Winter, Ben:

Schätzalgorithmen in Exponential Random Graph Modellen

Masterarbeit, Sommersemester 2017

Fakultät für Mathematik, Informatik und Statistik

Ludwig-Maximilians-Universität München

https://doi.org/10.5282/ubm/epub.41015

Ludwig-Maximilians-Universität München
Institut für Statistik

Masterarbeit von Ben Winter

Schätzalgorithmen in Exponential Random Graph Modellen

---

24. Mai 2017

Betreuer:
Prof. G. Kauermann

Zusammenfassung

Das Exponential Random Graph Modell (ERGM) ist ein aussichtsreicher Ansatz zur Modellierung von Netzwerken. Als problematisch erweist sich jedoch die Parameterschätzung aufgrund der grundsätzlich unberechenbaren Normalisierungskonstante, welche Teil der Log-Likelihood Funktion ist. Zur Lösung dieses Problems wurden einige simulationsgestützte Schätzalgorithmen entwickelt, die es ermöglichen, eine Berechnung der Normalisierungskonstante zu umgehen.

Das Ziel der vorliegenden Masterarbeit ist es, die Performance der wichtigsten Schätzalgorithmen bei der Parameterschätzung zu analysieren und zu vergleichen. Dafür werden aus verschiedenen ERGMs mehrere Netzwerke simuliert, wobei auch die Parameterwerte variiert werden. Anschließend werden mit allen zu vergleichenden Schätzalgorithmen auf den simulierten Netzwerken ERGMs mit denselben Netzwerk Statistiken, die bei der Simulation der Netzwerke verwendet wurden, geschätzt. Dies macht es möglich, die bei der Simulation zufälliger Netzwerke eingesetzten Parameterwerte mit den anschließend geschätzten Parameterwerten zu vergleichen und darüber hinaus die für eine Schätzung benötigte Zeit und die Quote, mit der die Schätzalgorithmen konvergieren, zu erfassen.

Die Simulationen zeigen, dass die Performance der Schätzalgorithmen von der Stabilität der Modelle, aus denen die zufälligen Netzwerke simuliert werden, abhängt. Bei Schätzungen von Netzwerken, die aus stabilen ERGMs simuliert wurden, führen alle Schätzalgorithmen zu sehr guten Schätzungen. Werden allerdings Netzwerke aus eher instabilen Modellen geschätzt, können deutliche Unterschiede vor allem bei der Quote, mit der die verschiedenen Schätzalgorithmen konvergieren, ausgemacht werden. Hier ist es der Stepping Algorithmus, bei dem am seltensten Konvergenzprobleme beobachtet werden konnten. Schätzungen mit dem oft standardmäßig verwendeten Markov Chain Monte Carlo Maximum Likelihood Verfahren führten dagegen am ehesten zu Konvergenzproblemen.

Inhaltsverzeichnis

1 Einleitung

Die Analyse von Netzwerken ist in vielen wissenschaftlichen Gebieten von großem Interesse. Klassische Beispiele findet man in den Sozialwissenschaften, in denen Freundesnetzwerke eine wichtige Rolle spielen, und in den Politikwissenschaften, in denen beispielsweise Netzwerke aus internationalen Beziehungen im Waffenhandel von Bedeutung sind. In der Finanzmathematik versucht man unter anderem mit Hilfe der Netzwerkanalyse eine optimale Portfolio Zusammensetzung zu erreichen. Auch in den Bereichen Informatik, Medizin oder Biologie findet man Fragestellungen, bei denen statistische Methoden der Netzwerkanalyse benötigt werden, um Netzwerke zu untersuchen.

Das Besondere an Netzwerken ist ihre Abhängigkeitsstruktur, da deren Beobachtungen, auch *Akteure* genannt, in einer Beziehung zu einander stehen, welche bei der Analyse von wesentlichem Interesse ist. Netzwerke beschränken sich demnach nicht lediglich auf Variablen, die einzelne Akteure betreffen, sondern bilden vor allem die Beziehungen zwischen ihnen ab. Somit können die Akteure eines Netzwerks grundsätzlich nicht als unabhängig voneinander betrachtet werden. Gewöhnliche statistische Analysen nehmen jedoch an, dass Beobachtungen unabhängig voneinander sind. Folglich werden für die Analyse von Netzwerken alternative Methoden der statistischen Modellierung benötigt.

Das Exponential Random Graph Modell ist eine solche alternative Methode, die zur Modellierung von Netzwerken geeignet ist. Dieses Modell ermöglicht es, neben den strukturellen Eigenschaften eines Netzwerkes auch erklärende Zusatzinformationen über die Akteure eines Netzwerkes als externe Variablen in die Modellierung mit aufzunehmen. Als problematisch erweist sich bei Exponential Random Graph Modellen jedoch die Maximum Likelihood Schätzung. Diese ist, aufgrund der grundsätzlich unberechenbaren Normalisierungskonstante in der Log-Likelihood Funktion, nicht durchführbar. Als Folge wurden zur Lösung dieses Problems einige Schätzmethoden entwickelt, die eine Berechnung der Normalisierungskonstante umgehen.

Im Kontext der Exponential Random Graph Modelle gibt es demnach einige unterschiedliche Herangehensweisen bei der Schätzung der Modellparameter. Bei einer Analyse mit der Softwareumgebung R (R Development Core Team (2008)) liefert das Paket ergm (Handcock et al. (2017)) alle notwendigen Utensilien für eine Netzwerkanalyse. Dieses Paket bietet für die Schätzung der Modellparameter eine Auswahl zwischen vier der wohl wichtigsten Schätzmethoden: der Maximum Pseudo-Likelihood Schätzung, dem Markov Chain Monte Carlo Maximum Likelihood Estimation Verfahren, dem Stepping Algorithmus und der Stochastischen Approximation nach

Robbins-Monro. Das Anliegen dieser Arbeit ist es, diese Schätzmethoden vorzustellen und ihre Performance an simulierten Netzwerken zu testen und miteinander zu vergleichen.

Diese Arbeit beschäftigt sich zunächst mit einigen relevanten Grundlagen der Graphentheorie. Anschließend wird das Exponential Random Graph Modell vorgestellt. In diesem umfangreichen Abschnitt werden auch die vier bereits erwähnten Schätzmethoden erläutert, wobei auch auf die Ziehung von zufälligen Netzwerken eingegangen wird, welche für drei der vier Schätzmethoden unumgänglich ist. Darüber hinaus beschäftigt sich dieser Abschnitt auch mit der Interpretation der geschätzten Parameter, möglichen Statistiken, dem Problem der Degeneration und einem möglichen Lösungsansatz durch gewichtete Statistiken. Abschließend wird in Abschnitt 4 anhand einer Simulationsstudie die Performance der zuvor vorgestellten Schätzmethoden untersucht und untereinander verglichen. Dabei dienen die von den Schätzalgorithmen für eine Schätzung benötigte Zeit, die Quote mit der die Schätzalgorithmen konvergieren und allgemein die Qualität der Schätzungen als Vergleichsmerkmale.

2 Graphentheorie

Ein Netzwerk ist eine Gruppe oder ein System von miteinander verbundenen Akteuren, wobei ein Akteur dabei sehr vieles sein kann, wie z.B. eine Person, eine Stadt oder eine Aktie. Bei Netzwerkdaten handelt es sich demnach um *relationale Daten*, da sie die Beziehungen zwischen den einzelnen Akteuren abbilden. Die Graphentheorie ist eine mathematische Theorie, die es ermöglicht, solche relationalen Daten zu beschreiben. Im Folgenden sollen nun alle für diese Arbeit relevanten Aspekte der Graphentheorie erläutert werden.

Dargestellt werden Netzwerke grundsätzlich durch Graphen. Ein *Graph* $G = (V, E)$ ist die mathematische Beschreibung eines Netzwerkes. Er besteht aus einer Knotenmenge V und einer Kantenmenge E. Ein *Knoten* verkörpert einen Akteur eines Netzwerkes. Üblicherweise wird davon ausgegangen, dass die Anzahl an Knoten $N_v = |V|$ kleiner unendlich ist. Eine *Kante* $\{i, j\}$ ist ein Element der Menge E und verbindet zwei Akteure miteinander. Sie Beschreibt die Beziehung zwischen den Knoten i und j. Existiert eine Kante zwischen den Knoten i und j, so kennzeichnet sie eine Beziehung zwischen ihnen (Kolaczyk (2009)).

Graphen können *gerichtet* oder *ungerichtet* sein. Bestehen ausschließlich symmetrische Beziehung zwischen den Akteuren, so handelt es sich um einen ungerichteten Graphen. Eine Beziehung zwischen zwei Akteuren ist symmetrisch, falls Akteur i zu Akteur j in derselben Beziehung steht wie Akteur j zu Akteur i. Ein Beispiel für eine symmetrische Beziehung ist eine Freundschaft in einem sozialen Netzwerk. Hat eine Beziehung zwischen zwei Akteuren jedoch eine Richtung, so handelt es sich um einen gerichteten Graphen der auch *Digraph* genannt wird. Um die Richtung der Beziehung zu verdeutlichen, wird eine Kante in einem gerichteten Graphen durch einen Pfeil dargestellt. Somit ist die Kante $\{i, j\}$ von der Kante $\{j, i\}$ zu unterscheiden. Ein Beispiel für einen solchen Digraph ist eine Nahrungskette, bei der zum Beispiel ein Braunbär den Lachs frisst, jedoch umgekehrt der Lachs nicht den Bären frisst.

Ein Graph enthält keine Schleifen. Eine *Schleife* ist eine Kante $\{i, j\}$, dessen Anfangs- und Endpunkt identisch sind ($i = j$) (Kolaczyk (2009)). Ein Graph enthält folglich nur Kanten die zwei nicht identische Knoten verbinden. Des Weiteren enthält ein Graph keine multiplen Kanten. Von einer *multiplen Kante* ist die Rede, wenn zwischen zwei Knoten mehrere Kanten existieren. Existieren in einem Netzwerk multiple Kanten, so spricht man von einem *Multigraphen*, ansonsten von einem *einfachen Graphen* (Kolaczyk (2009)). Multigraphen können zum Beispiel benutzt werden, um verschiedene Arten von Beziehungen in einem sozialen Netzwerk zu modellieren. Dabei könnte in einem sozialen Netzwerk beispielsweise unterschieden

werden, ob man im realen Leben miteinander befreundet ist, oder ob es sich nur um eine Internetbekanntschaft handelt. Darüber hinaus könnte in einem solchen Multigraphen zum Beispiel auch die Schulzugehörigkeit miteinbezogen werden.

Zwei Knoten $i, j \in V$ eines Graphen heißen *benachbart* beziehungsweise *adjazent*, wenn eine Kante $i, j \in E$ in dem Graphen G existiert, die diese beiden Knoten verbindet (Kolaczyk (2009)). Analog dazu heißen zwei Kanten adjazent, wenn beide über einen gemeinsamen Knoten aus V verbunden sind.

Die Struktur eines Graphen wird durch seine symmetrische Nachbarschaftsmatrix (auch *Adjazenzmatrix* genannt) bestimmt. Die *Nachbarschaftsmatrix* Y eines Graphen ist eine Matrix vom Format $N_v \cdot N_v$, deren Elemente (y_{ij}) die Anzahl der Kanten zwischen den Knoten i und j angeben (Kolaczyk (2009)). Für einen einfachen Graphen sind diese Elemente ausschließlich 0 oder 1. Besteht eine Kante zwischen Knoten i und Knoten j so ist $(y_{ij}) = 1$, ansonsten $(y_{ij}) = 0$. Ein Beispiel für eine solche Nachbarschaftsmatrix mit dazugehörigem Graphen ist in Abbildung 1 zu sehen.

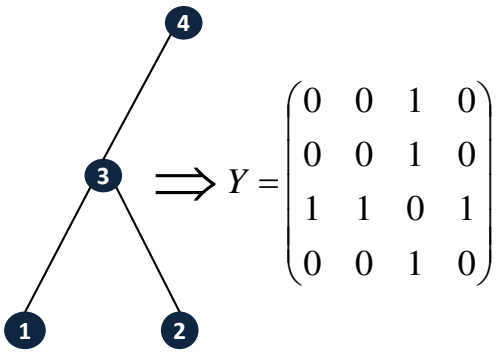

Abbildung 1: Beispiel eines Graphen mit dazugehöriger Nachbarschaftsmatrix.

