Stochastic actor oriented model with random effects

Simulation based estimation, model evaluation, and implementation of dynamic sets and multisets

Doctoral Dissertation submitted to the

Faculty of Informatics of the Università della Svizzera italiana

in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

presented by

Giacomo Ceoldo

under the supervision of

Ernst C. Wit

July 2023

Dissertation Committee

Igor Pivkin	Università della Svizzera italiana, Switzerland
Stefan Wolf	Università della Svizzera italiana, Switzerland
Tom Snijders	University of Oxford, United Kingdom
Alessandro Lomi	Università della Svizzera italiana, Switzerland

Dissertation accepted on 20 July 2023

Research Advisor	PhD Program Director
Ernst C. Wit	**Walter Binder, Stefan Wolf**

I certify that except where due acknowledgement has been given, the work presented in this thesis is that of the author alone; the work has not been submitted previously, in whole or in part, to qualify for any other academic award; and the content of the thesis is the result of work which has been carried out since the official commencement date of the approved research program.

Giacomo Ceoldo
Lugano, 20 July 2023

Abstract

This thesis advances the field of social network analysis by generalizing the stochastic actor oriented model (SAOM) to allow for the inclusion of random effects, so that the heterogeneity of the individuals can be modelled more accurately. The method of moments estimation, with the model evaluation procedure that are commonly used in the SAOM, are generalized to being able to estimate the parameters of the model when there are random effects. A social network of workers in a tailor shop has been analysed using the SAOM with random effects. The focus of the analysis has been the comparison of different models, with or without random effects, and the difference in the interpretation of the parameters.

The algorithm developed for the SAOM has been studied in more detail in a regression set up, also when the parameters are estimated with generalized method of moments. The focus of the research has been on the comparison between different methods to increase the power of test statistics, when the statistics used to estimate the parameters are correlated.

Algorithms that are used in social network analysis are often based on simulating the underline network process, that is represented by a discrete dynamic data structure. An efficient R implementation of sets and multisets, based on hash tables, is discussed and applied to network processes whose state is represented by a set, and whose sufficient statistics are stored in a multiset.

Acknowledgements

I would like to extend my deepest gratitude to my supervisor, Ernst Wit. Our scholarly journey together, traversing the landscape of Bachelor's, Master's, and now culminating in the Ph.D. thesis, has been an intellectually enriching and transformative experience. Ernst's consistent guidance has been an invaluable help, supporting me at every step of my academic journey. His profound knowledge and unwavering dedication have not only influenced the research I present today, but have also shaped my growth as a scholar. His patience and commitment have fostered an environment of learning and curiosity that has nurtured my academic aspirations and have led me to the completion of this doctoral study. It is under Ernst's tutorship that I have explored the depths of our field and found my own place within it. For this, I am deeply thankful.

Special thanks are due to Tom Snijders, who helped us greatly in developing our generalization of the model. Your vast knowledge, keen insights, and unwavering support have significantly enriched my academic journey. Your collaborative spirit and dedication to the project were inspiring, and have undeniably contributed to my success in this research.

To my colleagues and friends, I express my heartfelt appreciation for the moments we have shared. Your individual and collective inputs, criticisms, and motivations have been instrumental in the completion of this study.

Lastly, I extend my appreciation to to everyone who indirectly contributed to this project. This includes the members of the dissertation committee, to whom I am grateful in advance for dedicating their time and applying their expertise to the review of this thesis.

Contents

Chapter 1

Introduction

An interconnected system is kept alive by the complex and dynamical *structure of relationships* between who is part of the system. *Network data science* [42, 29], particularly *social network analysis* [10], provides powerful tools to make sense of these intricate networks of relationships among individuals. By modelling networks, we can gain a deeper understanding of collective behavior. In particular, a model can be often viewed as a parametrized stochastic algorithm that simulates a network (or a sequence of networks, a process within a network, ...). Simulation can be then used to select and evaluate a good model, in which the simulated process mimics as much as possible the observed data.

A *parametrized stochastic simulation algorithm* is a method to sample from a (parametrized) probability distribution. Statistical inference can be used to estimate the parameters of this distribution, that is, to select one or some distributions among many, from the set of distributions defined by the model. In *simulated statistical inference* [18] the parameters are adjusted until we are able to simulate values which match on average the observed ones. In detail, a set of functions called statistics, is assumed to summarize the “important" information about the network. The set of statistics is used to define which characteristics of the observed dataset should be matched on average by the simulated process. In the *simulated method of moments* [39] this set of statistics is used to solve stochastically an equation between their expected value, which depends on the parameters, and the observed value of the same statistics, which are computed from the dataset that is analysed.

The major contributions of this thesis are on the *stochastic actor oriented model* [62, 60], known as the *SAOM*, which is one of the most used methods in social network analysis. The SAOM is a *dynamic network model* in which the individuals (entities) have the ability to modify their connections to others based

on how much they like their “placement" in the overall network structure. The model is very flexible in defining the linear evaluation function of the individuals. This function encodes as a linear combination of statistics weighted by parameters, how much individuals “like" the possible configurations of connections. The statistics, which in this context are usually called effects, can also depend on attributes of the individuals, viewed as (independent, exogenous) covariates of the network evolution. The connections are modified by stochastic choices, to account for the element of randomness and uncertainty in how choices are made by the entities (e.g., by humans). In a given moment, an individual has the possibility to change one of its outgoing connections, the evaluation function determinates the probability of the allowed choices. Consequently, the SAOM provides a method for analyzing complex, evolving networks, offering valuable insights into the mechanisms driving social interactions and network development.

Our contributions partially address the current limitation in the frequentist version of the SAOM, in which individuals are assumed to be *homogeneous*, in the sense that two individuals would evaluate “equivalent placements" in the network in the same way. Therefore they will also evaluate changes from “equivalent configurations" in the same way. Mathematically, all individuals share the same parameters (weights of the effects in the evaluation function), meaning that the importance of each effect in the evaluation function is the same for everyone. The homogeneity assumption is often reasonable when the attributes about the individuals in the dataset characterize them well enough. So if the attributes contain enough information, their inclusion as interaction effects with some subgraph counts can be enough to model accurately the observed dynamics. However, relevant attributes are not always available. Moreover, the researcher might want to use a “simpler" evaluation function, with possibly much less effects common to all individuals, here considered *heterogeneous* in the sense that some of the effects are modelled as random, meaning that every individual has its own random parameter for the given effect. The price to pay for the dropping of the homogeneity assumption is higher randomness in the model, causing more variable estimates. There is then a dichotomy between considering a complex social dynamics between homogeneous individuals, or a simpler social dynamics between heterogeneous individuals. Therefore, the use of random effects can provide different explanations for patterns that are modelled with the SAOM. Our generalization is close “in spirit" to [57] and [55].

The simulation approach used in the SAOM to compute the solution of the moment equation is very flexible, so some of its properties are discussed in this thesis in the context of *regression with mixed effects* [46, 38]. The purpose is

to study the estimation algorithm that has been developed in parallel with the SAOM, in different set-ups. The model evaluation part has been investigated the most, mainly by discussing *Neyman's orthogonalization* procedure [43] in simulated inference based on, "standard" and "generalized" method of moments. This procedure is used to derive test statistics with good power when the statistics that are used to estimate the parameters with method of moments are correlated. Different orthogonalization procedures have been compared with a simulation study.

Many if not most limitations in network data science are computational. In the definition of the model, which starts from specifying, and eventually implementing, the algorithm which simulates the network dynamics, the computational complexity of the implementation must be taken into account. A large component of network dynamics is often *discrete* in nature. The computational representation of the current state of the process includes for example having to store in memory sets of relations, or multisets of statistics such as the counts of configurations used in the evaluation functions. When the updates to the network are *local* (few elements are updated) these sets or multisets should be updated quickly. We developed the *hset* package of the R language, of sets and multisets that is based on the hash table data structure implemented in [11]. The implementation is efficient in computing some relations and operations involving sets and multisets, as values in an hash table can be included, excluded, accessed and modified in constant time.

In Chapter 2 the SAOM with random effects is explained in detail. The study of its estimation algorithm in regression with random effects is discussed in Chapter 3, which is based on [17]. Finally, our R implementation of sets and multisets is discussed in Chapter 4, based on [16] Section 1.1 of this chapter describes some background material that is helpful to understand the other chapters. The contributions of our research are summarized in Section 1.2. In Section 1.3, the content of the Chapters 2, 3, and 4, is summarized in detail.

1.1 Background material

In this section, some background material that is useful to understand the following chapters is introduced.

1.1.1 Stochastic actor oriented model

The *stochastic actor oriented model* (*SAOM*) provides a way of understanding the dynamics of *longitudinal social networks*. It is rooted in the concept that individuals have the capacity to adjust their relationships with others based on their satisfaction with their current position within the network structure. The network evolution is modelled as a *Markov chain* over the state space of all possible networks of relations between the individuals in the study. Each actor or entity in the network can control its outgoing relations. Transitions in state space can happen when an actor has the possibility to modify one of its outgoing relations.

For a network of N nodes, the state space of the process is $\mathscr{X} \cong \{0,1\}^{N(N-1)}$. The probability of transitions are defined by specifying the *linear evaluation function* $u_i : \mathscr{X} \to \mathbb{R}$, such that $u_i(\boldsymbol{x}) = \boldsymbol{\beta}^\top \boldsymbol{s}_i(\boldsymbol{x})$, for each individuals $i \in \{1,...,N\}$. The evaluation function is a linear combination of the *effects* in $\boldsymbol{s}_i(\boldsymbol{x})$, that are typically counts of sub-graphs of which individual i is part of, weighted by a parameter vector $\boldsymbol{\beta}$. The effects may also be *interactions* between sub-graph counts and individual attributes, acting as exogenous, independent variables. These attributes, viewed as *covariates*, could be characteristics of the actors like age, gender, or other relevant properties. This flexibility in defining the effects, and so the evaluation function, enables the model to be applied across diverse network analyses.

The evaluation function u_i determinates the probability that the network transition from the current state $\boldsymbol{x}_0$ to $\boldsymbol{x}$, when individual i has the opportunity to modify the network. In the SAOM the probability is

$$P(\boldsymbol{x}|i,\boldsymbol{x_0}) = \frac{e^{u_i(\boldsymbol{x})} I(\boldsymbol{x} \in \mathscr{A}_i(\boldsymbol{x}_0))}{\sum_{\xi \in \mathscr{A}_i(\boldsymbol{x}_0)} e^{u_i(\xi)}}, \tag{1.1.1}$$

where $\mathscr{A}_i(\boldsymbol{x}_0)$ is the set of networks that differs from $\boldsymbol{x}_0$ in at most one tie that starts from i.

The temporal dynamics of the SAOM is regulated by the *rate functions*, $\lambda_i(t)$ for $i \in \{1,...,N\}$. The function $\lambda_i(t)$ determines how frequently actor i has opportunities to modify its outgoing ties at a given time t. Often $\lambda_i(t) = \lambda$ (constant rate) for all i and t, between two consecutive observations. The opportunities are assumed to arise by sampling for actor $i \in \{1,...,N\}$ a random exponentially distributed time with rate $\lambda_i(t)$. The actor for which the sampled time is lower has the opportunity to change the network. The rate functions therefore dictate the frequency of network updates and control the overall pace of the network evolution.

The *method of moment* estimation, in which the estimated parameter is the one for which the expected value of a set of statistic is equal to the observed one, is often used in the SAOM for computational reasons. In the following $\boldsymbol{x}(0)$ and $\boldsymbol{x}(1)$ denote the observed network of relations at times 0 and 1, respectively. For a given vector of parameters, the process can be simulated with starting point $\boldsymbol{x}(0)$, the simulated value $\hat{\boldsymbol{x}}(1)$, can be compared with $\boldsymbol{x}(1)$ to adjust the parameters, until the effects of the simulated networks are equal on average to the effects of the observed one. In this way, the moment equation is solved stochastically. Simulation (from the estimated parameter) is also used for the model evaluation procedure where the quality of the estimate is assessed, for example by computing standard errors, or by testing statistical hypothesis of irrelevance of some effects.

Over the years, the SAOM has been employed across various research fields due to its ability to model complex and evolving networks. In social science, the SAOM has been utilized to study friendship networks among adolescents, helping to shed light on how such networks develop and change over time [66]. In organizational studies, the model has been used to analyze communication or collaboration networks within and across organizations, revealing insights into factors driving information exchange and teamwork [36]. Additionally, in health research, the SAOM has been applied to understand the spread of health-related behaviors, such as smoking or exercise, across social networks [54]. The flexibility and comprehensiveness of the SAOM makes it a powerful tool for understanding the evolution and dynamics of various types of social networks.

1.1.2 Mixed effect models

A *mixed effect model* [46, 38] is a statistical model in which some “fixed" parameters are shared across all experimental units, while other “random" parameters can vary. A *regression with mixed effects* extends the “usual" regression by including mixed effects in the *linear predictor*. In formulas

$$Y \sim \text{Dist}(\mu(\eta)),\ \eta = X\beta + Zv,\ v \sim \text{Norm}(0, I_n \otimes \Sigma(\theta)), \tag{1.1.2}$$

where the fixed parameters are β and θ, μ is a function that maps uniquely the linear predictor to the parameters of the distribution Dist of the response Y, v contains the random parameters of all n individuals in the study and its variance is parametrized by the function Σ.

The fixed parameters of a mixed effect are shared by all individuals in the population, and they are typically the primary interest of the analysis. On the

other hand, the random parameters are used to model the intrinsic variability between the individuals in the population that is studied, specified by the function Σ. So the random parameters can be used to account for the fact that the observations, differently then in the “usual" regression, can be dependent because there might be multiple observations of the same individual.

Various methods can be used to estimate the parameters in a mixed effect models. The main R package to fit mixed effect models, called *lme4* [5], uses maximum likelihood and restricted maximum likelihood (REML). The methods that we use mimic the method of moment estimation algorithm of the SAOM [62] and its generalized version [1].

1.1.3 Sets, multisets, and hash tables in the R language

Sets and *multisets* are defined as *unordered collections* of elements. The difference between them is that for sets the number of times an element is contained in the collection (*multiplicity*) is irrelevant: an element is either in or out a set. For multisets, the multiplicity of elements is extended to strictly positive real numbers, so they can be viewed as weights of the elements.

The mathematical concept of set and multiset, when viewed as dynamic container of elements, can be implemented efficiently using an *hash table*, that is a data structure containing *key-value pairs*, which can be viewed also as maps from keys to values. An hash function is used to compute an index into an array of buckets or slots, which contain the values. The goal of a hash function is to distribute the keys as uniformly as possible across the array. So the keys are mapped to indices of an array that contain the values. An ideal hash function generates unique output (index in the array) for every unique input, but most hash functions used in practice can produce *hash collisions*, in which different inputs are mapped to the same output. Hash tables offer fast average-case time complexity for search, insert, and delete operations, although the efficiency depends on the hash function used.

In the R programming language, hash tables from the package *hash* [11] have *reference semantics*. This is the behaviour in which computational objects are manipulated indirectly through their memory addresses or references, rather than through the objects themselves. This approach is efficient in memory management, especially for mutable data structures like hash tables. When an operation involves the modification of a few key-value pairs, reference semantics ensures that only the memory locations corresponding to these pairs are updated. This selective updating of memory contributes to speedy operations, as it eliminates the need for unnecessary memory alterations or duplications.

In various programming languages, like C++ and Java, set and multiset data structures are usually implemented using a hash table. These are commonly referred to as *unordered sets* and *unordered multisets* due to the absence of a specific sequence order in which elements are stored. This arrangement doesn't correspond to any inherent notion of "order" in the mathematical definitions of sets and multisets, which are defined as unordered collections. Thus, data structures like ordered sets, which are typically implemented using *red-black trees* instead of hash tables, can also appropriately represent the mathematical concept of a set, regardless of the ordered storage of their elements.

1.2 Contributions

In this section the contributions described later in the thesis are summarized.

Stochastic actor oriented model with random effects.

- The evaluation function of the stochastic actor oriented model is generalized so that some effects can be random: with individual specific random parameters.

- Method of moments estimation procedures of the SAOM is adapted by introducing a new moment equation in the system, so that the variance of the random effects can be estimated with the other fixed parameters. This new equation provides a way to update the variance parameter in the stochastic optimization algorithm (Robbins-Monro) used to estimate the parameters (solve the system of moment equations), but the positive definiteness of the variance parameter has to be enforced.

- The model evaluation procedure generalizes the score test used in the SAOM to test the null hypothesis in which one effect is not statistically significant. The generalization of the theory is described for hypotheses of overdispersion of an effect, in which under the null the considered effect is not random, focusing of the important example of a random out-degree effect.

- The computation of the standard errors of the estimates is also discussed, for models with independent random effects (diagonal variance parameter).

- Evaluation of score tests and standard errors require the approximation of the derivative of the statistics with respect to the fixed parameters (rate, evaluation and variance random effects). This derivative is computed by using the chain rule, where the variance parameter is mapped first to the individual random parameters, which are then used to simulate the model, and so their contribution to the score function can be computed as if they are fixed parameters.

- The quality of the developed model and the interpretation of the estimated parameters is discussed in an application using the Kapferer's Taylor Shop dataset. The emphasis is in the difference between a model with only fixed effects, but a more complicated dynamics (inclusion of out-degree activity effect), and a model with the random out-degree, and so heterogeneous individuals.

Simulated method of moments in mixed effect models

- Iterative estimation algorithms based on simulation of the model from the generative process are studied in the method of moments framework and in its generalized version.

- The Neyman's orthogonalization procedure that is used to increase the power of statistical tests when the statistics are correlated, is discussed in detail, and in the generalized method of moments.

Efficient implementation of sets and multisets in R using hash tables

- Sets and multisets of numbers (R objects of type `numeric` and length 1) are implemented in R using an hash table from the package `hash`.

- The implementation provides an interface specified by the S4 class `hset`, in which sets and multisets can be accessed and manipulated by hiding the details of the implementation.

- The computational advantages of this implementation are evident when sets or multisets are viewed as dynamic "containers", as the hash table data structure allows the possibility to include, exclude or access elements in constant time. The performances are compared with the library `sets`.

1.3 Technical summary

In this section, the content of the following chapters is summarized by discussing “directly" the relevant topics of the chapters. Therefore context and background material, that will be given in the chapters, are not discussed here, as the goal is to describe in a brief but detailed way the research contributions explained in length later in the thesis.

1.3.1 Random effects in the stochastic actor oriented model

Chapter 2 focuses on addressing one of current limitations in the SAOM: the assumption that all individuals share the same evaluation function. The assumption, that limits the model's ability to account for the heterogeneity of individuals, is dropped by allowing some effects to be random, as described below.

The evaluation function is the main component of the SAOM, as it specifies how much the individuals like their “placement" in the network. Consequently, when individual i has the opportunity to modify something in the network $\boldsymbol{x}_0$, the function determinates the probabilities of the allowed modifications, as

$$P(\boldsymbol{x}|i,\boldsymbol{x_0}) = \frac{e^{u_i(\boldsymbol{x})} I(\boldsymbol{x} \in \mathscr{A}_i(\boldsymbol{x}_0))}{\sum_{\xi \in \mathscr{A}_i(\boldsymbol{x}_0)} e^{u_i(\xi)}}, \tag{1.3.1}$$

where the function u_i is defined below, and $\mathscr{A}_i(\boldsymbol{x}_0)$ is the set of networks that differs from $\boldsymbol{x}_0$ in at most one tie that starts from i.

As the evaluation function is linear, it can be decomposed into a fixed part (current evaluation function of the SAOM), and a random part with individual-specific random parameters, with variability that is determined by one or more variance parameters. In formulas, with $\boldsymbol{s}_i(\boldsymbol{x})$ and $\boldsymbol{r}_i(\boldsymbol{x})$ vectors of statistics summarizing the placement of i in the network $\boldsymbol{x}$, the linear evaluation function of the SAOM is generalized as

$$u_i(\boldsymbol{x}) = \boldsymbol{\beta}^\top \boldsymbol{s}_i(\boldsymbol{x}) + \boldsymbol{b}_i^\top \boldsymbol{r}_i(\boldsymbol{x}) = \sum_{k=1}^{p} \beta_k s_{ik}(\boldsymbol{x}) + \sum_{h=1}^{q} b_{ih} r_{ih}(\boldsymbol{x}), \tag{1.3.2}$$

where the first component parametrized by $\boldsymbol{\beta} \in \mathbb{R}^p$ is the current evaluation function used in the SAOM, whereas the “new" second component uses as weights the *individual-specific random parameter* $\boldsymbol{b}_i \in \mathbb{R}^q$ with distribution

$$\boldsymbol{b}_i \sim \mathcal{N}(\boldsymbol{0}_q, \boldsymbol{\Sigma}), \tag{1.3.3}$$

where the q-dimensional positive definite matrix $\boldsymbol{\Sigma}$ is a parameter that has to be estimated (together with $\boldsymbol{\beta}$), and each $\boldsymbol{b}_i$ is an independent sample. An important example that will be described in detail, is the model with *random out-degree*, which relaxes the assumption that all individuals in the network would like to have the same number of outgoing connections, other things being equal. In this model, $q = 1$, $s_{i1}(\boldsymbol{x}) = r_{i1}(\boldsymbol{x})$ is the out-degree of actor i, and so the evaluation function can be written as

$$u_i(\boldsymbol{x}) = (\beta_1 + b_i)s_{i1}(\boldsymbol{x}) + \sum_{k=2}^{p} \beta_k s_{ik}(\boldsymbol{x}), \tag{1.3.4}$$

where $b_i \sim \mathcal{N}(0, \sigma^2)$ i.i.d., $\sigma^2 \in (0, \infty)$.

The method of moment estimation method, that is the most used one in the frequentist version of the stochastic actor oriented model, is generalized by including the moment equations necessary to estimate the (positive definite) variance of the random effects with all other parameters. Assuming that there are two observed networks $\boldsymbol{x}(t_1)$ and $\boldsymbol{x}(t_2)$, in the model with random out-degree, the moment equation that is included in the system for σ^2 is

$$E\Big(\sum_{i=1}^{N} \big(s_{i1}(\boldsymbol{X}(t_2)) - \bar{s}_1(\boldsymbol{X}(t_2))\big)^2 \,\Big|\, \boldsymbol{x}(t_1) \Big) = \sum_{i=1}^{N} \big(s_{i1}(\boldsymbol{x}(t_2)) - \bar{s}_1(\boldsymbol{x}(t_2))\big)^2, \tag{1.3.5}$$

where the expected value depends on $\boldsymbol{\beta}$, σ^2, and the rate parameters. The system of moment equations is solved stochastically with the *Robbins-Monro algorithm*, each iteration consists of two main steps. The network is first simulated from the current parameters. Then, the difference between the statistics (used in the moment equations) computed in the simulated and observed networks, is used to update the parameters.

In our generalization of the Robbins-Monro algorithm, the first step is modified by including the sampling of the random parameters before simulating the network with the SAOM. The second step of the iteration is similar to the original one, however the positive definiteness of the variance parameter must be enforced. In the model with random out-degree, the network $\hat{\boldsymbol{x}}$ is simulated as

$$\boldsymbol{b}_i \sim \mathcal{N}(0, \sigma^2), \quad \text{then} \quad \hat{\boldsymbol{x}} \sim \mathcal{S}(\boldsymbol{\beta}, \boldsymbol{b}). \tag{1.3.6}$$

The update step for the variance parameter is

$$\sigma^2 \leftarrow \max\big(\sigma^2 - \zeta d\big(\hat{w}(\boldsymbol{\beta}, \boldsymbol{b}(\sigma^2)) - w\big),\, \sigma^2_{\min}\big), \tag{1.3.7}$$

where w is the right hand side of equation (1.3.5), $\hat{w}(\boldsymbol{\beta}, \boldsymbol{b}(\sigma^2))$ is a simulated value that is an unbiased estimate of the left hand side of (1.3.5), ζ and d are positive values, $\sigma^2_{\min} > 0$ is the minimum allowed value for σ^2 in the optimization.

The model evaluation part required more substantial modifications than the estimation method. We describe how to compute the variance of the estimator, and how to test the null hypothesis $H_0 : \sigma^2 = 0$ of the *absence of overdispersion in the out-degree*, against $H_1 : \sigma^2 > 0$ with a *score-type test*. The combination of the estimation functions for $\boldsymbol{\beta} \in \mathbb{R}^p$ and σ^2 is denoted by $\boldsymbol{g}(\boldsymbol{x}) = (\boldsymbol{s}(\boldsymbol{x}), w(\boldsymbol{x}))$, the estimated parameter under the null is $\hat{\boldsymbol{\beta}}$. Developing a score-type test statistic with good power, requires however the ability to approximate the $(p+1) \times p$ dimensional *Jacobian matrix*

$$\boldsymbol{J}(\hat{\boldsymbol{\beta}}) = \frac{\partial E_{\boldsymbol{\beta}}(\boldsymbol{g}(\boldsymbol{X}))}{\partial \boldsymbol{\beta}^\top}\bigg|_{\boldsymbol{\beta}=\hat{\boldsymbol{\beta}}} = E_{\boldsymbol{\beta}}\Big(\boldsymbol{g}(\boldsymbol{X}) \frac{\partial}{\partial \boldsymbol{\beta}^\top} \log f(\boldsymbol{X} \mid \boldsymbol{\beta})\Big)\bigg|_{\boldsymbol{\beta}=\hat{\boldsymbol{\beta}}}, \tag{1.3.8}$$

where $f(\boldsymbol{x} \mid \boldsymbol{\beta})$ is the probability mass function of the network $\boldsymbol{x}$ for parameter $\boldsymbol{\beta}$. This derivative is approximated with *Monte-Carlo integration* as

$$\hat{\boldsymbol{J}}(\hat{\boldsymbol{\beta}}) = \frac{1}{\mathsf{T}} \sum_{\mathtt{t}=1}^{\mathsf{T}} (\hat{\boldsymbol{g}}_\mathtt{t} - \boldsymbol{g}) \hat{\boldsymbol{l}}_\mathtt{t}, \tag{1.3.9}$$

where $\hat{\boldsymbol{g}}_\mathtt{t}, \boldsymbol{g} \in \mathbb{R}^{p+1}$ are the simulated and observed statistics, respectively, and $\hat{\boldsymbol{l}}_\mathtt{t} \in \mathbb{R}^{1 \times p}$ are, respectively, the simulated statistics and the row-vector of contributions to the score function, that is the simulated version of $\partial \log f(\boldsymbol{X} \mid \boldsymbol{\beta}) / \partial \boldsymbol{\beta}^\top$. For the model evaluation procedure, the quantities $\hat{\boldsymbol{g}}_\mathtt{t}$ and $\hat{\boldsymbol{l}}_\mathtt{t}$ are computed from networks simulated from the estimated parameter.

A test statistic for H_0 obtained *orthogonalizing* the statistics $w(\boldsymbol{X})$ and $\boldsymbol{s}(\boldsymbol{X})$ is

$$Y(\hat{\boldsymbol{\beta}}) = w(\boldsymbol{X}) - \boldsymbol{\Gamma}(\hat{\boldsymbol{\beta}}) \boldsymbol{s}(\boldsymbol{X}), \quad \boldsymbol{\Gamma}(\hat{\boldsymbol{\beta}}) = \boldsymbol{J}_2(\hat{\boldsymbol{\beta}}) \boldsymbol{J}_1(\hat{\boldsymbol{\beta}})^{-1}, \tag{1.3.10}$$

where the Jacobian $\boldsymbol{J}(\hat{\boldsymbol{\beta}})$ is decomposed in the block $\boldsymbol{J}_1(\hat{\boldsymbol{\beta}})$, and the bottom row $\boldsymbol{J}_2(\hat{\boldsymbol{\beta}})$. The *p-value* for the test is then computed as

$$\hat{\alpha} = \frac{1}{\mathsf{T}} \sum_{\mathtt{t}=1}^{\mathsf{T}} I(\hat{y}_\mathtt{t} > y), \tag{1.3.11}$$

where $y = w - \hat{\boldsymbol{\Gamma}} \boldsymbol{s}$ and $\hat{y}_\mathtt{t} = \hat{w}_\mathtt{t} - \hat{\boldsymbol{\Gamma}} \hat{\boldsymbol{s}}_\mathtt{t}$ are the observed and simulated (from values in iteration $\mathtt{t}$) test statistics, respectively, and $\hat{\boldsymbol{\Gamma}}$ is the matrix computed from $\hat{\boldsymbol{J}}(\hat{\boldsymbol{\beta}})$.

The estimated parameter $(\hat{\boldsymbol{\beta}}, \hat{\sigma}^2)$ for the model with random out-degree, computed with method of moments, has asymptotic covariance

$$\hat{\boldsymbol{C}}(\hat{\boldsymbol{\beta}}, \hat{\sigma}^2) = \hat{\boldsymbol{J}}(\hat{\boldsymbol{\beta}}, \hat{\sigma}^2)^{-1} \hat{\boldsymbol{V}}(\hat{\boldsymbol{\beta}}, \hat{\sigma}^2) \hat{\boldsymbol{J}}(\hat{\boldsymbol{\beta}}, \hat{\sigma}^2)^{-\top}, \tag{1.3.12}$$

where the matrices $\hat{\boldsymbol{V}}$ and $\hat{\boldsymbol{J}}$ are Monte-Carlo approximations of

$$\begin{aligned} \boldsymbol{V}(\hat{\boldsymbol{\beta}}, \hat{\sigma}^2) &= E_{\hat{\boldsymbol{\beta}}, \hat{\sigma}^2}\big((\boldsymbol{g}(\boldsymbol{X}) - E_{\hat{\boldsymbol{\beta}}, \hat{\sigma}^2}(\boldsymbol{g}(\boldsymbol{X}))(\boldsymbol{g}(\boldsymbol{X}) - E_{\hat{\boldsymbol{\beta}}, \hat{\sigma}^2}(\boldsymbol{g}(\boldsymbol{X}))^\top\big), \\ \boldsymbol{J}(\hat{\boldsymbol{\beta}}, \hat{\sigma}^2) &= E_{\boldsymbol{\beta}, \sigma^2}\Big(\boldsymbol{g}(\boldsymbol{X}) \frac{\partial}{\partial(\boldsymbol{\beta}^\top \sigma^2)} \log f(\boldsymbol{X} \mid \boldsymbol{\beta}, \boldsymbol{b}(\sigma^2))\Big)\Big|_{\boldsymbol{\beta}=\hat{\boldsymbol{\beta}}, \sigma^2=\hat{\sigma}^2}. \end{aligned} \tag{1.3.13}$$

The derivative of the log-density can be written as

$$\begin{aligned} &\frac{\partial}{\partial(\boldsymbol{\beta}^\top \sigma^2)} \log f(\boldsymbol{X} \mid \boldsymbol{\beta}, \boldsymbol{b}(\sigma^2)) = \frac{\partial}{\partial(\boldsymbol{\beta}^\top \sigma^2)} \log f(\boldsymbol{X}, \boldsymbol{u} \mid \boldsymbol{\beta}, \sigma^2) = \\ &= \left(\frac{\partial}{\partial(\boldsymbol{\beta}^\top \boldsymbol{b}^\top)} \log f(\boldsymbol{X} \mid \boldsymbol{\beta}, \boldsymbol{b})\right)\left(\frac{\partial}{\partial(\boldsymbol{\beta}^\top \sigma^2)}\left[\begin{pmatrix}\boldsymbol{\beta}\\ \sigma^2\end{pmatrix} \mapsto \begin{pmatrix}\boldsymbol{\beta}\\ \boldsymbol{b}\end{pmatrix}\right]\right), \end{aligned} \tag{1.3.14}$$

with $\boldsymbol{b} = \sigma^2 \boldsymbol{u}$, because the density of $\boldsymbol{u}$ does not depend on the parameter. The derivative of the function $(\boldsymbol{\beta}, \sigma^2) \mapsto (\boldsymbol{\beta}, \boldsymbol{b})$ is a $(p+N) \times (p+1)$ block diagonal matrix, with top-right and bottom-left blocks equal to $\boldsymbol{I}_p$ and $\boldsymbol{b}/2\sigma^2$, respectively. The matrix $\boldsymbol{J}(\hat{\boldsymbol{\beta}}, \hat{\sigma}^2)$ is then approximated as

$$\hat{\boldsymbol{J}}(\hat{\boldsymbol{\beta}}, \hat{\sigma}^2) = \frac{1}{\mathsf{T}} \sum_{\mathsf{t}=1}^{\mathsf{T}} \begin{pmatrix} (\hat{\boldsymbol{s}}_\mathsf{t} - \boldsymbol{s}) \hat{\boldsymbol{l}}_{\boldsymbol{\beta}\mathsf{t}} & \frac{1}{2\hat{\sigma}^2}(\hat{\boldsymbol{s}}_\mathsf{t} - \boldsymbol{s}) \hat{\boldsymbol{l}}_{\boldsymbol{b}\mathsf{t}} \boldsymbol{b}_\mathsf{t} \\ (\hat{w}_\mathsf{t} - w) \hat{\boldsymbol{l}}_{\boldsymbol{\beta}\mathsf{t}} & \frac{1}{2\hat{\sigma}^2}(\hat{w}_\mathsf{t} - w) \hat{\boldsymbol{l}}_{\boldsymbol{b}\mathsf{t}} \boldsymbol{b}_\mathsf{t} \end{pmatrix}, \tag{1.3.15}$$

where $(\hat{\boldsymbol{s}}_\mathsf{t}, \hat{w}_\mathsf{t}) = \boldsymbol{g}_\mathsf{t}$ is simulated from $(\hat{\boldsymbol{\beta}}, \boldsymbol{b}_\mathsf{t})$, $\boldsymbol{b}_\mathsf{t}$ is simulated from $\hat{\sigma}^2$, $\hat{\boldsymbol{l}}_{\boldsymbol{\beta}\mathsf{t}}$ and $\hat{\boldsymbol{l}}_{\boldsymbol{b}\mathsf{t}}$ are the contribution to the score functions of the parameters $\boldsymbol{\beta}$ and $\boldsymbol{b}$, respectively, for the simulated network in iteration t.

We assess the SAOM with random out-degree in the *Kapferer's Tailor Shop dataset*. This longitudinal network comprises two waves with 39 individuals of varying status (low or high). The out-degree frequencies exhibit significant variability, suggesting the presence of a *random out-degree* effect. The (fixed) *out-degree activity* effect also contributes to explaining this observed variability. However, estimating a model that includes both out-degree activity and random out-degree is unfeasible due to optimization convergence issues. The quality of various models has been checked with score tests between pairs of "nested" models. Also from this procedure, the two best models are the ones containing all effects considered in the analysis, and either one of out-degree activity, or

random out-degree, but not both simultaneously. These two models cannot be directly compared and provide distinct interpretations of individuals' behavior in tie formation or dissolution. In the model with a positive estimated parameter for out-degree activity, individuals with large out-degree are more likely to form ties. Conversely, random out-degree is interpreted as capturing the heterogeneity among individuals in their preference for forming different number of ties. Incorporating random effects increases the variability of the underlying stochastic process, so higher standard errors are the price to pay for dropping the homogeneity assumption. In this dataset however, the number of simulated tie flips necessary to simulate a network "similar" to the observed one is lower on average when the random out-degree is included Therefore, the random effect aids in achieving a more parsimonious dynamics, characterized by a reduced number of transitions.

1.3.2 Generalizing methods to regression with mixed effects

Chapter 3 extends the generalization of the SAOM to linear models with mixed effects. The aim here is to explore the properties of these models and gain a deeper understanding of their capabilities and potential applications.

Part of the statistical analysis consists in estimating the values of the fixed parameters $\beta \in \mathbb{B} = \mathbb{R}^p$ and $\theta \in \mathbb{T} \subseteq \mathbb{R}^r$, for the mixed effect model defined as in equation (1.1.2). For the method of moment, and the generalized method of moment procedures defined in the chapter, a vector of statistics for the parameter is $s = (s_\beta, s_\theta)$, whose components are defined as

$$\begin{aligned} s_\beta(y) &= (X^\top X)^{-1} X^\top y \in= \mathbb{R}^p, \\ s_v &= (Z^\top Z)^{-1} Z^\top (y - X s_\beta(y)) \in \mathbb{R}^{qn}, \\ s_\Sigma &= \frac{1}{n} \sum_{j=1}^{n} (s_{vj} - \bar{s}_v)(s_{vj} - \bar{s}_v)^\top \in \mathbb{S}_q^+, \\ s_\theta(y) &= \Sigma^{-1}(s_\Sigma) \in \mathbb{R}^r, \end{aligned} \tag{1.3.16}$$

where s_{vj} contains the elements of s_v in positions from $(q-1)j+1$ to qj, with q as the number of random effects, and $\bar{s}_v$ is the mean of s_{vj} across individuals j.

