18

We conclude that the structural influences are of much greater importance in determining the network dynamics than the covariate influences.

sienaRI

These measures are implemented in the function **`sienaRI`**.

Consult the help page!

Also see Section 13.5 of the RSiena manual and script `vdB_effsize.R`.

Effects in RSiena – coding aspects

In the **data** directory of the source code

`https://github.com/snlab-nl/rsiena/blob/main/data`

there is the file **allEffects.csv**,

which in the RSiena package is available as an
internal data frame used to construct effect objects:

allEffects

In RSiena , you can request
dim(allEffects)
and view part of the variables in this data frame in the browser through
effectsDocumentation()

The `xxxxxx, yyyyyy, zzzzzz` are names of variables to be filled in;
the ♯ is the internal effect parameter to be filled in.

effectGroups

The first column of **`allEffects`** is the **`effectGroup`**. By
`unique(allEffects$effectGroup)`
you can see that, currently, there are 59 effectGroups.

These are meaningfully ordered by row in **`effectsDocumentation()`**.

The **`effectGroup`** is based on combining the type of dependent variable
`(oneMode - symmetric - bipartite - behavior - continuous)`
with the kind of explanatory variable
(type of dependent variable & `actor covariate - dyadic covariate`)

This is done by **`getEffects`**, defined in the file
`https://github.com/snlab-nl/rsiena/blob/main/R/effects.r`
which calls internal function **`createEffects`**
for each **`effectGroup`** separately,
handling all variables in the **`Siena`** data set.

Creating new effects – R

The manual (Chapter 18) contains a tutorial about creating new effects.

If, for a new effect, you wish to create a new **`effectGroup`**,
you have to modify **`getEffects`** in `effects.r` accordingly,
and specify its rank order in `effectsDocumentation.r`.

But usually you can employ one of the existing effectGroups.
The RSiena manual (Section 18.2) contains a note about
how various effectGroups handle two-mode networks differently.

Coding effects in C++

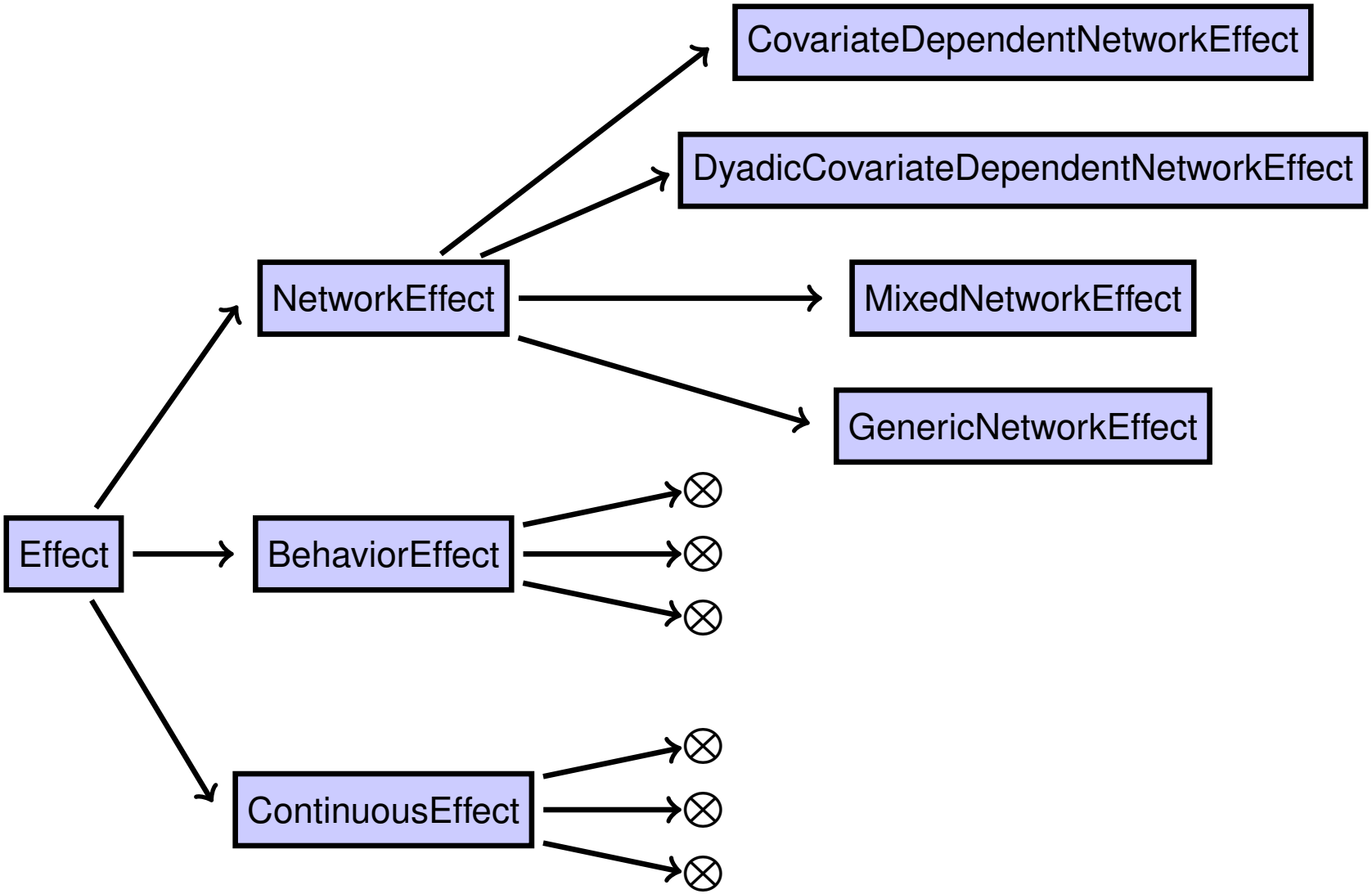
The **`EffectsFactory`** takes the included effects and constructs the computing machinery.