Aus der Nachbarschaftsmatrix können weitere nützliche Informationen über den zugrunde liegenden Graphen gewonnen werden. Bildet man zum Beispiel die Zeilensumme der i-ten Zeile $Y_{i+} = \sum_{j=1}^{N_v} Y_{ij}$, erhält man den Knotengrad von Knoten i. Der *Knotengrad* eines Knotens, welcher auch *Degree* eines Knotens genannt wird, entspricht der Anzahl an Kanten, die in diesem Knoten enden oder beginnen. In einem ungerichteten Graphen entspricht der Knotengrad d_v eines Knotens somit der Anzahl an Kanten, die den Knoten v enthalten. Bei einem gerichtetem Graphen ist die Nachbarschaftsmatrix jedoch nur in Sonderfällen symmetrisch, da eine Kante von i nach j nicht unbedingt auch eine Kante von j nach i impliziert. Bei gerichteten Graphen unterscheidet man deshalb zwischen Außen- und Innengrad eines Knotens. Der

Außengrad eines Knotens zählt die Kanten, die von ihm ausgehen und entspricht Y_{i+}. Der *Innengrad* hingegen entspricht der Anzahl an Kanten die in ihm enden und kann für Knoten j durch $Y_{+j} = \sum_{i=1}^{N_v} Y_{ij}$ bestimmt werden. Der Knotengrad eines Knotens ist somit ein intuitives Maß, um seine Eingebundenheit in das Gesamtnetzwerk zu beschreiben. Als zentrale Akteure eines Netzwerkes gelten Knoten mit einem hohen Knotengrad. In Abbildung 1 hat Knoten Nummer 3 einen Knotengrad von 3, und ist damit der zentrale Akteur des kleinen beispielhaften Netzwerkes. Sortiert man die Knotengrade aller Knoten eines Netzwerks in aufsteigender Reihenfolge, erhält man die *Degree-Sequenz* des Netzwerks. Ein weiterer wichtiger Begriff in diesem Zusammenhang ist die *Degree-Verteilung*. Sei f_d der Anteil an Knoten $v \in V$ mit einem Knotengrad $d_v = d$. Die Sammlung $\{f_d\}_{d \geqslant 0}$ wird Degree-Verteilung genannt. Einfacher ausgedrückt, entspricht die Degree-Verteilung dem Histogramm der Degree-Sequenz. Die Degree-Sequenz des Netzwerks aus Abbildung 1 entspricht $\{1, 1, 1, 3\}$. Für die daraus resultierende Degree-Verteilung erhält man daher $\{f_1 = \frac{3}{4}, f_3 = \frac{1}{4}\}$. Die Definitionen in diesem Absatz stammen aus Kolaczyk (2009).

Ein weiterer Begriff, der im Laufe dieser Arbeit benützt wird, ist die *Dichte* eines Netzwerkes. Diese lässt sich mit der Formel

$$dichte(Netzwerk) = \frac{|E_{Netzwerk}|}{|V_{Netzwerk}|(|V_{Netzwerk}| - 1)/2} \tag{1}$$

berechnen. Die Dichte eines Netzwerkes entspricht somit der Anzahl an Kanten im Netzwerk, geteilt durch die Anzahl möglicher Kanten des Netzwerkes. Dies gilt auch für gerichtete Netzwerke, weshalb die Dichte für ein gerichteten Netzwerkes mit der Formel

$$dichte(Netzwerk) = \frac{|E_{Netzwerk}|}{|V_{Netzwerk}|(|V_{Netzwerk}| - 1)}$$

bestimmt werden kann.

Es sei angemerkt, dass eine Nachbarschaftsmatrix für große Netzwerke auch Probleme mit sich bringen kann. So ist der Speicherbedarf einer solchen Matrix quadratischer Natur, was für große Netzwerke einen hohen Speicherbedarf bedeutet. Insbesondere bei nicht eng vernetzten Netzwerken ist der Speicherbedarf groß, obwohl viele Einträge der Nachbarschaftsmatrix in solch einem Fall Nulleinträge sind. Besonders für solche Netzwerke bietet es sich an, anstatt der Nachbarschaftsmatrix eine *Adjazenzliste* zu betrachten. Hierbei handelt es sich um eine Liste, in welcher der i-te Eintrag alle Knoten aufführt, zu denen der i-te Knoten eine direkte Kante hat. Diese Art der Repräsentation von Netzwerkdaten geht außer bei dichten Netzwerken mit einer Reduzierung des Speicheraufwands einher.

3 Exponential Random Graph Modelle

Bei Netzwerkdaten sind die Beziehungen zwischen den einzelnen Akteuren von zentralem Interesse. So interessiert man sich grundsätzlich für das Existieren oder Fehlen von Kanten zwischen verschiedenen Akteuren eines Netzwerkes und mögliche Einflussvariablen die das Zustandekommen eines Netzwerkes beeinflussen. Falls Beziehungen zwischen Beobachtungen bestehen, können diese jedoch nicht als unabhängig von einander betrachtet werden. So hängt die Existenz einer Kante zwischen zwei Akteuren explizit vom Zustand des restlichen Netzwerkes ab. Ist man beispielsweise an der Wahrscheinlichkeit für die Existenz einer Kante zwischen zwei Personen in einem Netzwerk aus Freundschaften interessiert, so könnte zum Beispiel die Anzahl gemeinsamer Freunde einen großen Einfluss auf diese Wahrscheinlichkeit haben. Die Anzahl gemeinsamer Freunde ist dabei dem restlichen Netzwerk zu entnehmen. Demnach lässt sich in diesem Beispiel die Wahrscheinlichkeit für die Existenz einer Kante zwischen zwei Akteuren nur unter Berücksichtigung des restlichen Netzwerkes vernünftig modellieren. Gängige statistische Analysen nehmen jedoch an, dass Beobachtungen unabhängig voneinander sind und können deshalb grundsätzlich nicht bei der Analyse von Netzwerkdaten eingesetzt werden. Ein sehr vielversprechender Ansatz zur Modellierung von Netzwerken ist hingegen das Exponential Random Graph Modell, welches im Folgenden vorgestellt wird.

3.1 Definition des Exponential Random Graph Modells

Das Exponential Random Graph Modell beschreibt die Wahrscheinlichkeitsverteilung für alle Graphen beziehungsweise Netzwerke mit einer festen Knotenzahl. Sei Y eine matrixwertige Zufallsvariable welche die Nachbarschaftsmatrix eines Netzwerkes repräsentiert. Die Wahrscheinlichkeitsverteilung der Zufallsvariable Y lässt sich in folgender Form ausdrücken:

$$P_{\theta,\mathcal{Y}}(Y = y) = \frac{1}{k(\theta, \mathcal{Y})} \cdot exp\{\theta^T \cdot g(y)\} \tag{2}$$

mit

- $\mathcal{Y}$ = Menge aller möglichen Nachbarschaftsmatrizen.
- y = Konkrete Ausprägung eines Netzwerks, wobei y $\in \mathcal{Y}$.
- $g(y)$ = Vektor bestehend aus Netzwerk Statistiken.

- θ = Vektor mit Gewichten bzw. Modellparametern.
- $k(\theta)$ = Normalisierungskonstante.

$\mathcal{Y}$ ist die Menge aller möglichen Nachbarschaftsmatritzen mit einer festen Knotenzahl N_v. Deren Elemente dürfen jedoch keine multiplen Kanten oder Schleifen enthalten. Bei y handelt es sich um eine konkrete Ausprägung eines Netzwerkes, wobei diese ein Element aus $\mathcal{Y}$ sein muss. Der Vektor $g(y)$ enthält Netzwerk Statistiken, welche auf der Nachbarschaftsmatrix Y basieren. Es gibt keine allgemeingültige Antwort auf die Frage, welche Netzwerk Statistiken aufgenommen werden sollen, um ein gutes Modell zu erhalten. Einen Überblick möglicher Netzwerk Statistiken gibt Abschnitt 3.6. Möchte man erklärende Zusatzinformationen als externe Variablen X in das Modell mit aufnehmen, muss man in Gleichung (2) $g(y)$ durch $g(y, X)$ ersetzen:

$$P_{\theta,\mathcal{Y}}(Y = y) = \frac{1}{k(\theta, \mathcal{Y})} \cdot exp\{\theta^T \cdot g(y, X)\}.$$

Mögliche erklärende Zusatzinformationen X könnten z.B. das Alter oder das Geschlecht der Akteure sein. Die Modellparameter θ messen auf gewisse Art und Weise die Stärke der Effekte der dazugehörigen erklärenden Variablen $g(y)$. Die Normalisierungskonstante

$$k(\theta) = \sum_{y \in \mathcal{Y}} exp\{\theta^T \cdot g(y)\}$$

ist ein Faktor welcher gewährleistet, dass es sich bei Gleichung (2) um eine Wahrscheinlichkeitsverteilung mit einem Wertebereich [0,1] handelt (Kolaczyk (2009) und Hummel et al. (2012)).

Eine genaue Berechnung von $k(\theta)$ ist außer für sehr kleine Netzwerke problematisch, da die in der Normalisierungskonstante enthaltene Summe über alle möglichen Ausprägungen der Zufallsvariable Y summiert. Die Anzahl möglicher Netzwerke $\mathcal{Y}$ wird jedoch schnell unglaublich groß. So sind für ein ungerichtetes Netzwerk mit n Knoten $2^{\frac{n(n-1)}{2}}$ verschiedene Netzwerke möglich. Bei nur 15 Knoten gibt es in einem ungerichteten Netzwerk demnach bereits $1.57 \cdot 10^{57}$ mögliche Ausprägungen der Zufallsvariable Y. Deshalb ist es in der Regel nicht möglich, die Normalisierungskonstante numerisch zu bestimmen. Damit wird der Umgang mit der Normalisierungskonstante zum entscheidenden Problem des Exponential Random Graph Modells (Hummel et al. (2012)), denn für eine klassische Maximum-Likelihood-Schätzung ist eine Berechnung der Normalisierungskonstante unumgänglich.

Angenommen man hat ein beobachtetes Netzwerk y_{obs} und einen Vektor mit erklärenden Variablen $g(y_{obs})$. Nun möchte man den Vektor θ mit der Maximum-

Likelihood-Methode schätzen. Dafür muss die Log-Likelihood Funktion

$$\ell(\theta) = (\theta)^T \cdot g(y_{obs}) - log(k(\theta)) \tag{3}$$

maximiert werden. Eine analytische Lösung dieser Gleichung ist jedoch grundsätzlich nicht möglich, da auf Grund der großen Anzahl an möglichen Netzwerken die Normalisierungskonstante $k(\theta)$ nicht berechnet werden kann. Eine Ausnahme bilden lediglich wie bereits erwähnt besonders kleine Netzwerke. Für größere Netzwerke ist es dennoch möglich mit alternativen Schätzmethoden geeignete Schätzer zu entwickeln. Einige der wichtigsten werden in Abschnitt 3.4 über die Schätzung von Exponential Random Graph Modellen vorgestellt. Die meisten dieser Schätzmethoden benutzen simulationsgestützte Verfahren, um die Berechnung der Normalisierungskonstante zu umgehen. Auf die Simulation zufälliger Netzwerke soll nun genauer eingegangen werden.

3.2 Simulation zufälliger Netzwerke

Üblicherweise basiert die Simulation zufälliger Netzwerke aus einer Zielverteilung $P_\theta(y)$ auf dem Markov Chain Monte Carlo (MCMC) Algorithmus. Hierfür ist es sinnvoll das Exponential Random Graph Modell umzuformulieren, so dass sich die bedingte Wahrscheinlichkeit für die Existenz einer Kante zwischen zwei Knoten damit berechnen lässt. Das Exponential Random Graph Modell für die Existenz einer Kante soll nun vorgestellt werden.