For $\alpha = (\beta, \theta) \in \mathbb{A} \subseteq \mathbb{R}^d$, with $d = p + r$, method of moments estimators are defined as the solution of

$$L(E_\alpha(s(Y)) - s(y)) = 0_d, \tag{1.3.17}$$

where $L \in \mathbb{R}^{d \times d'}$ is a full rank matrix, and s is a d' dimensional vector of statistics, with $d' = d$ in the "usual" method of moments, and $d' > d$ in the generalized version. The equation is solved stochastically with the iterative algorithm

$$\alpha \leftarrow \text{proj}(\alpha - \epsilon H L(s(Y) - s(y)), \mathbb{A}), \tag{1.3.18}$$

where Y is the simulated response variable from the current value α of the parameter, y is the observed response variable, ϵ and H are the learning rate and the preconditioning matrix of the algorithm, L is defined by the moment equation that is solved. Its optimal value is $L \propto GV^{-1}$, where

$$\begin{aligned} G &= E_{\alpha^*}(\partial (E_\alpha(s(Y)) - s(y))/\partial \alpha^\top) \in \mathbb{R}^{d' \times d}, \\ V &= E_{\alpha^*}((E_\alpha(s(Y)) - s(y))(E_\alpha(s(Y)) - s(y))^\top) \in \mathbb{S}^+_{d'}, \end{aligned} \tag{1.3.19}$$

and α^* is the real value of the parameter. In practise L is set at the beginning of the algorithm by approximating with simulation G and V with the corresponding values at the starting parameter α_0. After the initial time period until iteration b, in which the iterative process (1.3.18) has not yet reached stationarity, the values α_k, for $k \in \{b+1, ..., m_{\text{est}}\}$ are averaged to compute the estimator

$$\hat{\alpha} = \frac{1}{m_{\text{est}} - b} \sum_{k=b+1}^{m_{\text{est}}} \alpha_k. \tag{1.3.20}$$

The matrices G and V defined in (1.3.19) are estimated with

$$\hat{G} = \frac{1}{m_{\text{sim}}} \sum_{k=1}^{m_{\text{sim}}} \Big(\big(s(Y_k(\zeta_k)) - s(y)\big) \Big(\frac{\partial}{\partial \zeta_k^\top} \log p(Y_k(\zeta_k) | \zeta_k) \Big) \frac{\partial f_k(\alpha)}{\partial \alpha^\top} \Big|_{\alpha = \hat{\alpha}} \Big), \tag{1.3.21}$$

and

$$\hat{V} = \frac{1}{m_{\text{sim}}} \sum_{k=1}^{m_{\text{sim}}} \big(s(Y_k(\zeta_k)) - \hat{s}\big) \big(s(Y_k(\zeta_k)) - \hat{s}\big)^\top, \tag{1.3.22}$$

respectively, where for mixed effect models $\zeta_k = f_k(\hat{\beta}, \hat{\theta}) = (\hat{\beta}, v_k) \in \mathbb{B} \times \mathbb{R}^{qn}$, with v_k sampled from $\hat{\theta}$, and $\hat{s}$ is the mean of the m_{sim} simulated statistics. These matrices are used to compute the covariance of $\hat{\alpha}$, and to evaluate test statistics on the fixed parameters. In particular, test statistics for α, when the parameter is estimated with method of moments, are derived from

$$\begin{aligned} w_k &= s_2(Y_k) - \hat{\Lambda} s_1(Y_k), \\ w &= s_2(y) - \hat{\Lambda} s_1(y), \end{aligned} \tag{1.3.23}$$

that are the simulated (at iteration k) and the observed test statistics, respectively, where $\hat{\Lambda} = \hat{G}_{21}\hat{G}_{11}^{-1}$ according to the decomposition of s (and G) as

$$s = \begin{pmatrix} s_1 \\ s_2 \end{pmatrix}, \quad G = \begin{pmatrix} G_{11} \\ G_{21} \end{pmatrix}, \tag{1.3.24}$$

with s_1 that is used to estimate the parameters under the null, and s_2 that is assumed to contain information about the hypothesis that is tested. We generalized the statistic w_k to

$$w'_k = s_2(Y_k) - \hat{\Lambda}'(s'_1(Y_k) - \hat{b}_1), \quad \hat{b}_1 = \hat{s}'_1 - s'_1(y), \tag{1.3.25}$$

where $\hat{s}'_1$ is the mean of the simulated statistics that are used to estimate the model with generalized method of moments. The use of the estimated "bias" $\hat{b}_1$ is necessary because under the alternative hypothesis the simulated statistics are not equal to the observed ones on average, as only the linear combinations of them determined by L, are equal on average. Moreover, $\hat{\Lambda}'$ is computed using the left inverse of $\hat{G}'_{11}$, rather than the inverse, because with generalized method of moments the number d of parameters under the null is smaller than the number d' of statistics that are used to estimate them.

A simulation study has been done to check whether and how much the orthogonalization procedure is useful to increase the power of the test. We found that orthogonalization is useful when the statistics s_1 (or s'_1) that are used to estimate the parameters are correlated with s_2 from which the test statistic is derived. If there is no correlation between the statistics, orthogonalization reduces the power of the test because of the new variability introduced on the test statistic by using the approximated $\hat{\Lambda}$ and the simulated $s_1(Y_k)$.

1.3.3 Implementing sets and multisets in R

Chapter 4 addresses a more computational aspect: an efficient implementation in the R language of sets and multisets, which can be used as containers in algorithms. We present the R package, *hset*, which offers an implementation of a class for sets and multisets of numbers.

The S4 class *hset* specifies an interface in which sets and multisets can be manipulated. It contains two slots, the first with the hash table that contains the elements (and multiplicities for multisets), the second one contains the information about whether the object is a set, or a multiset. The elements that can be stored in our implementation are either numbers, or sets of numbers.

Mathematically, the set S of possible sets that can be stored is recursively defined as

$$\begin{aligned} &X = \{a_1, a_2, ...\} \in \mathtt{S}, \\ &a_i \in \mathtt{S} \uplus \mathtt{N}, \end{aligned} \tag{1.3.26}$$

where N is the set of `numeric` vectors of length 1, without some values such as `Inf`, that are excluded. Multisets are defined similarly.

The hash table from the package *hash* uses C, that denotes here the set of character vectors of length one, as the set of possible keys. So elements can be encoded as keys by the injection $k : \mathtt{S} \uplus \mathtt{N} \to \mathtt{C}$, which labels uniquely each possible element that is "allowed" in our implementation. For sets, the value associated with the key $k_i = k(a_i)$ is `""` when a_i is contained in the set, otherwise k_i is not the key of a pair in the table. The element a_i is contained in the multiset with multiplicity $m_i > 0$, if the pair (k_i, m_i) is contained in the hash table that implements the multiset.

The *inclusion relation* between an element a and a set X defined mathematically as

$$\in : (\mathtt{N} \uplus \mathtt{S}) \times \mathtt{S}, \quad a \in X \iff X = \{a, ...\}, \tag{1.3.27}$$

and the inclusion relation between an element a and a multiset Y, parametrized by the multiplicity m and the type of relation $\sim$, defined mathematically as

$$\in_{\sim}^{m} : (\mathtt{N} \uplus \mathtt{S}) \times \mathtt{M}, \quad a \in_{\sim}^{m} Y \iff Y = \{a[n], ...\},\ m - n \sim 0, \tag{1.3.28}$$

can be both evaluated by a predicate with signature

$$\mathtt{C} \times (\mathtt{S} \uplus \mathtt{M}) \times \mathtt{N}_+ \times \{\leq, <, =\} \to \{\mathtt{TRUE}, \mathtt{FALSE}\}, \tag{1.3.29}$$

where $\mathtt{N}_+ = \mathtt{N} \cap (0, \infty)$. The four arguments of the predicate are the label of the element, the set or the multiset of which the element might be in relation to, the multiplicity, and the type of the relation, respectively. The last two arguments are ignored when the second one is a set. *Subset relations* of different types between two sets or multisets are evaluated by a predicate with signature

$$(\mathtt{S} \uplus \mathtt{M}) \times (\mathtt{S} \uplus \mathtt{M}) \times \{\mathtt{TRUE}, \mathtt{FALSE}\} \times \{\mathtt{TRUE}, \mathtt{FALSE}\} \to \{\mathtt{TRUE}, \mathtt{FALSE}\}, \tag{1.3.30}$$

where the third and fourth arguments specifies whether the inclusion relation is strict and exact, respectively. The fourth one is used only if one of the first two arguments is a multiset.

The five *operations* of *intersection*, *union*, *difference*, *symmetric difference* and

sum, have all signature

$$\approx\colon (\mathtt{S} \uplus \mathtt{M}) \times (\mathtt{S} \uplus \mathtt{M})^* \to (\mathtt{S} \uplus \mathtt{M}), \tag{1.3.31}$$

where $Z^* = \uplus_{k \geq 0} Z^k$. These five operations are all implemented by functions with the common signature

$$(\mathtt{S} \uplus \mathtt{M}) \times (\mathtt{S} \uplus \mathtt{M})^* \times \{\texttt{"refer"}, \texttt{"value"}\} \to (\mathtt{S} \uplus \mathtt{M}), \tag{1.3.32}$$

where the second argument contains all operands except the first one, and the third argument specifies the semantic. These five functions are all computed using only two basic functions with signature

$$\begin{aligned} &\mathtt{S} \times \mathtt{S}^* \times (\mathtt{B}^* \to \mathtt{B}) \times \mathtt{B} \times \{\texttt{"refer"}, \texttt{"value"}\} \to \mathtt{S}, \\ &(\mathtt{S} \uplus \mathtt{M}) \times (\mathtt{S} \uplus \mathtt{M})^* \times ((\mathtt{N}_+ \uplus \mathtt{0})^* \to \mathtt{N}_+ \uplus \mathtt{0}) \times \mathtt{B} \times \{\texttt{"refer"}, \texttt{"value"}\} \to \mathtt{M}, \end{aligned} \tag{1.3.33}$$

in the case in which all operands are sets, or in the case in which at least one operand is a multiset, respectively. In the signatures above, $\mathtt{B} = \{\mathtt{TRUE}, \mathtt{FALSE}\}$. The third argument of these functions specifies which of the five operations is considered. The fourth argument is the *identity element* of the operation, that is the universe set or multiset for the intersection, and the empty set or multiset for all other operations. The identity element determinates whether it is necessary to cycle through all elements of the first operand in order to either compute the result (value semantic), or to transform the first operand into the result (reference semantic).

The performance of our implementation are compared with the package *sets* [40]. Relations between an element and a set can be evaluated in constant time with respect to the size of the set. For inclusion relations the computational complexity is linear with respect to the size of the first component of the relation, but constant with respect to the size of the second one. On the other hands in *sets*, the computational complexity for evaluating whether an element is included into a set is linear with respect to the size of the set. For binary operations, when the identity element is the empty set (union, difference, symmetric difference, sum) and reference semantic is used, the computational complexity to modify the first operand into the result of the operation is constant with respect to the size of the first operand, and linear with respect to the size of the second one, allowing an important speed up when the first operand is large. Whereas in *sets* the computational complexity for computing the result is linear in the size of the

largest operand, as it is in our implementation if reference semantic is not used. For evaluating the intersection between two sets our implementation does not provide an advantage, as evaluating this operation requires traversing through all elements of the first operand, which is assumed here to be the largest. The same analysis is valid when multisets are used instead of sets.

Markov processes can have a state space that is discrete or partially discrete. An example, is a process with state $(X_t, Z_t) \in \mathscr{X} \times \mathscr{Z}$, where X_t is the set of edges of a graph and Z_t is the multiset that contains the degree frequencies (number of nodes with given degree), that is a statistic, of the same graph. The network process evolves by tie flips in which one non-edge becomes an edge, or the opposite. In particular, a tie flip is proposed, and accepted with probability computed with the Metropolis-Hastings ratio of probability distributions over graphs. The distribution used in this example is the beta model

$$P(X = x|\beta) = \prod_{1\le i<j\le n} \frac{e^{\beta_i+\beta_j}}{1+e^{\beta_i+\beta_j}}, \tag{1.3.34}$$

where $\beta_i \in \mathbb{R}$ is the parameter for the vertex $i \in \{1, ..., n\}$. Over time, the process converges to its stationary distribution that depends on the parameter $\beta \in \mathbb{R}^n$.

The set of proposed flips F is used to compute the set $\mathscr{I}_F$ of vertices that are part of at least one proposed tie flip, and the proposed degrees $\tilde{d}_i$ for $i \in \mathscr{I}_F$. If the flip is accepted, the state of the process is updated as

$$\begin{aligned} &X_{t+1} \leftarrow X_t \triangle F, \\ &Z_{t+1} \leftarrow (Z_t - \{d_i[m_i] : i \in \mathscr{I}_F\}) + \{\tilde{d}_i[\tilde{m}_i] : i \in \mathscr{I}_F\}, \end{aligned} \tag{1.3.35}$$

where $\tilde{m}_i$ and m_i are the multiplicities of proposed $\tilde{d}_i$ and current d_i degrees respectively, for $i \in \mathscr{I}_F$. The update uses the symmetric difference (xor) operation $\triangle : \mathtt{S}\times\mathtt{S} \to \mathtt{S}$ between sets, with the sum $+ : \mathtt{M}\times\mathtt{M} \to \mathtt{M}$ and difference $- : \mathtt{M}\times\mathtt{M} \to \mathtt{M}$ between multisets. In our implementation, if X_t and Z_t are used as first operand of the respective operations with reference semantics, the complexity of these three operations will not depend on the size of X_t and Z_t, but only on the size of F and of $\mathscr{I}_F$, where $|\mathscr{I}_F| \le 2|F|$. So the complexity for updating the state is $O(|F|)$. The updates in equation (1.3.35) can be coded in our implementation

with reference semantic as

```
state$edge.set %xor% hset(flips$id)
state$degree.frequencies %-% hset(names(table.old.degrees),
 as.integer(table.old.degrees))
state$degree.frequencies %+% hset(names(table.new.degrees),
 as.integer(table.new.degrees))
```

(1.3.36)

where `state$edge.set` and `state$degree.frequencies` are `hset` objects representing X_t and Z_t respectively, the constructor `hset` is used to create the set F (with default second argument), and the multisets (with multiplicities as second argument) containing the updates for the degree frequencies.

Chapter 2

Stochastic actor oriented model with random effects

2.1 Introduction

Modeling the behavior of a group of people when interactions between them emerge and dissolve, is one the most ambitious goals of social network analysis. The first overview of statistical models used to analyze social network data appeared in the 1990s [72], following pioneering work on the quantitative study of social networks in the 1970s [e.g. 26, 27]. During this development, various basic continuous-time Markov chains were proposed to describe the evolution of a social network under a number of simplifying assumptions [71]. This formed the basis for the stochastic actor-oriented model for network change [59, 60, 62], which has become known as the SAOM. The SAOM parametrizes the evolution of the stochastic network of relations. The parameters of the model quantify how much a given configuration of the network, called effect, influences the choice of forming or dissolving ties. These parameters are shared by all actors, so differences between individual rules of network change have to be defined by means of observed covariates.

The study of social behaviour often either focuses on modelling the choices of heterogeneous actors in a relatively simple environment, or on modelling the choices of relatively homogeneous actors in a complex environment. The generalization we are proposing is to allow some parameters to be random, to model more accurately the heterogeneity of the actors in the choices they make. The heterogeneity of the actors is then parametrized by the variance of the random parameters.

In Section 2.2, longitudinal network data are introduced along with the stochas-

tic actor oriented model. The SAOM with random effects is discussed as generalization of the standard SAOM, following a formulation of the model similar to Schweinberger [55]. As an example we describe a model with random out-degree.

The estimation method described in Section 2.3 is an extension of the Robbins-Monro algorithm that is used in the "standard" SAOM, the main difference being that the positive definiteness of the variance parameter must be enforced. Various alternatives for modelling the variance parameter when there is more than one random effect are discussed, together with various other topics relevant to the estimation algorithm. Finally it is explained how to estimate the model with random out-degree using the current implementation of the SAOM.

Section 2.4 explains how to test hypotheses on the variance parameters and how to compute the standard error of the estimates. The theory is described in detail for the hypothesis of overdispersion, i.e., heterogeneity in the out-degrees of the actors not explained by the model with fixed effects only. It is also shown how to compute the standard error of the estimated parameters in this model.

In Section 2.5 we apply the derived method to a tailor shop social network, reanalyzing a study by Kapferer [31]. The study focuses on the changes in a social network of 39 individuals in a tailor shop in Zambia at the start and the end of a 7-month period. Previous analyses have shown a marked effect of various transitive effects in the interactions between the individuals. Various models, with and without random out-degree, are estimated and compared with the algorithm and model evaluation procedures derived in previous sections. Several parameters are interpreted in detail to discuss how, and possibly why, the inclusion of the random out-degree affects the estimation of the other parameters. Finally, we attempt to provide some guidelines on using random effects in the stochastic actor oriented model.

2.2 Stochastic actor oriented model with random effects

In this section we sketch the *Stochastic Actor Oriented Model* (*SAOM*) with the generalization that we propose. The estimation method will be discussed in detail in Section 2.3.

A *longitudinal network study* (or *panel network study*) is an observational study where, in its simplest form, a relational network on a fixed set of actors is observed at two or more time points. We assume here that the network is directed. An example is a friendship network between students in a classroom.

The *stochastic actor oriented model* (*SAOM*), is a method to analyse data from

a longitudinal network study. This is a parametric model for a stochastic process in continuous time on the outcome space of directed graphs, assuming that changes, when they occur, consist of a change of only one tie variable. The process is observed at discrete time points, and unobserved in between. The SAOM was first introduced in [60]. For a review of the model, including a discussion of various generalizations, see [62]. An extensive discussion of the model is in the manual of the package *RSiena* [49]. The SAOM is not the only model for longitudinal networks. An important alternative is the *temporal exponential random graph model* (*TERGM*) described in [23] and [34]. An extensive comparison of the two models is in [8] and [9]. *Relational event models* [12, 45] can also be used for longitudinal network data. However, this approach requires the observation of individual relational events, while in SAOM and TERGMs the history of events between the observations is not available.

We first describe the evolution of the process in detail in a simplified set-up. We assume that a simple directed graph without self-loops with a node set consisting of N actors is observed at $M = 2$ time points t_1 and t_2. The *state space* of the process (possible values for the dependent variable) is $\mathscr{X} \cong \{0,1\}^{N(N-1)}$, which can be represented as the set of binary matrices with values 0 in the diagonal. States $\boldsymbol{x} \in \mathscr{X}$ are adjacency matrices with elements $x_{ij} = 1$ in row i and column j if the tie (ij) is in the graph, otherwise $x_{ij} = 0$; moreover, $x_{ii} = 0$ for all i. The observed networks are $\boldsymbol{x}(t_1)$ and $\boldsymbol{x}(t_2)$, respectively. The unobserved evolution of the network between t_1 and t_2 is the right continuous random function $t \mapsto \boldsymbol{X}(t)$ such that $\boldsymbol{X}(t) \in \mathscr{X}$, $\boldsymbol{X}(t_1) = \boldsymbol{x}(t_1)$ and $\boldsymbol{X}(t_2) = \boldsymbol{x}(t_2)$. If the process changes at time t, an actor, denoted here by $i \in \{1,...,N\}$, has the opportunity to add or remove one outgoing tie, or to leave the configuration as it is. Therefore, if $\boldsymbol{x}_0 = \boldsymbol{x}(t^-)$ is the value of the network (state of the process) immediately before the possible change, the process can jump to the state $\boldsymbol{x} \in \mathscr{A}_i(\boldsymbol{x}_0) \subset \mathscr{X}$, where the *adjacency set* $\mathscr{A}_i(\boldsymbol{x}_0)$ is the set of networks that differ from $\boldsymbol{x}_0$ in at most one tie that starts from i.

The model for the opportunity of transitions in the state space is now introduced. In the stochastic actor oriented model, $\boldsymbol{X}(t)$ is a random variable with probability distribution conditional on actor i making a network change

$$P(\boldsymbol{X}(t) = \boldsymbol{x} \mid i, \boldsymbol{X}(t^-) = \boldsymbol{x}_0) \propto \exp(u_i(\boldsymbol{x}))I(\boldsymbol{x} \in \mathscr{A}_i(\boldsymbol{x}_0)), \tag{2.2.1}$$

where $I : \{F,T\} \to \{0,1\}$ is the indicator function, and $u_i : \mathscr{X} \to \mathbb{R}$ is the *linear evaluation function*

$$u_i(\boldsymbol{x}) = \boldsymbol{\beta}^\top \boldsymbol{s}_i(\boldsymbol{x}) = \sum_{k=1}^{p} \beta_k s_{ik}(\boldsymbol{x}), \tag{2.2.2}$$

in which $\boldsymbol{\beta} \in \mathbb{R}^p$ is a parameter that weights the p dimensional vector of statistics $\boldsymbol{s}_i(\boldsymbol{x})$, which contains information on the "position" of the focal actor i in the network. It is assumed that $\boldsymbol{x}_0 \in \mathscr{A}_i(\boldsymbol{x}_0)$ for all $\boldsymbol{x}_0$, so that the "trivial" transition in which the network does not change has always a positive probability. The components $s_{ik}(\boldsymbol{x})$ are called *effects*. Some examples are the following:

- *out-degree* (also called *density*) *effect*: number of ties starting from the focal actor $x_{i+} = \sum_j x_{ij}$,
- *reciprocity effect*: number of reciprocated ties $\sum_j x_{ij}x_{ji}$,
- *transitive triplet effect*: number of ordered pairs (j,h) of actors such that the three ties (ij), (ih) and (hj) are all present in the graph, computed as $\sum_{j,h} x_{ij}x_{ih}x_{hj}$,
- *out-degree related popularity effect*: sum of the out-degrees of the actors to whom i is tied, computed as $\sum_j x_{ij}x_{j+}$.

Effects can also depend on actor covariates or dyadic covariates.

The generalization proposed in this paper, is to decompose the evaluation function in a *fixed* and a *random part*, so that the actors in the network can have individual parameters. The linear evaluation function (2.2.2) is replaced with

$$u_i(\boldsymbol{x}) = \boldsymbol{\beta}^\top \boldsymbol{s}_i(\boldsymbol{x}) + \boldsymbol{b}_i^\top \boldsymbol{r}_i(\boldsymbol{x}) = \sum_{k=1}^{p} \beta_k s_{ik}(\boldsymbol{x}) + \sum_{h=1}^{q} b_{ih} r_{ih}(\boldsymbol{x}), \tag{2.2.3}$$

where $\boldsymbol{b}_i \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma})$ is the vector of *random parameters* for actor i, $\boldsymbol{\Sigma}$ is a q dimensional positive definite matrix, q is the number of *random effects* $r_{ih}(\boldsymbol{x})$, which are statistics like the ones given above. The values $s_{ik}(\boldsymbol{x})$ are now called *fixed effects*. It is possible that some statistics are both fixed and random effects. In applications it is wise to avoid the use of random effects not present in the model also as fixed effects.

To make the discussion more concrete we will illustrate the approach using a simple example, where the out-degree is the only random effect. The parameter for the out-degree effect balances creation of new ties with termination of existing ties and plays a role similar to the intercept in logistic regression. The *stochastic actor oriented model with random out-degree* is defined by (2.2.3) with $q = 1$ and $s_{i1}(\boldsymbol{x}) = r_{i1}(\boldsymbol{x}) = x_{i+}$, yielding the evaluation function

$$u_i(\boldsymbol{x}) = (\beta_1 + b_i)s_{i1}(\boldsymbol{x}) + \sum_{k=2}^{p} \beta_k s_{ik}(\boldsymbol{x}), \tag{2.2.4}$$

where $b_i \sim \mathcal{N}(0, \sigma^2)$. In addition to the fixed parameter $\boldsymbol{\beta}$, the variance σ^2 has to be estimated, therefore the parameter space is $\mathbb{R}^p \times (0, \infty)$.

For every network $\boldsymbol{x} \in \mathscr{X}_i(\boldsymbol{x}_0)$, the distribution (2.2.1) can be written with $u_i(\boldsymbol{x})$ replaced by the *evaluation difference*

$$d_i(\boldsymbol{x}, \boldsymbol{x}_0) = u_i(\boldsymbol{x}) - u_i(\boldsymbol{x}_0) = \boldsymbol{\beta}^\top (\boldsymbol{s}_i(\boldsymbol{x}) - \boldsymbol{s}_i(\boldsymbol{x}_0)) + \boldsymbol{b}_i^\top (\boldsymbol{r}_i(\boldsymbol{x}) - \boldsymbol{r}_i(\boldsymbol{x}_0)), \quad (2.2.5)$$

or by the *change statistic*

$$\begin{aligned} c_i(j, \boldsymbol{x}_0) = (-1)^{I((ij) \in \boldsymbol{x}_0)} \Big(&+ \boldsymbol{\beta}^\top \big(\boldsymbol{s}_i(\boldsymbol{x}_0 \cup (ij)) - \boldsymbol{s}_i(\boldsymbol{x}_0 \setminus (ij)) \big) \\ &+ \boldsymbol{b}_i^\top \big(\boldsymbol{r}_i(\boldsymbol{x}_0 \cup (ij)) - \boldsymbol{r}_i(\boldsymbol{x}_0 \setminus (ij)) \big) \Big), \end{aligned} \quad (2.2.6)$$

where $\boldsymbol{x}_0 \cup (ij)$ and $\boldsymbol{x}_0 \setminus (ij)$ are the graphs where the tie (ij) is forced to be in the network, or to be excluded from it, respectively. Equivalently $\boldsymbol{x}_0$, $\boldsymbol{x}_0 \cup (ij)$ and $\boldsymbol{x}_0 \setminus (ij)$ are the binary matrices, where the last two are equal to $\boldsymbol{x}_0$ in all elements but the one in row i and column j, which is forced to be either 1 or 0, respectively. Note that if i decides to keep the configuration as it is, the evaluation difference is $d_i(\boldsymbol{x}_0, \boldsymbol{x}_0) = 0$. The change statistic in (2.2.6) is convenient because the tie (ij) that is evaluated appears directly in the formula. In particular, if $\boldsymbol{x}$ and $\boldsymbol{x}_0$ are the evaluated and current network, and the only tie that can be different between them is (ij), the conditional transition probability from $\boldsymbol{x}_0$ to $\boldsymbol{x}$ as a result of actor i can be written in one of the following three equivalent ways

$$\begin{aligned} P(\boldsymbol{X}(t) = \boldsymbol{x} \mid i, \boldsymbol{X}(t^-) = \boldsymbol{x}_0) &= e^{u_i(\boldsymbol{x})} I(\boldsymbol{x} \in \mathscr{A}_i(\boldsymbol{x}_0)) / \textstyle\sum_{\xi \in \mathscr{A}_i(\boldsymbol{x}_0)} e^{u_i(\xi)} \\ &= e^{d_i(\boldsymbol{x}, \boldsymbol{x}_0)} I(\boldsymbol{x} \in \mathscr{A}_i(\boldsymbol{x}_0)) / \textstyle\sum_{\xi \in \mathscr{A}_i(\boldsymbol{x}_0)} e^{d_i(\xi, \boldsymbol{x}_0)} \\ &= e^{c_i(j, \boldsymbol{x}_0)} / \textstyle\sum_{k \neq i} e^{c_i(k, \boldsymbol{x}_0)}. \end{aligned} \quad (2.2.7)$$

The first two expressions are used in the interpretation of the parameters, the last form is convenient for computations.

We now briefly describe the temporal component of the stochastic actor oriented model. The generative process is assumed to be *Markovian in continuous time*, with exponentially distributed time between transitions. If a transition happens at time $t_0 \in (t_1, t_2)$, the next transition can occur at a random time computed in the following way. For all actors j in $\{1, \ldots, N\}$, random times ΔT_j are sampled independently from an exponential distribution with parameter λ_j. Then the next transition will happen at time $t = t_0 + \min_j(\Delta T_j)$, if $t < t_2$ (otherwise the process end), and the focal actor that has the possibility to flip one of the outgoing ties ties is $i = \operatorname{argmin}_j(\Delta T_j)$. Often $\lambda_j = \lambda$ for all j, but λ_j can also

depend on *temporal effects* depending on covariates and/or network position.

The first idea of using random effects in the stochastic actor oriented model is in [55, Chapter 4], where the estimation method, frequentist or Bayesian, is based on the likelihood function. However, this method is not implemented in *RSiena*. The algorithm that will be explained here is completely different as it is based on the method of moments, which is the estimation algorithm most used in the SAOM. In [57], a related model is developed in which some parameters are shared by members of the same group while varying between groups, but its Bayesian estimation method is also based on the likelihood function. In exponential random graph models, random effects have been introduced in a Bayesian framework in [68], but we are not aware of the inclusion of random effects in temporal ERGMs.

2.3 Estimation method

In this section we describe the joint estimation of the parameters $\boldsymbol{\beta}, \boldsymbol{\Sigma} \in \mathbb{R}^p \times \mathbb{S}_q^+$, where $\mathbb{S}_q^+$ is the set of q dimensional positive definite matrices, by means of an extension of the *simulated method of moments* (MoM). For the standard stochastic actor oriented model, the MoM is discussed in [60] and [64]. For simplicity of notation, we will focus in this section on *conditional* estimation of the rate parameters [60, see Section 4.2]. However, in Section 2.5 the models will be estimated with unconditional estimation. Other estimators developed for the SAOM include a Bayesian estimator [33], a *simulated maximum likelihood* estimator [65], and a *simulated generalized method of moments estimator* [1, 2]. We base our estimation method on the MoM because likelihood-based algorithms require computation times which are more than 10 times longer.

2.3.1 Simulated method of moments with random effects

We first consider the SAOM without random effects. The p dimensional *moment equation* for $\boldsymbol{\beta}$ is

$$E\Big(\sum_{i=1}^{N} \boldsymbol{s}_i(\boldsymbol{X}(t_2)) \,\Big|\, \boldsymbol{X}(t_1) = \boldsymbol{x}(t_1), \boldsymbol{\beta}\Big) = \sum_{i=1}^{N} \boldsymbol{s}_i(\boldsymbol{x}(t_2)), \tag{2.3.1}$$

the right side of the equation is called *target*. The simulated method of moments procedure for solving this equation is based on simulating the generative Markov process from t_1 to t_2 for a given parameter $\boldsymbol{\beta}$, starting from the network at time

t_1. The difference between the simulated and observed networks at time t_2 is used to update the parameter. The network at time t_2 simulated for parameter $\boldsymbol{\beta}$ is denoted by $\hat{\boldsymbol{x}}(\boldsymbol{\beta})$, and its statistics are $\hat{\boldsymbol{s}}(\boldsymbol{\beta}) = \sum_i \boldsymbol{s}_i(\hat{\boldsymbol{x}}(\boldsymbol{\beta}))$. The observed network at time t_2 and the target (observed statistics) are denoted by $\boldsymbol{x}$ and $\boldsymbol{s} = \sum_i \boldsymbol{s}_i(\boldsymbol{x})$, respectively. The estimation algorithm is iterative, so a chain $(\boldsymbol{\beta}_t)_t$ of parameters is generated via the procedure

$$\boldsymbol{\beta}_{t+1} \leftarrow \boldsymbol{\beta}_t - \epsilon \boldsymbol{D}\big(\hat{\boldsymbol{s}}(\boldsymbol{\beta}_t) - \boldsymbol{s}\big), \tag{2.3.2}$$

where ϵ is a positive *learning rate*, and $\boldsymbol{D}$ is an invertible *preconditioning matrix*, computed at the beginning of the algorithm to compensate for the different magnitude and sensitivity of the components of $\hat{\boldsymbol{s}}(\boldsymbol{\beta})$ with respect to variations of $\boldsymbol{\beta}$. This update rule is known as the *Robbins-Monro algorithm* [50].

The parameter that solves (2.3.1) is approximated by the tail average of the chain of parameters $(\boldsymbol{\beta}_t)_t$. This method is based on Polyak and Juditsky [47]. A small learning rate in (2.3.2) ensures that the solution is approximated more accurately, assuming that the number of iterations is large enough so that the process has converged stochastically. However, a small learning rate increases the auto-correlations between the elements in the chain $(\boldsymbol{\beta}_t)_t$ so that more iterations are required to obtain a "good" approximation. In the algorithm used in the *RSiena* package, the problem is solved with multiple sub-phases, each with constant learning rate, and a "provisional" estimate is used as starting point for the next sub-phase, after that the number of iterations, and the learning rate, for next sub-phase are increased, and decreased, respectively. The last sub-phase produces the estimator $\hat{\boldsymbol{\beta}}$. After the estimator is computed, whether (2.3.1) is sufficiently well approximated is checked by simulating the process multiple times for the resulting value $\hat{\boldsymbol{\beta}}$. These simulations are also used to approximate the variance of the estimator. Details about the algorithm are in [63].

In the stochastic actor oriented model with a single random effect, the moment equation for $\boldsymbol{\beta}$ remains the same as (2.3.1), but a further equation is required for estimating the variance σ^2 of the random out-degree parameter. The moment equations are

$$\begin{aligned}
&E\Big(\sum_{i=1}^{N} \boldsymbol{s}_i(\boldsymbol{X}(t_2)) \,\Big|\, \boldsymbol{x}(t_1), \boldsymbol{\beta}, \sigma^2 \Big) = \sum_{i=1}^{N} \boldsymbol{s}_i(\boldsymbol{x}(t_2)), \\
&E\Big(\sum_{i=1}^{N} \big(s_{i1}(\boldsymbol{X}(t_2)) - \bar{s}_1(\boldsymbol{X}(t_2))\big)^2 \,\Big|\, \boldsymbol{x}(t_1), \boldsymbol{\beta}, \sigma^2 \Big) = \sum_{i=1}^{N} \big(s_{i1}(\boldsymbol{x}(t_2)) - \bar{s}_1(\boldsymbol{x}(t_2))\big)^2,
\end{aligned} \tag{2.3.3}$$

that are to be solved for $\boldsymbol{\beta}, \sigma^2$ simultaneously. The first component is the i-th out-degree $s_{i1}(\boldsymbol{x}) = x_{i+}$, and $\bar{s}_1(\boldsymbol{x})$ is their average.

In the simulations for the parameter updates, model (2.2.4) is used and random effects b_i are independently sampled from $\mathcal{N}(0, \sigma_t^2)$ for each parameter update step t. The algorithm for $\boldsymbol{\beta}$ is the same as (2.3.2), replacing $\hat{\boldsymbol{s}}(\boldsymbol{\beta}_t)$ with $\hat{\boldsymbol{s}}(\boldsymbol{\beta}_t, \boldsymbol{b}(\sigma_t^2)) = \boldsymbol{s}(\hat{\boldsymbol{x}}(\boldsymbol{\beta}_t, \boldsymbol{b}(\sigma_t^2))$. The update step for σ^2 is

$$\sigma_{t+1}^2 \leftarrow \max\Big(\sigma_t^2 - \zeta d\big(\hat{w}(\boldsymbol{\beta}_t, \boldsymbol{b}(\sigma_t^2)) - w\big), \sigma_{\min}^2\Big), \tag{2.3.4}$$

where ζ is a positive learning rate, d is a preconditioning value which is set and kept fixed in all sub-phases, the observed statistic for the variance is denoted by $w = \sum_{i=1}^{N} \big(s_{i1}(\boldsymbol{x}) - \bar{s}_1(\boldsymbol{x})\big)^2$, and the simulated statistic $\hat{w}(\boldsymbol{\beta}_t, \boldsymbol{b}(\sigma_t^2))$ is defined in a similar way by replacing the observed network $\boldsymbol{x}$ with the simulated network $\hat{\boldsymbol{x}}(\boldsymbol{\beta}_t, \boldsymbol{b}(\sigma_t^2))$. The map $\tilde{\sigma}_{t+1}^2 \mapsto \sigma_{t+1}^2 = \max(\tilde{\sigma}_{t+1}^2, \sigma_{\min}^2)$ is used to force the variance to be positive, where $\sigma_{\min}^2$ is a very small positive value, for example 10^{-4}.

The generalization to multiple random effects is straightforward. In the evaluation function (2.2.3) the number of random effects q is larger than one. Assume first that the variance is unconstrained, so that $\boldsymbol{\Sigma} \in \mathbb{S}_q^+$ has $q(q+1)/2$ parameters that are free to vary. The moment equations are

$$\begin{aligned}
&E\Big(\sum_{i=1}^{N} \boldsymbol{s}_i(\boldsymbol{X}(t_2)) \,\Big|\, \boldsymbol{x}(t_1), \boldsymbol{\beta}, \boldsymbol{\Sigma}\Big) = \sum_{i=1}^{N} \boldsymbol{s}_i(\boldsymbol{x}(t_2)), \\
&E\Big(\sum_{i=1}^{N} (\boldsymbol{r}_i(\boldsymbol{X}) - \bar{\boldsymbol{r}}(\boldsymbol{X}))(\boldsymbol{r}_i(\boldsymbol{X}) - \bar{\boldsymbol{r}}(\boldsymbol{X}))^\top \,\Big|\, \boldsymbol{x}(t_1), \boldsymbol{\beta}, \boldsymbol{\Sigma}\Big) = \sum_{i=1}^{N} (\boldsymbol{r}_i(\boldsymbol{x}) - \bar{\boldsymbol{r}}(\boldsymbol{x}))(\boldsymbol{r}_i(\boldsymbol{x}) - \bar{\boldsymbol{r}}(\boldsymbol{x}))^\top,
\end{aligned} \tag{2.3.5}$$

where in the lower equation $\boldsymbol{X}$ and $\boldsymbol{x}$ stand for $\boldsymbol{X}(t_2)$ and $\boldsymbol{x}(t_2)$, respectively. Note that the first equation is the same as the one in (2.3.3). Instead, the second equation is between two q dimensional positive definite matrices. The observed covariance matrix on the right hand side of the equation is denoted by $\boldsymbol{W}$. If the simulated process is $\hat{\boldsymbol{x}}(\boldsymbol{\beta}, \boldsymbol{b}(\boldsymbol{\Sigma}))$, the simulated statistic is denoted succinctly as $\hat{\boldsymbol{W}}$, or more extensively as $\hat{\boldsymbol{W}}(\boldsymbol{\beta}, \boldsymbol{b}(\boldsymbol{\Sigma}))$ when the emphasis is on the parameters that have simulated the process. The algorithm that generalizes (2.3.4) is

$$\boldsymbol{\Sigma}_{t+1} \leftarrow \mathrm{proj}(\boldsymbol{\Sigma}_t - \zeta d(\hat{\boldsymbol{W}}(\boldsymbol{\beta}_t, \boldsymbol{b}(\boldsymbol{\Sigma}_t)) - \boldsymbol{W}), \sigma_{\min}^2), \tag{2.3.6}$$

where the new value is projected (if necessary) to the space of positive definite

matrices. There are various ways to define the projector operator. A formal way is to use the method discussed in [25], where the matrix outside the parametric space is projected to the closest positive definite matrix (using the distance induced by a matrix norm). Effectively, the method changes negative or zero eigenvalues to $\sigma^2_{\min}$, before recomputing $\boldsymbol{\Sigma}_{t+1}$.