Note that for each effect, we need two things:

1. its contribution to the objective function (exception: `gmom`-type effects);
2. its estimation statistic for MoM.

The next page shows the structure of the main effect classes.

Effect classes in C++



Structure of effect classes, without the further descendants

NetworkEffect: change statistic

The NetworkEffect class has a given **`ego()`**.

The contribution to the objective function for network effects
is defined by function **`calculateContribution(alter)`**,
and calculations that are not specific to `alter`
are done in **`preprocessEgo(int ego)`**
(called elsewhere, therefore `ego` is specified).

`calculateContribution(alter)` computes the *change statistic*.

NetworkEffect: estimation function

More details are given in **`Siena_algorithms.pdf`**.
Here only the case for effects depending only on the network is treated.

The statistic used for estimation is

$$\sum_{m=2}^{M} s_k^X \left(x(t_m) \right) , \tag{6}$$

where s_k^X is the sum of the effect over all actors, defined by

$$s_k^X(x) = \sum_i s_{ik}^X(x) . \tag{7}$$

and s_{ik}^X is effect k for actor i.

The terms in (7) are the function **`egoStatistic()`**, and these are computed for some (not all) effects as

$$\text{egoStatistic}(i) = \sum_j x_{ij}\, \text{tieStatistic}(i,j)\ . \tag{8}$$

Effect-specific instances of **`egoStatistic()`** or **`tieStatistic()`** are defined in all functions defining specific effects.

It makes no sense to define **`egoStatistic()`** as well as **`tieStatistic()`** for any specific effect, because (8) is computed in **`NetworkEffect.cpp`**, and will be valid unless **`egoStatistic()`** is defined for the specific effect (which then will avoid the use of **`tieStatistic()`**).

The endowment and creation effects will be computed by applying **`egoStatistic()`** to the network of lost or (respectively) newly created ties; only if this is to be replaced by something else, should **`endowmentStatistic()`** (or **`creationStatistic()`**) be defined.

Generic effects

The class **`GenericNetworkEffect`** may be used to specify an effect for a network variable X if its change statistic is given by

$$f_{ij}(x)$$

and the estimation statistic for the evaluation function by

$$\sum_{i,j} x_{ij} f^0_{ij}(x) \ ;$$

with the appropriate modifications for the endowment and creation functions. The functions $f_{ij}(x)$ and $f^0_{ij}(x)$ should be specified as instances of the **`AlterFunction`** class and passed as parameters when creating the effect.

The distinction between f_{ij} and f^0_{ij} is made mainly to allow possibilities for taking missing covariate data into account.

Composition of **`AlterFunction`**s is allowed, e.g., sum, product, square root, which opens the possibility of simple general definitions.

This can be studied by looking for `GenericNetworkEffect` in `https://github.com/snlab-nl/rsiena/blob/main/src/model/effects/EffectFactory.cpp`

Two examples

Example of a **`NetworkEffect`**:

`https://github.com/snlab-nl/rsiena/blob/main/src/model/effects/AverageDegreeEffect.cpp`

Example of a **`GenericNetworkEffect`**:

Search for `outAct_ego` in

`https://github.com/snlab-nl/rsiena/blob/main/src/model/effects/EffectFactory.cpp`

and then

`https://github.com/snlab-nl/rsiena/blob/main/src/model/effects/generic/EgoOutDegreeFunction.cpp`


The horserace of Siena
Il palio was originated in the 13th