3.2.1 ERGM für die Existenz einer Kante

Vorab einige notwendige Notationen:
Y_{ij} sei eine binäre Variable, welche angibt, ob eine Kante zwischen den Knoten i und j existiert ($Y_{ij} = 1$) oder nicht ($Y_{ij} = 0$). Y_{-ij} bezeichnet den Status aller übrigen Kanten aus Y, also allen Kanten außer der zwischen den Knoten i und j. Das Netzwerk welches aus Y_{-ij} entsteht wenn man eine Kante zwischen den Knoten i und j hinzufügt ($Y_{ij} = 1$) sei y_{ij}^+. Andernfalls, wenn keine Kante zwischen den Knoten i und j existiert ($Y_{ij} = 0$), so ergibt sich das Netzwerk y_{ij}^-. Für den bedingten Logit ergibt sich nun Folgendes (Hummel et al. (2012) und Lusher et al. (2013)):

$$
\begin{aligned}
\text{logit}[P_\theta(Y_{ij} = 1|Y_{-ij} = y_{-ij})] &= \log(\frac{P_\theta(Y_{ij} = 1|Y_{-ij} = y_{-ij})}{P_\theta(Y_{ij} = 0|Y_{-ij} = y_{-ij})}) \\
&= \log(\frac{P_\theta(Y = y_{ij}^+)}{P_\theta(Y = y_{ij}^-)}) \\
&= \log(\frac{exp\{\theta^T \cdot g(y_{ij}^+)\}/k(\theta)}{exp\{\theta^T \cdot g(y_{ij}^-)\}/k(\theta)}) \\
&= \log(exp\{\theta^T \cdot [g(y_{ij}^+) - g(y_{ij}^-)]) \\
&= \theta^T \cdot [g(y_{ij}^+) - g(y_{ij}^-)] \\
&= \theta^T \cdot \delta_g(y)_{ij} \qquad (4)
\end{aligned}
$$

Der Vektor θ^T ist dabei erneut ein Vektor mit Modellparametern. Der Teil des Ergebnisses, bei dem die Netzwerk Statistiken des Netzwerkes y_{ij}^- von denen des Netzwerkes y_{ij}^+ subtrahiert werden, also $[g(y_{ij}^+) - g(y_{ij}^-)]$, wird auch *Change-Statistik* genannt. Die Change-Statistik $\delta_g(y)_{ij}$ entspricht demnach der Veränderung in $g(y)$ ausgehend von einem Netzwerk, bei dem es keine Kante zwischen den beiden Knoten i und j gibt $(g(y_{ij}^-))$, hin zu einem Netzwerk, bei dem eine Kante zwischen den Knoten i und j existiert $(g(y_{ij}^+))$, wobei der Rest des Netzwerkes bei beiden identisch ist $(Y_{-ij} = y_{-ij})$. Entscheidend ist in Gleichung (4), dass diese keine Normalisierungskonstante enthält. Dies wird bei der Simulation von zufälligen Netzwerken ausgenutzt. Bevor genauer darauf eingegangen wird, soll jedoch zunächst die Change-Statistik ausführlicher erläutert werden.

3.2.2 Change-Statistik

Um ein besseres Verständnis für die Change-Statistik $\delta_g(y)_{ij}$ aus Gleichung (4) zu ermöglichen, soll diese nun explizit für die Anzahl an Kanten, 2-stars und triangles beschrieben werden. Diese sind Beispiele für häufig genutzte Netzwerk Statistiken. Ein *2-star* ist definiert als ein Knoten, der zu zwei anderen Knoten eine Kante besitzt. Ein Beispiel für einen 2-star ist in Abbildung 2 links zu sehen. Ein *triangle*, welcher in Abbildung 2 auf der rechten Seite abgebildet ist, entspricht der allgemeinen Vorstellung eines Dreiecks. Es handelt sich dabei also um drei Knoten die jeweils direkt miteinander verbunden sind.

Um die Change-Statistik zu bestimmen, betrachtet man die Veränderung ausgehend von einem Netzwerk, bei dem es keine Kante zwischen den beiden Knoten i und

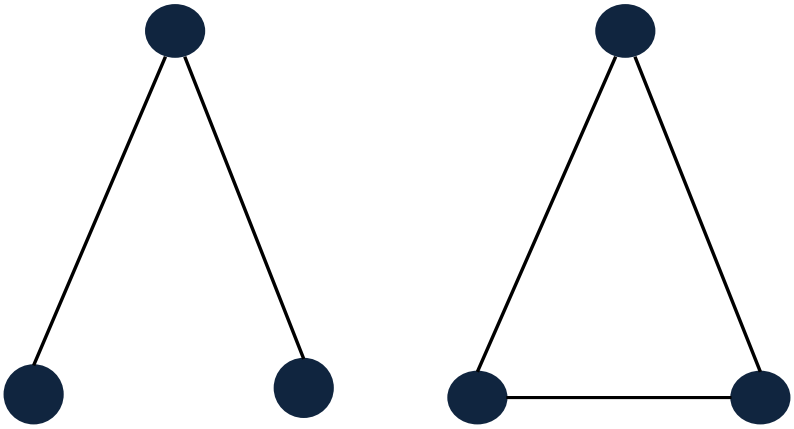

Abbildung 2: Abbildung eines 2-star (links) und eines triangles (rechts).

j gibt, hin zu einem Netzwerk, bei dem eine Kante zwischen den Knoten i und j existiert, wobei der Rest des Netzwerkes bei beiden identisch ist. Die Change-Statistik für die Anzahl Kanten nimmt demnach immer den Wert 1 an, da sich die Anzahl Kanten durch das Hinzufügen einer Kante zwischen i und j genau um 1 erhöht.

Die Change-Statistik für die Anzahl 2-stars wird mit folgender Formel berechnet (Snijders et al. (2006)):

$$\tilde{y}_{i+}(i,j) + \tilde{y}_{j+}(i,j),$$

wobei $\tilde{y}$ dem Netzwerk entspricht, welches aus Y_{-ij} entsteht, wenn man keine Kante zwischen den Knoten i und j hinzufügt ($\tilde{y} = y_{ij}^{-}$). Der Ausdruck $\tilde{y}_{i+}(i,j)$ steht für den Knotengrad, den der Knoten i im Netzwerk $\tilde{y}$ hat, und $\tilde{y}_{j+}(i,j)$ für den des Knoten j im selben Netzwerk. Demnach entspricht die Change-Statistik für die Anzahl 2-stars bei Betrachtung eines Netzwerkes von Freundschaften der Anzahl der Freunde von i addiert mit der Anzahl an Freunden von j. Dies erklärt sich dadurch, dass man mit der neu hinzugefügten Kante zwischen i und j mit jeder anderen Kante von i und j, also jedem Freund von i und j, einen weiteren 2-star bilden kann.

Die Change-Statistik für die Anzahl triangles lässt sich mit folgender Formel bestimmen (Snijders et al. (2006)):

$$\sum_{h \neq i,j} y_{ih} \cdot y_{hj} \ .$$

Dies entspricht der Anzahl an two-paths zwischen i und j, was in einem Netzwerk von Freundschaften wiederum der Anzahl gemeinsamer Freunde von i und j entspricht.

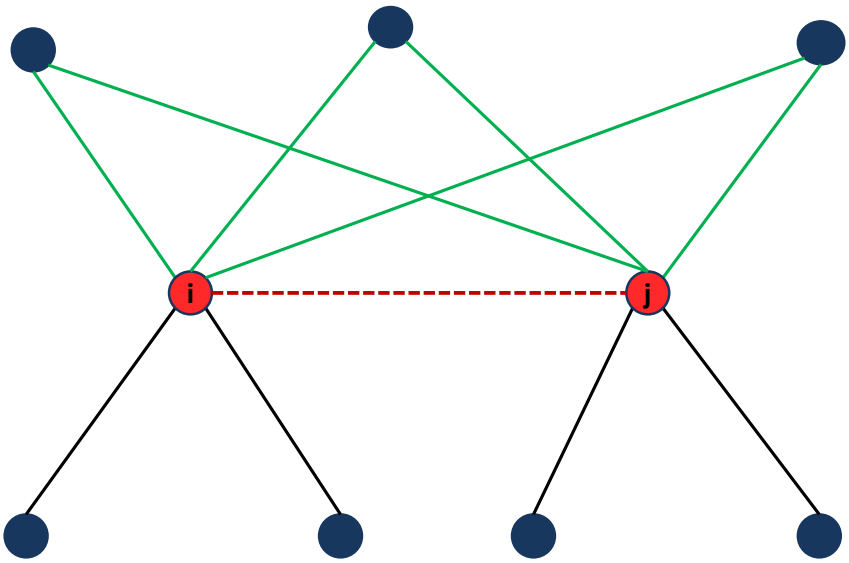

Abbildung 3: Visualisierung der Change-Statistik für die Anzahl triangles.

Diese Tatsache soll anhand von Abbildung 3 verdeutlicht werden.

Die beiden roten Knoten in Abbildung 3 stehen für die Knoten i und j. Demnach geht es um die Existenz der rot gestrichelten Kante zwischen i und j, gegeben das restliche Netzwerk. Um nun die Change-Statistik für die Anzahl triangles zu ermitteln, muss man bestimmen, wie viele neue triangles durch das Hinzufügen der rot gestrichelten Kante im Netzwerk entstehen. In dem Netzwerk aus Abbildung 3 entstehen genau drei neue triangles durch das Hinzufügen der rot gestrichelten Kante. Diese sind in der Abbildung grün markiert und entstehen genau dann, wenn die beiden Knoten, zwischen denen eine Kante hinzugefügt werden soll, einen gemeinsamen Freund haben und demnach das Hinzufügen der roten Kante das Dreieck „schließt“.

3.2.3 Markov-Chain-Monte-Carlo Algorithmus

Bei der Simulation von zufälligen Netzwerken, beziehungsweise Zufallsgraphen, aus einer Zielverteilung $P_\theta(y)$ durch den Markov-Chain-Monte-Carlo Algorithmus, wird eine Sequenz von Netzwerken generiert, bei der die Netzwerke nacheinander durch kleine Veränderungen aktualisiert werden. Genauer gesagt, wird ein Metropolis Algorithmus verwendet, wie u.a. in Lusher et al. (2013) beschrieben, um zufällige Netzwerke aus einem Exponential Random Graph Modell zu ziehen. Dieser Algorithmus tauchte erstmals in dem Paper von Metropolis et al. (1953) auf. Anfangs wurde er

hauptsächlich in der Physik und der Chemie verwendet; erst mit der Veröffentlichung des Papers von Hastings (1970) fand er seinen Weg in die Statistik Literatur (Geyer und Thompson (1992))).

Startpunkt ist dabei ein beliebiges Netzwerk mit fester Knotenzahl N_v. Als möglicher Startpunkt würde demnach beispielsweise auch das leere Netzwerk dienen. Anschließend wird ein Element aus der Nachbarschaftsmatrix Y per Zufall ausgewählt, um an diesem eine Änderung vorzunehmen. Es wird also bei einem zufällig gewählten Knotenpaar eine Verbindung hinzugefügt, beziehungsweise falls bereits eine Verbindung besteht, wird diese entfernt. Besitzt das so entstandene Netzwerk y^{neu} eine höhere Wahrscheinlichkeit als das vorherige y^{alt}, so wird es akzeptiert. In diesem Fall ist das nächste Netzwerk in der Sequenz das neue Netzwerk y^{neu}. Besitzt das neue Netzwerk allerdings eine geringere Wahrscheinlichkeit als das vorherige, wird es mit einer Wahrscheinlichkeit akzeptiert, die proportional zum Verhältnis der Wahrscheinlichkeiten von vorherigem und neuem Netzwerk ist. Falls das neue Netzwerk y^{neu} nicht akzeptiert wird, ist das nächste Netzwerk in der Sequenz erneut das vorherige Netzwerk y^{alt}.

Man erstellt demnach eine Sequenz von Netzwerken $y^{(0)}$, $y^{(1)}$, ..., $y^{(M-1)}$, $y^{(M)}$. Im m-ten Iterationsschritt wird dabei vorgegangen wie in Lusher et al. (2013) beschrieben:

1. Aus dem aktuellen Netzwerk $y^{(m-1)}$ wird ein Knotenpaar i, j per Zufall ausgewählt, wobei $i, j \in 1, ..., N_v$.

2. Das durch Hinzufügen bzw. Entfernen einer Verbindung an dem Knotenpaar i, j entstandene Netzwerk y^{neu} entspricht $y^{(m-1)}$ bis auf $y_{ij}^{neu} = 1$ - $y_{ij}^{(m-1)}$

3. Das vorgeschlagene Netzwerk y^{neu} wird mit der Wahrscheinlichkeit $\min\left\{1, \frac{P_\theta(Y = y^{neu})}{P_\theta(Y = y^{(m-1)})}\right\}$ akzeptiert.