When estimating a model, it is assumed that the parameters are *identifiable*. For estimation by the MoM, this means that the moment equation (2.3.5) has a unique solution with probability 1. Proving identifiability or its absence for a given set of effects is challenging in the SAOM, except in a few known cases in which it is known that some combinations of effects cannot be estimated together by the MoM in any data set. For the SAOM with random effects it will be even more difficult to prove identifiability. Our experience is that, for models with sensible specifications, the algorithm to solve (2.3.5) indeed leads to a unique solution for most data sets. Sometimes, however, the combination of the model specification and the data set has not enough information for estimating the parameter. In this case some components of the process $(\boldsymbol{\beta}_t, \boldsymbol{\Sigma}_t)_t$ fail to converge, whereas if some of the "problematic" effects are removed, the process converges. Therefore, failures of identifiability are often identified during estimation.

2.3.2 Summary estimation algorithm and implementation of the simulating function

Here we give the structure of the full algorithm for estimating the parameters and to compute the quantities needed to evaluate the estimate, for the stochastic actor oriented model with random effects.

- Phase 1. Compute preconditioning matrix $\boldsymbol{D}$ for $\boldsymbol{\beta}$, and preconditioning value d for $\boldsymbol{\Sigma}$. Compute starting point $\bar{\boldsymbol{\beta}}_0$ and $\bar{\boldsymbol{\Sigma}}_0$.
- Phase 2. For sub-phase $s \in \{1, \ldots, S\}$:
  - Set starting point $\boldsymbol{\beta}_1 = \bar{\boldsymbol{\beta}}_{s-1}$, $\boldsymbol{\Sigma}_1 = \bar{\boldsymbol{\Sigma}}_{s-1}$.
  - For $t \in \{1, \ldots, T_s - 1\}$, repeat the following iteration:

$$\begin{aligned}
&\text{sample random parameters } \boldsymbol{b}_i \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma}_t) \text{ i.i.d. for } i \in \{1, \ldots, N\},\\
&\text{simulate process } \hat{\boldsymbol{x}} \sim \mathcal{S}(\boldsymbol{\beta}_t, \boldsymbol{b}),\ \text{compute simulated statistics } \hat{\boldsymbol{s}},\ \hat{\boldsymbol{S}},\\
&\text{update } \boldsymbol{\beta}_{t+1} \leftarrow \boldsymbol{\beta}_t - \epsilon \boldsymbol{D}(\hat{\boldsymbol{s}} - \boldsymbol{s}),\\
&\text{update } \boldsymbol{\Sigma}_{t+1} \leftarrow \text{proj}(\boldsymbol{\Sigma}_t - \zeta d(\hat{\boldsymbol{W}} - \boldsymbol{W}), \sigma^2_{\min}).
\end{aligned} \tag{2.3.7}$$

– Compute tail averages $\bar{\boldsymbol{\beta}}_s$ and $\bar{\boldsymbol{\Sigma}}_s$, decrease learning rates ϵ and ζ.

Estimates are $\hat{\boldsymbol{\beta}} = \bar{\boldsymbol{\beta}}_s$ and $\hat{\boldsymbol{\Sigma}} = \bar{\boldsymbol{\Sigma}}_s$.

- Phase 3. For $t \in \{1, ..., T\}$, repeat the following iteration:

$$\begin{aligned} &\boldsymbol{b}_i \sim \mathcal{N}(\mathbf{0}, \hat{\boldsymbol{\Sigma}}) \text{ i.i.d. for } i \in \{1, ..., N\}, \\ &\hat{\boldsymbol{x}}_t \sim \mathcal{S}(\hat{\boldsymbol{\beta}}, \boldsymbol{b}), \\ &\text{store simulated statistics } \hat{\boldsymbol{s}}_t \text{ and } \hat{\boldsymbol{W}}_t, \\ &\text{store contribution to the score function } \hat{\boldsymbol{l}}_t. \end{aligned} \tag{2.3.8}$$

Evaluate convergence of $\hat{\boldsymbol{\beta}}$ and $\hat{\boldsymbol{\Sigma}}$, compute their variability.

The algorithm is a straightforward generalization of the one that is currently used in the SAOM [60]. The contribution to the score function is needed for the model evaluation procedure explained in Section 2.4. Note that the rate parameters are not included in this algorithm, because they are either estimated separately (conditional estimation), or estimated "together" with the parameters of the evaluation function. In the latter case, the algorithm above is kept unchanged, but with $\boldsymbol{\beta}$ always replaced by $\boldsymbol{\theta} = (\boldsymbol{\lambda}, \boldsymbol{\beta})$ and $\boldsymbol{s}$ is replaced by $(\boldsymbol{s}_{\boldsymbol{\lambda}}, \boldsymbol{s})$, where $\boldsymbol{x} \mapsto \boldsymbol{s}_{\boldsymbol{\lambda}}(\boldsymbol{x})$ are the estimation statistics for the rate parameters.

The simulating function $\mathcal{S}(\boldsymbol{\beta})$ that is currently used in the SAOM, can be generalized to $\mathcal{S}(\boldsymbol{\beta}, \boldsymbol{b})$, in which the actors can also have individual parameters. Consider the set of *individual dummy covariates*

$$\boldsymbol{\delta}^{(i)} = (\delta_n^{(i)})_{1 \le n \le N} = I(n = i), \tag{2.3.9}$$

for all actors $i \in \{1, ..., N\}$. The dummy covariates of actor i are $\delta_1^{(i)}, ..., \delta_N^{(i)}$, which are all equal to 0, except for $\delta_i^{(i)} = 1$. The random out-degree for actor i, which is x_{i+}, can be then obtained with the *covariate-ego effect* $x_{i+}\delta_i^{(i)}$, which is the interaction between the out-degree x_{i+} and the dummy covariate $\delta_i^{(i)}$. The latter formula has the advantage that is not only defined for the focal actor i, but also for all the other actors, taking value $x_{i+}\delta_n^{(i)} = 0$ when $n \neq i$. This allows us to "treat" random parameters $\boldsymbol{b}$ as the fixed parameters $\boldsymbol{\beta}$ in the simulation, considering $\tilde{\boldsymbol{\theta}} = (\boldsymbol{\beta}, \boldsymbol{b})$, so that each actor contains the individual parameter of everyone, but only its own will be relevant in evaluating tie flips, because all other random parameters will be multiplied by an effect equal to 0. In particular,

the evaluation function for actor i is

$$\begin{aligned} u_i(\boldsymbol{x}) &= \beta_1 x_{i+}(\boldsymbol{x}) + \sum_{n=1}^{N} b_n (x_{i+}\delta_n^{(i)}) + \sum_{k=2}^{p} \beta_k s_{ik}(\boldsymbol{x}) \\ &= (\beta_1 + b_i) x_{i+} + \sum_{k=2}^{p} \beta_k s_{ik}(\boldsymbol{x}), \end{aligned} \tag{2.3.10}$$

which is equivalent to (2.2.4), as $s_{i1} = x_{i+}$.

Then the first two steps of equations (2.3.7) and (2.3.8), for the model random out-degree, are implemented in the following way:

$$b_i \sim \mathcal{N}(0, \sigma^2), \quad \tilde{\boldsymbol{\theta}} = (\boldsymbol{\beta}, b_1, ..., b_N), \quad \hat{\boldsymbol{x}} \sim \mathcal{S}(\tilde{\boldsymbol{\theta}}), \tag{2.3.11}$$

where the current implementation of the SAOM is used in the last step. The first p elements of the vectors of simulated and observed statistics are used to update $\boldsymbol{\beta}$, while the last N elements are used to compute the sufficient statistic for σ^2, therefore the optimization is carried on in the same way as in the algorithm described. The same procedure can be used when there are more than one random effects.

This implementation was used in the application that will be described in Section 2.5. However, an important disadvantage of this method is its computational inefficiency, as evaluating every proposed tie flip involves the "useless" computation of $q(N-1)$ multiplications with an operand equal to 0, where q is the number of random effects. This can be avoided, however, in a future *RSiena* implementation.

2.3.3 Restricted models for the variance parameter

The algorithms discussed so far are for the *unrestricted* model for the variance in which $\boldsymbol{\Sigma}$ belongs to a $q(q+1)/2$ dimensional parametric space. However restricted models are possible, for example the *unrestricted diagonal* variance $\boldsymbol{\Sigma} = \text{diag}(\sigma_h^2) \in \text{diag}(\mathbb{S}_q^+) \cong (0, \infty)^q$, with a q dimensional parametric space. In this case, the second moment equation in (2.3.5) is replaced by the set of q equations

$$E\left(\sum_{i=1}^{N} (r_{ih}(\boldsymbol{X}) - \bar{r}_h(\boldsymbol{X}))^2 \,\middle|\, \boldsymbol{x}(t_1), \boldsymbol{\beta}, \boldsymbol{\Sigma} \right) = \sum_{i=1}^{N} (r_{ih}(\boldsymbol{x}) - \bar{r}_h(\boldsymbol{x}))^2, \tag{2.3.12}$$

for $h \in \{1, ..., q\}$, the update step that generalizes (2.3.6) is

$$\boldsymbol{\Sigma}_{t+1} \leftarrow \text{proj}\big(\boldsymbol{\Sigma}_t - \zeta d\big(\text{diag}(\hat{\boldsymbol{W}}(\boldsymbol{\beta}_t, \boldsymbol{b}(\boldsymbol{\Sigma}_t))) - \text{diag}(\boldsymbol{W})\big), \sigma^2_{\min}\big), \tag{2.3.13}$$

and the projection is much simpler.

Other examples for restricted models are the following:

- *scalar*: $\boldsymbol{\Sigma} = \sigma^2 \boldsymbol{I}$, 1 degree of freedom;
- *two-parameters*: $\boldsymbol{\Sigma} = \sigma^2 \boldsymbol{I} + \varrho(\boldsymbol{J} - \boldsymbol{I})$, $\boldsymbol{J}$ is the matrix of ones, $\varrho \in (-\sigma^2/(q-1), \sigma^2)$, 2 degrees of freedom;
- *autoregressive-1*: $\boldsymbol{\Sigma} = (\sigma_{hh'})$ such that $\sigma_{hh'} = \phi^{|h-h'|}\sigma^2/(1-\phi^2)$, $\phi \in (-1, 1)$, 2 degrees of freedom;
- *block-diagonal*: $\boldsymbol{\Sigma} = \text{blockdiag}(\boldsymbol{\Sigma}_l)$, where each $\boldsymbol{\Sigma}_l$ can have its own model, for example one of the models described above, the total degrees of freedom are the sum of the ones in each block.

Note that in the AR(1) model, the order of the effects in $\boldsymbol{r}_i$ is important. An example in which we may want to leverage this fact is the case of many waves, and the random parameter for the effect at given wave m, depends on its values in the previous waves. Another example is a model with a random effect in an ordered set of networks, which is encoded as a set of graphs in which the tie (ij) can become one only if the same tie is already 1 in the "lower" graphs. In this case the random parameter in a given level, can depend on its values in the lower levels.

The models that are used as default are often *diagonal*, which in the examples above are unrestricted diagonal, scalar, or block-diagonal with scalar blocks. The reason is that they are advantageous computationally, but also statistically. The methods for model evaluation that will be described in Section 2.4 for the case with one random effects, are easily generalizable to deal with multiple random effects, when the variance is diagonal. Otherwise the generalization can be quite difficult, and possibly specific for a given model of the variance.

2.4 Model evaluation

Model evaluation procedures for the SAOM with random out-degree and conditional estimation of the rate parameter, are introduced in Sections 2.4.1 and 2.4.2. In Section 2.4.3, some generalizations to more complex models are briefly

discussed. The model evaluation procedure presented here is a generalization of the test proposed by Schweinberger [56], which requires the Monte-Carlo approximation of the derivative of the expected statistics with respect to the parameters, developed in [58].

2.4.1 Score-type test for overdispersion

The combination of the estimation functions for $\boldsymbol{\beta}$ and σ^2 is denoted by $\boldsymbol{g}(\boldsymbol{x}) = (\boldsymbol{s}(\boldsymbol{x})^\top, w(\boldsymbol{x}))^\top$. The null hypothesis of the absence of overdispersion in the out-degree is

$$H_0 : \sigma^2 = 0, \tag{2.4.1}$$

and can be tested against $H_1 : \sigma^2 > 0$ with a *score-type test* that will be soon described. In linear random effects models, the score test is also known to be a good test for testing variance components; see [6].

Expected value and variance of the statistics, which are

$$\boldsymbol{m}(\boldsymbol{\beta}) = E_{\boldsymbol{\beta}}(\boldsymbol{g}(\boldsymbol{X})), \quad \boldsymbol{V}(\boldsymbol{\beta}) = E_{\boldsymbol{\beta}}\big((\boldsymbol{g}(\boldsymbol{X}) - \boldsymbol{m}(\boldsymbol{\beta}))(\boldsymbol{g}(\boldsymbol{X}) - \boldsymbol{m}(\boldsymbol{\beta}))^\top\big), \tag{2.4.2}$$

are approximated with Monte-Carlo integration as

$$\bar{\boldsymbol{m}}(\boldsymbol{\beta}) = \frac{1}{\mathsf{T}}\sum_{\mathsf{t}=1}^{\mathsf{T}} \hat{\boldsymbol{g}}_{\mathsf{t}}, \quad \hat{\boldsymbol{V}}(\boldsymbol{\beta}) = \frac{1}{\mathsf{T}}\sum_{\mathsf{t}=1}^{\mathsf{T}} (\hat{\boldsymbol{g}}_{\mathsf{t}} - \bar{\boldsymbol{m}}(\boldsymbol{\beta}))(\hat{\boldsymbol{g}}_{\mathsf{t}} - \bar{\boldsymbol{m}}(\boldsymbol{\beta}))^\top, \tag{2.4.3}$$

respectively, where $\hat{\boldsymbol{g}}_{\mathsf{t}} \in \mathbb{R}^{p+1}$ are statistics simulated with parameters $\boldsymbol{\beta} \in \mathbb{R}^p$. Developing a score-type test statistic with good power, requires however the ability to approximate the $(p+1) \times p$ dimensional matrix

$$\begin{aligned} \boldsymbol{J}(\boldsymbol{\beta}) = \frac{\partial \boldsymbol{m}(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}^\top} &= \sum_{\boldsymbol{x} \in \mathscr{X}} \boldsymbol{g}(\boldsymbol{x}) \Big(\frac{\partial}{\partial \boldsymbol{\beta}^\top} f(\boldsymbol{x} \mid \boldsymbol{\beta}) \Big) \frac{f(\boldsymbol{x} \mid \boldsymbol{\beta})}{f(\boldsymbol{x} \mid \boldsymbol{\beta})} \\ &= E_{\boldsymbol{\beta}}\Big(\boldsymbol{g}(\boldsymbol{X}) \frac{\partial}{\partial \boldsymbol{\beta}^\top} \log f(\boldsymbol{X} \mid \boldsymbol{\beta}) \Big), \end{aligned} \tag{2.4.4}$$

where $f(\boldsymbol{x} \mid \boldsymbol{\beta})$ is the probability mass function of the network $\boldsymbol{x}$ according to parameter $\boldsymbol{\beta}$. This derivative is approximated with Monte-Carlo integration as

$$\hat{\boldsymbol{J}}(\boldsymbol{\beta}) = \frac{1}{\mathsf{T}}\sum_{\mathsf{t}=1}^{\mathsf{T}} \hat{\boldsymbol{g}}_{\mathsf{t}} \hat{\boldsymbol{l}}_{\mathsf{t}}, \text{ or more stably as } \hat{\boldsymbol{J}}(\boldsymbol{\beta}) = \frac{1}{\mathsf{T}}\sum_{\mathsf{t}=1}^{\mathsf{T}} (\hat{\boldsymbol{g}}_{\mathsf{t}} - \boldsymbol{g}) \hat{\boldsymbol{l}}_{\mathsf{t}}, \tag{2.4.5}$$

where $\hat{\boldsymbol{g}}_{\mathsf{t}} \in \mathbb{R}^{p+1}$ and $\hat{\boldsymbol{l}}_{\mathsf{t}} \in \mathbb{R}^{1 \times p}$ are, respectively, the simulated statistics and the

row-vector of contributions to the score function; and $\boldsymbol{g}$ is the vector of observed statistics. Note that $\hat{\boldsymbol{l}}_t$ is the same row-vector that is computed for the model without random effects, because under the null hypothesis there are no random effects. The only difference is that the variance of the out-degrees $\hat{w}_t$, which is the last component of $\hat{\boldsymbol{g}}_t$, has to be also computed, even though this statistic is not used to estimate the parameters.

The derivation of the test statistic is based on Neyman's *orthogonalization method* [43], which is now described. The procedure is a linear transformation that produces the test statistic from $\boldsymbol{g}(\boldsymbol{x})$, to obtain a test with high statistical power isolating the effect of σ^2 on the test statistic. The procedure is presented here under the simplifying assumption of normality of the test statistic, however the p-value $\hat{\alpha}$ derived below does not require normality, as the only important assumption is the finiteness of the first two moments $\boldsymbol{m}(\boldsymbol{\beta})$ and $\boldsymbol{V}(\boldsymbol{\beta})$ of the test statistic $\boldsymbol{g}(\boldsymbol{X})$.

Assume that

$$\boldsymbol{g}(\boldsymbol{X}) \sim \mathcal{N}(\boldsymbol{m}(\boldsymbol{\beta}), \boldsymbol{V}(\boldsymbol{\beta})), \tag{2.4.6}$$

where the vector $\boldsymbol{m}(\boldsymbol{\beta})$ and the matrices $\boldsymbol{J}(\boldsymbol{\beta})$ and $\boldsymbol{V}(\boldsymbol{\beta})$ are decomposed, in correspondence to $\boldsymbol{g}(\boldsymbol{x}) = (\boldsymbol{s}(\boldsymbol{x})^\top, w(\boldsymbol{x}))^\top$, as

$$\boldsymbol{m}(\boldsymbol{\beta}) = \begin{pmatrix} \boldsymbol{m}_1(\boldsymbol{\beta}) \\ m_2(\boldsymbol{\beta}) \end{pmatrix}, \ \boldsymbol{J}(\boldsymbol{\beta}) = \begin{pmatrix} \boldsymbol{J}_1(\boldsymbol{\beta}) \\ \boldsymbol{J}_2(\boldsymbol{\beta}) \end{pmatrix}, \ \boldsymbol{V}(\boldsymbol{\beta}) = \begin{pmatrix} \boldsymbol{V}_{11}(\boldsymbol{\beta}) & \boldsymbol{v}_{12}(\boldsymbol{\beta}) \\ \boldsymbol{v}_{12}^\top(\boldsymbol{\beta}) & v_{22}(\boldsymbol{\beta}) \end{pmatrix}. \tag{2.4.7}$$

The component of $w(\boldsymbol{X})$ that is uncorrelated with $\boldsymbol{s}(\boldsymbol{X})$ is

$$Y(\boldsymbol{\beta}) = w(\boldsymbol{X}) - \boldsymbol{\Gamma}(\boldsymbol{\beta})\boldsymbol{s}(\boldsymbol{X}) \sim \mathcal{N}(m_2(\boldsymbol{\beta}) - \boldsymbol{\Gamma}(\boldsymbol{\beta})\boldsymbol{m}_1(\boldsymbol{\beta}), \xi(\boldsymbol{\beta})), \tag{2.4.8}$$

where

$$\begin{aligned} &\boldsymbol{\Gamma}(\boldsymbol{\beta}) = \boldsymbol{J}_2(\boldsymbol{\beta})\boldsymbol{J}_1(\boldsymbol{\beta})^{-1}, \\ &\xi(\boldsymbol{\beta}) = v_{22}(\boldsymbol{\beta}) - 2\boldsymbol{\Gamma}(\boldsymbol{\beta})\boldsymbol{v}_{12}(\boldsymbol{\beta}) + \boldsymbol{\Gamma}(\boldsymbol{\beta})\boldsymbol{V}_{11}(\boldsymbol{\beta})\boldsymbol{\Gamma}(\boldsymbol{\beta})^\top. \end{aligned} \tag{2.4.9}$$

If $\boldsymbol{s}$ and w are the observed statistics, the estimated parameter $\hat{\boldsymbol{\beta}}$ solves the moment equation $\boldsymbol{m}_1(\boldsymbol{\beta}) = \boldsymbol{s}$, but under the null hypothesis also $m_2(\boldsymbol{\beta}) = w$, as $\sigma^2 = 0$. With equation (2.4.6) this shows that

$$\xi(\hat{\boldsymbol{\beta}})^{-1/2}(Y(\hat{\boldsymbol{\beta}}) - y(\hat{\boldsymbol{\beta}})) \sim \mathcal{N}(0,1) \tag{2.4.10}$$

under H_0, where $y(\hat{\boldsymbol{\beta}}) = w - \boldsymbol{\Gamma}(\hat{\boldsymbol{\beta}})\boldsymbol{s}$.

The *score-type test statistic* z derived from [56], and the associated *p-value* $\tilde{\alpha}$

are

$$z(\hat{\boldsymbol{\beta}}) = \hat{\xi}(\hat{\boldsymbol{\beta}})^{-1/2}(y(\hat{\boldsymbol{\beta}}) - \bar{y}(\hat{\boldsymbol{\beta}})), \quad \tilde{\alpha} = 1 - \Phi(z(\hat{\boldsymbol{\beta}})), \tag{2.4.11}$$

where $z \mapsto \Phi(z)$ is the cumulative distribution function of the standard normal, $\bar{y}(\hat{\boldsymbol{\beta}}) = \bar{m}_2 - \hat{\boldsymbol{\Gamma}}(\hat{\boldsymbol{\beta}})\bar{\boldsymbol{m}}_1$ is the average of the simulated statistics, $y(\hat{\boldsymbol{\beta}})$ and $\hat{\xi}(\hat{\boldsymbol{\beta}})$ are computed with $\boldsymbol{\Gamma}(\hat{\boldsymbol{\beta}})$ replaced by $\hat{\boldsymbol{\Gamma}}(\hat{\boldsymbol{\beta}})$, substituting $\hat{\boldsymbol{J}}(\hat{\boldsymbol{\beta}})$ for $\boldsymbol{J}(\hat{\boldsymbol{\beta}})$, and $\hat{\boldsymbol{V}}(\hat{\boldsymbol{\beta}})$ for $\boldsymbol{V}(\hat{\boldsymbol{\beta}})$ in equation (2.4.9). The computation of $\tilde{\alpha}$ relies on the normality in (2.4.10), derived from the asymptotic normality in (2.4.6). The generalization proposed here relaxes the normality assumption, and the empirical distribution in the simulations is used to compute the p-value. From the statistics $(\hat{\boldsymbol{g}}_t)_t$ simulated in Phase 3 with parameter $\hat{\boldsymbol{\beta}}$, the matrices $\hat{\boldsymbol{V}}$ and $\hat{\boldsymbol{J}}$ are computed, and then used to compute the simulated test statistics $(\hat{y}_t = \hat{w}_t - \hat{\boldsymbol{\Gamma}}\hat{\boldsymbol{s}}_t)_t$, and their mean $\bar{y} = \bar{m}_2 - \hat{\boldsymbol{\Gamma}}\bar{\boldsymbol{m}}_1$. The dependence of these quantities on the estimate $\hat{\boldsymbol{\beta}}$ is implicit, to simplify the notation. Then the *empirical p-value* for testing (2.4.1) is defined as

$$\hat{\alpha} = \frac{1}{T}\sum_{t=1}^{T} I(\hat{y}_t > y) = \frac{1}{T}\sum_{t=1}^{T} I(\hat{z}_t > z), \tag{2.4.12}$$

where $\hat{z}_t = \hat{\xi}^{-1/2}(\hat{y}_t - \bar{y})$, and $z = z(\hat{\boldsymbol{\beta}})$ is defined in (2.4.11).

2.4.2 Standard errors for SAOM with random out-degree

For computing the standard error of the estimates $(\hat{\boldsymbol{\beta}}, \hat{\sigma}^2)$, the quantities $\bar{\boldsymbol{m}}(\hat{\boldsymbol{\beta}}, \hat{\sigma}^2)$ and $\hat{\boldsymbol{V}}(\hat{\boldsymbol{\beta}}, \hat{\sigma}^2)$ are approximated in the same way as in equation (2.4.3), but the simulated statistics $\hat{\boldsymbol{g}}_t$ used to compute these quantities are simulated with parameters $\hat{\boldsymbol{\beta}}$ and $\hat{\sigma}^2$. However, the procedure for calculating $\hat{\boldsymbol{J}}(\hat{\boldsymbol{\beta}}, \hat{\sigma}^2)$ is slightly more challenging than the one in the preceding section. Using the *delta method* [70, Theorem 5.13], the asymptotic covariance of the method of moment estimator $(\hat{\boldsymbol{\beta}}, \hat{\sigma}^2)$ is approximated by

$$\hat{\boldsymbol{C}}(\hat{\boldsymbol{\beta}}, \hat{\sigma}^2) = \hat{\boldsymbol{J}}(\hat{\boldsymbol{\beta}}, \hat{\sigma}^2)^{-1}\hat{\boldsymbol{V}}(\hat{\boldsymbol{\beta}}, \hat{\sigma}^2)\hat{\boldsymbol{J}}(\hat{\boldsymbol{\beta}}, \hat{\sigma}^2)^{-\top}, \tag{2.4.13}$$

see [60] and [70, Theorem 9.6]. The standard errors are the square root of the diagonal elements of this matrix.

The $(p+1) \times (p+1)$ dimensional matrix $\boldsymbol{J}(\boldsymbol{\beta}, \sigma^2)$ can be written as

$$\begin{aligned} \boldsymbol{J}(\boldsymbol{\beta}, \sigma^2) &= \frac{\partial \boldsymbol{m}(\boldsymbol{\beta}, \sigma^2)}{\partial(\boldsymbol{\beta}^\top, \sigma^2)} = \sum_{\boldsymbol{x} \in \mathscr{X}} \boldsymbol{g}(\boldsymbol{x}) \Big(\frac{\partial}{\partial(\boldsymbol{\beta}^\top, \sigma^2)} \log f(\boldsymbol{x} \mid \boldsymbol{\beta}, \sigma^2) \Big) \\ &= \int_{\mathbb{R}^N} \sum_{\boldsymbol{x} \in \mathscr{X}} \boldsymbol{g}(\boldsymbol{x}) \Big(\frac{\partial}{\partial(\boldsymbol{\beta}^\top, \sigma^2)} \log f(\boldsymbol{x}, \boldsymbol{u} \mid \boldsymbol{\beta}, \sigma^2) \Big) d\boldsymbol{u}, \end{aligned} \tag{2.4.14}$$

where $\boldsymbol{u} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_N)$ is the random vector from which random parameters are computed as $\boldsymbol{b}(\sigma^2) = \sigma\boldsymbol{u}$. The joint log-density can be written as

$$\log f(\boldsymbol{x}, \boldsymbol{u} \mid \boldsymbol{\beta}, \sigma^2) = \log f(\boldsymbol{x} \mid \boldsymbol{\beta}, \boldsymbol{b}(\sigma^2)) + \log \phi_N(\boldsymbol{u}), \tag{2.4.15}$$

where the density of the multivariate standard Gaussian $\phi_N(\boldsymbol{u})$ does not depend on the fixed parameters, and the density of the data depends on σ^2 only through the random function $\sigma^2 \mapsto \boldsymbol{b}(\sigma^2) = \sigma\boldsymbol{u}$. Then, the row-vector $\boldsymbol{l} \in \mathbb{R}^{1\times(p+1)}$ of contributions to the score function is written using the chain rule in equation (A.1.3) from Appendix A.1, as

$$\begin{aligned}
&\frac{\partial}{\partial(\boldsymbol{\beta}^\top, \sigma^2)} \log f(\boldsymbol{x}, \boldsymbol{u} \mid \boldsymbol{\beta}, \sigma^2) \\
&= \left(\frac{\partial}{\partial(\boldsymbol{\beta}^\top, \boldsymbol{b}^\top)} \log f(\boldsymbol{x} \mid \boldsymbol{\beta}, \boldsymbol{b})\right)\left(\frac{\partial}{\partial(\boldsymbol{\beta}^\top, \sigma^2)}\left[\begin{pmatrix}\boldsymbol{\beta}\\ \sigma^2\end{pmatrix} \mapsto \begin{pmatrix}\boldsymbol{\beta}\\ \boldsymbol{b}\end{pmatrix}\right]\right) \\
&= \begin{pmatrix}\boldsymbol{l}_\beta & \boldsymbol{l}_b\end{pmatrix}\begin{pmatrix}\boldsymbol{I}_p & \boldsymbol{0}_p \\ \boldsymbol{O}_{N\times p} & \frac{1}{2\sigma^2}\boldsymbol{b}\end{pmatrix} = \left(\boldsymbol{l}_\beta, \frac{1}{2\sigma^2}\boldsymbol{l}_b\boldsymbol{b}\right) = (\boldsymbol{l}_\beta, l_{\sigma^2}).
\end{aligned} \tag{2.4.16}$$

As $[\sigma^2 \mapsto \boldsymbol{b}] \in [(0,\infty) \to \mathbb{R}^N]$ is a random function, its derivative with respect to σ^2, which is $\frac{1}{2\sigma^2}\boldsymbol{b} \in \mathbb{R}^N$, computed using equation (A.2.4) from Appendix A.2, is also a random variable, and so is l_{σ^2}.

The matrix $\hat{\boldsymbol{J}}(\boldsymbol{\beta}, \sigma^2)$ that approximates $\boldsymbol{J}(\boldsymbol{\beta}, \sigma^2)$ is computed using the simulated statistics $\hat{\boldsymbol{g}}_t = (\hat{\boldsymbol{s}}_t^\top, \hat{w}_t)$ from the model with parameters $\boldsymbol{\beta}$ and σ^2. In each simulation, the vector of random parameter $\boldsymbol{b}_t = \sigma\boldsymbol{u}_t$ is also simulated, and it is then used to compute the second component of the row-vector of contributions to the score function $\hat{\boldsymbol{l}}_t$. The generalization of the second matrix of equation (2.4.5) is then

$$\begin{aligned}
\hat{\boldsymbol{J}}(\boldsymbol{\beta}, \sigma^2) &= \frac{1}{T}\sum_{t=1}^{T}\begin{pmatrix}(\hat{\boldsymbol{s}}_t - \boldsymbol{s})\hat{\boldsymbol{l}}_{\beta t} & (\hat{\boldsymbol{s}}_t - \boldsymbol{s})\hat{l}_{\sigma^2 t} \\ (\hat{w}_t - w)\hat{\boldsymbol{l}}_{\beta t} & (\hat{w}_t - w)\hat{l}_{\sigma^2 t}\end{pmatrix} \\
&= \frac{1}{T}\sum_{t=1}^{T}\begin{pmatrix}(\hat{\boldsymbol{s}}_t - \boldsymbol{s})\hat{\boldsymbol{l}}_{\beta t} & \frac{1}{2\sigma^2}(\hat{\boldsymbol{s}}_t - \boldsymbol{s})\hat{\boldsymbol{l}}_{bt}\boldsymbol{b}_t \\ (\hat{w}_t - w)\hat{\boldsymbol{l}}_{\beta t} & \frac{1}{2\sigma^2}(\hat{w}_t - w)\hat{\boldsymbol{l}}_{bt}\boldsymbol{b}_t\end{pmatrix}.
\end{aligned} \tag{2.4.17}$$

This matrix approximates $\boldsymbol{J}(\boldsymbol{\beta}, \sigma^2)$ in equation (2.4.14), in which the integral with respect to $\boldsymbol{u} \in \mathbb{R}^n$, and the sum with respect to $\boldsymbol{x} \in \mathscr{X}$, are approximated "together" by simulating $\boldsymbol{u}_t$ and $\hat{\boldsymbol{x}}_t \mid \boldsymbol{u}_t$ in each iteration.

2.4.3 Generalizations

Different random effect. As long as there is only one random effect in the model, all equations in Sections 2.4.1 and 2.4.2 are the same even when the random effect is not the out-degree, using $w(\boldsymbol{x}) = \frac{1}{N}\sum_i (r_i(\boldsymbol{x}) - \bar{r}(\boldsymbol{x}))^2$, the sample variance of the subgraph counts of the chosen random effect $\boldsymbol{r}$.

Unconditional estimation. The generalization of the model evaluation procedure to the more common case in which the rate is estimated unconditionally, is also straightforward. In this case $\boldsymbol{\beta}$, p and $\boldsymbol{s}$, should be replaced in all equations above by $\boldsymbol{\theta} = (\boldsymbol{\lambda}^\top, \boldsymbol{\beta}^\top)^\top$, $p^* = p_{\boldsymbol{\lambda}} + p$, and $\boldsymbol{s}^* = (\boldsymbol{s}_{\boldsymbol{\lambda}}^\top, \boldsymbol{s}^\top)^\top$, respectively, where $\boldsymbol{\lambda}$ is the $p_{\boldsymbol{\lambda}}$ dimensional vector of rate parameters, and $\boldsymbol{s}_{\boldsymbol{\lambda}}(\boldsymbol{x})$ is the statistic used to estimate $\boldsymbol{\lambda}$.

Reparametrization. Instead of computing the standard errors for $\boldsymbol{\beta}$ and σ^2, we might be interested on computing them for a different parametrization, e.g., for $\boldsymbol{\beta}$ and σ, where the second parameter is a standard deviation. The derivative of $\sigma \mapsto \boldsymbol{b} = \sigma\boldsymbol{u}$ with respect to σ is equal to $\boldsymbol{u} = \boldsymbol{b}/\sigma$. Therefore, $\hat{\boldsymbol{J}}(\boldsymbol{\beta}, \sigma)$ can be computed by replacing $(\boldsymbol{b}/2\sigma^2)$ with $\boldsymbol{u}$, and then σ^2 with σ, in all equations (2.4.14 - 2.4.17). Note that the matrix $\hat{\boldsymbol{J}}(\boldsymbol{\beta}, \sigma)$ obtained by this "replacement rule" is

$$\begin{aligned}\hat{\boldsymbol{J}}(\boldsymbol{\beta}, \sigma) &= \hat{\boldsymbol{J}}(\boldsymbol{\beta}, \sigma^2)\boldsymbol{H} = \hat{\boldsymbol{J}}(\boldsymbol{\beta}, \sigma^2)\begin{pmatrix} \boldsymbol{I}_p & \boldsymbol{0}_p \\ \boldsymbol{0}_p^\top & 2\sigma \end{pmatrix} \\ &= \hat{\boldsymbol{J}}(\boldsymbol{\beta}, \sigma^2)\left(\frac{\partial}{\partial(\boldsymbol{\beta}^\top, \sigma)} \left[\begin{pmatrix} \boldsymbol{\beta} \\ \sigma \end{pmatrix} \mapsto \begin{pmatrix} \boldsymbol{\beta} \\ \sigma^2 \end{pmatrix} \right] \right),\end{aligned} \tag{2.4.18}$$

as $2\sigma \cdot (\boldsymbol{b}/2\sigma^2) = \boldsymbol{u}$. Moreover $\hat{\boldsymbol{C}}(\hat{\boldsymbol{\beta}}, \hat{\sigma})$ computed using equation (2.4.13) with $\hat{\boldsymbol{J}}(\hat{\boldsymbol{\beta}}, \hat{\sigma})$, could have been derived directly with the delta method as $\hat{\boldsymbol{C}}(\hat{\boldsymbol{\beta}}, \hat{\sigma}) = \boldsymbol{H}^{-1}\hat{\boldsymbol{C}}(\hat{\boldsymbol{\beta}}, \hat{\sigma}^2)\boldsymbol{H}^{-\top}$, because $\boldsymbol{H}^{-1}$ is the derivative of $(\boldsymbol{\beta}^\top, \sigma^2)^\top \mapsto (\boldsymbol{\beta}^\top, \sigma)^\top$ with respect to $(\boldsymbol{\beta}^\top, \sigma^2)$. Therefore, the model evaluation procedure developed in the previous sections is coherent with the statistical theory of reparametrizations, which is important when the interest is on a particular interpretation of the estimated parameter.