4. Wird das vorgeschlagene Netzwerk akzeptiert, entspricht das nächste Netzwerk in der Sequenz $y^{(m)} = y^{neu}$, ansonsten bleibt es das alte: $y^{(m)} = y^{(m-1)}$.

Um das Verhältnis von $\frac{P_\theta(Y = y^{neu})}{P_\theta(Y = y^{(m-1)})}$ zu bestimmen, muss ähnlich wie in Gleichung (4), keine Normalisierungskonstante $k(\theta)$ bestimmt werden, da die Berechnung der Change-Statistiken ausreicht:

$$\begin{aligned} log\left\{\frac{P_\theta(Y = y^{neu})}{P_\theta(Y = y^{(m-1)})}\right\} &= log\left\{\frac{P_\theta(Y_{ij} = 1 - y^{(m-1)} | Y_{-ij} = y_{-ij}}{P_\theta(Y_{ij} = y^{(m-1)} | Y_{-ij} = y_{-ij}}\right\} \\ &= \theta^T \cdot [g(y^{neu}) - g(y^{(m-1)})] \end{aligned}$$

So kann man durch simulationsgestützte Verfahren die Berechnung der Normalisierungskonstante umgehen.

Das Ergebnis nach einer ausreichend großen Anzahl an Iterationen ist ein zufälliges Netzwerk, welches als Ziehung aus der Zielverteilung $P_\theta(y)$ gesehen werden kann. Alle vorherigen Ziehungen werden als *Burn In* bezeichnet und bleiben unberücksichtigt. Dieser Burn In ist notwendig, damit die Markov-Kette nicht mehr von ihrem beliebig gewählten Anfangszustand abhängt, denn nur bei ausreichend großem Burn In ist ein gezogenes Netzwerk unabhängig vom Startpunkt. Jedes anschließend gezogene Netzwerk kann auch als Ziehung aus der Zielverteilung angesehen werden, weshalb mehrere zufällige Netzwerke aus der gleichen Kette gezogen werden können. Zwei direkt nacheinander gezogene Netzwerke sind jedoch offensichtlich stark korreliert, da in jeder Iteration maximal eine Kante im Netzwerk verändert wird. Deshalb lässt man zwischen den gezogenen Netzwerken jeweils t Iterationen aus. Der Wert von t wird *Thinning* genannt und sollte möglichst groß sein, um die Autokorrelation zwischen den gezogenen Netzwerken möglichst gering zu halten (Lusher et al. (2013)).

3.3 Degeneration

Ein nützliches stochastisches Modell sollte nach Handcock (2003) u.a. den größten Teil seiner Wahrscheinlichkeitsmasse auf die Netzwerke legen, welche eine hohe Wahrscheinlichkeit haben, durch den tatsächlich zugrundeliegenden realen Entstehungsprozess generiert zu werden. Wenn ein Modell hingegen einen unangemessen großen Anteil seiner Wahrscheinlichkeitsmasse auf nur sehr wenige der möglichen Netzwerke $\mathcal{Y}$ legt, so spricht man von einem degenerierten Modell (Handcock (2003)). Genauer gesagt wird hierbei oft ein großer Teil der Wahrscheinlichkeitsmasse auf unrealistische Netzwerke, wie dem leeren Netzwerk (d.h., $Y_{ij} = 0 \ \forall i, j$) oder dem vollen Netzwerk (d.h., $Y_{ij} = 1 \ \forall i, j$) gelegt. Zwar mag es reale Netzwerke geben, die beinahe leer oder voll sind, doch wie in Handcock (2003) beschrieben, werden diese wohl nicht Gegenstand einer Netzwerkanalyse sein.

Einige Modelle zeigen darüber hinaus ein sogenanntes nahezu-degeneriertes Verhalten. Modelle die ein solches Verhalten zeigen, legen einen großen Teil ihrer Wahrscheinlichkeitsmasse auf das volle oder das leere Netzwerk oder eine Mischung aus beiden. Ein nahezu-degeneriertes Modell liegt vor, wenn die erwartete Statistik $E_\theta(g(Y))$ in der Nähe des Randes der konvexen Hülle C aller möglichen Statistiken liegt. Die Menge aller Netzwerke auf dem Rand der konvexen Hülle ist dabei $deg(\mathcal{Y}) = \{y \in \mathcal{Y} : g(y) \in rbd(C)\}$ (Handcock (2003)).

Mit Vorliebe in ERGMs verwendete, traditionelle Zählstatistiken, wie zum Beispiel die bereits in Abschnitt 3.2.2 vorgestellten 2-stars und triangles, führen häufig zur Degeneration. Grund dafür ist, dass bei solchen Statistiken das Hinzufügen einer einzelnen Kante den Wert der Change-Statistiken für andere Kanten enorm vergrößern kann, wodurch eine Kettenreaktion ausgelöst werden kann. Dabei vergrößert die Hinzunahme einer Kante die Change-Statistiken anderer möglicher Kanten des Netzwerkes. Diese werden nun, falls der entsprechende Parameter positiv ist, aufgrund der größeren Change-Statistik wahrscheinlicher gebildet und beeinflussen ihrerseits wiederum die Change-Statistiken anderer Kanten wodurch es zum sogenannten Lawineneffekt kommt (Snijders et al. (2006)).

Exponential Random Graph Modelle sind folglich anfällig für das Problem der Degeneration (Snijders et al. (2006), Schweinberger (2011)). Die Problematik dabei ist, dass degenerierte Modelle zu einer sehr schlechten Approximation der wahren Wahrscheinlichkeitsfunktion führen. Darüber hinaus können degenerierte Modelle die Konvergenz der simulationsgestützten Schätzmethoden aus Abschnitt 3.4 verhindern (Handcock (2003)). In einem solchen Fall schaffen es die simulationsgestützten Schätzmethoden nicht den Maximum Likelihood Schätzer zu finden, woraufhin es auch passieren kann, dass Schätzungen der Parameterwerte ausgegeben werden, bei denen es sehr unwahrscheinlich ist, dass ein ERGM mit diesen Parameterwerten Netzwerke generiert, die dem zu schätzenden Netzwerk ähnlich sind (Caimo und Friel (2011)). Dies liegt vor allem daran, dass degenerierte Modelle instabil sind und schon extrem kleine Veränderungen der Parameterwerte große Veränderungen in den simulierten Netzwerken bewirken können. Dadurch kann die Konvergenz des Algorithmus insofern gestört werden, als während der Simulation zufälliger Netzwerke volle oder leere Netzwerke generiert werden.

Demnach sind Modelle erwünscht, die vom Bereich der Degeneration weit entfernt und stabil sind. Ein Modell ist stabil, wenn kleine Veränderungen der Parameterwerte nur kleine Veränderungen in der probabilistischen Struktur des Modells bewirken. Bei einem instabilen Modell können hingegen schon sehr geringe Veränderungen der Parameterwerte zu großen Unterschieden in den Netzwerkstrukturen führen. Somit ist es möglich, dass sehr ähnliche Parameterwerte für deutlich unterschiedliche Netzwerkstrukturen stehen. Ein Modell gilt als instabil, wenn es nah an $rbd(C)$ ist (Handcock (2003)). Somit sind die Bedingungen für Stabilität und Degeneration eng miteinander verbunden.

Einen Ansatz, um das Problem der Degeneration zu entschärfen, bietet die Verwendung von gewichteten Statistiken. Diese werden in Abschnitt 3.7 vorgestellt. Nun

wird zunächst auf die Schätzung der Modellparameter eingegangen.

3.4 Schätzung von Exponential Random Graph Modellen

Man gehe von der generellen Definition eines Exponential Random Graph Modells aus Gleichung (2) aus. Der Maximum Likelihood Schätzer (MLE) für den Vektor der Modellparameter θ ist definiert als $\hat{\theta} = argmax_\theta \ \ell(\theta)$, wobei $\ell(\theta)$ die Log-Likelihood Funktion aus Gleichung (3) ist. Eine solche direkte Maximierung der Log-Likelihood ist jedoch aufgrund der unberechenbaren Normalisierungskonstante $k(\theta)$ in der Log-Likelihood Funktion grundsätzlich nicht möglich.

Als Folge gibt es im Kontext der Exponential Random Graph Modelle einige unterschiedliche Herangehensweisen bei der Schätzung der Modellparameter. Die große Herausforderung für alle alternativen Methoden ist es, eine Lösung für den Umgang mit der Normalisierungskonstante zu finden. Im Folgenden sollen nun einige der wichtigsten Schätzmethoden vorgestellt werden.

3.4.1 Maximum Pseudo-Likelihood Schätzung

Die wohl einfachste und mit dem geringsten Rechenaufwand verbundene Herangehensweise für die Schätzung der Parameter des Exponential Random Graph Modells aus Gleichung (2), ist die Maximum Pseudo-Likelihood Schätzung, welche in Strauss und Ikeda (1990) vorgestellt wird. Diese basiert auf der sogenannten Pseudo-Likelihood Funktion, welche folgendermaßen definiert ist:

$$\prod_{i,j \in \{1,2,\ldots,N_v\}, j>i} P_\theta(Y_{ij} = y_{ij} | Y_{-ij} = y_{-ij}) \quad . \tag{5}$$

Demnach wird für jede potentielle Kante des Netzwerks die bedingte Wahrscheinlichkeit gegeben das restliche Netzwerk aus Gleichung (4) berechnet. Das Produkt dieser einzelnen bedingten Wahrscheinlichkeiten ergibt die Pseudo-Likelihood Funktion. Der Maximum Pseudo-Likelihood Schätzer ist der Wert θ, der die logarithmierte Pseudo-Likelihood Funktion

$$\ell_p(\theta) = \sum_{i,j \in \{1,2,\ldots,N_v\}, j>i} log[P_\theta(Y_{ij} = y_{ij} | Y_{-ij} = y_{-ij})]$$

maximiert. Demnach ist der Maximum Pseudo-Likelihood Schätzer definiert als:

$$\theta_p = argmax_\theta \ \ \ell_p(\theta) \quad . \tag{6}$$

Die Maximierung der logarithmierten Pseudo-Likelihood Funktion kann dabei, wie von Strauss und Ikeda (1990) gezeigt, durch einfache logistische Regression erfolgen. Der Vorteil einer Maximum-Pseudo-Likelihood Schätzung im Kontext der Exponential Random Graph Modelle ist, dass selbst kompliziertere Modelle ohne große Probleme und mit vergleichsweise geringem Rechenaufwand gefittet werden können (Robins et al. (2007)).

Allerdings wird bei einer logistischen Regression angenommen, dass die Beobachtungen unabhängig voneinander sind. Diese Annahme ist bei Netzwerkdaten jedoch grundsätzlich nicht gegeben, da es sich dabei, wie bereits in Abschnitt 2 beschrieben, um relationale Daten handelt. Folglich können die Schätzungen der Parameter bei einer Maximum Pseudo-Likelihood Schätzung verzerrt sein, da die Abhängigkeitsstruktur der Daten ignoriert wird. Nur wenn die Variablen für die Existenz einer Kante Y_{ij} bedingt unabhängig voneinander sind, ist der Maximum Pseudo-Likelihood Schätzer ein unverzerrter Schätzer. Falls nur schwache Abhängigkeitsstrukturen vorliegen, sollten die Schätzungen auch einigermaßen gute Approximationen für den Maximum Likelihood Schätzer liefern (Kolaczyk (2009)). Liegen dagegen starke Abhängigkeitsstrukturen im Netzwerk vor, kann der Maximum Pseudo-Likelihood Schätzer erheblich verzerrt sein.