Multiple random effects. If there are more random effects, and they are independent, for example when $\boldsymbol{\Sigma} = \text{diag}(\sigma_1^2, \sigma_2^2)$, the model evaluation procedure can be easily generalized for testing $H_0 : \sigma_h^2 = 0$, or for computing the

standard error of $(\boldsymbol{\beta}, \sigma_1^2, \sigma_2^2)$. In the latter case, the matrix $\hat{\boldsymbol{V}}(\boldsymbol{\beta}, \boldsymbol{\Sigma})$ is computed as in equation (2.4.3), but with the simulated statistic $\hat{\boldsymbol{g}}_t$ computed with $\boldsymbol{g}(\boldsymbol{x}) = (\boldsymbol{s}(\boldsymbol{x})^\top, \text{diag}(\boldsymbol{S}(\boldsymbol{x}))^\top)^\top = (\boldsymbol{s}(\boldsymbol{x})^\top, w_1(\boldsymbol{x}), w_2(\boldsymbol{x}))^\top$. The matrix $\hat{\boldsymbol{J}}(\boldsymbol{\beta}, \boldsymbol{\Sigma})$ can also be computed with a similar procedure, where $\boldsymbol{g}(\boldsymbol{x}) \in \mathbb{R}^{p+2}$ is simulated, and the contribution to the score function is

$$\boldsymbol{l} = (\boldsymbol{l}_\beta, \boldsymbol{l}_\Sigma) = (\boldsymbol{l}_\beta, l_{\sigma_1^2}, l_{\sigma_2^2}) = \left(\boldsymbol{l}_\beta, \frac{1}{2\sigma_1^2}\boldsymbol{l}_{\boldsymbol{b}_1}\boldsymbol{b}_1, \frac{1}{2\sigma_2^2}\boldsymbol{l}_{\boldsymbol{b}_2}\boldsymbol{b}_2\right) \in \mathbb{R}^{1\times(p+2)}, \tag{2.4.19}$$

where $\boldsymbol{b}_1$ and $\boldsymbol{b}_2$ are the vectors of random parameters for the two effects, so that $\hat{\boldsymbol{J}}(\boldsymbol{\beta}, \boldsymbol{\Sigma})$ can be computed with a formula similar to (2.4.17). For testing the null hypothesis of no overdispersion in one random effect, for example $H_0 : \sigma_2^2 = 0$, the same procedure of Section 2.4.1 is used, but with $\boldsymbol{l} = (\boldsymbol{l}_\beta, l_{\sigma_1^2}) \in \mathbb{R}^{1\times(p+1)}$ and $\boldsymbol{g}(\boldsymbol{x}) = (\boldsymbol{s}(\boldsymbol{x})^\top, w_1(\boldsymbol{x}), w_2(\boldsymbol{x}))^\top \in \mathbb{R}^{p+2}$ used to approximate $\boldsymbol{J}(\boldsymbol{\beta}, \sigma_1^2) \in \mathbb{R}^{(p+2)\times(p+1)}$, which can be used to transform the sufficient statistics into a test statistic with the orthogonalization procedure. If the random effects are still independent, but some (variance) parameters are shared between different effects, it is still possible to extend easily the estimation and the model evaluation procedure to this case. The equations for $\boldsymbol{J}$ and $\boldsymbol{V}$ will depend on which parameters are shared, so we do not discuss them in detail. The generalization of the model evaluation procedure is however much more difficult in the case in which the random effects are not independent. The reason is that when $\boldsymbol{\Sigma}$ is diagonal, the maps $\boldsymbol{\Sigma} \mapsto \boldsymbol{\Sigma}^{1/2}$ and $\boldsymbol{\Sigma}^{1/2} \mapsto \boldsymbol{b}$ needed to correlate the random parameters, can be written simply as vector-to-vector functions $(0, \infty)^q \to (0, \infty)^q$ and $(0, \infty)^q \to \mathbb{R}^{qN}$, respectively. By contrast, if some random effects are correlated, $\boldsymbol{\Sigma} \mapsto \boldsymbol{\Sigma}^{1/2}$ and $\boldsymbol{\Sigma}^{1/2} \mapsto \boldsymbol{b}$ must be treated as a matrix-to-matrix, and a matrix-to-vector functions, respectively, complicating significantly the model evaluation procedure.

Composite hypotheses. Finally, composite hypotheses on $\boldsymbol{\beta}$, $\boldsymbol{\Sigma}$, or a combination of the two are discussed. Consider the hypothesis $H_0 : \boldsymbol{\beta}_2 = \boldsymbol{0}_{p_2}$ for $\boldsymbol{\beta} = (\boldsymbol{\beta}_1^\top, \boldsymbol{\beta}_2^\top)^\top$, when there are random effects in the model, e.g., in the SAOM with random out-degree. The simulated quantities under the null hypothesis are used to approximate the positive definite matrix $\boldsymbol{V}(\boldsymbol{\beta}_1, \sigma^2) \in \mathbb{S}^+_{p+1}$, and the Jacobian $\boldsymbol{J}(\boldsymbol{\beta}_1, \sigma^2) \in \mathbb{R}^{(p+1)\times(p-p_2+1)}$. These quantities are used to compute

$$\boldsymbol{\Gamma} = \boldsymbol{J}_2\boldsymbol{J}_1^{-1} \in \mathbb{R}^{p_2\times(p-p_2+1)}, \quad \boldsymbol{\Xi} = \boldsymbol{V}_{22} - (\boldsymbol{\Gamma}\boldsymbol{V}_{12} + \boldsymbol{V}_{12}^\top\boldsymbol{\Gamma}^\top) + \boldsymbol{\Gamma}\boldsymbol{V}_{11}\boldsymbol{\Gamma}^\top \in \mathbb{S}^+_{p_2}, \tag{2.4.20}$$

where in order to keep the notation simple, the dependence of these quantities on $(\boldsymbol{\beta}_1, \sigma^2)$, the parameter under H_0, is implicit. Therefore $\boldsymbol{Z} = \boldsymbol{\Xi}^{-1/2}(\boldsymbol{Y} - \boldsymbol{y}) \sim \mathcal{N}(\mathbf{0}_{p_2}, \boldsymbol{I}_{p_2})$ under the null hypothesis, for $\boldsymbol{Y} = \boldsymbol{g}_2(\boldsymbol{X}) - \boldsymbol{\Gamma}\boldsymbol{g}_1(\boldsymbol{X})$, where $\boldsymbol{x} \mapsto \boldsymbol{g}_2(\boldsymbol{x})$ are the simulated statistics associated with the parameters that are 0 under the null, and $\boldsymbol{y}$ is computed orthogonalizing the observed statistics. The test statistic used in [56], with its associated p-value is

$$z^2 = (\boldsymbol{y} - \bar{\boldsymbol{y}})^\top \hat{\boldsymbol{\Xi}}^{-1}(\boldsymbol{y} - \bar{\boldsymbol{y}}), \quad \tilde{\alpha} = 1 - \chi^2_{p_2}(z^2), \tag{2.4.21}$$

where $\chi^2_{p_2}(z^2)$ is the cumulative distribution function of a Chi-squared distribution with p_2 degrees of freedom, evaluated at z^2, and $\bar{\boldsymbol{y}}$ is the average of the simulated test statistics $\hat{\boldsymbol{y}}_t$. The non-parametric version of $\tilde{\alpha}$ that is based on the empirical distribution of the simulated test statistics, rather than assuming that their distribution is normal, is

$$\hat{\alpha} = \frac{1}{T}\sum_{t=1}^{T} I\big((\hat{\boldsymbol{y}}_t - \bar{\boldsymbol{y}})^\top \hat{\boldsymbol{\Xi}}^{-1}(\hat{\boldsymbol{y}}_t - \bar{\boldsymbol{y}}) > z^2\big), \tag{2.4.22}$$

where z^2 is defined in (2.4.21). The same theory can be used for hypotheses as $H_0 : \sigma_h^2 = 0$, for $h \in \{q_1+1, \ldots, q\}$, in which the last $q_2 = q - q_1$ random effects are not significant, assuming $\boldsymbol{\Sigma}$ is diagonal, or for hypotheses $H_0 : \beta_k = 0, \sigma_h^2 = 0$, for $k \in \{p_1+1, \ldots, p\}$ and h as before. Therefore the theory developed in this section, generalizing [56], can be used to test most hypotheses in which a group of effects, fixed or random, is not included in the model under the null hypothesis, as long as the variance $\boldsymbol{\Sigma}$ of the random effects is diagonal.

2.5 Analysing social interactions in a tailor shop

In this section a dataset is analyzed, mainly with the purpose of illustrating the proposed model, its estimation, and its evaluation procedure. The interpretation of some estimated parameters is described in detail, to show how the inclusion of the random out-degree effect can lead to different estimates of other parameters of interest.

2.5.1 Kapferer's tailor shop dataset

Between June 1965 and January 1966 Kapferer [31] carried out a study of a tailor shop in Kabwe, Zambia, in a period when the workers were negotiating for

higher wages. There were 39 employees; networks between them were observed at two times, seven months apart. The dataset used is available in the *Ucinet* data repository http://vlado.fmf.uni-lj.si/pub/networks/data/Ucinet/UciData.htm. We study the "sociational" (friendship, socioemotional) network.

An important covariate is job status (low or high). We shall not consider the specific job of the individuals in our analysis. Figure 2.1 shows the two network snapshots with the out-degrees of the individuals. Each individual is in the same position in the two plots. Status is indicated by color (light blue for high, orange for low). Out-degrees in the first graph range from 0 to 12 with an average of 2.8, whereas in the second graph they range from 0 to 21 with an average of 3.8.

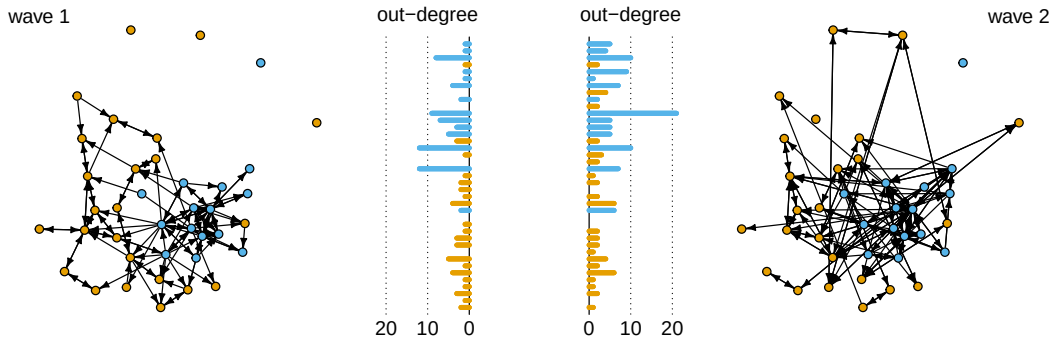

Figure 2.1: Kapferer data set with out-degrees of the nodes.

The dataset was described first in Kapferer [31], and has been the subject of a variety of analyses. The stability of groups in the network is modelled in [15]. The dataset is analysed in [44] using a *stochastic block model*. In [51] the network at time 1 is used to check how the estimate of the parameters in an *exponential random graph model* is influenced by various patterns of missing relations. In [14] the data set used to illustrate their implementation of Bayesian ERGMs. We cannot compare the outcomes of our models with those of earlier analyses, because the objectives and models that we found in the literature are very dissimilar from our own.

2.5.2 Definition estimated models

The aim of the analysis is to discover what has been driving changes of ties in the network. We are particularly interested in discovering to what extent triadic effects, such as transitivity and out-degree activity, as well as work status

(low/high) might be confounded with heterogeneity in out-degree. In order to answer these questions, we consider the following fixed effects,

- *basic rate*: rate of tie flips between the two waves;
- *out-degree* (*density*): number of ties starting from the focal actor i, $s_{i1}(\boldsymbol{x}) = \sum_j x_{ij}$;
- *reciprocity*: number of reciprocated ties, $s_{i2}(\boldsymbol{x}) = \sum_j x_{ij}x_{ji}$;
- *transitive triplets*: ordered pairs of actors (jh) to both of whom i is tied, while also j is tied to h, $s_{i3}(\boldsymbol{x}) = \sum_{j,h} x_{ij}x_{ih}x_{hj}$;
- *out-degree activity*: squared out-degree of the actor, $s_{i4}(\boldsymbol{x}) = (\sum_j x_{ij})^2$;
- *status alter*: sum of the covariate over all actors to whom i has a tie, calculated as $s_{i5}(\boldsymbol{x}) = \sum_j x_{ij}v_j$, where v_j is the binary covariate status, of actor j;
- *status ego*: out-degree of actor i weighted by its status, $s_{i6}(\boldsymbol{x}) = v_i \sum_j x_{ij}$;
- *status similarity*: sum of centered similarity scores between i and the other actors j to whom he is tied, calculated as

$$s_{i7}(\boldsymbol{x}) = \sum_j x_{ij}\big(\mathrm{sim}_{ij}(\boldsymbol{v}) - \overline{\mathrm{sim}}(\boldsymbol{v})\big),$$

  where, for a binary covariate, $\mathrm{sim}_{ij}(\boldsymbol{v}) = 1 - |v_i - v_j|$, and $\overline{\mathrm{sim}}(\boldsymbol{v})$ is their mean.

We consider four models with different selections of fixed effects. They are called *standard*, *no-transitivity*, *no-status* and *full-transitivity*, and summarized in Table 1. The four models are estimated with and without random out-degree, to compare the estimates.

2.5.3 Estimation and interpretation of the results

The models without random out-degree are estimated first using the function `siena07` of the package *RSiena*. Then the estimates are used as starting points for the models with random out-degree, using $\sigma^2_{min} = 10^{-4}$ as starting value for σ^2. Parameter update steps (2.3.2) and (2.3.4) were used, with in (2.3.2) the replacement of $\boldsymbol{\beta}$ by $\boldsymbol{\theta} = (\lambda, \boldsymbol{\beta})$, because the rate parameter is estimated unconditionally. To have a direct comparison between the algorithms, in all models

fixed effects	standard	no-transitivity	no-status	full-transitivity
basic rate	X	X	X	X
out-degree (density)	X	X	X	X
reciprocity	X	X	X	X
transitive triplets	X		X	X
out-degree activity				X
status alter	X	X		X
status ego	X	X		X
status similarity	X	X		X

Table 2.1: Fixed effects in the four models.

the same learning rate ϵ is used, and in each sub-phase the learning rate is decreased in the same way. This employs the default values of `siena07`, i.e., in the first sub-phase $\epsilon = 0.2$ and after the end of a sub-phase the learning rate is decreased by the factor 2. For the models with random out-degree, the learning rate for σ^2 is the same, i.e., $\zeta = \epsilon$, and updated in the same way at the end of the sub-phase. For each of the four models without random effects, the preconditioning matrix $\boldsymbol{D}$ used for updating $\boldsymbol{\theta}$ is computed with the default procedure of `siena07`, and used also for the equivalent model with random effect. The preconditioning value d used in the updating step for σ^2 is set so that the magnitude of the updates of σ^2 is comparable with the updates of $\boldsymbol{\theta}$.

Phase 2 consists of 2,100 iterations in total, divided in four sub-phases of 100, 100, 200 and 1,700 iterations. The number of iterations used to compute the tail average used as starting point of the next sub-phase is 20, 40 and 80, for the first three sub-phases, respectively, and the last 1,500 iterations are averaged to compute the estimated parameter. In Phase 3, 25,000 iterations are used in all models.

The full-transitivity model with random out-degree did not converge because of near collinearity, although the average simulated statistics for all effects were very near the observed ones. The near collinearity is caused by the presence among the estimation statistics of the sum of out-degrees (for the fixed out-degree parameter), the sum of squared outdegrees (out-degree activity effect) and the variance of the out-degrees (random out-degree parameter). The latter is a deterministic, although not linear, function of the other two. This implies that there are different combinations of $\beta_{\text{out-degree activity}}$, $\beta_{\text{status ego}}$, and σ^2 that produce similar simulated statistics, so these three parameters cannot be estimated together, making the full-transitivity model with random out-degree "almost non-identifiable".

In the left plot of Figure 2.2, the chain of the variance parameter for the standard model with random out-degree is plotted. The dashed vertical lines are the iterations where the learning rate is decreased, and the filled vertical line denotes the end of the burn-in period, so all values on the right of this line are averaged to compute the estimated variance, which is the black dot on the right of the plot. The right plot shows the distribution of the variances of the out-degrees of the networks for wave 2 as simulated in Phase 3, for all estimated models. Filled and dashed lines are for the models with and without random out-degree, respectively. The colours denote the models, they are black, red, green and blue for the standard, no-transitivity, no-status and full-transitivity models, respectively. The vertical grey line is the variance of the out-degrees in the observed network in wave 2. The distributions for the models with random out-degree are all similar, and their average is the sample variance of the observed network, whereas for the models without random out-degree, only the full-transitivity model explains the observed overdispersion.

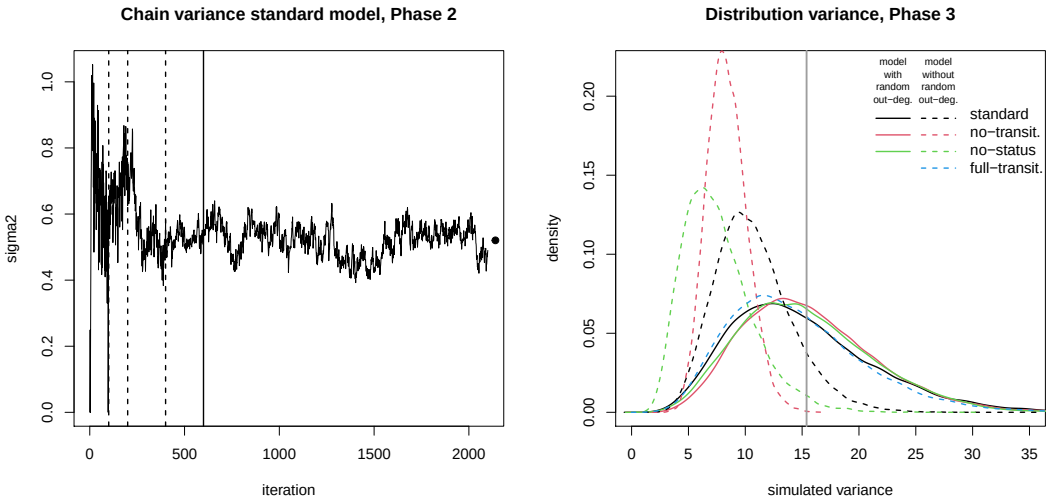

Figure 2.2: Left: chain of the variance parameter for the standard model with random out-degree. Right: distribution of the variance of the out-degrees computed from the simulations in Phase 3.

Estimated parameters and standard errors are reported in Table 2. The models without random out-degree have, in all cases, an estimated rate higher than the corresponding models with random out-degree. This means that for the models with random out-degree, fewer steps are required, on average, to simulate a network with statistics that match the data.

Comparison of the parameter estimates for the fixed effects between the models with and without random out-degree shows that the absolute values for all parameters, except transitive triplets, are higher in the models with random out-degree. To explain this, note that it was argued for logistic regression models by Mood [41, p. 68-69] that, since the error term is included implicitly in the non-linear link function, extending a given model by extra uncorrelated explanations will lead to approximately scaling the other parameters and their standard errors with a factor greater than 1. The same holds for multinomial logistic models such as (2.2.1). Therefore the main interpretation of the greater absolute values of the parameters and standard errors in the models with random out-degree is the increase in explanatory power of this model (reflected also by the lower estimated rate parameters). The exception is the transitive triplets parameter, implying that the random out-degree effect partially explains the triangles in the model. Special mention should be made here of the status ego effect, because ego effects of covariates are the deterministic, i.e., explained, analogues of the random out-degree effect. The increase in the standard error of status ego is more than proportional with the other increases, but the effect stays significant.

Table 2 also gives the estimates $\hat{\sigma}^2$ and its positive root $\hat{\sigma}$ of the variance and the standard deviation of the random out-degree effect, respectively, with their standard errors. The estimated variance parameters are smaller for the models containing more effects, which suggests that omitting effects will lead to higher estimated residual heterogeneity. The standard errors of $\hat{\sigma}$ are computed with the delta method from those of $\hat{\sigma}^2$. Considering the ratios of these parameters to their standard errors highlights that these ratios should not be used in a conventional t-test for testing the null hypothesis that $\sigma^2 = 0$. The better strategy is to use the score-type test, as demonstrated in the next section.

2.5.4 Model comparison

In this section, we compare different models using the theory developed in Section 2.4. Figure 2.3 describes the nesting relations between the eight models, where arrows point from models to models that are their extensions. If two models are connected by a path of arrows, the null hypothesis that the smaller model is the generative model can be tested. We formally compared five models, indicated by red arrows with their empirical p-values. These are computed using 25,000 simulations from the estimated model under the null hypothesis.

1. Random out-degree in the standard model. We test the null hypothesis that there is no random out-degree effect, $H_0 : \sigma^2 = 0$, versus the alternative

Without random out-degree	standard	no-transitivity	no-status	full-transitivity
effects / parameters	est. (s.e.)	est. (s.e.)	est. (s.e.)	est. (s.e.)
basic rate	21.39 (4.81)	19.05 (3.78)	16.68 (2.94)	22.94 (5.18)
out-degree (density)	-2.66 (0.24)	-2.52 (0.23)	-2.08 (0.11)	-2.97 (0.25)
reciprocity	3.26 (0.41)	3.37 (0.41)	2.23 (0.21)	3.34 (0.40)
transitive triplets	0.19 (0.05)	/	0.21 (0.04)	0.11 (0.05)
out-degree activity	/	/	/	0.04 (0.01)
status alter	-1.15 (0.25)	-0.98 (0.24)	/	-1.06 (0.24)
status ego	1.45 (0.28)	1.77 (0.29)	/	1.24 (0.27)
status similarity	0.30 (0.13)	0.42 (0.13)	/	0.40 (0.13)

With random out-degree	standard	no-transitivity	no-status	full-transitivity
effects / parameters	est. (s.e.)	est. (s.e.)	est. (s.e.)	est. (s.e.)
basic rate	17.30 (4.47)	15.78 (3.96)	10.86 (2.51)	
out-degree (density)	-2.98 (0.37)	-3.03 (0.40)	-2.76 (0.52)	
reciprocity	3.64 (0.50)	3.87 (0.53)	3.16 (0.60)	
transitive triplets	0.13 (0.07)	/	0.16 (0.08)	
out-degree activity	/	/	/	
status alter	-1.17 (0.28)	-1.06 (0.29)	/	
status ego	1.83 (0.49)	2.16 (0.54)	/	
status similarity	0.48 (0.18)	0.61 (0.20)	/	
variance random out-degree	0.52 (0.46)	0.92 (0.71)	1.92 (1.89)	
std.dev. random out-degree	0.72 (0.32)	0.96 (0.37)	1.39 (0.68)	

Table 2.2: Estimated parameters with standard errors in brackets for the models with and without random out-degree. Full-transitivity model with random out-degree is missing, as the algorithm has not converged.

$H_1 : \sigma^2 > 0$. The empirical p-value (2.4.1) is $\hat{\alpha} = 2 \cdot 10^{-4}$. If the asymptotic normal distribution of the test statistic is used, the p-value is $\tilde{\alpha} < 4 \cdot 10^{-5}$. Note that in the right-hand plot of Figure 2.2, the observed value of the out-degree variance aligns moderately well with the distribution of the simulated variances of the out-degrees obtained from Phase 3 (the proportion of higher simulated values is 0.09). However, this correspondence does not take into account that parameters are estimated. This shows the higher power associated with the orthogonalization.

2. Random out-degree in the full-transitivity model. Similarly, we test the null hypothesis of no random out-degree in the full-transitivity model. The p-values computed with the empirical and asymptotic distribution are both $\hat{\alpha} = 0.31$, so there is no evidence against the null hypothesis in this model. This is expected because when activity and overdispersion in the out-degree are both included, the model becomes almost no-identifiable.

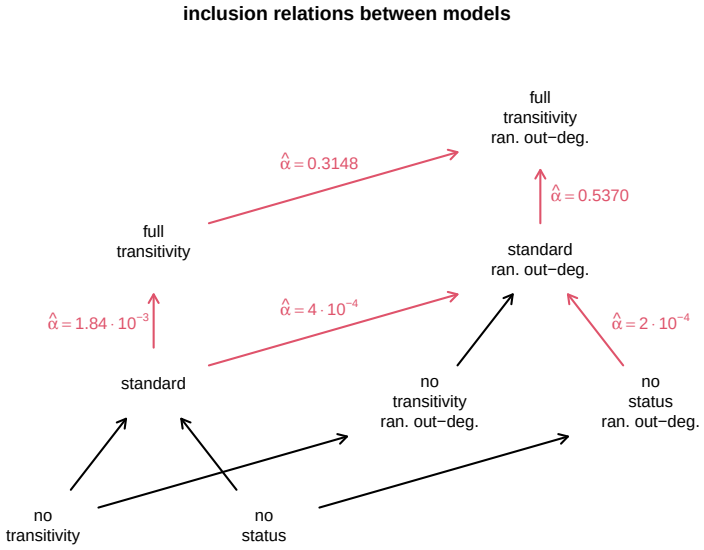

Figure 2.3: Arrow Model 1 $\longrightarrow$ Model 2 *if the parameter space of* Model 1 *is a subset of the one of* Model 2. *Red arrows were formally tested, with null hypothesis* H_0 : Model 1 is true model*;* $\hat{\alpha}$ *are the empirical* p*-values of the tests.*

3. Relevance of out-degree activity effect for the model without random out-degrees. We test the null hypothesis of the out-degree activity effect, $H_0 : \beta_{\text{out-d. activity}} = 0$, against the two-sided alternative $\beta_{\text{out-d. activity}} \neq 0$. The empirical p-value for this test is $\hat{\alpha} = 0.002$, and $\tilde{\alpha} = 7.4 \cdot 10^{-4}$ for the test that relies on the normal approximation. Therefore, if the random out-degree is not included in the model, the out-degree activity effect should be included to explain the observed network dynamics.

4. Irrelevance of out-degree activity effect for the model with random out-degrees. Similarly, we test the null hypothesis $H_0 : \beta_{\text{out-d. activity}} = 0$ in the model with random out-degrees. The p-values obtained without and with the assumption of normality are $\hat{\alpha} = 0.54$ and $\tilde{\alpha} = 0.61$, respectively. Therefore, if the random out-degree is included in the model, the out-degree activity effect does not need to be included to explain the observed data. This is coherent with test (2).

5. Relevance of status for the model with random out-degrees. We would like to see whether the heterogeneity in the out-degree is able to account for the

effect of status, testing the hypothesis

$$H_0 : \beta_{\text{status alter}} = \beta_{\text{status ego}} = \beta_{\text{status similarity}} = 0.$$

This is the only composite hypothesis tested. Mean, variance of the simulated test statistics after orthogonalization, and the observed orthogonalized test statistics, are respectively:

$$\bar{\boldsymbol{y}} = \begin{pmatrix} 22.30 \\ 22.57 \\ 10.99 \end{pmatrix}, \quad \hat{\boldsymbol{\Xi}} = \begin{pmatrix} 75.02 & 92.80 & -14.39 \\ 92.80 & 153.79 & -23.87 \\ -14.39 & -23.87 & 48.38 \end{pmatrix}, \quad \boldsymbol{y} = \begin{pmatrix} 9.98 \\ 40.09 \\ 14.52 \end{pmatrix}.$$

The empirical p-value of the test is $\hat{\alpha} = 2 \cdot 10^{-4}$, while the p-value of the test that uses the asymptotic normal distribution is $\tilde{\alpha} < 4 \cdot 10^{-5}$. So, clearly the random out-degree is not able to capture fully the effect of status.

2.5.5 Out-degree activity and overdispersion in Tailor shop network

The main purpose of the statistical analysis was the illustration of the theory developed in the previous sections in an empirical dataset. The emphasis has been on the dichotomy between modelling the observed variability of the out-degree distribution including either the out-degree activity effect or the random out-degree effect. These two differ in their interpretation as drivers for social behaviour of the actors when forming and removing ties. The choice of effect also changes the estimates and the standard deviations of the other parameters in the evaluation function. The model containing both effects included is overparametrized, and the estimation algorithm does not converge because of identifiability issues. The two models that emerge as the best ones from the estimating and testing procedures are not nested and therefore cannot be formally compared by a hypothesis test.

The interpretation of a positive parameter for the out-degree activity effect is that actors with a larger out-degree are more likely to form ties. The interpretation of the random out-degree, on the other hand, is unobserved heterogeneity between actors in their tendency to create outgoing ties. The former model provides an explanation for the observed variability together with a homogeneity assumption. In the absence of a formal information criterion, the choice will have to be made on conceptual grounds.

2.6 Overview and discussion

In this paper we have generalized the stochastic actor-oriented model to include random effects in the evaluation function. This generalization overcomes the limitation that all actors in the study must have the same parameters, allowing actors to react in their own idiosyncratic way to certain circumstances. The generalization is important because random effects are commonly used in longitudinal studies, as the main method to parametrize the heterogeneity of the actors in the study if no suitable covariates are available.

The distribution of the random parameters is parametrized by their variance, which is shared by all actors, and it is estimated with the method of moments, as are the other global parameters. This estimation method is less time-consuming than likelihood-based methods. The generalizations of the evaluation function presented in Section 2.2, and of the estimation algorithm, described in Section 2.3, were relatively straightforward. The score-type test and the standard error of estimation were derived in Section 2.4. These methods allow the comparison of nested models. Further work on information criteria for comparing models would be a welcome addition. Perhaps it is possible to adapt the "explained variation" idea of [61] to the stochastic actor-oriented model with random effects.

In Section 2.5 we fitted and compared various models, with and without random out-degree, on Kapferer's tailor shop data [31]. In some models, some important effects were purposely left out, to test whether the random effect can partially overcome the excluded information. We found that when important information is left out, the estimated variance of the out-degree parameter is higher, meaning that heterogeneity of actors is one way to fit the observed network data. Two of the models considered explain the observed data significantly better than the others. The first one includes the out-degree activity effect and has no random effects. The second model includes the random out-degree effect, and not the outdegree-activity effect. We cannot formally test which of the two models explains the data better, as they are not nested. Therefore, the choice between them must depend on substantive theory and the preferred interpretation.

This was an elaborate example of a model with random out-degree effect. Other effects could also be postulated to have parameters randomly varying between actors. This would be interesting to explore. For future research, it will be interesting to elaborate the generalization to models with multiple correlated random effects, as mentioned in Section 2.4.3.

Stochastic actor-oriented models with random effects provide the flexibility to

choose which parameters should be global, thus common to all actors, and which ones are random, so as to allow heterogeneity between actors. We have described an efficient test for the presence of a random effect, extending the approach for fixed effects. Therefore the researcher can evaluate different nested models, and choose the ones that are theoretically and empirically most appealing.

Chapter 3

Simulated method of moments in mixed effect models

3.1 Introduction

Algorithms for estimation and model evaluation that have been discussed in the previous chapter, are studied here in a different set-up, that is regression with random effects. The generalized version of the method of moments is also discussed. In estimation, the main focus has been the inclusion of updating schemes commonly used in machine learning, for the preconditioning matrix in the iterative algorithm used to estimate the parameters. In model evaluation, the focus has been on discussing various generalizations of the orthogonalization procedure, some of which are compared in a simulation study.

In recent years, the use of mixed-effect models [46, 38] has become increasingly important in statistical analyses for their capacity to incorporate both fixed and random effects, effectively addressing the variability within and between the individuals (or groups) in the study. However, the complexity of these models can make the inference challenging, in particular when the stochastic process that is modelled is not too amenable to analytic simplifications. Simulation-based inference [18], has emerged as a powerful tool to overcome such limitations. This inference method is particularly useful when the stochastic process that is modelled has distribution that determined by a parametrized algorithm that can be used to forward simulate the model. Simulation-based inference blend well together with method of moments estimation, as statistics of the simulated process can be used to infer the parameters by solving stochastically a moment equation.

In Section 3.2, regression with mixed effects are introduced first with the focus of deriving a set of statistics that can be used to estimate the parameter. Then,

the simulated method of moments with its generalized version are discussed. Finally there is a discussion on how to approximate derivatives, that can be used in moment-based estimation algorithms.

Section 3.3 deepens the discussion on stochastic iterative methods to estimate the parameters by adjusting them until the simulated statistics match on average the observed ones. The discussion is based on extending the stochastic gradient descent method, commonly used in machine learning, with the aim of using similar algorithms in our case. It is then discuss how to update the state of the optimization to make it more stable, how the final estimator is computed, and how the convergence of the optimization is evaluated.

In Section 3.4 it is shown how to approximate the quantities that are used to evaluating the quality of the estimator, for example by computing its standard error, or by testing hypothesis on the parameters. Various orthogonalization procedures that can improve the power of the test are discussed, also when the parameters under the null are estimated with generalized method of moments. These orthogonalization procedures are useful when the statistics used in the estimation are correlated with the ones used to derive the test statistic.

In Section 3.5 the theory described is used in a Poisson regression model with mixed effects. The main goal is to assess by simulation the quality of different orthogonalization procedures.

3.2 Simulation based inference for mixed effect models

3.2.1 Regression with mixed effects

A *mixed effect model* [46, 38] is a statistical model in which some *fixed parameters* are shared across all experimental units, while others, called *random* or *individual parameters*, can vary. The *regression with mixed effects* is a model of this type, in which the *response variable* y_i for the i-th observation, has a distribution that depends on a *linear predictor*, that is a linear combination of exogenous information weighted by some of the fixed, and the individual parameters whose distribution depends on other fixed parameters. The number of observations is N and the number of individuals in the study is $n < N$. Formally,

$$Y_i \sim \text{Dist}(\mu_i) \in \mathbb{Y}, \quad \mu_i = \text{link}(\eta_i) \in \mathbb{M}, \quad \eta_i = x_i^\top \beta + z_i^\top \nu(\theta) \in \mathbb{R}, \tag{3.2.1}$$

where $\beta \in \mathbb{B} \subseteq \mathbb{R}^p$ and $\theta \in \mathbb{T} \in \mathbb{R}^r$ are the fixed parameters, $\nu(\theta) \in \mathbb{R}^{qn}$ is the vector containing the random parameters of all individuals, $x_i \in \mathbb{R}^p$ and $z_i \in \mathbb{R}^{qn}$ are exogenous information about observation i called *covariates* and *design* re-

spectively, link is an invertible function and Dist is the parametric distribution of the response variable that is completely specified by μ_i in some cases, otherwise it depends on other parameters that must be estimated. The model above can be written in matrix form as

$$Y \sim \text{Dist}(\mu(\eta)) \in \mathbb{Y}^N, \quad \eta = X\beta + Z\nu(\theta) \in \mathbb{R}^N. \tag{3.2.2}$$

The full definition of the model involves specifying the distribution of the random parameters, we consider

$$\nu = (I_n \otimes \Gamma(\theta))u \sim \text{Norm}(0_{qn}, I_n \otimes \Sigma(\theta)) \in \mathbb{R}^{qn}, \tag{3.2.3}$$

where 0_{qn} and I_n are the qn dimensional vector of zeros and the n dimensional identity matrix respectively, $\otimes$ is the *Kronecker product*, $\Gamma(\theta) \in \mathbb{G} \subseteq \mathbb{R}^{q\times q}$ is an invertible matrix, $\theta \mapsto \Gamma(\theta)$ is a bijection, $\Sigma(\theta) = \Gamma(\theta)\Gamma(\theta)^\top \in \mathbb{S}_q^+$ is a q dimensional positive definite matrix. The vectors ν and $u \sim \text{Norm}(0_{qn}, I_{qn})$ contain the (correlated and uncorrelated, respectively) random parameters of all individuals $j \in \{1, ..., n\}$. In particular the random parameters of individual j are in the positions from $(j-1)q+1$ to jq of $\nu = (\nu_1, ..., \nu_n)$. The Gaussian distribution for the random parameters whose variance is constructed using the Kronecker product, implies that the distribution of ν_j is independent to the random parameters of all other individuals. If observation i is about individual j, only ν_j contributes to η_i as only the values in positions from $(j-1)q+1$ to jq of the design z_i can be different than 0.

Part of the statistical analysis consists in estimating the values of the fixed parameters β and θ. The method of moment estimation that is introduced below with its generalized version, uses a set of statistics, that are functions of the data y containing information about the parameters, to estimate them. A statistic for β that is commonly used in regression is

$$s_\beta(y) = (X^\top X)^{-1} X^\top y \in \mathbb{R}^p. \tag{3.2.4}$$

A statistic for θ is computed in two steps, starting from

$$s_\nu = (s_{\nu 1}, ..., s_{\nu n}) = (Z^\top Z)^{-1} Z^\top (y - X s_\beta(y)) \in \mathbb{R}^{qn}, \tag{3.2.5}$$

and finally

$$s_\theta(y) = \Sigma^{-1}(s_\Sigma) \in \mathbb{T}, \;\; s_\Sigma = \frac{1}{n}\sum_{j=1}^{n}(s_{vj} - \bar{s}_v)(s_{vj} - \bar{s}_v)^\top \in \mathbb{S}_q^+, \;\; \bar{s}_v = \frac{1}{n}\sum_{j=1}^{n} s_{vj} \in \mathbb{R}^q, \tag{3.2.6}$$

where $\Sigma^{-1} : \mathbb{S}_q^+ \to \mathbb{T}$ is the inverse of $\theta \mapsto \Sigma(\theta)$. A simple example is the model with *independent random effects* with standard deviations $\theta \in \mathbb{T} = \mathbb{R}_+^q$, $\Gamma(\theta) = \text{diag}(\theta)$ and $s_\theta(y)$ can contain either the diagonal elements of s_Σ (variances), or their square roots (standard deviations).

3.2.2 Simulated method of moments

The vector of fixed parameters is denoted by $\alpha = (\beta, \theta) \in \mathbb{A} = \mathbb{B} \times \mathbb{T} \subseteq \mathbb{R}^d$, with dimension $d = p + r$. The *method of moments* estimator $\hat{\alpha} \in \mathbb{A}$ is the value that solves the d dimensional *moment equation*

$$E_\alpha(s(Y)) = s(y). \tag{3.2.7}$$

Then, $\hat{\alpha}$ is the value for which the expected value of the statistics, that depends on α, is equal to the observed value of the statistic $s(y) = (s_\beta(y), s_\theta(y))$. It is assumed here that the moment equation is well defined and that it has a unique solution. This is the case when all components of the expected value are well defined, when they are finite, and when $\alpha_h \mapsto E_\alpha s_h(Y)$ is a monotonic function, for all $h \in \{1, ..., d\}$.

For most mixed effect models, the expected value of the statistics can not be computed analytically, meaning that the function $\alpha \mapsto E_\alpha(s(Y))$ is not available. However, it is often the case that the simulated statistic $s(Y(\alpha))$, computed from the value simulated from α, is an unbiased estimate of $E_\alpha(s(Y))$. This estimation method is called *simulated method of moments*, as the moment equation is solved stochastically by adjusting the parameter until the simulated statistics are equal on average to the observed ones. The basic iterative method is the following:

- the current fixed parameter is $\alpha = (\beta, \theta)$,
- sample $u \sim \text{Norm}(0_{qn}, I_{qn})$ and compute $v = (I_n \otimes \Gamma(\theta))u$,
- compute the linear predictor $\eta(\beta, v)$, the mean function $\mu(\eta)$ and sample $Y \sim \text{Dist}(\mu)$,

- update the fixed parameter as

$$\alpha \leftarrow \text{proj}(\alpha - \epsilon H(s(Y) - s(y)), \mathbb{A}), \tag{3.2.8}$$

  where $\epsilon \in \mathbb{R}_+$ is a positive learning rate , $H \in \mathbb{R}^{d\times d}$ is an invertible preconditioning matrix, and the projection to the parametric space is required to force $\alpha \in \mathbb{A}$.

After an initial period, the iterative algorithm will reach stationarity when simulated and observed statistics are equal on average. Various estimation algorithms can be defined by modifying this basic iterations, the most common ones are obtained by decreasing the learning rate ϵ over various iterations, or by updating iteratively the preconditioning matrix H, so that over time it approaches a value that is "optimal" under some assumptions on the model.

The simulated method of moments has been originally developed for estimating discrete choice models in [39].

3.2.3 Generalized method of moments

In the traditional or in the simulated method of moments, the number of statistics and parameters of the model are equal. However, it is possible to use a moment equation with more statistics than parameters.

Consider the statistic $s : \mathbb{Y} \to \mathbb{R}^{d'}$, with $d' > d = \dim(\mathbb{A})$. The *generalized method of moments estimator* is

$$\hat{\alpha}_W = \underset{\alpha\in\mathbb{A}}{\text{argmin}}\ c_W(y, \alpha), \tag{3.2.9}$$

where y is the observed sample and the *cost function*

$$c_W(y, \alpha) = (E_\alpha(s(Y)) - s(y))^\top W (E_\alpha(s(Y)) - s(y)), \tag{3.2.10}$$

is defined so that the d' dimensional moment equation is solved when α minimizes c_W. If $W \in \mathbb{S}^+_{d'}$, meaning a d' dimensional positive definite matrix, the derivative of the cost function with respect to the fixed parameter is

$$\frac{\partial}{\partial \alpha^\top} c_W(y, \alpha) = 2(E_\alpha(s(Y)) - s(y))^\top W J(\alpha) \in \mathbb{R}^{1\times d}, \tag{3.2.11}$$

with $J(\alpha) = \partial E_\alpha(s(Y))/\partial \alpha^\top \in \mathbb{R}^{d'\times d}$.

The quality of the estimator $\hat{\alpha}_W$ depends on W, as its asymptotic variance is

$$C(\alpha^*) = (G^\top W G)^{-1} G^\top W V W^\top G (G^\top W^\top G)^{-1} \in \mathbb{S}_d^+, \tag{3.2.12}$$

where α^* is the real value of the parameter,

$$G = E_{\alpha^*}(\partial (E_\alpha(s(Y)) - s(y))/\partial \alpha^\top) \in \mathbb{R}^{d' \times d}, \tag{3.2.13}$$

and

$$V = E_{\alpha^*}((E_\alpha(s(Y)) - s(y))(E_\alpha(s(Y)) - s(y))^\top) \in \mathbb{S}_{d'}^+. \tag{3.2.14}$$

The most efficient estimator is obtained with $W \propto V^{-1}$, in this case $C(\alpha^*) = (G^\top V^{-1} G)^{-1}$.

The method of moments and its generalized counterpart can be also distinguished by the fact that the former allows for the direct solution of the moment equation, whereas the latter requires an optimization, such as the one in equation (3.2.9). We can define the optimization implicity by solving directly the d' dimensional equation $E_\alpha(s(Y)) = s(y)$ for $\alpha \in \mathbb{A} \subseteq \mathbb{R}^d$, or equivalently

$$L(E_\alpha(s(Y)) - s(y)) = 0_d, \tag{3.2.15}$$

for a matrix $L \in \mathbb{R}^{d \times d'}$ of rank d. If α^* is the real value of the parameter, the asymptotic variance of $s(Y)$ is V defined in equation (3.2.14), then the asymptotic variance of the solution $\tilde{\alpha}_L$ of equation (3.2.15) is

$$C(\alpha^*) = (LG)^{-1} L V L^\top (LG)^{-\top} \in \mathbb{S}_d^+, \tag{3.2.16}$$

[22], where G and V are defined in equations (3.2.13) and (3.2.14) respectively, and $(LG)^{-\top} = ((LG)^{-1})^\top = ((LG)^\top)^{-1}$. The most efficient $\tilde{\alpha}_L$ is obtained when $L \propto G^\top V^{-1}$, so that $C(\alpha^*) = (G^\top V^{-1} G)^{-1}$, that is the same variance of $\hat{\alpha}_W$ for $W \propto V^{-1}$. Note that in the "non generalized" method of moment, L and G are both invertible matrices, so $(LG)^{-1} = G^{-1} L^{-1}$, and $C(\alpha^*) = G^{-1} V G^{-\top}$ for all L.

The basic iteration (3.2.8) is modified as

$$\alpha \leftarrow \text{proj}(\alpha - \epsilon H L(s(Y) - s(y)), \mathbb{A}), \tag{3.2.17}$$

to compute $\tilde{\alpha}_L$ by solving the moment equation (3.2.15) stochastically. The advantage of the implicit method is that the derivative (3.2.11) is not required to compute the estimator.

3.2.4 Approximation of derivatives in simulated inference

The estimation and the model evaluation procedure are sometimes improved by using derivatives of the expected value of the statistics. For the linear regression with mixed effects described above these derivatives can be computed, but for more complex models this is not the case. In particular, for models in which the probability density is not available, the expected value of the statistics is not an analytic function of the parameters, so it can not be differentiated. One example is when the stochastic process being modelled is largely unobserved, save for a few timestamps.

Method of moment procedures rely on the ability to compute the expected value $E_\alpha(s(Y))$ of the statistic $s : \mathbb{Y} \to \mathbb{R}^{d'}$, as function of the parameters $\alpha \in \mathbb{A} \subseteq \mathbb{R}^d$. If the function

$$\alpha \mapsto E_\alpha(s(Y)) = \int_{\mathbb{Y}} s(y) dP(y|\alpha) \in \mathbb{R}^{d'} \tag{3.2.18}$$

is analytic, it can be differentiated with respect to α. Under some regularity conditions [35, Theorem 1], if $dP(y|\alpha) = p(y|\alpha)dy$, then

$$\begin{aligned} J(\alpha) = \frac{\partial}{\partial \alpha^\top} E_\alpha(s(Y)) = \frac{\partial}{\partial \alpha^\top} \int_{\mathbb{Y}} s(y) p(y|\alpha) dy = \int_{\mathbb{Y}} s(y) \Big(\frac{\partial}{\partial \alpha^\top} p(y|\alpha) \Big) dy = \\ = \int_{\mathbb{Y}} s(y) \Big(\frac{\partial}{\partial \alpha^\top} \log p(y|\alpha) \Big) p(y|\alpha) dy = E_\alpha \Big(s(Y) \frac{\partial}{\partial \alpha^\top} \log p(Y|\alpha) \Big) \in \mathbb{R}^{d' \times d}. \end{aligned} \tag{3.2.19}$$

The derivative of the log-density of the data is called *score function*. The *likelihood ratio method* [35, Section 2.4] for approximating the expected value of the derivative is based on a stochastic approximation of the expected value in equation (3.2.19). This technique is useful especially when $p(y|\alpha)$ is not an analytic function when y is the observed sample, for example in the case in which the generative process is not fully observed. When Y is simulated, however, the density $p(Y|\alpha)$ may be calculated since the entire simulated process is accessible.

Monte Carlo integration, which is based on the law of large numbers, can be used to approximate an integral when it cannot be solved analytically. The integral in equation (3.2.18), for example, can be approximated by simulating many samples $Y_1(\alpha), ..., Y_m(\alpha)$ from $dP(y|\alpha)$, computing the value of $s(Y_k(\alpha))$ for all s, and then the arithmetic mean $\sum_k s(Y_k(\alpha))/m$ is used as approximation

of the integral. The expected value in equation (3.2.19) can be approximated by

$$\hat{J}(\alpha) = \frac{1}{m}\sum_{k=1}^{m} s(Y_k(\alpha))\Big(\frac{\partial}{\partial \alpha^\top}\log p(Y_k(\alpha)|\alpha)\Big), \tag{3.2.20}$$

and both functions can be computed because the samples $Y_k(\alpha)$ are simulated.

Depending on the parametrization of the model, it is possible that the samples are not simulated directly from the fixed parameters of interest. In mixed effect models for example, θ is first transformed to an invertible matrix $\Gamma(\theta)$, that is used to generate the random parameters v from which the model is simulated. Consider the function $\alpha \mapsto \zeta = f(\alpha)$, where α is the parameter of interest and ζ is used to simulate the process. If the function $\alpha \mapsto \zeta$ is deterministic and analytic, equation (3.2.20) is rewritten using the chain rule as

$$\hat{J}(\alpha) = \Big(\frac{1}{m}\sum_{k=1}^{m} s(Y_k(\zeta))\Big(\frac{\partial}{\partial \zeta^\top}\log p(Y_k(\zeta)|\zeta)\Big)\Big)\frac{\partial f(\alpha)}{\partial \alpha^\top}, \tag{3.2.21}$$

with $\zeta = f(\alpha)$. On the other hand, if ζ is a random parameter sampled from α as $\zeta = F(\alpha)$, where F is a random function, $f_k(\alpha)$ is sampled from $F(\alpha)$, $\zeta_k = f_k(\alpha)$, and

$$\hat{J}(\alpha) = \frac{1}{m}\sum_{k=1}^{m}\Big(s(Y_k(\zeta_k))\Big(\frac{\partial}{\partial \zeta_k^\top}\log p(Y_k(\zeta_k)|\zeta_k)\Big)\frac{\partial f_k(\alpha)}{\partial \alpha^\top}\Big), \tag{3.2.22}$$

because $s(Y_k(\zeta_k))\partial \log p(Y_k(\zeta_k)|\zeta_k)/\partial \zeta_k^\top$ and $\partial f_k(\alpha)/\partial \alpha^\top$ are not independent. Note that

$$\frac{\partial}{\partial \alpha^\top}E_\alpha(s(Y)) = \frac{\partial}{\partial \alpha^\top}(E_\alpha(s(Y)) - s(y)) = \frac{\partial}{\partial \alpha^\top}E_\alpha(s(Y) - s(y)), \tag{3.2.23}$$

because $\alpha \mapsto s(y)$ and $Y \mapsto s(y)$ are constant functions, but usually the approximation of $\hat{J}(\alpha)$ is more stable if in the three equations (3.2.20 - 3.2.22), $s(Y)$ is replaced by $(s(Y) - s(y))$, where Y is $Y_k(\alpha)$, $Y_k(\zeta)$ and $Y_k(\zeta_k)$ in the three cases.

3.3 Simulation based estimation

The usage of derivatives of the function being optimized, as well as the number of derivatives employed, can be used to roughly categorize optimization techniques. In the simulated method of moments, a zeroth-order optimization method is used when the moment equation is solved “directly", meaning that in the basic itera-

tion (3.2.17), the matrices H and L are either not updated, or they are updated by using only the variance of the simulated statistics. Whereas in a first-order optimization method L is updated using the first derivative of the statistics.

3.3.1 Stochastic and simulated gradient descent

We are interested in developing a theory that is general enough to be used in models in which the expected value of the statistics, and its derivative with respect to the parameters, can not be computed. The optimization becomes stochastic if the statistics are sampled. The *stochastic optimization* algorithm is due to Robbins and Monro [50], in which the basic iteration (3.2.17) is used but the learning rate is decreased in each iteration. If the learning rate in iteration k is $\epsilon_k = O(1/k)$, then as $k \to \infty$, the current parameter α_k converges to $\hat{\alpha}$, that is the solution of the moment equation, for every positive definite preconditioning matrix H. The algorithm of Polyak and Juditsky [47] uses a larger learning rate, on the order of $\epsilon_k = O(1/k^c)$, with $c \in (1/2, 1)$, but the current value of the parameter at iteration m is $\bar{\alpha}_m = \sum_{k=1}^{m} \alpha_k/m$, that is guaranteed to converge to $\hat{\alpha}$, for every positive definite H. For $c \in (0, 1/2]$, $\bar{\alpha}_m$ is also guaranteed to converge to the solution of the moment equation, but for a finite number of iterations, the quality of the approximation of $\hat{\alpha}$ with $\bar{\alpha}_m$ depends on the preconditioning matrix H. In more modern approaches, mainly coming from machine learning, $\epsilon_k = O(1)$, so that a much quicker convergence can be achieved, but the approximation relies heavily on the updating scheme for H, and on assumptions about the optimized functions [37].

If the derivative of the function that is to be optimized can be computed explicitly, the parameter can be estimated with a *gradient descent* iterative method

$$\alpha \leftarrow \text{proj}(\alpha - \epsilon H \nabla(y, \alpha), \mathbb{A}), \tag{3.3.1}$$

where the gradient $\nabla(y, \alpha)$ is the transpose of the derivative of the function to be optimized with respect to α, evaluated at the current value of the parameter, and the observed data, and the projector operator is required to keep α in its parametric space. For the generalized method of moment estimator defined in equation (3.2.9) the gradient is $\nabla(y, \alpha) = \partial c_W(y, \alpha)/\partial \alpha$, that is the transpose of the value in equation (3.2.11).

The computation of $\nabla(y, \alpha)$ is often linear on the size of the sample y, in the *stochastic gradient descent* the iteration (3.3.1) is replaced with

$$\alpha \leftarrow \text{proj}\Big(\alpha - \frac{n\epsilon}{u} H \nabla(y_U, \alpha), \mathbb{A}\Big), \tag{3.3.2}$$

where the random vector $\nabla(y_U, \alpha)$ is such that $E(\nabla(y_U, \alpha)) = u\nabla(y, \alpha)/n$, and y_U is an u dimensional random subsample of the n dimensional observed dataset y. The advantage of this approach is that in large datasets the computation of the gradient can be costly, so the optimization is made stochastic by computing the gradient in a random subsample, in each iteration.

If the function $\alpha \mapsto \nabla(y, \alpha)$ can not be computed explicitly, the optimization can be made stochastic using simulation, replacing $\nabla(y, \alpha)$ by their simulated version $\hat{\nabla}(Y(\alpha), \alpha)$, as long as $E(\hat{\nabla}(Y(\alpha), \alpha)) = \nabla(y, \alpha)$, where $Y(\alpha)$ is a simulation of the process wiht parameter α. For the computation of $\hat{\alpha}_W$ in equation (3.2.9), the simulated gradients

$$\hat{\nabla}(Y(\alpha), \alpha) = 2\Big(\frac{\partial}{\partial \alpha^\top} \log p(Y(\alpha)|\alpha)\Big)^\top s(Y(\alpha))^\top W(s(Y(\alpha)) - s(y)), \tag{3.3.3}$$

and

$$\hat{\nabla}(Y(\alpha), \alpha) = 2\Big(\frac{\partial}{\partial \alpha^\top} \log p(Y(\alpha)|\alpha)\Big)^\top (s(Y(\alpha)) - s(y))^\top W(s(Y(\alpha)) - s(y)), \tag{3.3.4}$$

have both expected value $\nabla(y, \alpha)$, then $\hat{\alpha}_W$ can be computed with the simulated gradient descent

$$\alpha \leftarrow \text{proj}(\alpha - \epsilon H \hat{\nabla}(Y(\alpha), \alpha), \mathbb{A}). \tag{3.3.5}$$

The two stochastic approaches can be combined by using the gradients $n\hat{\nabla}(Y_U(\alpha), \alpha)/u$, in which $Y(\alpha)$ and $y(\alpha)$ in the equations above are replaced by $Y_U(\alpha)$ and $y_U(\alpha)$ respectively. The advantage is that we have to simulate the process only for a subset of the data, but the unbiasedness of the simulated gradient relies on the fact that the generative process of a smallest subsample is representative of the one in the full sample. This is the case when it is assumed that the observed data are independent.

3.3.2 Update state optimization

With large learning rates, e.g. $\epsilon_k = O(1)$ for all iterations k, the quality of the approximation depends on the updating scheme for the preconditioning matrix H. The goal is to “standardize" the variability of the components of the gradient, so that the size of the learning rate has a comparable effect to all components. Usually diagonal preconditioning matrices are used for computational reasons. Some examples common in the literature are now described.

In the basic updating scheme, called *AdaGrad* [19], a matrix $\tilde{H}$ is updated in each iteration as $\tilde{H} \leftarrow \tilde{H} + \nabla\nabla^\top$, and then the preconditioning matrix is $H =$

$\text{diag}(\tilde{H})^{-1/2}$. In *RMSProp* [69], $H = \text{diag}(1/\sqrt{h_i})$, and the updates for h_i are

$$h_i \leftarrow \gamma h_i + (1-\gamma)\nabla_i^2, \tag{3.3.6}$$

for all i, with forgetting factor $\gamma \in (0,1)$. In *Adam* [32], the gradients in the previous iterations are also used to update the preconditioning, the updates in iteration k are

$$\begin{array}{rcl} \bar{\nabla} & \leftarrow & (\gamma_1 \bar{\nabla} + (1-\gamma_1)\nabla)/(1-\gamma_1^k), \\ h_i & \leftarrow & (\gamma_2 h_i + (1-\gamma_2)\nabla_i^2)/(1-\gamma_2^k), \ \forall i, \\ \alpha & \leftarrow & \text{proj}(\alpha - \epsilon \text{diag}(1/(\sqrt{h_i}+\delta))\bar{\nabla}, \mathbb{A}), \end{array} \tag{3.3.7}$$

where $\delta \approx 0$, $\gamma_1, \gamma_2 \in (0,1)$.

All these methods for updating the preconditioning matrix H can be used in all "types" of moment-based estimations discussed in Section 3.2. The difference between the estimation method is in how ∇ is computed. For computing $\hat{\alpha}_W$ in equation (3.2.9) the simulated gradients (3.3.4) and (3.3.5) are required. However the "simulated score function" $\partial / \log p(Y(\alpha)|\alpha)/\partial \alpha^\top$ can vary widely, making the algorithm unstable. Moreover the matrix $W \in \mathbb{R}^{d' \times d'}$ has to be set. The advantage of solving directly equation (3.2.15), and so of computing $\tilde{\alpha}_L$, is the score function is not used to update the parameter, so that the algorithm is more stable. However $L \in \mathbb{R}^{d \times d'}$ has still to be set, and its optimal values depends on the matrices $G \in \mathbb{R}^{d \times d'}$ and $V \in \mathbb{S}_{d'}^+$ defined in equations (3.2.13) and (3.2.14). The approach that we follow is to use the matrix L that is "optimal at the starting point" of the optimization. Therefore, the statistics $s(Y_1(\alpha)), ..., s(Y_m(\alpha))$ that are simulated at the starting point α, are used to approximate their variance $\hat{V}(\alpha)$, and their Jacobian $\hat{J}(\alpha)$ with the method in Section 3.2.4. Then the matrix $L = \hat{J}(\alpha)\hat{V}(\alpha)^{-1}$ that is used in the whole optimization with iteration (3.2.17).

In Figure 3.1 are shown the chains of parameters during an estimation in the same dataset, with fixed and variable preconditioning H. In the second case H is updated with RMSprop (3.3.6), with $\gamma = 0.9$. It can be seen that when the preconditioning H is not updated, the oscillations are different for the various parameters. For example, in the left plot the blue chain of θ_2 (standard deviation of a random effect) has much smaller oscillations than the red chain of β_2 (regression parameter), whereas in the right plot all chains have oscillations with more similar variance.

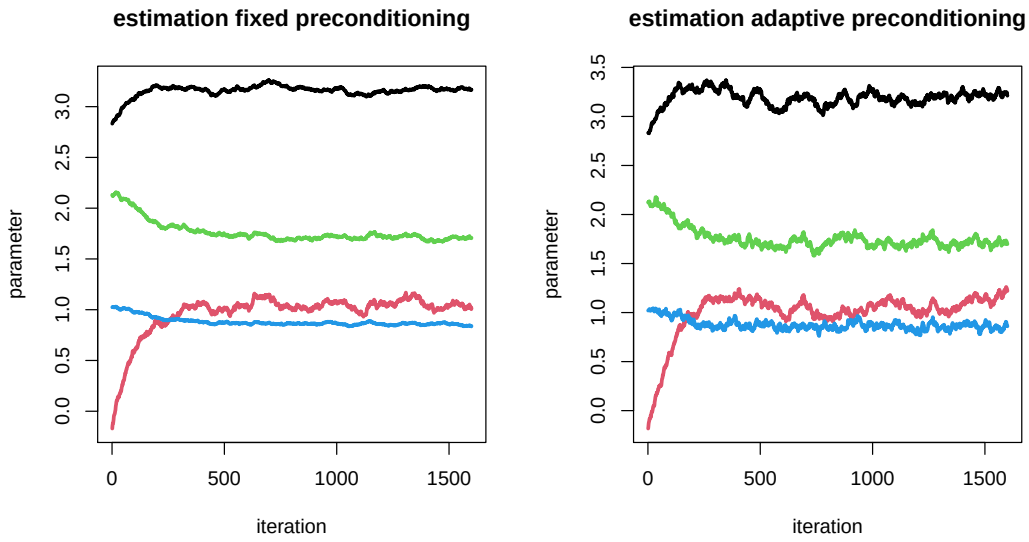

Figure 3.1: Chain paramerters during estimation with fixed (left) and variable (right) preconditioning.

3.3.3 Computation of the estimator and evaluation of convergence

Depending on whether the learning rate is decreased between iterations, and if so on how quickly it goes to 0, the estimator is computed differently. We consider the case in which the learning rate is kept constant, therefore the current value of α does not tend to the solution $\tilde{\alpha}_L$ of the moment equation. However when stationarity is reached, the current value oscillates around $\tilde{\alpha}_L$, so that the values of α after stationarity is reached can be averaged to approximate $\tilde{\alpha}_L$. Formally, if $\alpha_1, ..., \alpha_m$ are the values that are used to simulate the process at iterations $1, ..., m$ respectively, the final estimator is defined as

$$\hat{\alpha} = \frac{1}{m-b} \sum_{k=b+1}^{m} \alpha_k, \tag{3.3.8}$$

where $0 \leq b < m$ is a value that ends the “burn-in" period, in which it is not assumed that the process has reached stationarity. Note that $\hat{\alpha} \in \mathbb{A}$ as $\alpha_k \in \mathbb{A}$ for all k, and $\mathbb{A}$ is a convex space.

There are two possible issues that determinates whether $\hat{\alpha}$ can be used instead of the solution of the moment equation $\tilde{\alpha}_L$. First, the stationarity of the chain $(\alpha_k)_k$ for $k \in \{b+1, ..., m\}$ has to be assessed. This can be done in various ways, as every method that can be used to test stationarity of a stochastic process can

be used. A simple approach is to check for all statistics s_h where $h \in \{1, ..., d'\}$, for how many $k \in \{b+1, ..., m\}$, the difference $s_h(Y(\alpha_k)) - s_h(y)$ is positive. If α_k oscillates around the solution of the moment equation, the sign of the differences should have mean approximately 0 for all statistics. This approach is more convenient than considering the differences $\alpha_{hk} - \alpha_{h(k-1)}$, as the projection to $\mathbb{A}$ can force some of them to be 0 or approximately 0, when the parameter is at the boundary of $\mathbb{A}$. A second related issue is to check the approximation error $\hat{\alpha} - \tilde{\alpha}_L$. This is done by evaluating the average of the simulated differences $s_h(Y(\alpha_k)) - s_h(y)$ for $k \in \{b+1, ..., m\}$ that should be approximately 0 for all $h \in \{1, ..., d'\}$. Note that for assessing stationarity the average of the sign of the simulated differences is used, whereas for assessing the approximation error the average of the simulated differences is used.

In Figure 3.2, the estimated values of β_1 and θ_1 for 100 simulated datasets of the model that will be discussed in Section 3.5, are plotted with the proportion of times simulated and observed statistics are positive, and their associated preconditioning at the end of the estimation. The vertical lines are the real values for β_1 and θ_1. It can be seen that the counts of signs of differences between simulated and observed statistics, after the end of the burn-in period can be used to evaluate the convergence of the optimization. Another important consideration is that when the optimization does not converge, the preconditioning shrink.

3.4 Simulation based model evaluation

3.4.1 Monte-Carlo approximation of quantities for model evaluation

Under assumptions [24], the asymptotic distribution of the generalized method of moment estimator is

$$\sqrt{N}(\hat{\alpha}(Y) - \alpha^*) \xrightarrow{d} \text{Norm}(0, C(\alpha^*)), \tag{3.4.1}$$

as $n \to \infty$, where N is the size of the sample Y, that is generated from the real value α^*, and $C(\alpha^*)$ is equal to (3.2.12) when $\hat{\alpha} = \hat{\alpha}_W$, or it is equal to (3.2.16) when $\hat{\alpha} = \tilde{\alpha}_L$. The computation of $C(\alpha^*)$ requires the avaliability of the functions

$$\begin{aligned} \tilde{\alpha} &\mapsto G(\tilde{\alpha}) = E_\alpha(\partial(E_\alpha(s(Y)) - s(y))/\partial\alpha^\top)|_{\alpha=\tilde{\alpha}}, \\ \tilde{\alpha} &\mapsto V(\tilde{\alpha}) = E_\alpha((E_\alpha(s(Y)) - s(y))(E_\alpha(s(Y)) - s(y))^\top)|_{\alpha=\tilde{\alpha}}, \end{aligned} \tag{3.4.2}$$

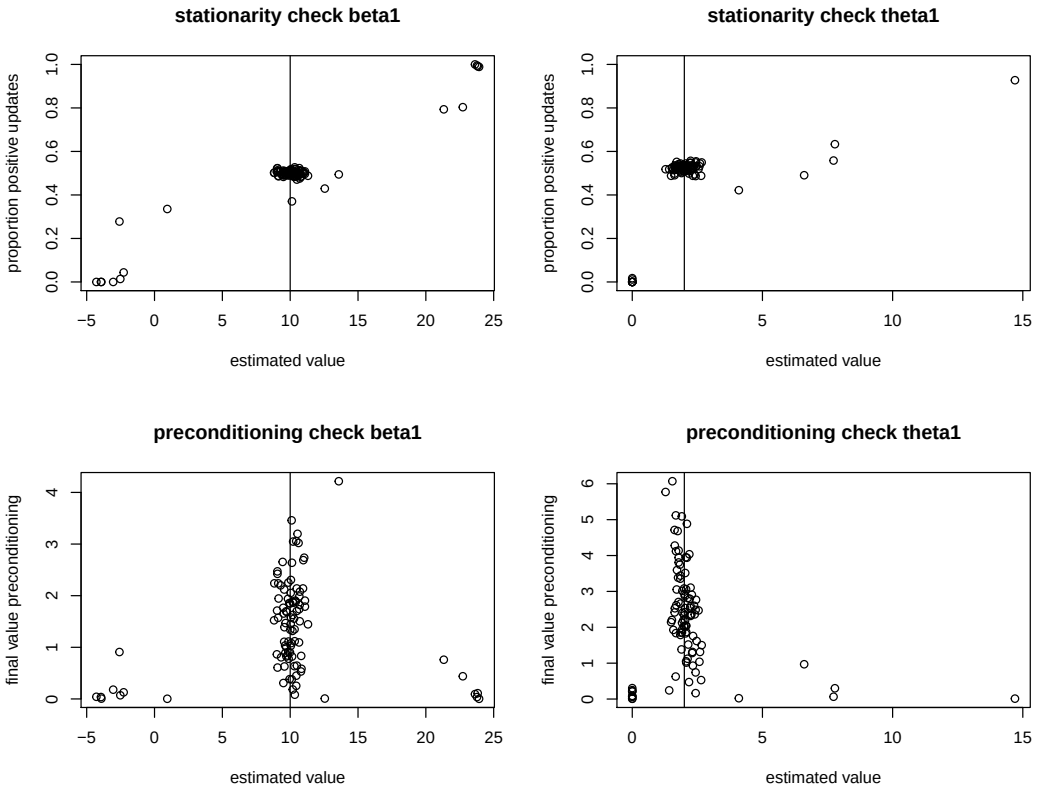

Figure 3.2: Top row with estimated values in a simulation, plotted against the sign of differences between simulated and observed statistics. Bottom row estimated values are plotted against the final value of the preconditioning in the optimization.

that are used in equations (3.2.13) and (3.2.14) respectively, when $\tilde{\alpha} = \alpha^*$. The first problem is that α^* is unknown, but *Slutsky's theorem* implies that the asymptotic distribution of the generalized method of moment estomator is the same if $C(\alpha^*)$ is replaced by $C(\hat{\alpha})$, that is computed by replacing $\tilde{\alpha}$ with $\hat{\alpha}$ in the functions (3.4.2), so that the matrices $G(\hat{\alpha})$ and $V(\hat{\alpha})$ are used to compute $C(\hat{\alpha})$. However, we are interested in cases in which the functions $\alpha \mapsto G(\alpha)$ and $\alpha \mapsto V(\alpha)$ are not available, and so $C(\hat{\alpha})$ is approximated by $\hat{C}$, that is obtained by replacing $G(\hat{\alpha})$ and $V(\hat{\alpha})$ with their approximated values $\hat{G}$ and $\hat{V}$ respectively.

The matrices $\hat{G}$ and $\hat{V}$ are computed with Monte-Carlo integration. The generative process is simulated m times from the estimated parameter $\hat{\alpha}$. We consider the general case in which there are random parameters in the model. Therefore the k-th simulated value is $Y_k(\zeta_k)$, where ζ_k is simulated from a random function, usually degenerate in some of its components, parametrized by $\hat{\alpha}$, that is $\zeta_k = f_k(\hat{\alpha})$. The Monte-Carlo approximation of $V(\hat{\alpha})$ is

$$\hat{V} = \frac{1}{m}\sum_{k=1}^{m}(s(Y_k(\zeta_k)) - \hat{s})(s(Y_k(\zeta_k)) - \hat{s})^\top, \tag{3.4.3}$$

where the vector of average statistics

$$\hat{s} = \frac{1}{m}\sum_{k=1}^{m} s(Y_k(\zeta_k)), \tag{3.4.4}$$

is used in the computation of the sample variance of the simulated statistics $\hat{V}$. The matrix of expected derivatives is

$$\hat{G} = \frac{1}{m}\sum_{k=1}^{m}\left(\left(s(Y_k(\zeta_k)) - s(y)\right)\left(\frac{\partial}{\partial \zeta_k^\top}\log p(Y_k(\zeta_k)|\zeta_k)\right)\left(\frac{\partial f_k(\alpha)}{\partial \alpha^\top}\Big|_{\alpha=\hat{\alpha}}\right)\right), \tag{3.4.5}$$

so that $\hat{G} = \hat{J}(\hat{\alpha})$ that was defined in equation (3.2.22). Note that the first term of the summands is $s(Y_k(\zeta_k)) - s(y)$ instead of $s(Y_k(\zeta_k))$ that was used in equation (3.2.22), because the approximation is more stable. Therefore, the approximated variance matrix for $\hat{\alpha}_W$ or $\tilde{\alpha}_L$ are

$$\begin{aligned}\hat{C}_W &= (\hat{G}^\top W \hat{G})^{-1}\hat{G}^\top W \hat{V} W^\top \hat{G}(\hat{G}^\top W^\top \hat{G})^{-1},\\ \hat{C}_L &= (L\hat{G})^{-1} L\hat{V}L^\top (L\hat{G})^{-\top},\end{aligned} \tag{3.4.6}$$

respectively.

3.4.2 Score tests with Neyman's orthogonalization procedure

The vector $\hat{s}$, with the matrices $\hat{V}$ and $\hat{G}$, can be used to test statistical hypothesis on the parameter α. The d dimensional parametric space $\mathbb{A}$ is decomposed in $\mathbb{A} = \mathbb{A}_1 \cup \mathbb{A}_2$, where $\mathbb{A}_1$ and $\mathbb{A}_2 = \mathbb{A} \setminus \mathbb{A}_1$ have dimension d_1 and $d_2 = d - d_1$ respectively. The null hypothesis is in general

$$H_0 : \alpha \in \mathbb{A}_1, \tag{3.4.7}$$

but we consider a simpler case in which α is decomposed as $\alpha = (\alpha_1, \alpha_2)$, where α_1 is d_1 dimensional, and

$$\mathbb{A}_1 = \{\alpha = (\alpha_1, \alpha_2) \in \mathbb{A} : \alpha_2 = 0_{d_2}\}, \tag{3.4.8}$$

so that the null hypothesis can be written as

$$H_0 : \alpha_2 = 0_{d_2}. \tag{3.4.9}$$

The vector of statistics for α can be similarly decomposed as $s = (s_1, s_2)$, its expected values, variance, and derivative with respect to α are decomposed accordingly as

$$\mu(\alpha) = \begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix}, \quad V(\alpha) = \begin{pmatrix} V_{11} & V_{12} \\ V_{12}^\top & V_{22} \end{pmatrix}, \quad J(\alpha) = \begin{pmatrix} J_{11} & J_{12} \\ J_{21} & J_{22} \end{pmatrix}, \tag{3.4.10}$$

respectively.

The *Neyman's orthogonalization procedure* [43] is used to transform the statistic s in a test statistic for (3.4.9) with high power. The orthogonalization procedure when the estimate is computed with the method of moments is considered first, the procedure with generalized method of moments is discussed in Section 3.4.4. If the asymptotic mean and variance of s, are $\mu(\alpha)$ and $V(\alpha)$ respectively, when the real parameter is α, then

$$w(Y, \alpha) = s_2(Y, \alpha) - \Lambda(\alpha) s_1(Y, \alpha) \in \mathbb{R}^{d_2}, \tag{3.4.11}$$

where

$$\Lambda(\alpha) = J_{21}(\alpha) J_{11}(\alpha)^{-1} \in \mathbb{R}^{d_2 \times d_1}, \tag{3.4.12}$$

so that $w(Y, \alpha)$ has expected value and variance

$$\begin{aligned} &E(w(Y, \alpha)) = \mu_2(\alpha) - \Lambda(\alpha)\mu_1(\alpha), \\ &\Xi(\alpha) = V_{22}(\alpha) - (\Lambda(\alpha)V_{12}(\alpha) + V_{12}(\alpha)^\top \Lambda(\alpha)^\top) + \Lambda(\alpha)V_{11}(\alpha)\Lambda(\alpha)^\top, \end{aligned} \tag{3.4.13}$$

respectively.

We denote with $\hat{\alpha}$ the estimated parameter under the null hypothesis (3.4.9). As $\hat{\alpha}$ solves the moment equation, when the null hypothesis is true $E(w(Y,\hat{\alpha})) = 0_{d_2}$ and the test statistic

$$z^2(Y) = (w(Y,\hat{\alpha}) - w(y,\hat{\alpha}))^\top \Xi(\hat{\alpha})^{-1/2}(w(Y,\hat{\alpha}) - w(y,\hat{\alpha})) \in [0,\infty), \quad (3.4.14)$$

has asymptotic Chi-squared distribution, with d_2 degrees of freedom. We are interested in the general case in which $\alpha \mapsto E_\alpha(A(Y))$ is not analytic. The simulated values $Y_k(\zeta_k)$, $\zeta_k = f_k(\hat{\alpha})$, for $k \in \{1,...,s\}$, are then used to approximate $V(\hat{\alpha})$, $a(y,\hat{\alpha})$, and $J(\hat{\alpha})$ with $\hat{V}$, $\hat{a}$ and $\hat{G}$ respectively, these quantities were computed in equations (3.4.3 - 3.4.5). Then $z^2(Y)$ is approximated as

$$\hat{z}^2 = (\hat{w} - w)^\top \hat{\Xi}^{-1/2}(\hat{w} - w) = ||\hat{w} - w||^2_{\hat{\Xi}}, \quad (3.4.15)$$

where $\hat{w} = \hat{s}_2 - \hat{\Lambda}\hat{s}_1$, $w = s_2(y) - \hat{\Lambda}s_1(y)$, $\hat{\Lambda} = \hat{G}_{21}\hat{G}_{11}^{-1}$, and $\hat{\Xi}$ is computed replacing $\Lambda(\alpha)$ and $V(\alpha)$ with $\hat{\Lambda}$ and $\hat{V}$ in equation (3.4.13).