3.4.2 Markov Chain Monte Carlo Maximum Likelihood Estimation Verfahren

Die Maximierung der wahren Log-Likelihood Funktion aus Gleichung (3) ist einer Maximierung der Pseudo-Likelihood Funktion aus Gleichung (5) vorzuziehen, da das Verhalten der Maximum Pseudo-Likelihood Schätzung schwer einzuschätzen ist (Hummel et al. (2012)). Wie bereits bei der Einführung des Exponential Random Graph Modells in Abschnitt 3.1 beschrieben, gibt es aufgrund der grundsätzlich unberechenbaren Normalisierungskonstante $k(\theta)$ keine direkte Möglichkeit die Log-Likelihood Funktion nach θ zu maximieren.

Eine indirekte Möglichkeit, dieses Problem zu umgehen, bietet das Markov Chain Monte Carlo Maximum Likelihood Estimation (MCMCMLE) Verfahren aus Hunter und Handcock (2006), welches aus Geyer und Thompson (1992) hervorgeht. Hierbei wird ein willkürlicher Parameter θ_0 festgelegt und daraufhin die folgende

Log-Likelihood Formulierung betrachtet:

$$\begin{aligned}\ell(\theta) - \ell(\theta_0) &= (\theta)^T \cdot g(y_{obs}) - log(k(\theta)) - [(\theta_0)^T \cdot g(y_{obs}) - log(k(\theta_0))] \\ &= (\theta - \theta_0)^T \cdot g(y_{obs}) - log\left\{\frac{k(\theta)}{k(\theta_0)}\right\} \\ &= (\theta - \theta_0)^T \cdot g(y_{obs}) - log(E_{\theta_0}[exp\{(\theta - \theta_0)^T \cdot g(Y)\}]). \qquad (7)\end{aligned}$$

Als Startwert für θ_0 eignet sich dabei der Maximum Pseudo-Likelihood Schätzer aus Gleichung (6). Wie in Hunter und Handcock (2006) beschrieben, kann Gleichung (7) ausgenutzt werden, indem man zufällige Netzwerke $Y_1, Y_2, ..., Y_m$ aus der Zielverteilung $P_{\theta_0}(y)$ durch den Markov-Chain-Monte-Carlo Algorithmus generiert, um $\ell(\theta) - \ell(\theta_0)$ folgendermaßen zu approximieren:

$$\ell(\theta) - \ell(\theta_0) \approx (\theta - \theta_0)^T \cdot g(y_{obs}) - log\left[\frac{1}{m}\sum_{i=1}^{m} exp\{(\theta - \theta_0)^T \cdot g(Y_i)\}\right]. \qquad (8)$$

Das Gesetz der großen Zahlen garantiert dabei die Konvergenz von

$$E_{\theta_0}[exp\{(\theta - \theta_0)^T \cdot g(Y)\}]$$

aus Gleichung (7) gegen

$$\frac{1}{m}\sum_{i=1}^{m} exp\{(\theta - \theta_0)^T \cdot g(Y_i)\}$$

aus Gleichung (8) für $m \to \infty$ (Hunter und Handcock (2006)). Demnach kann durch die Simulation von m Zufallsgraphen und einer anschließenden Maximierung von Gleichung (8) nach θ der Populationsmittelwert beziehungsweise der wahre Parameter durch den Stichprobenmittelwert approximiert werden und man erhält somit eine Approximation für den Maximum Likelihood Schätzer $\hat{\theta}$.

Wie bereits von Geyer und Thompson (1992) angemerkt, funktioniert die Approximation der Likelihood aus Gleichung (7) am besten, wenn θ_0 nah am wahren Parameter Vektor θ ist. Wählt man dagegen ein θ_0 welches nicht nah am wahren Parameter Vektor θ ist, liefert dieses Vorgehen nur ungenaue oder sogar keinerlei Schätzungen. Um dieses Problem zu entschärfen, kann das sogenannte Importance Sampling nach Geyer und Thompson (1992) angewandt werden. Jedoch funktioniert auch diese Methode nur gut, wenn θ_0 nah am wahren Parameter Vektor θ liegt (Hummel et al. (2012)).

3.4.3 Stepping Algorithmus

Der sogenannte Stepping Algorithmus, welcher von Hummel et al. (2012) eingeführt wurde, bietet eine Lösung für das große Problem des MCMCMLE Vorgehens an. Wie bereits im vorherigen Abschnitt 3.4.2 angemerkt, liefert das MCMCMLE Verfahren nur gute Schätzungen, wenn θ_0 nah am wahren MLE $\hat{\theta}$ liegt.

Ziel des Stepping Algorithmus ist es deshalb, θ_0 schrittweise an den unbekannten Maximum Likelihood Schätzer $\hat{\theta}$ zu nähern. Hilfreich ist dabei die Eigenschaft, dass der Maximum Likelihood Schätzer jeder Exponentialfamilie, wenn er existiert, der einzige Vektor θ ist, welcher

$$E_{\hat{\theta}} g(Y) = g(y_{obs}) = \hat{\xi}$$

erfüllt (Barndorff-Nielsen (2014) und Brown (1987)). Die original ERGM Parametrisierung aus Gleichung (2), mit θ als Parameter, wird kanonische Parametrisierung genannt (Hummel et al. (2012)). Die sogenannte mean-value Parametrisierung wird hingegen definiert durch

$$\xi(\theta) = E_\theta(g(Y)).$$

Die Idee ist nun, sich dem MLE schrittweise in der mean-value Parametrisierung anzunähern, indem man vorgibt, dass der MLE nicht $g(y_{obs})$ sei, sondern irgendein Punkt zwischen $g(y_{obs})$ und der Schätzung von $\xi(\theta_0)$ (Hummel et al. (2012)). Dieser Punkt wird im t-ten Iterationsschritt mit $\hat{\xi}_t$ gekennzeichnet und wird im Laufe dieser Arbeit Pseudo-Observation genannt, da er die Rolle der beobachteten Daten anstelle von $g(y_{obs})$ einnimmt. Dabei wird im Laufe des Stepping Algorithmus immer wieder schrittweise von einer kanonischen Parametrisierung in eine mean-value Parametrisierung gewechselt und umgekehrt.

Genauer gesagt, setzt sich der Stepping Algorithmus aus mehreren Schritten zusammen. **Schritt 1** wird nur beim Start des allerersten Durchlaufs des Stepping Algorithmus ausgeführt. Hier wird die Iterationsnummer t gleich 0 gesetzt und ein Startwert für den Vektor der Modellparameter θ_0 festgelegt. Als Startwert für θ_0 wird üblicherweise der Maximum Pseudo-Likelihood Schätzer aus Gleichung (6) verwendet.

Anschließend werden in **Schritt 2** zufällige Netzwerke $Y_1, Y_2, ..., Y_m$ aus der Zielverteilung $P_{\theta_t}(y)$ durch den Markov-Chain-Monte-Carlo Algorithmus generiert. Daraufhin

wird in **Schritt 3** der Stichprobenmittelwert $\bar{\xi}_t$ berechnet:

$$\bar{\xi}_t = \frac{1}{m} \sum_{i=1}^{m} g(Y_i) \quad .$$

In **Schritt 4** wird eine neue Pseudo-Observation $\hat{\xi}_t$ durch Kombinieren von $g(y_{obs})$ und $\bar{\xi}_t$ bestimmt. Die Kombination erfolgt für ein $\gamma_t \in (0, 1]$ durch

$$\hat{\xi}_t = \gamma_t g(y_{obs}) + (1 - \gamma_t)\bar{\xi}_t \quad .$$

Es gilt dabei γ_t so zu wählen, dass es so nah wie möglich an 1 ist, ohne dass $\hat{\xi}_t$ das Innere der konvexen Hülle der Vektoren $g(Y_1), ..., g(Y_m)$ verlässt (Hummel et al. (2012)). Dadurch kann der Stepping Algorithmus der Degeneriertheit entgegenwirken und stabil sein, wohingegen andere Algorithmen wegen der potentiellen Degeneriertheit recht instabil sein können.

Abbildung 4 zeigt zusammenfassend den Fortschritt des Stepping Algorithmus nach Schritt 4. Zu diesem Zeitpunkt ist der Startpunkt θ_0 festgelegt worden, woraufhin durch die Simulation von zufälligen Netzwerken der Stichprobenmittelwert $\bar{\xi}_0$ bestimmt wurde. Daraufhin wurde mit $\hat{\xi}_0$ der ursprüngliche Stichprobenmittelwert $\bar{\xi}_0$ näher an $\hat{\xi}$ gerückt. Die eingekreisten Nummern teilen den Schritten des Stepping Algorithmus ihren Platz in der Abbildung zu.

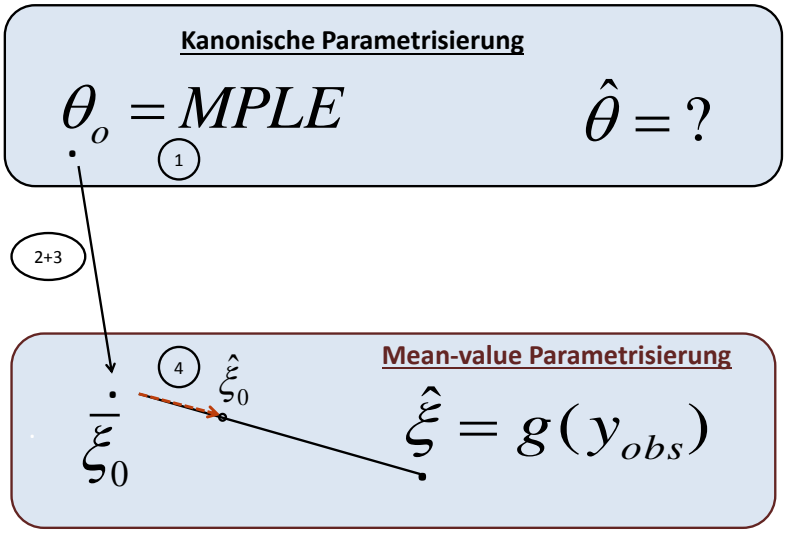

Abbildung 4: Fortschritt des Stepping Algorithmus, nachdem Schritt 4 das erste Mal abgeschlossen wurde.

In **Schritt 5** wird der Vektor θ_{t+1} aus $\hat{\xi}_t$ bestimmt. Dafür wird in Gleichung (8) $g(y_{obs})$ durch $\hat{\xi}_t$ ersetzt:

$$\theta_{t+1} = argmax_\theta \left\{ (\theta - \theta_t)^T \cdot \hat{\xi}_t - log\left[\frac{1}{m} \sum_{i=1}^{m} exp\{(\theta - \theta_t)^T \cdot g(Y_i)\} \right] \right\}. \qquad (9)$$

Der erste Durchlauf des Stepping Algorithmus ist damit beendet. Nun wird t um 1 erhöht und der Algorithmus kehrt zurück zu Schritt 2. Sobald es in Schritt 4 jedoch möglich ist $\gamma_t = 1$ zu setzen und gleichzeitig $\hat{\xi}_t$ im Inneren der konvexen Hülle der Vektoren $g(Y_1), ..., g(Y_m)$ ist, gilt der Stepping Algorithmus als konvergiert (Hummel et al. (2012)). Nun lässt sich, mit θ_{t+1} als Startwert in Gleichung (8), der Maximum Likelihood Schätzer approximieren.

Abbildung 5 veranschaulicht beispielhaft die einzelnen Schritte des Stepping Algorithmus bis hin zur Konvergenz und der damit verbundenen Approximation des Maximum Likelihood Schätzers $\hat{\theta}$. Um Abbildung 5 übersichtlich gestalten zu können, konvergiert der Algorithmus bereits nach dem zweiten Iterationsschritt. Grundsätzlich können jedoch deutlich mehr Iterationsschritte notwendig sein. Die eingekreisten Nummern teilen erneut den Schritten des Stepping Algorithmus ihren Platz in der Abbildung zu. Nach Schritt 5 ist der erste Durchlauf des Stepping Algorithmus beendet. Auf Nummerierungen für den zweiten Durchlauf wird in Abbildung 5 verzichtet.

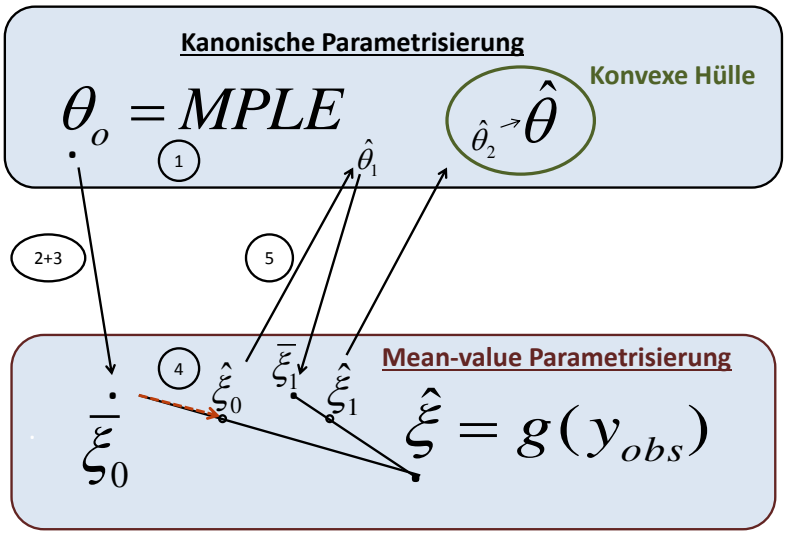

Abbildung 5: Beispielhafte Skizzierung der einzelnen Schritte des Stepping Algorithmus bis hin zur Konvergenz.

3.4.4 Stochastische Approximation (Robbins-Monro)

Eine weitere Möglichkeit, das Problem der unberechenbaren Normalisierungskonstante zu umgehen, ist die sogenannte stochastische Approximation aus Snijders (2002). Diese macht sich den Robbins-Monro Algorithmus (Robbins und Monro (1951)) zunutze, um Exponential Random Graph Modelle zu schätzen.

Ziel der Parameterschätzung ist es, die Verteilung der Statistiken der simulierten Netzwerke über den Statistiken des beobachteten Netzwerkes zu zentrieren. Als zentriert gilt die Verteilung der Statistiken der simulierten Netzwerke, wenn

$$E_\theta(g(Y)) - g(y_{obs}) = 0 \tag{10}$$

beziehungsweise $E_\theta(g(Y)) = g(y_{obs})$. Das Auflösen dieser Gleichung liefert denselben θ-Vektor wie die Maximum-Likelihood Schätzung, denn für die partielle Ableitung der Log-Likelihood Funktion aus Gleichung (3) nach θ erhält man:

$$\begin{aligned}
\frac{\partial}{\partial\theta}\ell(\theta) &= g(y_{obs}) - \frac{\partial}{\partial\theta} log(k(\theta)) \\
&= g(y_{obs}) - \frac{\partial}{\partial\theta} log(\sum_{y\in\mathcal{Y}} exp\{\theta^T \cdot g(y)\}) \\
&= g(y_{obs}) - \sum_{y\in\mathcal{Y}} g(y) \cdot P_\theta(y) \\
&= g(y_{obs}) - E_\theta(g(Y))
\end{aligned}$$

was wiederum Gleichung (10) entspricht. Die Tatsache, dass $\frac{\partial}{\partial\theta} log(k(\theta)) = E_\theta(g(Y))$, kann ausgenutzt werden, um durch Simulation von Zufallsgraphen aus der Zielverteilung die Score Funktion $\frac{\partial}{\partial\theta}\ell(\theta)$ zu approximieren.

Um dies zu ermöglichen, muss mit dem MCMC Algorithmus aus Abschnitt 3.2.3 für jeden iterativen Aktualisierungsschritt t ein einzelnes zufälliges Netzwerk $y^{(t)}$ aus der Zielverteilung $P_{\theta^{(t)}}(y)$ gezogen werden. Das simulierte Netzwerk $y^{(t)}$ basiert demnach auf der aktuellen Schätzung des Vektors mit Modellparametern $\theta^{(t)}$. Außerdem muss, wie bei dem MCMCMLE Verfahren, ein Startwert $\theta^{(0)}$ festgelegt werden. Hierfür eignet sich der Maximum Pseudo-Likelihood Schätzer aus Gleichung (6). Mit dem Ziel, den MLE $\hat{\theta}$ zu finden, für den Gleichung (10) erfüllt ist, erzeugt man nun eine Sequenz von Parametern $\theta^{(0)}$, $\theta^{(1)}$, ..., $\theta^{(N)}$ mit Hilfe eines Newton-Raphson ähnlichen

Verfahrens. Eine Aktualisierung der Sequenz von $\theta^{(t)}$ zu $\theta^{(t+1)}$ erfolgt durch

$$\theta^{(t+1)} = \theta^{(t)} - a_{(t)} D_0^{-1}(g(y^{(t)}) - g(y_{obs})) \quad . \tag{11}$$

Die Idee dabei ist, dass $\theta^{(t+1)}$ mit jedem Aktualisierungsschritt näher an den MLE gerückt wird. Die Skalierungsmatrix D_0 wird benutzt, um die Aktualisierungen der Elemente des Parametervektors zu skalieren und muss vor der Aktualisierung durch Gleichung (11) bestimmt werden. Dabei handelt es sich genauer gesagt um die Diagonalmatrix der durch simulierte Netzwerke geschätzten Kovarianzmatrix. Der Ausdruck a_n bezeichnet den sogenannten gain factor, welcher die Größe der Aktualisierungsschritte bei der Aktualisierung von $\theta^{(t)}$ kontrolliert (Lusher et al. (2013)). Der gain factor ist ein positiver Wert, welcher in jedem Iterationsschritt kleiner wird und letztendlich gegen 0 konvergiert (Snijders (2002)). Somit werden die Veränderungen innerhalb der Sequenz immer kleiner, da $g(y^{(t)})$ sich $g(y_{obs})$ annähert und $a_{(t)}$ gegen 0 konvergiert.

Wenn $\theta^{(t+1)}$ der wahre MLE ist, muss $E_{\theta^{(t+1)}}(g(Y)) = g(y_{obs})$ sein und somit für eine ausreichend große Stichprobe $\bar{g}_{\theta^{(t+1)}} - g(y_{obs}) \approx 0$ sein. Um $\bar{g}_{\theta^{(t+1)}}$ zu erhalten, müssen zufällige Netzwerke $Y_1, Y_2, ..., Y_m$ aus der Zielverteilung $P_{\theta^{(t+1)}}(y)$ durch den Markov-Chain-Monte-Carlo Algorithmus generiert werden. Anschließend kann der Stichprobenmittelwert $\bar{g}_{\theta^{(t+1)}} = \frac{1}{m}\sum_{i=1}^{m} g(Y_i)$ berechnet werden. Wenn $\bar{g}_{\theta^{(t+1)}} - g(y_{obs}) \approx 0$, kann man sagen, dass die erwarteten Werte den beobachteten entsprechen. In diesem Fall gilt der Schätzalgorithmus als konvergiert. Eine besonders ausführliche Beschreibung der stochastischen Approximation bietet Abschnitt 12.4.2 in Lusher et al. (2013).

3.5 Interpretation der Parameter

Nachdem die Modellparameter θ geschätzt wurden, soll nun kurz auf ihre Interpretation eingegangen werden. Dabei ist es sinnvoll, an die bedingte Wahrscheinlichkeit, dass zwei Knoten eine gemeinsame Kante besitzen, gegeben das restliche Netzwerk, aus Gleichung (4) zu denken, wonach

$$\text{logit}[P_\theta(Y_{ij} = 1 | Y_{-ij} = y_{-ij})] = \theta^T \cdot [g(y_{ij}^+) - g(y_{ij}^-)] \quad .$$

Demnach gilt

$$P_\theta(Y_{ij} = 1 | Y_{-ij} = y_{-ij}) = \text{logit}^{-1}(\theta^T \cdot [g(y_{ij}^+) - g(y_{ij}^-)]) \quad ,$$

mit

$$\text{logit}^{-1}(x) = \frac{exp(x)}{1 + exp(x)} \quad .$$

Da $\delta_g(y)_{ij} = g(y_{ij}^+) - g(y_{ij}^-)$ für die Change Statistik aus Gleichung (4) steht, erhält man für die Umrechnung in die entsprechende Wahrscheinlichkeit folgende Gleichung

$$P_\theta(Y_{ij} = 1 | Y_{-ij} = y_{-ij}) = \frac{exp(\theta^T \cdot \delta_g(y)_{ij})}{1 + exp(\theta^T \cdot \delta_g(y)_{ij})} \quad . \tag{12}$$

Angenommen man hat z.B. ein ERGM dessen einzige Netzwerk Statistik die Anzahl der Kanten ist und die Schätzung für den dazugehörigen Parameterwert ergab $\hat{\theta}_{Kanten} = -1.6$. Aus Abschnitt 3.2.2 ist bekannt, dass die Change-Statistik $\delta_g(y)_{ij}$ für die Anzahl an Kanten immer den Wert 1 annimmt. Somit lässt sich aus $\hat{\theta}_{Kanten}$ die Wahrscheinlichkeit für die Entstehung einer Kante zwischen den Knoten i und j aus Gleichung (12) berechnen:

$$P_\theta(Y_{ij} = 1 | Y_{-ij} = y_{-ij}) = \frac{exp(-1.6 \cdot 1)}{1 + exp(-1.6 \cdot 1)} = 0.168 \quad .$$

Darüber hinaus ist es möglich, die Odds ($\text{Odds(Ergeignis)} = \frac{P(\text{Ereignis})}{P(\text{Gegenereignis})}$) für die Entstehung einer Kante zwischen den Knoten i und j gegeben das restliche Netzwerk zu interpretieren. Diese lassen sich mit Hilfe der Ergebnisse aus Gleichung (12) berechnen:

$$\begin{aligned}
\frac{P_\theta(Y_{ij} = 1 | Y_{-ij} = y_{-ij})}{P_\theta(Y_{ij} = 0 | Y_{-ij} = y_{-ij})} &= \frac{P_\theta(Y_{ij} = 1 | Y_{-ij} = y_{-ij})}{1 - P_\theta(Y_{ij} = 1 | Y_{-ij} = y_{-ij})} \\
&= \frac{\frac{exp(\theta^T \cdot \delta_g(y)_{ij})}{1+exp(\theta^T \cdot \delta_g(y)_{ij})}}{1 - \frac{exp(\theta^T \cdot \delta_g(y)_{ij})}{1+exp(\theta^T \cdot \delta_g(y)_{ij})}} \\
&= \frac{\frac{exp(\theta^T \cdot \delta_g(y)_{ij})}{1+exp(\theta^T \cdot \delta_g(y)_{ij})}}{\frac{1+exp(\theta^T \cdot \delta_g(y)_{ij}) - exp(\theta^T \cdot \delta_g(y)_{ij})}{1+exp(\theta^T \cdot \delta_g(y)_{ij})}} \\
&= exp(\theta^T \cdot \delta_g(y)_{ij}) \quad .
\end{aligned}$$

Angenommen das Modell enthält u Netzwerk Statistiken und damit auch u dazugehörige Modellparameter θ. Da in diesem Fall $\theta^T \cdot \delta_g(y)_{ij} = \sum_{k=1}^{u} \theta_k \cdot \delta_{kg}(y)_{ij}$, erhält man:

$$\begin{aligned}
\frac{P_\theta(Y_{ij} = 1 | Y_{-ij} = y_{-ij})}{P_\theta(Y_{ij} = 0 | Y_{-ij} = y_{-ij})} &= exp(\sum_{k=1}^{u} \theta_k \cdot \delta_{kg}(y)_{ij}) = \prod_{l=1}^{u} exp(\theta_k \cdot \delta_{kg}(y)_{ij}) = \\
&= exp(\theta_1 \cdot \delta_{1g}(y)_{ij}) \cdot ... \cdot exp(\theta_u \cdot \delta_{ug}(y)_{ij}) \ .
\end{aligned}$$

Somit ist es möglich, die Modellparameter auf folgende Art und Weise zu interpretieren: erhöht sich die k-te Change-Statistik $\delta_{kg}(y)_{ij}$ um r, bei Festhalten aller anderen Change-Statistiken und Parameter, erhöhen beziehungsweise verringern sich die Odds für die Entstehung der Kante zwischen den Knoten i und j gegeben das restliche Netzwerk um den Faktor $exp(\theta_k \cdot r)$. Somit lässt sich, für ein positives r, zusammenfassend sagen:

- Falls $\theta_k > 0 \Rightarrow$ Erhöhung der Odds
- Falls $\theta_k = 0 \Rightarrow$ Kein Einfluss auf Odds
- Falls $\theta_k < 0 \Rightarrow$ Verringerung der Odds .