The *p-value* of the test (3.4.9) based on the asymptotic distribution of $z^2(Y)$ is approximated as

$$\tilde{p} = 1 - \chi^2_{d_2,\hat{z}^2} \in (0,1], \quad (3.4.16)$$

where the last term is the quantile of the Chi-squared distribution with d_2 degrees of freedom, evaluated at $\hat{z}^2$. The simulated values $Y_k(\zeta_k)$ can be used to compute a p-value that does not relies in the asymptotic distribution of the statistics. For each simulated value, compute the approximated statistic $\hat{w}_k = \hat{w}(Y_k(\zeta_k),\hat{\alpha})$, and their mean $\bar{w} = \sum_{k=1}^m \hat{w}_k/s$, then the p-value of the test (3.4.9) is approximated as

$$\begin{aligned} \hat{p} &= \frac{1}{m}\sum_{k=1}^m I(||\hat{w}_k - \bar{w}||^2_{\hat{\Xi}} > ||w - \bar{w}||^2_{\hat{\Xi}}) \\ &= \frac{1}{m}\sum_{k=1}^m I((\hat{w}_k - \bar{w})^\top \hat{\Xi}^{-1}(\hat{w}_k - \bar{w}) > \hat{z}^2) \in [0,1], \end{aligned} \quad (3.4.17)$$

where $\hat{z}^2$ is the same value that was used in (3.4.16), and I is the indicator function such that $I(\text{True}) = 1$ and $I(\text{False}) = 0$. Therefore, the difference between $\tilde{p}$ and $\hat{p}$ is that the latter is based on the simulated empirical distribution of the statistic $w(Y,\hat{\alpha})$, while for computing $\tilde{p}$, only the first two moments (approximated by simulation) of the distribution of $w(Y,\hat{\alpha})$ are used.

The idea of using orthogonalization to increase the power of score tests is an adaptation of the work presented in [56], which is itself based on [4, 3].

3.4.3 Different orthogonalizations procedures

In the previous section it was assumed that the vector of statistics s that is used to derive the test statistic z^2 has the same dimension d of the parametric space $\mathbb{A}$. For estimating the model under the null, s_2 is not used as $\alpha_2 = 0_{d_2}$ under H_0. Then J_{11} is a $d_1 \times d_1$ dimensional matrix that is invertible, so $\Lambda(\alpha)$ can be computed as in equation (3.4.12). However it is not required to use all statistics and all parameters (under the null) in the orthogonalization procedure.

The case in which all statistics are used in the orthogonalization, but not all parameters under the null, is discussed first. Without loss of generality suppose that only the first $d_1^\circ \leq d_1$ parameters under the null are used. The statistic $w(Y,\alpha)$ is replaced by

$$w(Y,\alpha)^\circ = s_2(Y,\alpha) - \Lambda^\circ(\alpha) s_1(Y,\alpha) \in \mathbb{R}^{d_2}, \tag{3.4.18}$$

where

$$\Lambda^\circ(\alpha) = J_{21}^\circ (J_{11}^{\circ\top} J_{11}^\circ)^{-1} J_{11}^{\circ\top} \in \mathbb{R}^{d_2 \times d_1}, \tag{3.4.19}$$

and the matrices $J_{11} \in \mathbb{R}^{d_1 \times d_1}$ and $J_{21} \in \mathbb{R}^{d_2 \times d_1}$ are decomposed as $J_{11} = (J_{11}^\circ \; J_{11}^{\circ\circ})$ and $J_{21} = (J_{21}^\circ \; J_{21}^{\circ\circ})$, respectively, with $J_{11}^\circ \in \mathbb{R}^{d_1 \times d_1^\circ}$ and $J_{21}^\circ \in \mathbb{R}^{d_2 \times d_1^\circ}$. Therefore, when computing $\Lambda^\circ(\alpha)$ instead of $\Lambda(\alpha)$, the left inverse of J_{11}° is used instead of the inverse of J_{11}.

Now the case in which fewer statistics than parameters are used in the orthogonalization procedure, is considered. In this case the statistic $w(Y,\alpha) \in \mathbb{R}^{d_2}$ is replaced by

$$w(Y,\alpha)^* = s_2(Y,\alpha) - \Lambda^*(\alpha) s_1^*(Y,\alpha) \in \mathbb{R}^{d_2}, \tag{3.4.20}$$

where $s_1^*(Y,\alpha)$ is d_1^* dimensional, for $d_1^* \leq d_1$, and $\Lambda^*(\alpha) \in \mathbb{R}^{d_2 \times d_1^*}$. Assume without loss of generality that only the first d_1^* statistics are used in the orthogonalization, so that J_{11} is decomposed as $J_{11}^\top = (J_{11}^{*\top} \; J_{11}^{**\top})^\top$, that is, the first d_1^* rows of J_{11} are selected. Then, $\Lambda^*(\alpha)$ is computed using the right inverse of J_{11}^*, as

$$\Lambda^*(\alpha) = J_{21} J_{11}^{*\top} (J_{11}^* J_{11}^{*\top})^{-1} \in \mathbb{R}^{d_2 \times d_1^*}. \tag{3.4.21}$$

Note that the whole J_{21} is used, as all parameters under the null are used in the orthogonalization procedure.

The two orthogonalization methods just discussed can be combined by considering the decompositions

$$J_{11} = \begin{pmatrix} J_{11}^{*\circ} & J_{11}^{*\circ\circ} \\ J_{11}^{**\circ} & J_{11}^{**\circ\circ} \end{pmatrix}, \quad J_{22} = \begin{pmatrix} J_{12}^\circ & J_{12}^{\circ\circ} \end{pmatrix}, \tag{3.4.22}$$

where

$$
\begin{aligned}
w(Y,\alpha)^{*\circ} &= s_2(Y,\alpha) - \Lambda^{*\circ}(\alpha) s_1^*(Y,\alpha) \in \mathbb{R}^{d_2}, \\
\Lambda^{*\circ}(\alpha) &= J_{21}^{\circ}(J_{11}^{*\circ})^{\pm} \in \mathbb{R}^{d_2 \times d_1^*},
\end{aligned}
\tag{3.4.23}
$$

where $(J_{11}^{*\circ})^{\pm}$ denotes the left or the right inverse of $J_{11}^{*\circ}$, depending on whether d_1° is lower than d_2°, respectively. The statistic $w(Y,\alpha)^{*\circ}$ can be then used instead of $w(Y,\alpha)$ to compute $z^2(Y,\alpha)$, that is Chi-squared distributed with d_2 degrees of freedom under the null.

Orthogonalization is useful when the statistics that are used to estimate the model and the ones that are used to test the null hypothesis are correlated. If the correlation is not important, orthogonalization can reduce the power of the test, because of the new variability introduced in w by the approximation of Λ and by the simulated statistics s_1, is not “related" with whether the null hypothesis is true or false. Therefore, orthogonalizations procedure with less statistics and parameters can increase the power of the test, whereas if all statistics and parameters would be used, the power of the test would also be reduced.

3.4.4 Orthogonalization in generalized method of moments

When the model under the null is estimated with generalized method of moments, s_1 contains more statistics than parameters. The d_1' dimensional statistic s_1' is used under the null to estimate the d_1 dimensional parameter, with $d_1' > d_1$. To derive a test statistic, the $d' = d_1' + d_2$ dimensional statistic s' is simulated under the estimated value, decomposed as $s' = (s_1', s_2)$, and w' is computed similarly to w^* in equation (3.4.20).

There is however a problem with the generalized method of moments estimator $\tilde{\alpha}_L$ when the alternative hypothesis is true. The estimator is the solution of the moment equation between expected and observed value of a linear transformation of the statistics in s_1', determined by the matrix $L \in \mathbb{R}^{d_1 \times d_1'}$ with full rank d_1 that is fixed at the beginning of the algorithm. Under the null, the estimated model is the true one, so all statistics simulated from the estimated value are equal on average to the observed ones. Under the alternative, only a linear transformation of the simulated sufficient statistics is equal on average to the observed ones. In detail, $\hat{\alpha}$ is the generalized method of moment estimator that solves $LE_{\alpha} s_1'(Y) = L s_1'(y)$, by solving the integral on the left hand side with simulation. The expected values of the statistics at $\hat{\alpha}$ under the null and the

alternatives are

$$\begin{array}{lll} H_0 \text{ is true} & LEs_1'(Y(\hat{\alpha})) = Ls_1'(y) & Es_1'(Y(\hat{\alpha})) = s_1'(y), \\ H_0 \text{ is false} & LEs_1'(Y(\hat{\alpha})) = Ls_1'(y) & Es_1'(Y(\hat{\alpha})) \neq s_1'(y), \end{array} \tag{3.4.24}$$

the inequality is due to the fact that the moment equation solves an equation of the transformed (into a lower dimensional space) statistics. This is not a problem under the null because y has been "sampled" from the true model.

The test statistic w is derived from a linear transformation of s_2 and s_1', but the information that is relevant for testing the hypothesis is in s_2. However under the alternative $\Lambda' s_1'$ can become the dominant part of w despite it might contain no information about the hypothesis. Therefore the statistic used is

$$w(Y, y, \alpha)' = s_2(Y, \alpha) - \Lambda'(\alpha)(s_1'(Y, \alpha) - b_1(y, \alpha)) \in \mathbb{R}^{d_2}, \tag{3.4.25}$$

where the vector b_1 of biases is

$$b_1(y) = Es_1'(Y, \alpha) - s_1'(y) \approx \frac{1}{m}\sum_{k=1}^{m} s_1'(Y_k(\alpha)) - s_1'(y) = \hat{b}_1 \in \mathbb{R}^{d_1'}. \tag{3.4.26}$$

The observed test statistic, and the simulated one at iteration k are then computed as

$$\begin{aligned} w' &= s_2(y) - \hat{\Lambda}'(s_1'(y) - \hat{b}_1), \\ w_k' &= s_2(Y_k) - \hat{\Lambda}'(s_1'(Y_k) - \hat{b}_1), \end{aligned} \tag{3.4.27}$$

respectively, where

$$\hat{\Lambda}' = \hat{J}_{21}'(\hat{J}_{11}'^{\top}\hat{J}_{11}')^{-1}\hat{J}_{11}'^{\top} \in \mathbb{R}^{d_2 \times d_1'}. \tag{3.4.28}$$

In the computation of $\hat{\Gamma}'$, $\hat{J}_{12}'$ and the left inverse of $\hat{J}_{11}'$ are used because there are more statistics than parameters.

If not all parameters or statistics are used in the orthogonalization, the methods developed in Section 3.4.3 can be combined with the one developed here when the parameters are estimated using generalized method of moments. The most general case is with the statistic

$$\begin{aligned} &w(Y, y, \alpha)'^{*\circ} = s_2(Y, \alpha) - \Lambda'^{*\circ}(\alpha)(s_1'^{*}(Y, \alpha) - b_1(y, \alpha)) \in \mathbb{R}^{d_2}, \\ &\Lambda'^{*\circ}(\alpha) = J_{21}'^{\circ}(J_{11}'^{*\circ})^{\pm} \in \mathbb{R}^{d_2 \times d_1'^{*}}, \end{aligned} \tag{3.4.29}$$

where the left or the right inverse of $J_{11}'^{*\circ}$ is used depending on the dimensions.

3.4.5 Score tests on models with random effects

For models with random effects such as the ones introduced in Section 3.2.1, there are various hypotheses that are commonly investigated. We always consider models in which under the null hypothesis the random effects are independent, meaning that $\theta \in \mathbb{R}^q$ and $\Gamma(\theta) = \text{diag}(\theta)$.

The null hypotheses that are first considered here are the *irrelevance of a fixed effect*, the *irrelevance of a random effect* (called also of *no-overdispersion of an effect*), or the *irrelevance of an effect*, written as

$$H_0 : \beta_p = 0, \;\; H_0 : \theta_q = 0, \;\; H_0 : \beta_p = \theta_q = 0, \tag{3.4.30}$$

respectively, where we assume without loss of generality that the effect that is tested is the last one (in position p or q). In the last case we assume that β_p and θ_q are the parameters of the same effect (one modelling its average, the other its dispersion), thus for all observations $i \in \{1, ..., N\}$, the last non-zero element of the design z_i is equal to the last value of the covariate x_i, that is $z_{i[j(i)\cdot q]} = x_{ip}$, where $j(i)$ is the individual in observation i. The decompositions for computing the statistic w defined in Sections 3.4.2, are

$$\begin{array}{lccc} & H_0 : \beta_p = 0 & H_0 : \theta_q = 0 & H_0 : \beta_p = \theta_q = 0 \\ s_1 : & (s_{\beta 1}, ..., s_{\beta[p-1]}, s_\theta) & (s_\beta, s_{\theta 1}, ..., s_{\theta[q-1]}) & (s_{\beta 1}, ..., s_{\beta[p-1]}, s_{\theta 1}, ..., s_{\theta[q-1]}), \\ s_2 : & s_{\beta p} & s_{\theta q} & (s_{\beta p}, s_{\theta q}), \end{array} \tag{3.4.31}$$

for the three cases, respectively. Thus the last values of the statistics s_β and s_θ defined in Section 3.2.1 are assumed to contain information about whether the hypotheses should be rejected. For the last hypothesis, the observed statistic w and the simulated ones $\hat{w}_k$ are used with their variance $\hat{\Xi} \in \mathbb{S}_2^+$ to compute the p-values in equations (3.4.16) or (3.4.16). The same method can be used for computing the p-values in the other two cases, but as w is a scalar, the p-values can also be computed as

$$\begin{array}{ll} \tilde{p} = 1 - 2\Phi(|w - \bar{w}|/\hat{\Xi}^{1/2}), & \hat{p} = \sum_k I(|\hat{w}_k - \bar{w}| > |w - \bar{w}|), \\ \tilde{p} = 1 - \Phi((w - \bar{w})/\hat{\Xi}^{1/2}), & \hat{p} = \sum_k I(\hat{w}_k > w), \end{array} \tag{3.4.32}$$

respectively, where Φ is the cumulative distribution function of the standard Gaussian, and for the second hypothesis has been used the fact that the alternative $H_1 : \theta_q > 0$ is unilateral.

The second class of null hypotheses that are considered are of *shared param-*

eters of a fixed effect, defined as

$$H_0 : \beta_p = \beta_{p-1}. \tag{3.4.33}$$

Note that the parametric space under the null has dimension $d-1 = (p-1)+q$. If the parameters under the first null hypothesis are estimated with generalized method of moments, the statistics $s_{\beta[p-1]}$ and $s_{\beta p}$ are both used to estimate the parameters. In this case $s' = (s_1', s_2)$ with $s_1' = (s_\beta, s_\theta) \in \mathbb{R}^d$, and s_2 is defined such that it is large when the null should be rejected. A possibility is to use

$$s_2 = (s_{\beta[p-1]} - s_{\beta p})^2, \tag{3.4.34}$$

that is used together with s_1' to compute w' as in Section 3.4.4, and the p-values are computed as in the bottom row of (3.4.32), as the null is rejected when the simulated statistic is larger. If the parameters under the null are estimated with method of moments, the two statistics $s_{\beta[p-1]}$ and $s_{\beta p}$ have to be combined into one statistic such as $s_{\beta 0} = s_{\beta[p-1]} + s_{\beta p}$, so that $s = (s_1, s_2)$ with $s_1 = (s_{\beta 1}, ..., s_{\beta[p-2]}, s_\theta, s_{\beta 0})$ and s_2 as in equation (3.4.34). The hypothesis $H_0 : \theta_q = \theta_{q-1}$ of *shared parameters of a random effect* is tested in the same way.

The last null hypothesis that is considered is the one of *independence of random effects*

$$H_0 : \Gamma(\theta) = \text{diag}(\theta), \tag{3.4.35}$$

with alternative in which at least two effects are correlated. Suppose for simplicity that $q = 2$, and that the model is estimated with method of moments. The statistic $s_v = (s_{v1}, ..., s_{vn})$ for v and their sample variance $s_\Sigma \in \mathbb{S}_2^+$ are computed as in Section 3.2.1. The statistic for θ under the null is $s_\theta = \text{diag}^{-1}(s_\Sigma) = (s_{\Sigma 11}, s_{\Sigma 22}) \in \mathbb{R}_+^2$, that is the vector of containing the diagonal elements of s_Σ. Then $s = (s_1, s_2)$ contains $s_1 = (s_\beta, s_\theta) \in \mathbb{R}^{p+q}$ that is used to estimate the parameters, and that the statistic s_2 that it is assumed to contain information about the null hypothesis is $s_2 = s_{\Sigma 12} \in \mathbb{R}$, that is the out-diagonal element of s_Σ. The null hypothesis is rejected when the observed value for this statistic is very far from 0, in comparison to the simulated values, so a bilateral p-value is used.

3.5 Poisson model with random effects and softplus link

3.5.1 Model

The model that is used in this section is

$$\begin{aligned} &Y_i \sim \text{Pois}(\mu_i) \in \mathbb{N}, \\ &\mu_i = \text{softplus}(\eta_i) = \log(1+\exp(\eta_i)) \in \mathbb{R}_+, \\ &\eta_i = x_i^\top \beta + z_i^\top \nu \in \mathbb{R}, \\ &\nu = (I_n \otimes \Gamma(\theta))u \sim \text{Norm}(0, (I_n \otimes \Gamma(\theta))(I_n \otimes \Gamma(\theta))^\top) \in \mathbb{R}^{qn}, \end{aligned} \tag{3.5.1}$$

for $i \in \{1, ..., N\}$ with fixed parameters

$$\beta \in \mathbb{B} \subseteq \mathbb{R}^p, \quad \Gamma(\theta) \in \mathbb{G} \subseteq \mathbb{R}^{q\times q}, \quad \theta \in \mathbb{T} \subseteq \mathbb{R}^r. \tag{3.5.2}$$

The *softplus link* is used because it has better property than the *canonical link* $\eta_i \mapsto \exp(\eta_i)$ and the *identity link* $\eta_i \mapsto \max(\eta_i, 0)$. These three links are plotted in black, red and green, respectively, in the left plot of Figure 3.3. The softplus link in the Poisson regression have been studied extensively in [74]. The advantage of the softplus with respect to the identity link is that the former is differentiable in its domain, so the optimization is simplified. The advantage with respect to the canonical link is that the softplus is more stable for large values, the advantage is significant when the moment equation is solved stochastically, as small variations of β and especially of θ are amplified significantly by the exponential.

In the middle plot of Figure 3.3, the histogram of the dependent variable of one simulated dataset from the model (3.5.1) is plotted. The simulated dataset contain two covariates and two independent random effects, the values of the parameter from which the dataset is sampled are $\beta_1 = 10$ and $\beta_2 = -0.3$ for the two covariates, and the standard deviations of the random effects are $\theta_1 = 2$ and $\theta_2 = 1$. In the right plot of Figure 3.3, the cumulative distribution function of the response variable is plotted for different simulated datasets. The black curves are for three simulations from the model with parameter $(\beta, \theta) = (10, -0.3, 1, 1)$ whereas for the red and the green curves, all parameters are the same, except for θ_1 that is equal to 4 and 8, respectively.

To approximate the matrix of derivatives $J(\alpha)$, where $\alpha = (\beta, \theta)$, it is neces-

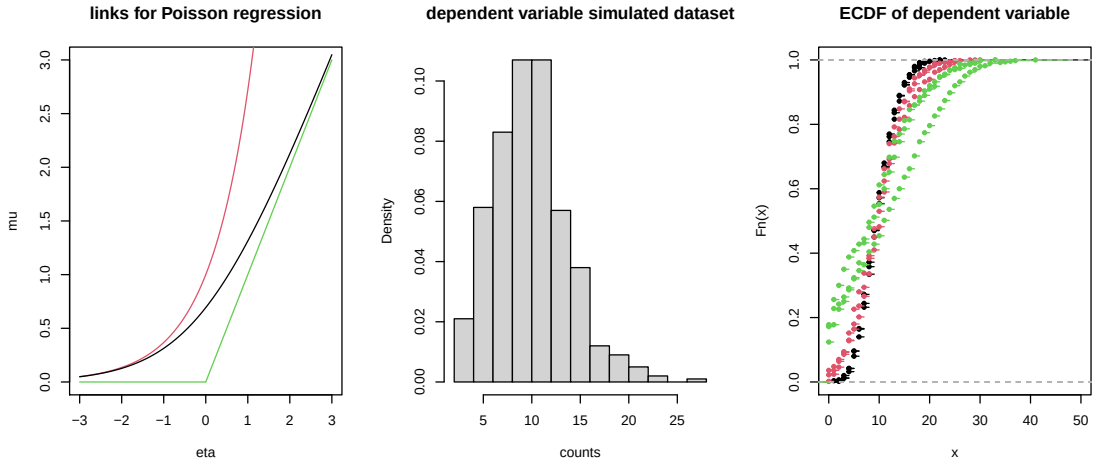

Figure 3.3: Left: softplus (black), exponential (red) and identity (green) link functions. Center: histogram of dependent variable simulated from model (3.5.1). Right: cumulative distribution functions of dependent variable simulated from different values of the parameter θ_1, black, red and green for θ_1 equal to 1, 4 and 8 respectively.

sary to compute the score function

$$\frac{\partial}{\partial(\beta^\top\ \theta^\top)}\log f(y|\beta,\theta) = \left(\frac{\partial \log f(y|\mu)}{\partial \mu^\top}\frac{\partial \mu(\eta)}{\partial \eta^\top}\frac{\partial \eta(\beta,\nu)}{\partial(\beta^\top\ \nu^\top)}\cdot \right. \\ \left. \cdot\frac{\partial[(\beta,\theta)\mapsto(\beta,\nu)]}{\partial(\beta^\top\ \theta^\top)}\right) \in \mathbb{R}^{N\times(p+r)}, \tag{3.5.3}$$

where the first three derivatives are

$$\begin{aligned}
\frac{\partial \log f(y|\mu)}{\partial \mu^\top} &= \operatorname{diag}((y_i-\mu_i)/\mu_i)_i \in \mathbb{R}^{N\times N},\\
\frac{\partial \mu(\eta)}{\partial \eta^\top} &= \operatorname{diag}(\operatorname{expit}(\eta_i))_i \in \mathbb{R}^{N\times N},\\
\frac{\partial \eta(\beta,\nu)}{\partial(\beta^\top\ \nu^\top)} &= (X\ \ Z) \in \mathbb{R}^{N\times(p+qn)},
\end{aligned} \tag{3.5.4}$$

with $\operatorname{expit}(\eta_i) = \exp(\eta_i)/(1+\exp(\eta_i))$. As $\theta \mapsto \nu$ does not depend on β, the last

derivative is

$$\frac{\partial[(\beta,\theta)\mapsto(\beta,v)]}{\partial(\beta^\top\ \theta^\top)}=\begin{pmatrix} I_p & O_{p\times r}\\ O_{qn\times p} & \partial v/\partial\theta^\top\end{pmatrix}\in\mathbb{R}^{(p+qn)\times(p+r)}. \tag{3.5.5}$$

We consider the simple case with independent random effects, in which $r=q$ and $\theta\in\mathbb{R}^q_+$ with

$$\Gamma(\theta)=\operatorname{diag}(\theta)\in\operatorname{diag}(\mathbb{R}^q_+)\cong\mathbb{R}^q_+. \tag{3.5.6}$$

Then

$$\frac{\partial v}{\partial\theta^\top}=\frac{\partial(I_n\otimes\Gamma(\theta))u}{\partial\theta^\top}=\begin{pmatrix}\operatorname{diag}(u_i)_{1\le i\le n}\\ \dots\\ \operatorname{diag}(u_i)_{((n-1)q+1)\le i\le qn}\end{pmatrix}\in\mathbb{R}^{qn\times q}. \tag{3.5.7}$$

as $v_i=\theta_j u_i$ for $i=(k-1)q+j$, $k\in\{1,\dots,n\}$, $j\in\{1,\dots,q\}$.

If the number of fixed and random effects is p and q respectively, but the parameter is $(\beta_0,\theta_0)\in\mathbb{R}^{p_0}\times\mathbb{R}^{q_0}_+$ with $p_0\le p$ and $q_0\le q$, the fixed parameter (β,θ) from which the model is simulated is computed with

$$[(\beta_0,\theta_0)\mapsto(\beta,\theta)]:\mathbb{R}^{p_0}\times\mathbb{R}^{q_0}_+\to\mathbb{R}^p\times\mathbb{R}^q_+. \tag{3.5.8}$$

whose derivative is

$$\frac{\partial[(\beta_0,\theta_0)\mapsto(\beta,\theta)]}{\partial(\beta_0^\top\ \theta_0^\top)}=\begin{pmatrix}\partial\beta/\partial\beta_0^\top & \partial\beta/\partial\theta_0^\top\\ \partial\theta/\partial\beta_0^\top & \partial\theta/\partial\theta_0^\top\end{pmatrix}\in\mathbb{R}^{(p+q)\times(p_0+q_0)}. \tag{3.5.9}$$

In all cases that we consider, $\partial\beta/\partial\theta_0^\top=O_{p\times q_0}$, $\partial\theta/\partial\beta_0^\top=O_{q\times p_0}$, and

$$\frac{\partial\beta}{\partial\beta_0^\top}=\left\{I_p \text{ if } \beta=\beta_0,\ \begin{pmatrix}I_{p_0}\\ e_{p_0}^\top\end{pmatrix} \text{ if } \beta=(\beta_0,\beta_{0p_0}),\ \begin{pmatrix}I_{p_0}\\ 0_{p_0}^\top\end{pmatrix} \text{ if } \beta=(\beta_0,0)\right\}\in\mathbb{R}^{p\times p_0}, \tag{3.5.10}$$

where $e_{p_0}=(0,\dots,0,1)\in\mathbb{R}^{p_0}$. The three cases are the standard one, the one in which the last two parameters β_p and β_{p-1} are shared, and the one in which the last parameter β_p is 0, respectively. Analogously for computing $\partial\theta/\partial\theta_0^\top$, the equations above are the same with β and p replaced by θ and q respectively. The last case is used when the model is estimated under the null hypothesis in which the last component (of β, of θ, or of both) is 0. The second case is used either when estimating the model under the hypothesis in which the last two component are shared, or when the model is estimated with generalized method of moment, so that the last two statistics for β (or for θ) are both used to estimate

β_{p-1} (or θ_{q-1}).

3.5.2 Simulation study

In this section a simulation study is used to evaluate the estimation method and different orthogonalization procedures for datasets generated from the model (3.5.1) with independent random effects. The simulated datasets have $n = 30$ individuals and $N = 500$ observations. The number of covariates is $p = 3$ and each individual has $q = 2$ random effects. The first covariate is the intercept, the two other contain correlated Gaussian random values, and the two non-zero values for the design are the same of the first two covariates. For the simulation, 100 datasets generated from the real model are used.

The hypothesis considered is

$$H_0 : \beta_3 = 0, \tag{3.5.11}$$

the model is estimated with method of moments, with statistics as in Section 3.2.1. The dataset are generated under the null hypothesis from the real parameter $\alpha = (\beta, \theta) \in \mathbb{R}^{p_0} \times \mathbb{R}_+^2$ with $p_0 = 2$, $\beta = (10, -.3)$, and with standard deviations for the random effects $\theta = (2, 1)$.

For each simulated dataset the starting point of the optimization is equal to the real parameter under the null, plus some added noise. The invertible matrix $L \in \mathbb{R}^{p_0 \times p_0}$ is estimated at the starting point using the method is Section 3.3.2, using 300 simulations at the starting point, the matrix L it is never updated in the optimization. For method of moments it is not required to approximate L, as every other invertible matrix would not change the solution of the moment equation. However, it is advantageous to use this matrix for numerical stability. To estimate the parameter, 1600 iterations consisting in a simulation from the current value of the parameter, and an update depending on the difference between simulated and observed statistics. The learning rate is kept constant equal to $\epsilon = 6/N = 0.012$, and the preconditioning is updated with RMSProp as in (3.3.6) with $\gamma = 0.9$. The first 400 iterations are excluded as “burn-in", so that 800 values of the parameter are averaged to compute the estimator. At the estimated value, 1000 simulations are used to compute the matrices needed for the model evaluation procedure and to test the hypothesis. Here the statistic that is assumed to contained information about β_3, and so that is used as test statistic, is also simulated.

For computing the p-values of the test (3.5.11), three different orthogonalization procedures are used. The first one is the “basic" case in which there is

no orthogonalization, meaning that the statistic associated with β_3 is used “directly", so the statistics that are used to estimate the parameters under the null are not used “orthogonalize" the test statistics. In the second case, all parameters (under the null) and statistics are used in the orthogonalization, as described in Section 3.4.2. Finally we consider the procedure in which only one statistics (but with all parameters) is used in the procedure, so the test statistic is w^* as defined in Section 3.4.3. The only statistic that is used in the orthogonalization is the one associated with β_2, because the second and the third covariate are correlated, so the second statistic is likely to be correlated with the test statistic.

We have considered two cases under the alternative, in the first one $\beta_3 = -1$, in the second one $\beta_3 = -2.5$. Some of the simulated datasets have been excluded because of convergence issues in the estimation. This has been done using the method described in Section 3.3.3, where the sign of the differences between simulated and observed statistics is used to check if the process has reached stationarity after the “burn-in" period In particular , under the null and under the two alternatives 78, 78 and 76 datasets are kept, respectively. The cumulative distribution functions of the p-values under the null, and under the two alternatives considered are plotted in Figure 3.4. The black curves are ones of the p-values without orthogonalization, in the red curves all statistics are used to orthogonalize the test statistic, and in the green curves only the second statistic is used. The latter approach is (slightly) the best one, whereas the orthogonalization using all statistics reduces the power of the test. The reason is that orthogonalization is useful only when the test statistic is correlated with the ones that are used in estimation, and only the second one has a significant correlation with the test statistic.

3.6 Conclusion

The purpose of this chapter was mainly to study algorithms similar to the ones used in the SAOM with random effects, in different set-ups. The original idea was to link the theory about stochastic gradient descent developed mostly in machine learning, with the theory of moment- and simulation-based statistical inference, by considering optimizations in which the stochasticity in the gradient or in its approximation, is due to the simulation of the generative process, in each iteration.

The approximations of the derivative of the expected statistics with respect to the fixed parameters has been studied in detail. The approximation of derivatives is based on the multivariate chain rule that is used to “break" the score function

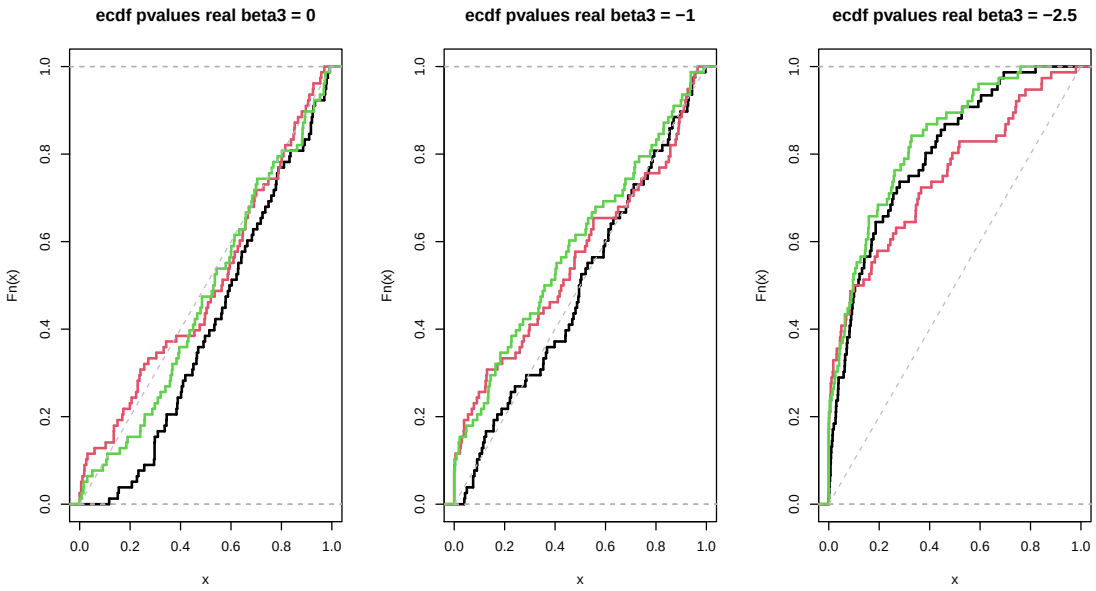

Figure 3.4: Cumulative distribution functions of p*-values under the three orthogonalization procedures: black, red, and green for orthogonalizations using none, all, and only the second statistic, respectively.*

into multiple components (functions that compose to the score), one for each “step" of the simulation. In models with random effects, some of these components are stochastic, as they depend on the sampling of the random parameters.

The theory developed can be used both with the “standard" and the generalized method of moments, focusing on the important differences between them in the estimation and especially in the model evaluation phases. In the method of moments the choice of the matrix L is relevant only for the stability of the algorithm, whereas in the generalized method of moments L determinates also the covariance of the estimator. Approximations of a good L are based on approximations of the derivative of the expected statistics with respect to the parameters, that are also used to approximate the matrix Λ that is used to orthogonalize the test statistic to increase its power when the statistic is correlated with the ones used to estimate the parameters. In this context there is a major difference between the “usual" and the generalized method of moments, because in the latter case it is not guaranteed that the statistics that are used in estimation are equal on average (at the estimated parameter) to the observed ones. This issue must be taken into account when orthogonalizing the test statistic, otherwise the orthogonalized statistic will be influenced too significantly by the statistics used in

estimation, even though they might contain none or few information about the hypothesis that is tested.

Different orthogonalization procedures have been compared with a simulation. We found that it is convenient to use only the estimating statistics that are significantly correlated with the test ones, because the variability in the simulation of the estimating statistics and in the approximation of the matrix Γ reduces the power of the test when these statistics are not correlated with the test ones.

Chapter 4

Efficient implementation of sets and multisets in R using hash tables

4.1 Introduction

Sets are the most basic and fundamental containers of objects in mathematics. According to set theory (almost) all objects in mathematics are, or can be described as, sets. Some objects have additional mathematical and computational structure, such as multisets, lists, vectors, stacks, etc. Sets and multisets have been less developed in programming languages than other objects, such as vectors and lists. The reason is that the latter ones are used very frequently in algorithms, so almost all programming languages have a built-in implementation of them. These implementations do not reflect the mathematical derivation of these objects from set theory, as their structure allows the use of much more computationally efficient implementations and algorithms. Sets and multisets as basic containers are nevertheless very important, especially for discrete probabilistic and statistical models. The aim of the paper is to provide an efficient implementation for algorithms that use sets and multisets as containers. Our implementation is based on the `hash` package [11]. It is efficient because the hash table data structure allows search, insertion and deletion of one element in the table in constant time.

The mathematical definition of sets and multisets is given in Section 4.2, where our implementation and its semantic is also discussed. In Section 4.3, relations and operations between sets and multisets are defined mathematically, and their implementation is described. The performance of our implementation is discussed in Section 4.4. In Section 4.5 an application of our implementation of sets and multisets as states of a Markov chain is provided.

4.2 Sets and multisets

Sets, multisets and some other constructions/containers derived from them, are introduced mathematically in Section 4.2.1. The emphasis is on how containers differ on how much structure is "imposed" on them. In sets elements are either in or out of them, in multisets it is also relevant how frequently an element is contained, for sequences the order of the elements is also important. The R package `hset` is introduced in Section 4.2.2, where it is also discussed which objects can be included as elements, and how they are stored in a data structure based on the package `hash`. The semantic of `hset` objects is described in Section 4.2.3, with some other functions that are used to control these objects, that depend on the chosen semantic.

4.2.1 Mathematical definition

Sets are defined as collections of objects called *elements*, or *members*. *Set theory* can be used as foundation of mathematics. The existence of the *empty set* $\emptyset = \{\}$ is postulated, $\emptyset$ is the only element such that

$$\{a, \emptyset\} = \{a\}, \ \forall a, \tag{4.2.1}$$

where equality between sets will be formally defined in Section 4.3.1. All elements of a set are considered to be sets themselves, and (almost) all objects in mathematics are constructed as sets.

An important example is the set $\mathbb{N}$ of *natural numbers* that can be defined recursively as

$$0 = \emptyset, \ 1 = \{0\} = \{\emptyset\}, \ 2 = \{0, 1\} = \{\emptyset, \{\emptyset\}\}, \ \ldots \tag{4.2.2}$$

Another important example is the *Cartesian product* of two sets, that is the set of ordered pairs with first element from X_1 and second element from X_2, defined as

$$X_1 \times X_2 = \{X_1, \{X_2\}\}, \tag{4.2.3}$$

which contains elements (a_1, a_2), for $\emptyset \neq a_1 \in X_1$ and $\emptyset \neq a_2 \in X_2$. Note that $\emptyset \times X = X \times \emptyset = \{\emptyset\}$ for all X, and that $(a_1, a_2) \neq (a_2, a_1)$ for $a_1 \neq a_2$ (the Cartesian product is not *commutative*).