Angenommen man möchte ein ERGM interpretieren, dessen Netzwerk Statistiken die Anzahl der Kanten und die Anzahl der triangles sind. Der Schätzung nach sei $\hat{\theta}_{Kanten} = -1.6$ und $\hat{\theta}_{triangles} = 0.15$. Wenn das Hinzufügen einer Kante zwischen den Knoten i und j eine Erhöhung der triangles im Netzwerk um 1 bewirkt, so sind ihre Odds:

$$exp(-1.6 \cdot 1) \cdot exp(0.15 \cdot 1) = 0.235$$

Würde das Hinzufügen der Kante zwischen den Knoten i und j keine Erhöhung der Anzahl der triangles im Netzwerk bewirken, so wären ihre Odds mit

$$exp(-1.6 \cdot 1) \cdot exp(0.15 \cdot 0) = 0.202$$

geringer. Erhöht sich die Change-Statistik für die triangles um 1, bei Festhalten aller anderen Change-Statistiken, erhöhen sich demnach die Odds für die Entstehung der Kante zwischen den Knoten i und j um den Faktor $exp(\theta_{triangles}) = exp(0.15) = 0.162$.

Auf Netzwerkebene könnte man interpretieren, dass das Netzwerk, welches durch das Hinzufügen der Kante zwischen den Knoten i und j entsteht (y_{ij}^{+}), plausibler ist als das Netzwerk, welches entsteht, wenn man die Kante zwischen den Knoten i und j weglässt (y_{ij}^{-}).

3.6 Überblick möglicher Netzwerk Statistiken

Selbstverständlich gibt es keine pauschale Antwort auf die Frage, welche Statistiken aufgenommen werden müssen, um ein gutes Modell zu erhalten. Dies hängt unter anderem stark von der Art und der Struktur des zu schätzenden Netzwerkes ab. Dieser Abschnitt soll einen kurzen Überblick grundlegender Statistiken für Exponential Random Graph Modelle und deren Implementierung in der Softwareumgebung R (R Development Core Team (2008)) bieten. Die aufgeführten Statistiken sind alle im *ergm* Paket (Handcock et al. (2017)) enthalten. Eine vollständige Aufzählung der im *ergm* Paket enthaltenen Statistiken bietet Morris et al. (2008).

Eine wichtige Statistik im Exponential Random Graph Modell ist die Anzahl der Kanten in einem Netzwerk, welche bereits in Abschnitt 3.2.2 vorgestellt wurde. Diese Statistik kann in der Softwareumgebung R durch die im *ergm* Paket enthaltene Statistik *edges* implementiert werden, welche die Anzahl an Kanten in einem Netzwerk zählt. Sie spielt die Rolle des Intercepts im ERGM und gibt die Grundwahrscheinlichkeit für die Bildung einer Kante im Netzwerk wieder. Da die Change Statistik für die Anzahl der Kanten in Gleichung 4 immer gleich 1 ist, beeinflusst der dazugehörige Parameter θ_{edge} jedes Netzwerk auf die selbe Art und Weise. Für gerichtete Netzwerke können die Statistiken *mutual* und *asymmetric* benutzt werden. Die Statistik mutual zählt die Anzahl an Knotenpaaren i, j in einem gerichteten Netzwerk, bei denen sowohl die Kante $\{i, j\}$ als auch die Kante $\{j, i\}$ existiert. Die Statistik asymmetric zählt dagegen nur diejenigen Kanten $\{i, j\}$, bei denen die Kante $\{j, i\}$ nicht existiert.

Die Anzahl an k-stars in einem Netzwerk lässt sich durch den Term *k-star(k)* im Modell berücksichtigen. Ein k-star ist ein Knoten, der zu k anderen Knoten eine Kante besitzt. Es sei erwähnt, dass k-star(1) in einem ungerichteten Netzwerk der Statistik edges entspricht. Bei gerichteten Netzwerken verwendet man die Statistiken *ostar(k)* und *istar(k)*. Die *ostar(k)* Statistik zählt die Anzahl an k-outstars in einem gerichteten Netzwerk. Ein k-outstar ist definiert als ein Knoten i, von dem aus es Kanten zu k anderen Knoten gibt. Die Statistik *istar(k)* zählt die Anzahl an k-instars. Ein k-instar ist ein Knoten i, in dem k Kanten in einem gerichteten Netzwerk enden. Zur Veranschaulichung ist in Abbildung 6 links ein 3-star, mittig ein 3-outstar und rechts ein 3-instar zu sehen. Darüber hinaus gilt es zu beachten, dass bei diesen Statistiken ein Knoten mehr als einmal gezählt werden kann. So enthält ein 3-star wie in Abbildung 6 links drei 2-stars.

Die Statistik *degree(d)* zählt die Knoten in einem ungerichteten Netzwerk, die einen Knotengrad von d aufweisen. Somit kann durch diese Statistik die in Abschnitt 2 beschriebene Degree-Verteilung im Modell berücksichtigt werden. Liegt ein gerich-

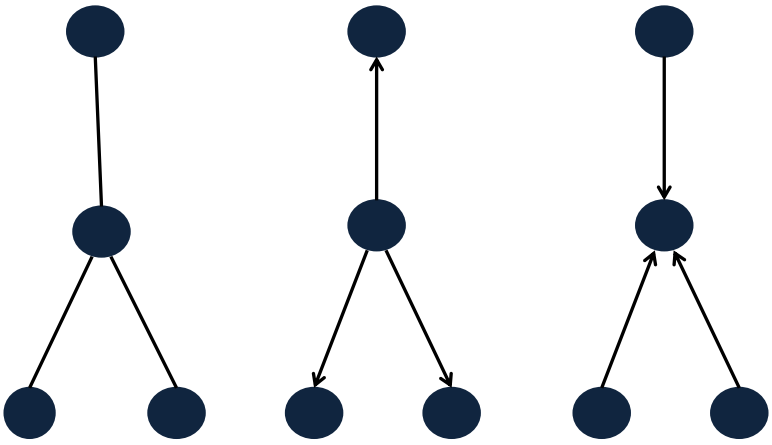

Abbildung 6: Abbildung eines 3-stars (links), eines 3-outstars (mittig) und eines 3-instars (rechts).

tetes Netzwerk vor, verwendet man für den Außengrad eines Knotens die Statistik *odegree(d)* und für den Innengrad eines Knotens die Statistik *idegree(d)*.

Die Anzahl an triangles, welche bereits in Abschnitt 3.2.2 vorgestellt wurden, lässt sich durch die Statistik *triangle* im Modell berücksichtigen. Für gerichtete Netzwerke ist ein triangle definiert als drei Knoten i, j und k, bei denen die beiden Kanten $(i \rightarrow j)$ und $(j \rightarrow k)$ sowie entweder $(k \rightarrow i)$ oder $(k \leftarrow i)$ existieren. Für den Fall, dass $(k \rightarrow i)$ das Dreieck schließt, spricht man von einem sogenannten transitive triple, welches in R durch die Statistik *ttriple* implementiert werden kann. Falls allerdings die Kante $(k \leftarrow i)$ das Dreieck schließt, spricht man von einem sogenannten cyclic triple. Dieses kann durch den Term *ctriple* in das Modell mit aufgenommen werden. Bei einem gerichteten Netzwerk können allerdings sinnvollerweise maximal zwei der drei Statistiken *triangle*, *ttriple* und *ctriple* in ein Modell mit aufgenommen werden, da *triangle* = *ttriple* + *ctriple*.

Wie bereits erwähnt, bietet das Exponential Random Graph Modell die Möglichkeit, erklärende Zusatzinformationen als externe Variablen in das Modell mit aufzunehmen. Dabei gibt es mehrere Möglichkeiten diese externen Variable je nach Fragestellung und Skalenniveau im Modell zu berücksichtigen. Mit der Statistik *nodecov* lässt sich der Haupteffekt einer externen Variable in das Modell mit aufnehmen. Dabei werden durch die Verwendung der Statistik nodecov alle Werte der externen Variablen von miteinander verbundenen Akteuren eines Netzwerkes addiert. Mit *nodematch* lassen sich die Anzahl an Kanten zählen, deren dazugehörige Knoten dieselbe Ausprägung der externen Variable aufweisen. So kann mit nodematch zum Beispiel

soziale Homophilie untersucht werden, also die Tendenz, eher mit den Individuen eine Freundschaft einzugehen, die einem ähnlich sind. Bei einer solchen Untersuchung könnte die Ähnlichkeit der Akteure bezüglich Kriterien wie zum Beispiel Geschlecht, ethnische Herkunft oder Bildungsgrad von Interesse sein.

3.7 Gewichtete Statistiken

Die im vorherigen Abschnitt 3.6 aufgeführten traditionellen Zählstatistiken, wie z.B. k-stars, triangles oder ctriples, können, wie in Abschnitt 3.3 erläutert, zur Degenerierung des modellierten Netzwerkes führen. Einen Ansatz, um dieses Problem in den Griff zu bekommen, bietet die Verwendung von gewichteten Statistiken. Diese stabilisieren die Modelle, doch ihre Interpretation ist besonders schwer. Snijders et al. (2006) schlagen drei solche Statistiken für ungerichtete Netzwerke vor: alternating k-stars, alternating k-triangles und alternating two paths. Diese und auch die später von Hunter (2007) vorgeschlagene alternative Formulierung für alternating k-triangles (GWESP) sollen nun kurz vorgestellt werden. Für eine ausführliche Erläuterung dieser gewichteten Statistiken wird auf Snijders et al. (2006) und Hunter (2007) verwiesen.

3.7.1 Alternating k-star

Wie bereits in Abschnitt 3.6 erwähnt, ist ein k-star definiert als ein Knoten, der zu k anderen Knoten eine Kante besitzt. Die Anzahl an k-stars in einem Netzwerk y kann für $k \leqslant 2$ mit folgender Formel berechnet werden:

$$S_k(y) = \sum_{1 \leqslant i \leqslant n} \binom{y_{i+}}{k}$$

Dabei steht y_{i+} für den Knotengrad des Knotens i. Die k-star Statistik ist allerdings besonders anfällig für das Problem der Degeneration.

Snijders et al. (2006) beschreiben die Möglichkeit, den Lawineneffekt durch die Aufnahme mehrerer k-star Statistiken S_k zu verringern. Die Idee hinter der alternating k-star Statistik AKS ist, dass positive Gewichtungen einiger k-star Statistiken durch negative Gewichtungen anderer k-star Statistiken ausgeglichen werden (Snijders et al. (2006)). Dies wird durch alternierende Vorzeichen der Gewichtungen der

k-star Statistiken erreicht:

$$\begin{aligned} AKS &= S_2 - \frac{S_3}{\lambda} + \frac{S_4}{\lambda^2} - ... + (-1)^{n-2}\frac{S_{n-1}}{\lambda^{n-3}} \\ &= \sum_{k=2}^{n-1}(-1)^k \frac{S_k}{\lambda^{k-2}} \end{aligned}$$

Der Ausdruck AKS heißt alternating k-star mit Paramater λ, wobei üblicherweise $\lambda = 2$ gesetzt wird (Robins et al. (2009)). Der Parameter λ sorgt für eine abnehmende Gewichtung der k-star Statistiken. Dadurch wird verhindert, dass besonders dichte Netzwerke eine hohe Wahrscheinlichkeit zugeordnet bekommen.

Die alternating k-star Statistik ist für eine Modellierung der Degree-Verteilung vorgesehen. Ein positiver Parameterwert für diese Statistik spricht für eine schiefe Degree-Verteilung, da das Netzwerk in einem solchen Fall auch Knoten mit einem sehr hohen Knotengrad enthält. Ein negativer Parameterwert suggeriert hingegen, dass Knoten mit einem sehr hohen Knotengrad unwahrscheinlich sind und sich demnach die Knotengrade der einzelnen Knoten nicht allzu sehr unterscheiden (Robins et al. (2009)).