Order and multiplicity of the elements of a set is not defined, that is

$$\{a, b\} = \{b, a\} = \{a, a, b\}, \ \forall a \neq b. \tag{4.2.4}$$

Multisets are defined as collections of elements with *multiplicities*, that are non-negative numbers. We will always assume that the multiplicities of the elements are finite. The order of the elements is not defined, but multisets of elements with different multiplicities are different:

$$\{a[m], b[n]\} \neq \{a[n], b[m]\},\ \forall n \neq m,\ \forall \emptyset \neq a \neq b \neq \emptyset \tag{4.2.5}$$

If the multiplicity of an element is 0, then the element is not contained in the multiset:

$$\{a[m], b[0]\} = \{a[m]\},\ \forall a, b. \tag{4.2.6}$$

A set can be considered equivalent to a multiset of the same elements, all with multiplicity 1:

$$X = \{a, b\} \cong \{a[1], b[1]\} = Y(X),\ \forall a, b, \tag{4.2.7}$$

The injective function $X \mapsto Y(X)$ converts a set to the "equivalent" multiset, but when applied to a multiset, it is the identity function: $Y = Y(Y)$. The surjective function $Y \mapsto X(Y)$ maps the multiset Y into its *support*, that is the set $X(Y)$ of its elements, if X is a set $X = X(X)$. The size of a multiset (or of a set), is the number of its elements, that is

$$\text{size}(Y) = |Y| = |X(Y)| \in \mathbb{N}. \tag{4.2.8}$$

The cardinality of a multiset $Y = \{a_i[m_i]\}_i$ is the sum of its multiplicities, that is

$$\text{card}(Y) = ||Y|| = \textstyle\sum_i m_i \in [0, \infty), \tag{4.2.9}$$

while for a set X, $\text{size}(X) = \text{card}(X)$.

Other constructions that will be used later on are based on the Cartesian product, that is used to define *powers* of the set X as

$$X^k = X \times X \times ... \times X, \tag{4.2.10}$$

for $k \in \mathbb{N}$, where X is repeated k times in the right hand side, $X^0 = \emptyset$ and $X^1 = X$. Powers of X are used themselves to define finite dimensional *sequences* (or *strings*) of elements in X as

$$A = (a_1, ..., a_l) \in X^* = \biguplus_{k \geq 0} X^k, \tag{4.2.11}$$

so a sequence of length l with elements in X is an element of X^l, and the set of

sequences X^*, that is the *disjoint union* of all powers, contains all sequences of all finite possible lengths, including the *empty sequence* (). Sequences have even more structure than multisets, as they are ordered, meaning that $(a_1, a_2, ...) \neq (a_2, a_1, ...)$ for $a_1 \neq a_2$.

Finite dimensional sequences are introduced because they are one of the most used objects in programming, so most languages have efficient built-in implementations of sequences with finite length. However these implementations are often inefficient when a sequence is modified locally by operations of inclusion or removal of elements from the list.

In R finite dimensional sequences are implemented as objects of type `vector`, with sub-types `atomic` and `list` [73, Chapter 3]. In the next sections, our R implementation of sets and multisets is discussed in detail.

4.2.2 Computational implementation

In mathematics, elements of sets are usually considered sets themselves. Then a formal definition of elements is redundant, once sets are defined. On the other hand, when sets are viewed as computational objects, a definition of elements is required because the elements are objects with, possibly various, data types stored in memory.

Sets and multisets are implemented in the *R package* `hset`, as objects of *S4 class* `"hset"`. These objects are containers of elements, that are either numbers, or sets of numbers. A more formal definition of objects that are valid elements will soon be given. An object of class `"hset"` contains two *slots*. The main one, called `@htable`, is an hash table from the package `hash` [11]. The second slot, called `@info`, is of class `"environment"`, which contains a Boolean value that distinguish sets from multisets. The reason why the second slot is a "trivial" (with one object) environment, rather than an object of class `"logical"`, is because environments and (logical) vectors have a different semantic, then it would be difficult to reason about "sameness" of objects (sets or multisets). The constructor for objects of class `"hset"` will be described at the end of this section, the semantic of our implementation will be discussed in detail in the next section.

The set `S` of possible sets that can be stored is recursively defined as

$$\begin{aligned} &X = \{a_1, a_2, ...\} \in \mathtt{S}, \\ &a_i \in \mathtt{S} \uplus \mathtt{N}, \end{aligned} \tag{4.2.12}$$

so the element a_i can be either a set, or a value in `N` that is the set of `numeric` vectors of length 1, without the values `Inf`, `-Inf`, `NaN`, `NA`, `NA_integer_` and

`NA_real_`, that are excluded. The symbol $\uplus$ in $\mathtt{S} \uplus \mathtt{N}$ is used to denote the disjoint union between the sets `S` and `N`. The inclusion relation $\in$ between an element, and a set or multiset, will be formally defined in Section 4.3.1.

The set `M` of possible multisets that can be stored is recursively defined as

$$\begin{aligned} &Y = \{a_1[m_1], a_2[m_2], ...\} \in \mathtt{M}, \\ &a_i \in \mathtt{S} \uplus \mathtt{N}, \\ &m_i \in \mathtt{N}_+, \end{aligned} \tag{4.2.13}$$

where `S` and `N` are defined as above, and $\mathtt{N}_+ \subset \mathtt{N}$ is the subset of values of `N` that are strictly positive. Note that in our current implementation multisets can not be elements of a set or a multiset, and that recursion occurs in the definition of `M` because it occurs in the definition of `S`. The `numeric` datatype includes `integer` and `double` as subtypes, so the elements can be also of these two types. Vectors of type `numeric` of length 0, together with the `NULL` object, are considered equivalent to the empty set, so they are not included in an object of class `"hset"` (even though formally an empty set is included in every set). Instead, `numeric` values of length at least 2 and `list` values of every length are converted to elements of `S`, that is to sets, before being included as elements.

The package `sets` [40] uses a different approach, the sets in this library, that can be of classes `set` (sets), `gset` (*generalized* sets), `cset` (*customizable* sets), can contain elements of every type. Two elements can be considered the same only if they have the same class, so for example the sets {2, 2L} and {2}, are not equal (the former has two elements), as 2 and 2L have class `"numeric"` and `"integer"` respectively. As a result, although our implementation is more limited because we only take into account sets and multisets of numbers, it is still somewhat closer to the mathematical definition in which 2L and 2 represent the same number.

The *hash table* implemented in `hash` is a data structure that contains *key-value pairs*. The *keys* are different objects of type `character`, they are unique labels of pointers to the *values*, that can be objects of every type except `NULL`. The advantage of using an hash table to implement sets and multisets is that various operations that are: adding and removing a key-value pair from the table, checking to see if a key is present in the table, and returning the value associated with a key; only require on average a constant number of elementary operations.

There is an injection $k : \mathtt{S} \uplus \mathtt{N} \to \mathtt{C}$, where `C` is the set of character vectors of length 1, that is used as set of keys to label uniquely each possible element of a set or a multiset. For example, the element

$$a_i = \{\texttt{-1}, \texttt{1}, \texttt{1L}, \{\}, \texttt{2}, \texttt{11}, \{\texttt{2}, \{\texttt{3}\}\}\}, \tag{4.2.14}$$

is mapped to

$$k_i = k(a_i) = \texttt{"\{-1,\{\},\{2,\{3\}\},1,11,2\}"}, \tag{4.2.15}$$

where the "sub-elements" `1` and `1L` are both mapped to `"1"`, and the components of the character k_i are in *lexicographic order*, which has the mathematical properties of a *total order*. Note that the order is important to guarantee that k is an injection. For sets, all values of the hash table are equal to the empty character `""`, that is $v(k_i) =$ `""` for all i, whereas for multisets $v(k_i) \in \mathbb{N}_+$.

Sets and multisets are created with the *constructor* `hset` with three arguments that are `members`, `multiplicities` and `generalized`. The first two arguments are `NULL` by default (empty set), the last one is `FALSE` by default. If the second input is not `NULL`, `generalized` is set to `TRUE` if it was not the case. Then it is checked that `members` and `multiplicities` are of the correct type, that they are coherent with themselves, and then they are included in the hash table. The function `is.hset`, with input `x`, returns `TRUE` when `x` is of class `"hset"`, and `FALSE` otherwise. The function `as.hset`, with input `x`, return `x` itself if it is of class `"hset"`, otherwise it applies the constructor `hset`, with `members` equal to `x`, `multiplicities` and `generalized` as default, so that the function creates a set with elements taken from `x`.

Size and cardinality defined in equations (4.2.8) and (4.2.9) are returned by the functions `size.support` and `cardinality`. A vector containing the labels of the elements is returned by the function `members`, while the vector of multiplicities of the elements are returned by the function `multiplicities`. The only input of these four functions is an object of class `"hset"`. The components of the vectors obtained by the last two functions are coherent, so that the i-th value of the vector is the multiplicity of the i-th element. If the function `multiplicities` is used on a set, a vector with all values equal to 1 is returned.

4.2.3 Semantic of hset

In R, objects can be accessed and modified with *reference*, or with *value semantic* (the two alternatives are described in Appendix A.3). In R objects of most classes have value semantic, but environments and hash tables from the package `hash` have reference semantics. An object of class `"hset"` contains a `hash` object in the first slot, and an `environment` object in the second. When a set is copied "directly", both slots of the copied object, refer to the same ones of the original object. The functions `clone.of.hset` and `refer.to.hset` are used to copy an `hset` with value and reference semantic, respectively.

The function `is.generalized`, with logical returned value, is used to dis-

tinguish sets and multisets. The map `as.generalized` transforms a set to a multiset by converting the values of all elements of the hash table to `1L`, while `as.not.generalized` transforms a multisets to a set by converting all values to `""`. These functions implement $X \mapsto Y(X)$ and $Y \mapsto X(Y)$ that were defined mathematically in Section 4.2.1. The sets are modified locally, i.e., with reference semantic, so if a multiset is transformed to a set the information about the multiplicities of the multiset that is passed as input is lost. The functions `clone.of.hset` and `refer.to.hset` can also be used to convert sets to multisets, and vice versa. They have as second argument the (empty, or logical) value called `generalized`, that is `NULL` by default, but it can be used to convert a multiset to a set, or a set to a multiset, when it is copied. Applying `refer.to.hset` with second argument equal to `TRUE` (respectively `FALSE`), is equivalent to applying the function `as.generalized` (respectively `as.not.generalized`). Whereas the application of `clone.of.hset` with second argument equal to `TRUE` or `FALSE`, creates a new `hset` with the same support as the original one, but with the multiplicities that are converted to `1L` or `""` in the two cases, and the `hset` that is passed as input of `clone.of.hset` is not modified.

In the next section relations and operations between sets and multisets will be described mathematically and computationally. In this section we describe how the chosen semantic can affect the computation of an operation. Relations are encoded as functions with Boolean codomain, so if two components that can be elements, sets or multisets, are in relation, the function returns `TRUE`, otherwise `FALSE`. As sets are not modified when checking whether two components are in relation, the semantic is irrelevant. Conversely for operations between sets or multisets, even though the result of the operation is not changed by the semantic used, one operand will be modified when reference semantic is used, while a new `hset` with the result of the operation is created. The operands do not change when value semantic is used.

All operations are computed with the function `hset.operation.numeric` if at least one of the operands is a multiset, otherwise `hset.operation.logical` is used. These two functions have the same signature in which the output is an object called `new.hset` of class `"hset"`. The first input `hset1` is the first operand, `...` contains all other operands (for operations with multiple arity), the arguments `operation` (function) and `identity.is.universe` (logical value) completely specify how the operands are combined. The last input `semantic`, that can be equal to `"refer"` (default) or `"value"`, specifies the semantic. The difference between the numeric and the logical operation is in how the multiplicities are handled. In the latter case we define the multiplicities of a set by the bijection that maps `NULL` to `FALSE` and `""` to `TRUE`, where the Boolean outcomes are used

to evaluate the operation. The evaluation is stored using the inverse function, as setting an element to `NULL` in an `hash` object is equivalent to removing the key-value pair from the hash table, or to not doing anything if the pair is not present. If the numeric operation is used, the multiplicities that are used in the operation are obtained by the surjective function of type $\texttt{NULL} \uplus \texttt{""} \uplus \texttt{N}_+ \rightarrow \texttt{N}_{+=}$, s.t. $\texttt{NULL} \mapsto \texttt{0}$, $\texttt{""} \mapsto \texttt{1}$, and $\texttt{m} \mapsto \texttt{m}$ for $\texttt{m} \in \texttt{N}_+$. The non-negative values are then used as operands, and the result is stored in the hash table with the bijection $\texttt{N}_{+=} \rightarrow \texttt{NULL} \uplus \texttt{N}_+$, s.t. $\texttt{0} \mapsto \texttt{NULL}$, and $\texttt{m} \mapsto \texttt{m}$ for all $\texttt{m} \in \texttt{N}_+$. Note that the output of a numeric operation is always a multiset, so `""` is never stored.

The function `create.new.hset` is used in `hset.operation.numeric` and `hset.operation.logical`, to create the object `new.hset` that will store the result. When reference or value semantics are used, `refer.to.hset` or `clone.of.hset` are used respectively, inside `create.new.hset`, with argument `hset1`. Therefore, `new.hset` and `hset1` will refer to the same object in memory with reference semantic, so that when the result is computed, will be stored both in `new.hset`, and in `hset1`. Whereas with value semantic, `new.hset` will be a reference to an object in memory that is a clone of `hset1`, so the latter will not change when the result is computed. The reference semantic is used by default for its computational advantages. In particular when the identity element of the operation is the empty set, that is when `identity.is.universe` is `FALSE`, computing the result does not require a complete scan through the elements of `hset1`.

The computational complexity of an operation between multisets Y_1, Y_2, ..., Y_a, where $\emptyset$ is the identity element of the operation, is $O(|Y_2|+...+|Y_a|)$ with reference semantic, and $O(|Y_1|+|Y_2|+...+|Y_a|)$ with value semantic. The advantage is significant when $|Y_1| \gg \max_{j \neq 1} |Y_j|$. Note that difference of $O(|Y_1|)$ operations between the two semantics, is due to the necessity of copying the first operand. However, if the first operand is a set, while some of the others are multisets, there is no difference between the two semantics, because the result of the operation is a multiset, so $O(|Y_1|)$ operations are required to convert the first operand to a multiset, even when reference semantic is used.

4.3 Sets algebra

Relations and operations involving `hset` objects are described in Sections 4.3.1 and 4.3.2, respectively. Relations of different types are described by their signature, definition, and implementation as functions with Boolean codomain. Operations are also described in the same way, but the functions that compute them return sets or multisets.

4.3.1 Relations

Inclusion of elements. The *inclusion* relation between an element and a set is defined mathematically as

$$\in : (\mathtt{N} \uplus \mathtt{S}) \times \mathtt{S}, \quad a \in X \iff X = \{a, ...\}, \tag{4.3.1}$$

meaning that the element $a \in (\mathtt{N} \uplus \mathtt{S})$ and the set $X \in \mathtt{S}$ are related by $\in$ if and only if a is an element of X. Note that $\emptyset \in X$ for all sets X, and that the symbol : is used in the signature of the relation to avoid the notation $\in\in (\mathtt{N} \uplus \mathtt{S}) \times \mathtt{S}$. This relation is extended trivially to multisets as

$$\in : (\mathtt{N} \uplus \mathtt{S}) \times \mathtt{M}, \quad a \in Y \iff Y = \{a[n], ...\},\ n \geq 1, \tag{4.3.2}$$

that is if the multiplicity of $a \in (\mathtt{N} \uplus \mathtt{S})$ in $Y \in \mathtt{M}$ is at least 1.

The straightforward generalization for multisets are relations between an element with a given multiplicity, and a multiset. Three relations of this type are defined as

$$\in_\sim : ((\mathtt{N} \uplus \mathtt{S}) \times \mathtt{N}_+) \times \mathtt{M}, \quad a[m] \in_\sim Y \iff Y = \{a[n], ...\},\ m - n \sim 0, \tag{4.3.3}$$

where $\sim$ can be $\leq$, $<$ and $=$, for the relations of, inclusion $\in = \in_\leq$, *strict inclusion* $\in_<$ and *exact inclusion* $\in_=$, respectively. Intuitively, in the three cases, $a[m]$ and Y are related if a in Y has a multiplicity greater or equal, greater, and equal to m respectively. Instead of defining relations between an element with a given multiplicity and a multiset, we could have equivalently defined the family of relations between elements and multisets parametrized by $m \in \mathtt{N}_+$:

$$\in^m_\sim : (\mathtt{N} \uplus \mathtt{S}) \times \mathtt{M}, \quad a \in^m_\sim Y \iff Y = \{a[n], ...\},\ m - n \sim 0, \tag{4.3.4}$$

so that, for all $a \in \mathtt{N} \uplus \mathtt{S}$, $Y \in \mathtt{M}$, and $m \in \mathtt{N}_+$,

$$a[m] \in_\sim Y \iff a \in^m_\sim Y. \tag{4.3.5}$$

All relations defined above can be encoded as one function with signature

$$((\mathtt{N} \uplus \mathtt{S}) \uplus ((\mathtt{N} \uplus \mathtt{S}) \times \mathtt{N}_+)) \times (\mathtt{S} \uplus \mathtt{M}) \times \{\leq, <, =\} \to \{\mathtt{TRUE}, \mathtt{FALSE}\}, \tag{4.3.6}$$

where the first argument is either an element, or a pair between an element and a multiplicity, the second argument is either a set or a multiset, whereas the last argument specifies the type of relation. The function returns TRUE if the first two

arguments are in relation, of the type specified by the third argument. In the package, a similar function, called `inclusion.member`, has signature

$$\mathtt{C} \times (\mathtt{S} \uplus \mathtt{M}) \times \mathtt{N}_+ \times \{\leq, <, =\} \to \{\mathtt{TRUE}, \mathtt{FALSE}\}. \tag{4.3.7}$$

The last two arguments, called `multiplicity` and `type.relation` with default values `1` and $\leq$ respectively, are ignored when the second argument is a set. The first argument, called `member` must be a vector of length 1, that is converted to a character `C` inside the function, if this value is not a valid element, as defined above, the function returns `FALSE` for every possible choice of the last two arguments. Then, `inclusion.member` returns `TRUE` if and only if the first two arguments are in relation specified by the last two arguments. The binary operator `%in%` uses `inclusion.member` where the last two arguments are set as default, so that it evaluates the relations (4.3.1) or (4.3.2), depending on whether the second argument is a set or a multiset respectively. If the first argument, i.e., the left operand of `%in%`, is a vector of characters, the operand returns a vector of Booleans with the result of the evaluated relation for each character.

Subsets and equalities. Now relations in which both objects are sets or multisets are considered. The *subset relation* between sets X_1 and X_2 is

$$\subseteq: \mathtt{S} \times \mathtt{S}, \quad X_1 \subseteq X_2 \iff (a \in X_1 \implies a \in X_2), \tag{4.3.8}$$

and the *strict subset relation* is

$$\subset: \mathtt{S} \times \mathtt{S}, \quad X_1 \subset X_2 \iff (X_1 \subseteq X_2 \text{ and } \exists b \in X_2 \text{ s.t. } b \notin X_1). \tag{4.3.9}$$

The *equality relation* between sets is defined as

$$=: \mathtt{S} \times \mathtt{S}, \quad X_1 = X_2 \iff (X_1 \subseteq X_2 \text{ and } X_2 \subseteq X_1). \tag{4.3.10}$$

For finite dimensional sets, the last two relations can also be written as

$$X_1 \approx X_2 \iff X_1 \subseteq X_2 \text{ and } |X_1| \sim |X_2|, \tag{4.3.11}$$

where for the strict inclusion, $\approx$ and $\sim$ are replaced by $\subset$ and $<$ respectively, while for the equality relation $\approx$ and $\sim$ are both replaced by $=$.

The relations above will be generalized in the case in which at least one component of the relation is a multiset. For $Y = \{a_i[m_i]\} \in \mathtt{M}$, the multiplicities are written as a function $v_Y : X(Y) \to \mathtt{N}_+$, such that $v_Y(a_i) = m_i$. The domain of this

function, that is the support of Y, is extended to all well defined elements, as

$$\nu_Y : \mathtt{S} \uplus \mathtt{N} \to \mathtt{N}_+ \cup \{0\}, \quad \text{s.t.} \quad \nu_Y(a_i) = \begin{cases} m_i & \text{if } a_i \in X(Y) \\ 0 & \text{otherwise} \end{cases}. \tag{4.3.12}$$

Note that the extension of this function is coherent with equation (4.2.6). The relations above are generalized as

$$\begin{aligned}
&Y_1 \subseteq Y_2 \iff \nu_{Y_1}(a) \leq \nu_{Y_2}(a)\ \forall a \in \mathtt{S} \uplus \mathtt{N}, \\
&Y_1 \subset Y_2 \iff Y_1 \subseteq Y_2 \text{ and } \exists a \in \mathtt{S} \uplus \mathtt{N} \text{ s.t. } \nu_{Y_1}(a) < \nu_{Y_2}(a), \\
&Y_1 \sqsubseteq Y_2 \iff Y_1 \subseteq Y_2 \text{ and } \nexists a \in \mathtt{S} \uplus \mathtt{N} \text{ s.t. } 0 \neq \nu_{Y_1}(a) < \nu_{Y_2}(a), \\
&Y_1 \sqsubset Y_2 \iff Y_1 \sqsubseteq Y_2 \text{ and } \exists a \in \mathtt{S} \uplus \mathtt{N} \text{ s.t. } 0 = \nu_{Y_1}(a) < \nu_{Y_2}(a), \\
&Y_1 = Y_2 \iff Y_1 \subseteq Y_2 \text{ and } Y_2 \subseteq Y_1 \iff \nu_{Y_1}(a) = \nu_{Y_2}(a)\ \forall a \in \mathtt{S} \uplus \mathtt{N},
\end{aligned} \tag{4.3.13}$$

The signature of these relation is $\approx$: $\mathtt{S} \uplus \mathtt{M} \times \mathtt{S} \uplus \mathtt{M}$, where $\approx$ is one of the five relations above. However, only the definition for multisets is given in (4.3.13), but this is not a problem, as equation (4.2.7) implies that if at least one argument, say the first one, is a set, the relation $X_1 \approx Y_2$ is equivalent to $Y(X_1) \approx Y_2$.

For finite dimensional multisets, the relations can be written as

$$\begin{aligned}
&Y_1 \subseteq Y_2 \iff \nu_{Y_1}(a) \leq \nu_{Y_2}(a)\ \forall a \in X(Y_1), \\
&Y_1 \subset Y_2 \iff Y_1 \subseteq Y_2 \text{ and } (\exists a \in X(Y_1)\ \nu_{Y_1}(a) < \nu_{Y_2}(a) \text{ or } |Y_1| < |Y_2|), \\
&Y_1 \sqsubseteq Y_2 \iff \nu_{Y_1}(a) = \nu_{Y_2}(a)\ \forall a \in X(Y_1) \text{ and } |Y_1| \leq |Y_2|, \\
&Y_1 \sqsubset Y_2 \iff \nu_{Y_1}(a) = \nu_{Y_2}(a)\ \forall a \in X(Y_1) \text{ and } |Y_1| < |Y_2|, \\
&Y_1 = Y_2 \iff \nu_{Y_1}(a) = \nu_{Y_2}(a)\ \forall a \in X(Y_1) \text{ and } |Y_1| = |Y_2|.
\end{aligned} \tag{4.3.14}$$

Other definitions are available, but these ones are the most efficient computationally, because it is not necessary to evaluates multiplicities in Y_2 for elements not in $X(Y_1)$. The function `hset1.included.to.hset2` with signature

$$(\mathtt{S} \uplus \mathtt{M}) \times (\mathtt{S} \uplus \mathtt{M}) \times \{\mathtt{TRUE}, \mathtt{FALSE}\} \times \{\mathtt{TRUE}, \mathtt{FALSE}\} \to \{\mathtt{TRUE}, \mathtt{FALSE}\} \tag{4.3.15}$$

where the four arguments are `hset1`, `hset2`, `strictly` and `exactly`, returns `TRUE` if the first two arguments are in one of the relations defined above, except for the equality, that is implemented with another function. If the first two arguments are both sets, the fourth argument is ignored because in $\mathtt{S} \times \mathtt{S}$ the relations $\subseteq$ and $\sqsubseteq$ are equivalent, and so are $\subset$ and $\sqsubset$. The function iterates through the members of `hset1`, computes the difference of multiplicities, if this difference

is negative `FALSE` is returned immediately, otherwise the difference is accumulated. For $\sqsubseteq$ and $\sqsubset$ the accumulated difference must be zero at the end of the loop, and the two relations are distinguished by the condition on the supports. For $\subset$, either the accumulated difference is strictly positive, or the support of the second set is larger. Whereas for $\subseteq$, no other conditions are required after the end of the loop. The evaluation of the equality relations is implemented in the function `hsets.are.equal`, in which only the functions `size.support`, `members` and `multiplicities` defined at the end of Section 4.2.1 are used to evaluate the relation.

Some generic operators that call the function `hset1.included.to.hset2` with different combinations of the last two inputs are defined. For $\subseteq$, `<=` and `>=` are used, where the latter is for the inverse relation $\supseteq$, obtained by reflecting the arguments. For $\subset$, the generic operators are `<` and `>`, for $\sqsubseteq$ they are `%=<=%` and `%=>=%`, and for $\sqsubset$, `%=<%` and `%=>%` are used. The equality operator `==` calls the function `hsets.are.equal` and `!=` returns its negation.

4.3.2 Operations

When discussing relations the semantic of the implementation was never mentioned, as the sets that were possibly be part of some relations were never modified, and returned by the functions used to check whether two objects are in relation. An *operation* is a ternary relation between sets or multisets, that is written as $X_1 \approx X_2 = X$ for a given $\approx$, however the interest here is computing X from the *operands* X_1 and X_2. Moreover all operations will be defined for general arity, that is for more than two operands. The *universe set*, denoted by U is a set such that $X \subseteq U$, for all $X \in \mathtt{S}$, that is the set containing all possible elements: $a \in U$ for all $a \in \mathtt{S} \uplus \mathtt{N}$. The *universe multiset* U is the multiset such that $Y \subseteq U$, for all $Y \in \mathtt{M}$. Then, U contains all elements in $\mathtt{S} \uplus \mathtt{N}$, each with infinite multiplicity. Note that the universe set and multiset are defined by the same symbol, as it will be clear from the context which “universe" is considered.

The signature of all operations defined below is

$$\approx\colon (\mathtt{S} \uplus \mathtt{M}) \times \biguplus_{k \geq 0} (\mathtt{S} \uplus \mathtt{M})^k \to (\mathtt{S} \uplus \mathtt{M}), \tag{4.3.16}$$

so that the operation is defined if there is at least one operand, the second argument is written as a disjoint union of all possible tuple of `hsets`, so that for a given k, there are $k+1$ operands. However, only binary operations are defined, as the extension to general arities is straightforward. The result of the operation

is of class `S` if and only if all operands are of class `S`, otherwise the result is of class `M`. In the latter case, if an operand X_i is a set, it is replaced by $Y_i = Y(X_i)$.

The *intersection*, *union*, *sum*, *difference* and *symmetric difference* between X_1 and X_2 are

$$\begin{aligned}
&X = X_1 \cap X_2 \iff (a \in X \implies a \in X_1 \text{ and } a \in X_2),\\
&X = X_1 \cup X_2 \iff (a \in X \implies a \in X_1 \text{ or } a \in X_2),\\
&X = X_1 + X_2 \iff X = X_1 \cup X_2,\\
&X = X_1 \setminus X_2 \iff (a \in X \implies a \in X_1 \text{ and } a \notin X_2),\\
&X = X_1 \triangle X_2 \iff X = (X_1 \setminus X_2) \cup (X_2 \setminus X_1),
\end{aligned} \tag{4.3.17}$$

respectively. The identity element for the intersection is $X_2 = U$, while for all other operations is $X_2 = \emptyset$. All operations, except for the difference, are commutative and associative, $X_1 \setminus X_2 \setminus X_3$ is defined to be equal to $(X_1 \setminus X_2) \setminus X_3$. Note that the sum has been defined to be the same as the union, but these operations will be different for multisets. The multiset version of the operations above is

$$\begin{aligned}
&Y = Y_1 \cap Y_2 \iff \forall a, v_Y(a) = \min(v_{Y_1}(a), v_{Y_2}(a)),\\
&Y = Y_1 \cup Y_2 \iff \forall a, v_Y(a) = \max(v_{Y_1}(a), v_{Y_2}(a)),\\
&Y = Y_1 + Y_2 \iff \forall a, v_Y(a) = v_{Y_1}(a) + v_{Y_2}(a),\\
&Y = Y_1 \setminus Y_2 \iff \forall a, v_Y(a) = \max(v_{Y_1}(a) - v_{Y_2}(a), 0),\\
&Y = Y_1 \triangle Y_2 \iff \forall a, v_Y(a) = |v_{Y_1}(a) - v_{Y_2}(a)|.
\end{aligned} \tag{4.3.18}$$

As for the set version, the identity elements are $Y_2 = U$ and $Y_2 = \emptyset$ for the intersection, and for all other operations respectively. When generalizing $\setminus$ and $\triangle$ to multisets, some properties that hold for sets are violated. For example, in general $(Y_1 \setminus Y_2) \cap Y_2 \neq \emptyset$, and $Y_1 \triangle Y_2 \triangle Y_3 \neq Y_2 \triangle Y_1 \triangle Y_3$.

The functions `hset.operation.numeric` and `hset.operation.logical` have signature

$$\begin{aligned}
&(\mathtt{S} \uplus \mathtt{M}) \times (\mathtt{S} \uplus \mathtt{M})^* \times ((\mathtt{N}_+ \uplus \mathtt{0})^* \to \mathtt{N}_+ \uplus \mathtt{0}) \times \mathtt{B} \times \{\texttt{"refer"}, \texttt{"value"}\} \to \mathtt{M},\\
&\mathtt{S} \times \mathtt{S}^* \times (\mathtt{B}^* \to \mathtt{B}) \times \mathtt{B} \times \{\texttt{"refer"}, \texttt{"value"}\} \to \mathtt{S},
\end{aligned} \tag{4.3.19}$$

respectively, where $\mathtt{Z}^*$ is computed as in equation (4.2.11), and $\mathtt{B} = \{\mathtt{FALSE}, \mathtt{TRUE}\}$. The first two arguments, called `hset1` and `...`, contain the operands. The third argument, called `operation`, is a function that computes the updated multiplicity, the choice of this function defines which of the operations above (intersection,

union, sum, difference, symmetric difference) is computed. The fourth argument called `identity.is.universe`, with default `FALSE`, specifies whether iterating through all elements of the first operand is needed. The last argument called `semantic`, with default `"refer"`, specifies whether the first operand is modified, or whether its clone is modified. For the intersection, union, sum, difference, and symmetric difference, with logical multiplicities, the functions that are used in the third input are `all`, `any`, `any`, `nimp` (for "not implies") and `niff` (for "not if and only if") respectively. Whereas, with numeric multiplicities, the functions are `min`, `max`, `sum`, `pdif` (for "positive difference") and `sdif` (for "symmetric difference") respectively. The functions `nimp`, `niff`, `pdif` and `sdif` have been implemented by us, while the others are primitive functions in R. In the intersection the fourth argument is `TRUE` (U is the identity element of the operation), while in all other operations, the fourth argument is `FALSE` ($\emptyset$ is the identity element).

The functions `intersection`, `union`, `setsum`, `difference`, `symmdiff` with signature

$$(\mathtt{S} \uplus \mathtt{M}) \times (\mathtt{S} \uplus \mathtt{M})^* \times \{\mathtt{"refer"}, \mathtt{"value"}\} \to (\mathtt{S} \uplus \mathtt{M}), \tag{4.3.20}$$

call `hset.operation.numeric` if at least one operand is a multiset, otherwise `hset.operation.logical` is used, with third and fourth arguments set by the operation. The infix operands for computing the binary intersection are `%&%`, `%&&%` and `%and%`, for the union, the operands are `%|%`, `%||%` and `%or%`, for the sum are `%+%` and `%sum%`, for the difference `%-%` and `%!implies%`, and for the symmetric difference `%--%` and `%xor%`. In all these operands reference semantic is used, for a binary operation with value semantic, all operands can be used, but a $\sim$ is added before the last `%`, e.g., for the union with value semantic, `%|`$\sim$`%` can be used.

4.4 Performance

Here we assess the performance of our implementation, by comparing its two semantics between themselves, and with the package `sets` [40].

4.4.1 Relations

Computing a relation results to a Boolean output indicating whether the two components are related. Therefore the sets are not modified and the semantic is irrelevant. The only comparison is then between our implementation based

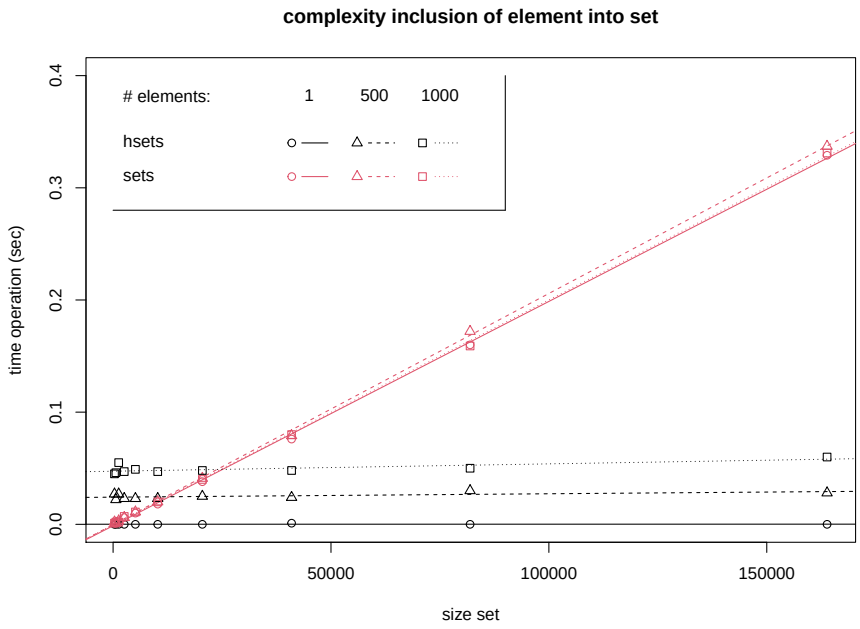

Figure 4.1: Time complexity for evaluating the inclusion relation.

on an hash table, denoted here by `hsets`, and the implementation of the library `sets`.

The comparison for the inclusion relation $a \in X$ from equation (4.3.1) between the element a and the set X, is in Figure 4.1. The x axis is $|X|$, which is the size of the set X, on the y axis the time of evaluating the relation `ai %in% X`, where `ai` is a vector of elements, and `X` have classes `hset` and `sets`, for the black and the red dots, respectively. The shape of the dots denotes for how many elements, that is the size of `ai`, the relation is evaluated. In our implementation, the complexity of the operation does not depend on the size of the set $|X|$, but it depends linearly on the size of `ai`, that is the number of elements that we want to check whether they are contained in the set. On the other hand, the complexity of the implementation of `sets`, depends linearly on $|X|$, but it seems not to depend on the size of `ai`, implying that the inclusion relations onto a set are parallellized in `sets`.

In our implementation, the subset and the equality relations between sets, written succinctly as $X_1 \approx X_2$, with $\approx$ in $\{\subseteq, \subset, =\}$, have the same complexity evaluated above, that is constant with respect to $|X_2|$, and linear with respect to $|X_1|$, where $X_2 = X$ and $X_1 = \{a_i\}_i$. The same complexity can be found if X_1 and X_2 are replaced by multisets Y_1 and Y_2, for all cases of $\approx$. The reason is that,

when evaluated using the formulas in (4.3.14), the complexity of the relation grows linearly with respect to $|Y_1|$ because of the component $\forall a \in X(Y_1)$, and the complexity is constant with respect to $|Y_2|$ because $|Y_1| \approx |Y_2|$ is evaluated in constant time, for various $\approx$.

4.4.2 Operations

The codomain of an operation is either a set or a multiset. In Figure 4.2 the complexity of the defined set operations is plotted in the left column. The x and y axes distinguish the size of the operands and the time for evaluating one operation in seconds. The shape of the points denotes the size of the second operands, their colours distinguish the semantics of our implementation and the operations computed with objects from the package `sets`. In Section 4.2.3, it has been explained that the reference semantic is helpful for some operations when the first operand is large. In particular, the time complexity of the operation is constant with respect to the size of the first operand, when $\emptyset$ is the identity element of the operation, that is for the union, difference and symmetric difference. The complexity cost of using the value semantic can be seen by the fact that the complexity grows linearly with the size of both operands, as the (large) first one has to be copied at the beginning of the operation. Whereas with the reference semantic the complexity of these operation is linear only with respect to the size of the second operand, while being constant with respect to the size of the first one. In the library `sets`, the complexity of the operations grows linearly with respect to the size of the first operand, but it seems not to depend on the size of the second one. We think that it actually depends on the size of the largest operands. For the interaction however our implementation is much more inefficient that `sets`, regardless of the semantic, although all implementations have a linear complexity with respect to the size of the first operand. The sum of two sets is defined to be equivalent to the union, so the complexity of the implementations is not computed.

In the right column of Figure 4.2 the same comparison has been done for operations between multisets. As in the previous case, the intersection is much more efficient in the `sets` package, whereas for all other operations, our implementation with reference semantic does not depend on the size of the first operand, while with value semantic it does, as the first operand is copied.

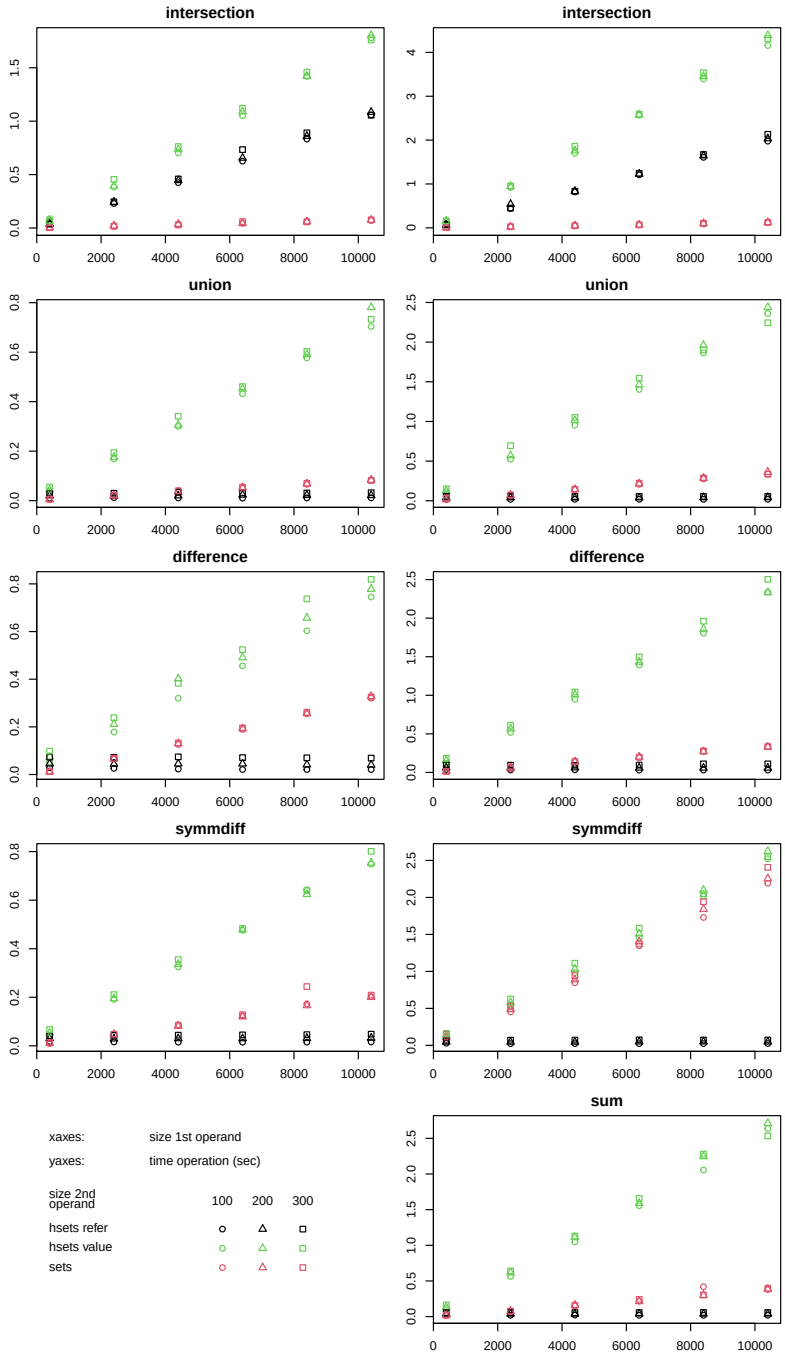

Figure 4.2: Time complexity of set (left) and multiset (right) operations for different implementations, and sizes of the operands.

4.5 MCMC with state space of undirected graphs

In this section we describe an application of the `hset` package to *Markov processes* with set- or multiset-valued (discrete) *state space*. The example considered here is a stochastic process with set-valued state space, and multiset-valued *sufficient statistic* for the distribution of the process. The stochastic process is defined over the $n(n-1)/2$ dimensional state space of n dimensional *undirected simple graphs*. Each graph is represented uniquely by the *edge set*, which contains all information (ties) about the graph. The state of the process is augmented with the *degree distribution* of the graph, that is a multiset and contains all relevant information about the distribution of the stochastic process.

The network process evolves by *tie flips* in which (typically few) non-edges become edges and vice versa. The tie flips are often *local*, e.g., when only one tie can be flipped at given time, in general at each time point only a number of ties that is small, in comparison with the size of the network, can be flipped. If few edges of the graph are flipped, the sufficient statistic does not have to be recomputed from the edge set, as only the degrees of the vertices adjacent to the flipped edges must be updated. Therefore the reference semantic in our implementation that is derived from the hash table data structure, has the computational advantage of being able to modify the state locally (in memory), without having to copy the state when its size changes.

We consider here the *Beta Model* [28, 7, 48], where vertex i has its own parameter β_i, and the tie ij between vertices i and j is in the graph with probability $p_{ij} = e^{\beta_i+\beta_j}/(1+e^{\beta_i+\beta_j})$. The model is equivalent to a sample of $n(n-1)/2$ independent Bernoulli random variables X_{ij} with probabilities p_{ij}. The distribution of the network can be written as

$$P(X = x|\beta) = \prod_{1\le i<j\le n} p_{ij} = \exp\Big(\sum_{i=1}^{n} d_i(x)\beta_i - \psi(\beta)\Big), \tag{4.5.1}$$

where $d_i(x)$ is the degree of vertex i, and $\psi(\beta) = \sum_{i<j}\log(1+e^{\beta_i+\beta_j})$ is the log-partition function. Thus the *degree vector* $d = (d_1,...,d_n)$ is a sufficient statistic for the model.

We consider a Markov chain in which at each time point some (typically few, in comparison with n) tie flips are proposed, and accepted with probability

$$Q(\tilde{x}|x,\beta) = \min\Big(\exp\Big(\sum\nolimits_{i\in\mathscr{I}}(\tilde{d}_i - d_i)\beta_i\Big), 1\Big), \tag{4.5.2}$$

where $\mathscr{I} \subseteq \{1,...,n\}$ and $i \in I$ if exist vertex j such that the tie ij is included in

the proposed tie flips (thus $i \in \mathscr{I} \iff j \in \mathscr{I}$), $d_i = d_i(x)$ and $\tilde{d}_i = d_i(\tilde{x})$ are the degrees of vertex i before and after the tie flips, respectively. If the proposed tie flip contains only the tie ij, then $\mathscr{I} = \{i, j\}$ and $|\mathscr{I}| = 2$, in general more than one tie can be flipped, but the algorithm is useful computationally when $|\mathscr{I}| \ll n$. Note that this process is a Markov chain with *Metropolis* updates, as we use a symmetric proposal distribution for the tie flips (uniform distribution over $n(n-1)/2$). Therefore, the stationary distribution of the Markov chain with acceptance probability $Q(\tilde{x}|x, \beta)$ in equation (4.5.2) is $P(X = x|\beta)$ in equation (4.5.1).

Three Markov chains $(X_t)_t$, $(X_t{}^-)_t$ and $(X_t{}^+)_t$ are considered, with same transition probability parametrized by β, but with different starting points. The chains $(X_t)_t$, $(X_t{}^-)_t$ and $(X_t{}^+)_t$ have *stationary*, *sparse* and *dense starting* point, respectively. Therefore X_0, $X_0{}^-$ and $X_0{}^+$ have degrees similar, lower and higher, respectively, than the expected degrees computed from the stationary distribution of the Markov chain. The real parameter is generated as $\beta \sim \text{Norm}(-1_n, I_n)$. At each iteration, a single tie flip is proposed, sampling $h \sim \text{Unif}(1, \ldots, n(n-1)/2)$, and the tie ij is computed from h.

With our implementation, the state of the chain with stationary starting point is (X_t, Z_t), that is composed by two objects of class `"hset"`, X_t is the network encoded as set of edges, and Z_t is the degree distribution of the network encoded as a multiset. In each iteration the `hset` of flips F is sampled, in our case $|F| = 1$ as only one tie is sampled. Then the set $\mathscr{I}_F$ of vertices that are part of at least one proposed tie flip is computed, and so are the proposed degrees $\tilde{d}_i$ for $i \in \mathscr{I}_F$. If the flip is accepted, the state is updated as

$$\begin{aligned} X_{t+1} &\leftarrow X_t \triangle F, \\ Z_{t+1} &\leftarrow (Z_t - \{d_i[m_i] : i \in \mathscr{I}_F\}) + \{\tilde{d}_i[\tilde{m}_i] : i \in \mathscr{I}_F\}, \end{aligned} \tag{4.5.3}$$

where $\tilde{m}_i$ and m_i are the multiplicities of proposed $\tilde{d}_i$ and current d_i degrees respectively, for $i \in \mathscr{I}_F$. The update uses the operations $\triangle : \mathtt{S} \times \mathtt{S} \to \mathtt{S}$, $+ : \mathtt{M} \times \mathtt{M} \to \mathtt{M}$ and $- : \mathtt{M} \times \mathtt{M} \to \mathtt{M}$, that have been defined in Section 4.3.2. In Section 4.4.2 it has been shown that for these three operations (that all have the empty set / multiset as the identity second operand), the complexity is not affected by the sizes of X_t and Z_t, but it depends linearly on the sizes of F and $\mathscr{I}_F$. The same algorithm is used with $(X_t{}^+, Z_t{}^+)$ and $(X_t{}^-, Z_t{}^-)$, in which the starting point is out of equilibrium .

The updates in equation (4.5.3) are coded with reference semantic as

```
state$edge.set %xor% hset(flips$id)
state$degree.frequencies %-% hset(names(table.old.degrees),
 as.integer(table.old.degrees))
state$degree.frequencies %+% hset(names(table.new.degrees),
 as.integer(table.new.degrees))
```

(4.5.4)

where `state$edge.set` and `state$degree.frequencies` are `hset` objects containing the current network and degree distribution, `flips$id` contains the ties that are flipped, `table.old.degrees` and `table.new.degrees` contain the old and new degrees for the vertices in $\mathscr{I}_{\mathscr{F}}$. Note that the constructor `hset` is used to create the set F, and the multisets containing the degree frequencies to be subtracted and added.

In Figure 4.3 the processes derived from $(X_t)_t$, $(X_t{}^-)_t$ and $(X_t{}^+)_t$ are plotted in red, green and blue respectively. In the left, the moving average of the acceptance ratio is plotted. In each iteration the proposed transition is either accepted, or it is not. This binary outcome is replaced in the plot by the average of 150 binary values around it, giving an indication on how probable are transitions of state in and out of equilibrium. The stationary and the sparse chain have a similar behaviour with transition probability approximately equal to 0.35 throughout their whole history. The dense chain starts with a larger transition probability, that is reduced toward the equilibrium value as the process approaches the stationary distribution. In the right plot the size of the state, that is the number of ties in the network is plotted in the three cases, showing how the distribution of $|X_t{}^-|$ and $|X_t{}^+|$ approach the one of $|X_t|$ as $t \to \infty$.

In Figure 4.4 are plotted, for all chains, the empirical cumulative distribution functions of the degree distribution of states at equally spaced iterations. For the stationary process (X_t, Z_t) these curves, that are plotted in red, are $\{\text{ecdf}(Z_t)\}$ for $t \in \{1, 1001, 2001, ..., 10001\}$. For the processes with sparse and dense starting points, the ECDFs are plotted in blue and green respectively. The width of the ECDFs is larger when t is as such, showing how the degree distributions of the networks $X_t{}^-$ and $X_t{}^+$ approach the degree distribution of X_t as $t \to \infty$.

Markov chains of the type discussed in this application are used to estimate the parameters of models for which the function that normalizes the stationary distribution of the process is not analytic. For example in *Exponential Random Graph Models* [52] computing the normalization constant requires summing over

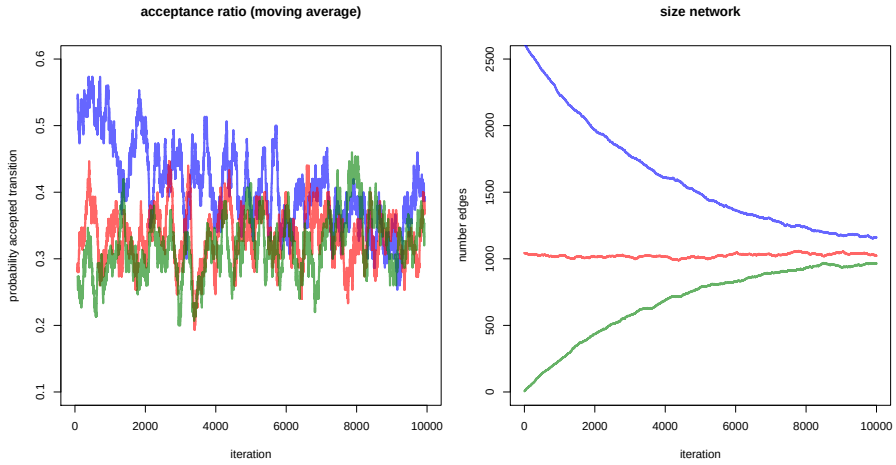

Figure 4.3: Left: acceptance ratio (moving average with 150 *filtered observations) of the Metropolis-Hastings algorithm. Right: number of ties of the network (state) at different iterations. Colours distinguish the different starting points: red for stationary, green for sparse, blue for dense.*

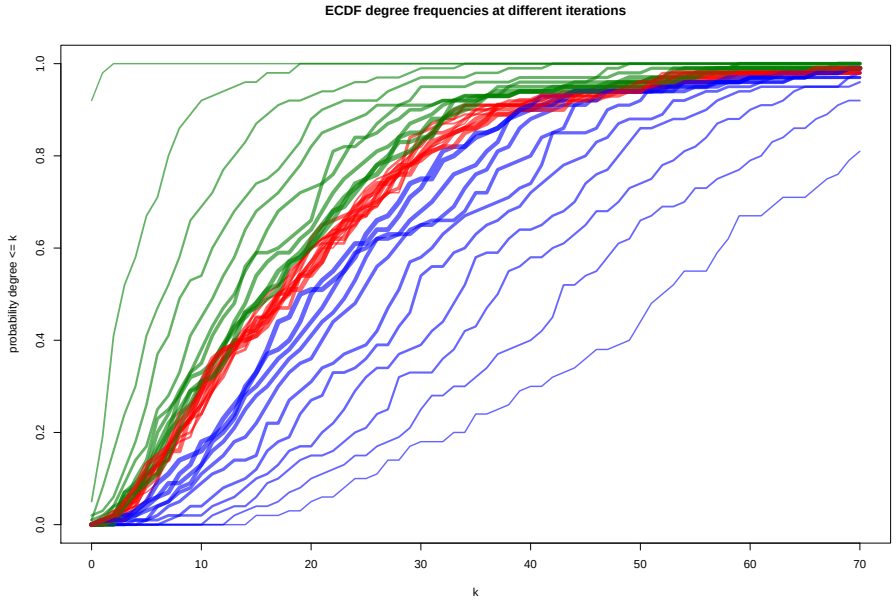

Figure 4.4: Empirical cumulative distribution function of the degree frequencies, at iterations 1 *(lowest line width),* 1001, 2001, ..., 10001 *(largest line width), for the three starting points (stationary - red, sparse - green, dense - blue).*

the set of all possible graphs of a given dimension, making the computation infeasible also for small networks. Frequentist and Bayesian estimation algorithms for ERGMs [67, 13] are based on the *Markov Chain Monte Carlo Maximum Liklihood Estimation* method (*MCMCMLE*), developed in [20] and [21], where gradients of the likelihood are approximated using a Markov chain that does not require the computation of the normalization constant.

A graph is usually represented by its *adjacency matrix*, with Boolean elements denoting whether two vertices are connected. Observed large networks are usually *sparse*, meaning that the number of edges grows linearly with respect to the number of vertices. If the adjacency matrix is stored as a *dense matrix*, the space required is $O(n^2)$, where n is the size of the network, but individual elements can be updated in constant time ($O(1)$ elementary operations). Whereas if a *sparse matrix* is used, the space required for storing a sparse network is $O(n)$, but flipping a tie might cost $O(n)$ elementary operations, as the sparse matrix might have to be re-constructed. The hash table used in `hset` can be helpful in these cases, as the space required to store the set is $O(n)$, but individual elements are updated with $O(1)$ operations.

4.6 Conclusion

The `hset` implementation of set operations is motivated by the efficacy of `hash` data structure when used with reference semantic, allowing significant computational advantages in algorithms in which a set or a multiset is used as container, and it is updated few components at the time. The implementation is currently restricted to sets or multisets with elements that are numbers (or sets of numbers), so that mathematical relations between integers and reals are respected, e.g., that `1L` and `1.0` both represents the number `1`. This approach differs from the library `sets` where the classes of two objects determinate whether they can be the same, and reference semantic is not used.

Basic parametrized relations and operations between sets and multisets are implemented, for most of them (all but the intersection) reference semantic can speed up algorithms significantly. In R, reference semantic is used for `environment` and `hash` data objects, whereas almost all other objects use value semantic. Objects with reference semantic are usually modified by functions with side effects. In [73] it is suggested to avoid the use of a function for both its side effects and its returned value. We partially follow this suggestion, as for the operations defined in Section 4.3.2 computed with reference semantic, the result of the operation is both returned, and the first operand is transformed to it. Therefore all terms

`X1%|%X2`, `X1 <- X1%|%X2`, and `X1 <- X1%| ~ %X2` evaluate to the same objects in memory. The first term is the most common approach to compute operations with reference semantic, where an operator is viewed as accumulator causing the first operand to be modified to the result of the operation with the second operand. However we suggest using the second approach, that is more coherent with the syntax used with value semantics, and so it is more similar to how code is usually written in R. Note that the suggestion of avoiding functions with both side effects and returned value is only partially followed, because there are no side effects that modify objects that are not returned.

Our implementation can be useful for simulating Markov chains with countable state space, as in simulation and estimation of temporal network models. Recent approaches to statistics and probability theory such as [30], emphasise the importance of the multiset mathematical structure of discrete probability distributions, especially when determining the properties of algorithms used to sample, estimate or learn probability distributions. These approaches on probability have been heavily influenced by theories of computation, so they will probably be influential on how algorithms in computational statistics and other disciplines will be developed and implemented.

Chapter 5

Conclusion

The inclusion of random effects in the stochastic actor oriented model, that is the main contribution of this thesis, is an important generalization of the SAOM because it allows various options for modelling accurately the heterogenity between the individuals in a social network. The model specifies the evolution for a stochastic process that can be simulated, and it is assumed to mimic the social dynamics between people. The focus of the thesis has been therefore on statistical inference based on simulation, where the parameters of the model control the algorithm that is used for simulating the process. The method for estimation and model evaluation that is used in our generalization of the SAOM has been also studied in a regression set-up. An implementation of sets and multisets is then discussed. This last topic is not related with the SAOM, although it that can be useful when simulating dynamic networks, that are fundamentally discrete objects.

In simulated statistical inference, estimation is done by adjusting the parameters until the process mimic in some sense the observed data, model evaluation is by simulating from the estimated values to check how well the models behave. Estimation and model evaluation in the SAOM have been developed, implemented and checked for models with independent random effects. However we have checked the algorithm without modifying the package RSiena, so the implementation is not efficient. Modifications on RSiena to allow simulations with random effects at approximately the same cost of the one without, are planned so that researchers can use models with random effects much more easily. The analysis on the tailor shop dataset showed how different assumptions for the generative process suggest different interpretations for the social dynamics that is modelled.

The simulated method of moments that is used in the SAOM with random

effects has been studied in regression, mainly to evaluate possible extensions of the algorithm, for example when the parameters are estimated with generalized (simulated) method of moments. Following the literature on stochastic gradient descent, we have studied an adaptive method to update the preconditioning to make the optimization more stable. Moreover models evaluation procedures based on orthogonalization to increase the power of score tests have been studied extensively also in the case in which the parameters under the null are estimated with generalized method of moments.

The R implemenation of sets and multisets based on the hash table data structure is useful when simulating processes that are at least in part fundamentally discrete. This implementation leverages the fact that elements in an hash table can be accessed, included or removed in constant time. The implemntation is advantageous for all set (and multiset) operations except for the intersection, when the first operand is large, as to evaluate the operation is not required to access all elements of the first operand. An example that has been discussed, is the representation with a set and a multiset for an evolving undirected graph with a set of sufficient statistics for its distribution.

Appendix A

Mathematical and computational background

A.1 Multivariable chain rule

Consider the functions $f : \mathbb{R}^n \to \mathbb{R}^m$, $g : \mathbb{R}^m \to \mathbb{R}^k$, and their composite $h = (g \circ f) : \mathbb{R}^n \to \mathbb{R}^k$. Let $D_a(h)$ be the *total derivative* of the function $h : \mathbb{R}^n \to \mathbb{R}^k$ evaluated at $a \in \mathbb{R}^n$, i.e.,

$$D_a(h) = \lim_{||\delta|| \to 0} \frac{||h(a+\delta) - h(a)||}{||\delta||}. \tag{A.1.1}$$

The *multivariable chain rule* [53, Chapter 9] is the formula that relates the total derivative of $h = g \circ f$ with the total derivatives of f and g, which is

$$D_a(h) = D_{f(a)}(g) \circ D_a(f). \tag{A.1.2}$$

The total derivative is a linear transformation, so the quantities in the last equation can be represented by *Jacobian matrices*, with the composition operation replaced by the matrix product, so that the last equation can be rewritten as

$$\boldsymbol{J}_h(a) = \boldsymbol{J}_g(f(a)) \cdot \boldsymbol{J}_f(a), \tag{A.1.3}$$

where the three matrices are $(k \times n)$, $(k \times m)$ and $(m \times n)$ dimensional, respectively. Note that the last equation is the generalization of the “usual" chain rule $h'(a) = g'(f(a))f'(a)$, which is obtained when $n = m = k = 1$.

A.2 Derivative of a Gaussian r.v. with respect to its parameters

Let X be a Gaussian random variable with mean μ and variance σ^2. Then

$$U = \frac{X-\mu}{\sigma} \sim \mathcal{N}(0,1), \tag{A.2.1}$$

and X can be written as $X = \sigma U + \mu \sim \mathcal{N}(\mu, \sigma^2)$. The standard deviation σ is a constant and the distribution of U does not depend on μ, then

$$\frac{\partial X}{\partial \mu} = \frac{\partial(\sigma U + \mu)}{\partial \mu} = \sigma \frac{\partial U}{\partial \mu} + \frac{\partial \mu}{\partial \mu} \sim \sigma\delta_0 + 1 \sim \delta_0 + 1 \sim \delta_1, \tag{A.2.2}$$

where δ_y is the degenerate random variable such that y is sampled with probability 1. Similarly the distribution of U does not depend on σ, then

$$\frac{\partial X}{\partial \sigma} = \frac{\partial(\sigma U + \mu)}{\partial \sigma} = \frac{\partial \sigma}{\partial \sigma} U + \frac{\partial \mu}{\partial \sigma} = U \sim \mathcal{N}(0,1), \tag{A.2.3}$$

and

$$\frac{\partial X}{\partial \sigma^2} = \frac{\partial((\sigma^2)^{1/2} U + \mu)}{\partial \sigma^2} = \frac{1}{2}(\sigma^2)^{-1/2} U + 0 = \frac{1}{2\sigma} U \sim \mathcal{N}\Big(0, \frac{1}{4\sigma^2}\Big). \tag{A.2.4}$$

A.3 Semantics in R language

In R, objects are generally modified with *value semantic*, meaning that whenever the object is accessed, a new copy of the object is first created, this new copy is then modified, and eventually stored with the name of the previous variable. More precisely, a new copy is not always created, because modifying the object locally has computational advantages, this behaviour is called *copy-on-modified* and it is explained in [73, Chapter 2], however when reasoning about the code, it can be assumed that the code behaves as if an object is copied every time it is accessed. The other approach is to use a *reference semantic*, where the name of the object refers to a pointer to a memory address, so that the modification is always local. However, operations in the two semantics behave differently. For example, in the code

```
a = 2; b = a; b = b + 1;
```
(A.3.1)

`a` is initialized to `2`, then `b` is defined to be equal to `a`, and `b` is incremented. With value semantic, after the three operations, `a` and `b` are equal to `2` and `3` respectively, whereas with reference semantic, they are both equal to `3`. The reason is that, with value semantic, in the second operation, the value that is referred by `a` is copied and stored elsewhere, this copy is referred by `b`, when `b` is modified, only the value to which `b` points to is modified, so `a` has not changed. On the other hand, if the code above would have been with reference semantic, the second operation would have copied the pointer labelled by `a`, and this new copied pointer is labelled by `b`, then `a` and `b` would have pointed to the same address in memory, so if `b` is modified, also `a` is.

Objects of class `"numeric"` (as for most types and classes in R), are accessed and updated with value semantic. Then after the operations above are evaluated, `a` and `b` are equal to `2` and `3` respectively. Whereas objects of class `"hash"` [11], are accessed and updated with reference semantic, therefore in the code

```
a = hash::hash(key="k1", values=2);
b = a;                                          (A.3.2)
b[["k1"]] = b[["k1"]] + 1;
```

`a` and `b` refers to the same hash table, so after the last command, `a[["k1"]]` is also equal to 3. An hash table can be copied with the following method

```
a = hash::hash(key="k1", values=2);
b = hash::hash(hash::keys(a), hash::values(a));  (A.3.3)
b[["k1"]] = b[["k1"]] + 1;
```

so that in the second line a new hash table, labelled by `b` is created, this hash table is a copy of `a`, as the keys and values are copied from it, but the modification of `b` in the last row modified, does not change `a`.

Bibliography

[1] Viviana Amati, Felix Schönenberger, and Tom A B Snijders. Estimation of stochastic actor-oriented models for the evolution of networks by generalized method of moments. *Journal de la Société Française de Statistique*, 156:140–165, 2015.

[2] Viviana Amati, Felix Schönenberger, and Tom A B Snijders. Contemporaneous statistics for estimation in stochastic actor-oriented co-evolution models. *Psychometrika*, 84:1068–1096, 2019.

[3] IV Basawa. Neyman-le cam tests based on estimating functions. In *Proceedings of the Berkeley conference in honor of Jerzy Neyman and Jack Kiefer*, volume 2, pages 811–825. Wadsworth Belmont, Calif, USA, 1985.

[4] IV Basawa. Generalized score tests for composite hypotheses. *Estimating functions*, pages 121–131, 1991.

[5] Douglas Bates, Martin Maechler, Ben Bolker, and Steve Walker. Fitting linear mixed-effects models using lme4. *Journal of Statistical Software*, 67(1): 1–48, 2015. doi: 10.18637/jss.v067.i01.

[6] Johannes Berkhof and Tom A B Snijders. Variance component testing in multilevel models. *Journal of Educational and Behavioral Statistics*, 26(2): 133–152, 2001.

[7] Joseph Blitzstein and Persi Diaconis. A sequential importance sampling algorithm for generating random graphs with prescribed degrees. *Internet mathematics*, 6(4):489–522, 2011.

[8] Per Block, Johan Koskinen, James Hollway, Christian Steglich, and Christoph Stadtfeld. Change we can believe in: Comparing longitudinal network models on consistency, interpretability and predictive power. *Social Networks*, 52:180–191, 2018.

[9] Per Block, Christoph Stadtfeld, and Tom A B Snijders. Forms of dependence: Comparing saoms and ergms from basic principles. *Sociological Methods & Research*, 48(1):202–239, 2019.

[10] Stephen P Borgatti, Ajay Mehra, Daniel J Brass, and Giuseppe Labianca. Network analysis in the social sciences. *science*, 323(5916):892–895, 2009.

[11] Christopher Brown. *hash: Full Feature Implementation of Hash/Associated Arrays/Dictionaries*, 2019. URL https://CRAN.R-project.org/package=hash. R package version 2.2.6.1.

[12] Carter T Butts. 4. a relational event framework for social action. *Sociological Methodology*, 38(1):155–200, 2008.

[13] Alberto Caimo and Nial Friel. Bayesian inference for exponential random graph models. *Social networks*, 33(1):41–55, 2011.

[14] Alberto Caimo, Nial Friel, et al. Bergm: Bayesian exponential random graphs in R. *Journal of Statistical Software*, 61(i02), 2014.

[15] Kathleen M Carley. Group stability: A socio-cognitive approach. *Advances in Group Processes*, 7(1):44, 1990.

[16] Giacomo Ceoldo and Ernst C Wit. Efficient implementation of sets and multisets in r using hash tables. *arXiv preprint arXiv:2304.09809*, 2023.

[17] Giacomo Ceoldo, Tom AB Snijders, and Ernst C Wit. Stochastic actor oriented model with random effects. *arXiv preprint arXiv:2304.07312*, 2023.

[18] Kyle Cranmer, Johann Brehmer, and Gilles Louppe. The frontier of simulation-based inference. *Proceedings of the National Academy of Sciences*, 117(48):30055–30062, 2020.

[19] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. In *Journal of Machine Learning Research*, volume 12, pages 2121–2159, 2011.

[20] Charles J Geyer. Markov chain monte carlo maximum likelihood. *Interface Foundation of North America*, 1991.

[21] Charles J Geyer and Elizabeth A Thompson. Constrained monte carlo maximum likelihood for dependent data. *Journal of the Royal Statistical Society: Series B (Methodological)*, 54(3):657–683, 1992.

[22] Andrew R. Hall. *Generalized Method of Moments*. Oxford University Press, Oxford, 2005.

[23] Steve Hanneke, Wenjie Fu, and Eric P Xing. Discrete temporal models of social networks. *Electronic Journal of Statistics*, 4:585–605, 2010.

[24] Lars Peter Hansen. Large sample properties of generalized method of moments estimators. *Econometrica: Journal of the Econometric Society*, pages 1029–1054, 1982.

[25] Nicholas J Higham. Matrix nearness problems and applications. In M J C Gover and S Barnett, editors, *Applications of Matrix Theory*, pages 1–27. Oxford University Press, Oxford, 1989.

[26] Paul W Holland and Samuel Leinhardt. A dynamic model for social networks. *Journal of Mathematical Sociology*, 5(1):5–20, 1977.

[27] Paul W Holland and Samuel Leinhardt. A method for detecting structure in sociometric data. *Social Networks*, pages 411–432, 1977.

[28] Paul W Holland and Samuel Leinhardt. An exponential family of probability distributions for directed graphs. *Journal of the american Statistical association*, 76(373):33–50, 1981.

[29] Petter Holme and Jari Saramäki. Temporal networks. *Physics reports*, 519 (3):97–125, 2012.

[30] Bart Jacobs. *Structured probabilistic reasoning*. Forthcoming book, 2019. URL http://www.cs.ru.nl/B.Jacobs/PAPERS/ProbabilisticReasoning.pdf.

[31] Bruce Kapferer. *Strategy and transaction in an African factory: African workers and Indian management in a Zambian town*. Manchester University Press, 1972.

[32] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[33] Johan H. Koskinen and Tom A. B. Snijders. Bayesian inference for dynamic social network data. *Journal of Statistical Planning and Inference*, 13:3930–3938, 2007.

[34] Pavel N Krivitsky and Mark S Handcock. A separable model for dynamic networks. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 76(1):29–46, 2014.

[35] Pierre L'Ecuyer. An overview of derivative estimation. In *Proceedings of the 1991 Winter Simulation Conference*. Institute of Electrical and Electronics Engineers (IEEE), 1991.

[36] Alessandro Lomi, Tom AB Snijders, Christian EG Steglich, and Vanina Jasmine Torló. Why are some more peer than others? evidence from a longitudinal study of social networks and individual academic performance. *Social science research*, 40(6):1506–1520, 2011.

[37] Stephan Mandt, Matthew D. Hoffman, and David M. Blei. Stochastic gradient descent as approximate bayesian inference. *Journal of Machine Learning Research*, 18(50):1–35, 2017.

[38] Charles E McCulloch and Shayle R Searle. *Generalized, linear, and mixed models*. John Wiley & Sons, 2004.

[39] Daniel McFadden. A method of simulated moments for estimation of discrete response models without numerical integration. *Econometrica: Journal of the Econometric Society*, pages 995–1026, 1989.

[40] David Meyer and Kurt Hornik. Generalized and customizable sets in R. *Journal of Statistical Software*, 31(2):1–27, 2009. doi: 10.18637/jss.v031.i02.

[41] Carina Mood. Logistic regression: Why we cannot do what we think we can do, and what we can do about it. *European Sociological Review*, 26: 67–82, 2010.

[42] Mark Newman. *Networks*. Oxford university press, 2018.

[43] Jerzy Neyman. Optimal asymptotic tests of composite hypotheses. In Ulf Grenander, editor, *Probability and Statistics*, pages 213–234. Wiley, 1959.

[44] Krzysztof Nowicki and Tom A B Snijders. Estimation and prediction for stochastic blockstructures. *Journal of the American Statistical Association*, 96(455):1077–1087, 2001.

[45] Patrick O Perry and Patrick J Wolfe. Point process modelling for directed interaction networks. *Journal of the Royal Statistical Society: SERIES B: Statistical Methodology*, pages 821–849, 2013.

[46] José C Pinheiro and Douglas M Bates. Linear mixed-effects models: basic concepts and examples. *Mixed-effects models in S and S-Plus*, pages 3–56, 2000.

[47] Boris T Polyak and Anatoli B Juditsky. Acceleration of stochastic approximation by averaging. *SIAM journal on control and optimization*, 30(4): 838–855, 1992.

[48] Alessandro Rinaldo, Sonja Petrović, and Stephen E. Fienberg. Maximum lilkelihood estimation in the β-model. *The Annals of Statistics*, 41(3):1085 – 1110, 2013. doi: 10.1214/12-AOS1078. URL https://doi.org/10.1214/12-AOS1078.

[49] Ruth M Ripley, Tom A B Snijders, Zsófia Boda, András Vörös, and Paulina Preciado. Manual for rsiena. *University of Oxford, Department of Statistics, Nuffield College*, 2023.

[50] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, 22:400–407, 1951.

[51] Garry Robins, Philippa Pattison, and Jodie Woolcock. Missing data in networks: exponential random graph (p*) models for networks with non-respondents. *Social Networks*, 26(3):257–283, 2004.

[52] Garry Robins, Pip Pattison, Yuval Kalish, and Dean Lusher. An introduction to exponential random graph (p*) models for social networks. *Social networks*, 29(2):173–191, 2007.

[53] Walter Rudin. *Principles of Mathematical Analysis*, volume 3. McGraw-Hill New York, 1964.

[54] David R Schaefer, Steven A Haas, and Nicholas J Bishop. A dynamic model of us adolescents' smoking and friendship networks. *American journal of public health*, 102(6):e12–e18, 2012.

[55] Michael Schweinberger. *Statistical methods for studying the evolution of networks and behavior*. PhD thesis, Rijksuniversiteit Groningen, 2007.

[56] Michael Schweinberger. Statistical modelling of network panel data: Goodness of fit. *British Journal of Mathematical and Statistical Psychology*, 65(2): 263–281, 2012.

[57] Michael Schweinberger. Statistical inference for continuous-time markov processes with block structure based on discrete-time network data. *Statistica Neerlandica*, 74(3):342–362, 2020.

[58] Michael Schweinberger and Tom A B Snijders. Markov models for digraph panel data: Monte Carlo-based derivative estimation. *Computational Statistics & Data Analysis*, 51:4465–4483, 2007.

[59] Tom A B Snijders. Stochastic actor-oriented models for network change. *Journal of Mathematical Sociology*, 21(1-2):149–172, 1996.

[60] Tom A B Snijders. The statistical evaluation of social network dynamics. *Sociological Methodology*, 31:361–395, 2001.

[61] Tom A. B. Snijders. Explained variation in dynamic network models. *Mathématiques, Informatique et Sciences Humaines / Mathematics and Social Sciences*, 168(4), 2004.

[62] Tom A B Snijders. Stochastic actor-oriented models for network dynamics. *Annual Review of Statistics and Its Application*, 4:343–363, 2017.

[63] Tom A B Snijders. Siena algorithms. Technical report, University of Groningen, University of Oxford, http://www.stats.ox.ac.uk/~snijders/siena/Siena_algorithms.pdf, 2023.

[64] Tom A. B. Snijders, Christian E. G. Steglich, and Michael Schweinberger. Modeling the co-evolution of networks and behavior. In Kees van Montfort, Han Oud, and Albert Satorra, editors, *Longitudinal models in the behavioral and related sciences*, pages 41–71. Mahwah, NJ: Lawrence Erlbaum, 2007.

[65] Tom A B Snijders, Johan Koskinen, and Michael Schweinberger. Maximum likelihood estimation for social network dynamics. *The Annals of Applied Statistics*, 4(2):567, 2010.

[66] Tom AB Snijders, Gerhard G Van de Bunt, and Christian EG Steglich. Introduction to stochastic actor-based models for network dynamics. *Social networks*, 32(1):44–60, 2010.

[67] Tom AB Snijders et al. Markov chain monte carlo estimation of exponential random graph models. *Journal of Social Structure*, 3(2):1–40, 2002.

[68] Stephanie Thiemichen, Nial Friel, Alberto Caimo, and Göran Kauermann. Bayesian exponential random graph models with nodal random effects. *Social Networks*, 46:11–28, 2016.

[69] Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude, 2012.

[70] Larry Wasserman. *All of statistics: a concise course in statistical inference*. Springer, New York, 2004.

[71] Stanley Wasserman. Analyzing social networks as stochastic processes. *Journal of the American Statistical Association*, 75(370):280–294, 1980.

[72] Stanley Wasserman and Katherine Faust. *Social network analysis: Methods and applications*. Cambridge University Press, New York and Cambridge, 1994.

[73] Hadley Wickham. *Advanced r*. CRC press, 2019. URL https://adv-r.hadley.nz/.

[74] Paul FV Wiemann, Thomas Kneib, and Julien Hambuckers. Using the softplus function to construct alternative link functions in generalized linear models and beyond. *arXiv preprint arXiv:2111.14207*, 2021.