3.7.2 Alternating k-triangle

Nach Snijders et al. (2006) ist es mit den traditionellen Zählstatistiken wie der triangle Statistik nahezu unmöglich, Netzwerke zu modellieren, die ein hohes Maß an Transitivität aufweisen, aber trotzdem nur eine eher niedrige Dichte haben. Die *Transitivität* eines Netzwerkes entspricht der Anzahl an enthaltenen triangles dividiert durch die Anzahl an 2-stars im Netzwerk (Csardi und Kolaczyk (2014)). Ein hohes Maß an Transitivität würde in einem Netzwerk aus Freundschaften bedeuten, dass das Eingehen einer Freundschaft zwischen zwei Personen, die mehrere gemeinsame Freunde haben, grundsätzlich einiges wahrscheinlicher ist als das Schließen einer Freundschaft zwischen zwei Personen ohne gemeinsame Freunde. Netzwerke, die trotz eines hohen Maßes an Transitivität eine eher geringe Dichte aufweisen, lassen sich jedoch häufig beobachten.

Deshalb schlagen Snijders et al. (2006) eine andere Möglichkeit vor, mit der sich die Transitivität modellieren lässt. Das Prinzip ist ähnlich dem der alternating k-stars, jedoch unter Verwendung der k-triangle Statistik. Ein k-triangle mit Basis (i, j) ist definiert als eine existierende Kante $\{i, j\}$, in Verbindung mit mindestens k weiteren Knoten, die sowohl mit i als auch mit j verbunden sind. Einfacher gesagt handelt es sich bei einem k-triangle, wie er in Abbildung 7 zu sehen ist, um k verschiedene

Dreiecke, die alle dieselbe Kante $\{i, j\}$ enthalten (Csardi und Kolaczyk (2014)).

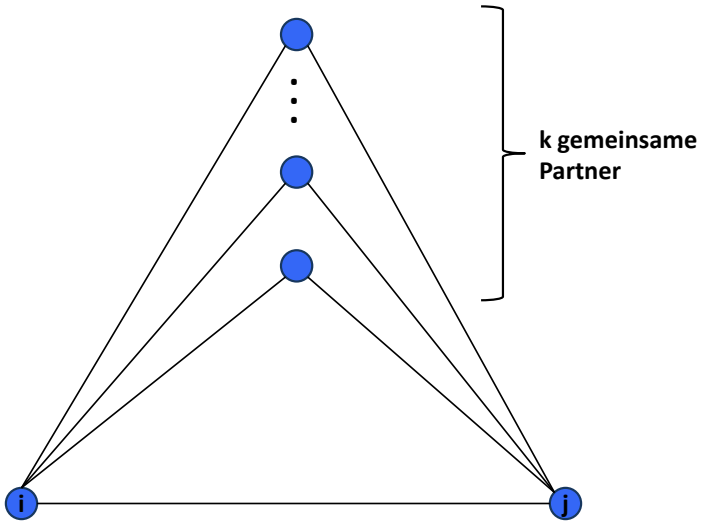

Abbildung 7: k-triangle für ein ungerichtetes Netzwerk.

Die Anzahl an k-triangles kann für $k \geqslant 2$ berechnet werden durch

$$T_k = \sum_{i<j} y_{ij} \binom{L_{2ij}}{k} \quad ,$$

wobei

$$L_{2ij} = \sum_{h \neq i,j} y_{ih} y_{hj}$$

der Anzahl an two-paths zwischen i und j entspricht. Aus dieser Formel wird darüber hinaus ersichtlich, dass ein 3-triangle drei 2-triangles enthält. Ein 1-triangle entspricht einem triangle. Die Anzahl an 1-triangles in einem Netzwerk ist gegeben durch:

$$T_1 = \frac{1}{3} \sum_{i<j} y_{ij} L_{2ij} \quad .$$

Für die Berechnung der alternating k-triangle Statistik werden, wie schon bei der alternating k-star Statistik, abnehmende Gewichte mit alternierenden Vorzeichen

verwendet:

$$
\begin{aligned}
AKT &= 3T_1 - \frac{T_2}{\lambda} + \frac{T_3}{\lambda^2} - ... + (-1)^{n-3}\frac{T_{n-2}}{\lambda^{n-3}} \\
&= \sum_{i<j} y_{ij} \sum_{k=1}^{n-2} \left(\frac{-1}{\lambda}\right)^{k-1} \binom{L_{2ij}}{k} \\
&= \lambda \sum_{i<j} y_{ij} \left\{ 1 - \left(1 - \frac{1}{\lambda}\right)^{L_{2ij}} \right\}
\end{aligned}
$$

3.7.3 Alternating independant two-path

Das Prinzip der alternating independant two-path Statistik entspricht erneut dem der bereits vorgestellten gewichteten Statistiken, jedoch unter Verwendung der k-independant two-path Statistik. Ein k-independant two-path, welcher in Abbildung 8 illustriert ist, entspricht einem k-triangle, bei dem die Basis (i, j) fehlen darf. Demnach handelt es sich bei einem k-independant two path um ein Knotenpaar (i, j), das genau k gemeinsame Nachbarn hat, wobei diese Statistik davon unabhängig ist, ob eine Kante zwischen i und j existiert (Snijders et al. (2006)).

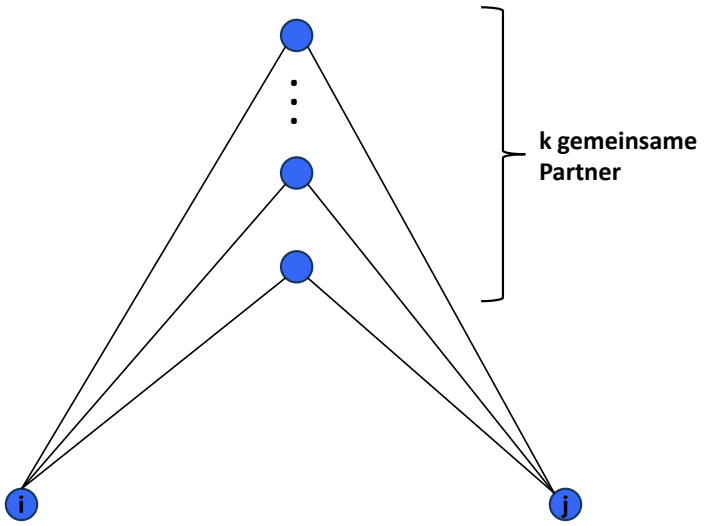

Abbildung 8: k-independant two-paths.

Die Anzahl an k-independant two-paths lässt sich für $k \neq 2$ berechnen durch

$$
U_k = \sum_{i<j} \binom{L_{2ij}}{k} \quad .
$$

Für $k = 2$ lässt sich die Anzahl an k-independant two-paths ermitteln durch

$$U_2 = \frac{1}{2} \sum_{i<j} \binom{L_{2ij}}{2} \quad .$$

Die sich daraus ergebende alternating independant two-path Statistik lässt sich nach Snijders et al. (2006) bestimmen durch

$$\begin{aligned} AITP &= U_1 + \frac{2}{\lambda} U_2 + \sum_{k=3}^{n-2} \left(\frac{-1}{\lambda} \right)^{k-1} U_k \\ &= \lambda \sum_{i<j} \left\{ 1 - \left(1 - \frac{1}{\lambda} \right)^{L_{2ij}} \right\} \quad . \end{aligned}$$

3.7.4 Geometrically weighted edgewise shared partner

Hunter (2007) präsentiert eine alternative Formulierung der alternating k-triangle Statistik. Hierzu werden die edgewise shared partner Statistiken $EP_0(y)$ ... $EP_{n-2}(y)$ eingeführt. $EP_k(y)$ entspricht der Anzahl an Kanten $\{i, j\}$, bei denen i und j genau k gemeinsame Nachbarn teilen. Die k-triangle Statistik lässt sich für $2 \leqslant k \leqslant n-2$ mit folgender Gleichung durch die edgewise shared Partner Statistik ausdrücken:

$$T_k(y) = \sum_{i=k}^{n-2} \binom{i}{k} EP_i(y) \quad . \tag{13}$$

Für $T_1(y)$, also der Anzahl an triangles gilt diese Formel jedoch nicht. Hier muss der Faktor $\frac{1}{3}$ hinzugefügt werden:

$$T_1(y) = \frac{1}{3} \sum_{i=k}^{n-2} i EP_i(y) \quad . \tag{14}$$

Der Unterschied zwischen der edgewise shared partner Statistik $EP_k(y)$ und der k-triangle Statistik T_k lässt sich am besten durch das einfache Netzwerk aus Abbildung 9, welches dem Beispiel aus Hunter (2007) ähnelt, veranschaulichen. Für dieses kleine Netzwerk mit fünf Knoten, ist die edgewise shared partner Verteilung (EP_0, EP_1, EP_2, EP_3) = (1, 4, 1, 0). Für die k-triangle Verteilung (T_1, T_2, T_3) ergibt sich hingegen (2, 1, 0). An diesem Beispiel lassen sich die Formeln (13) und (14) auch gut nachvollziehen. Nachdem der positive Parameter $\mu_t = \log \lambda$ eingeführt wurde, kann die k-triangle Statistik nun mit Hilfe der Gleichungen (13) und (14) durch EP_k

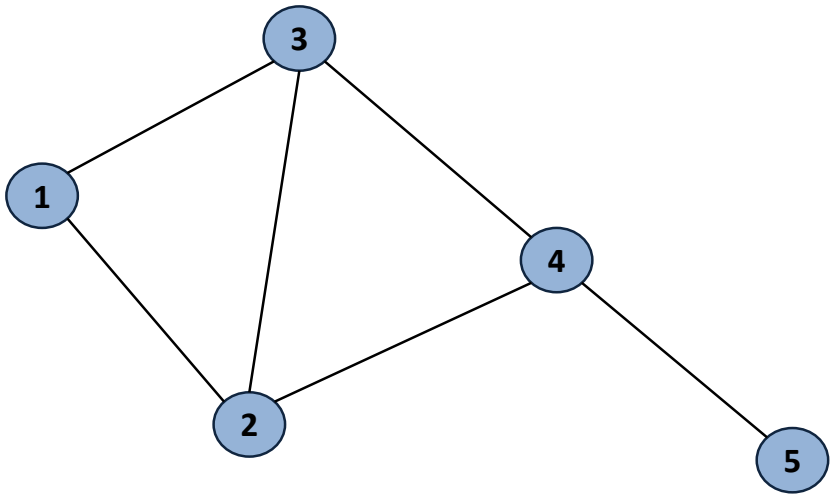

Abbildung 9: Veranschaulichung des Unterschieds zwischen der edgewise shared partner Statistik und der k-triangle Statistik anhand eines einfachen Netzwerkes.

ausgedrückt werden:

$$GWESP = e^{\mu_t} \sum_{i=1}^{n-2} \left\{ 1 - (1 - e^{\mu_t})^i \right\} EP_i(y) \quad .$$

GWESP heißt geometrically weighted edgewise shared partner. Die Interpretation dieser Statistik erweist sich jedoch als schwierig. Grundsätzlich lässt sich sagen, dass es bei einem positiven Parameterwert θ_{GWESP} eher eine Tendenz zum Hinzufügen von zusätzlichen edgewise shared Partnern gibt. Bei einem negativen Parameterwert θ_{GWESP} werden dagegen eher keine zusätzlichen edgewise shared Partner hinzugefügt.

Analog dazu schlägt Hunter (2007) auch eine alternative Formulierung der alternating independant two path Statistik (geometrically weighted dyadic shared partner) und der alternating k-star Statistik (geometrically weighted degree Statistik) vor. Da diese in dieser Arbeit jedoch nicht zum Einsatz kommen, wird für eine ausführliche Erklärung auf Hunter (2007) verwiesen.

4 Simulation

Um die Performance der unterschiedlichen Schätzmethoden bei der Parameterschätzung vergleichen zu können, werden im Folgenden mehrere Simulationen durchgeführt. Zu diesem Zweck werden zufällige Netzwerke aus ERGMs gezogen, wobei die eingesetzten Netzwerk Statistiken und ihre dazugehörigen Parameterwerte in den verschiedenen Simulationen variiert werden. Die Ziehung eines zufälligen Netzwerkes aus einer gegebenen Wahrscheinlichkeitsverteilung wird mit Hilfe des in Abschnitt 3.2.3 vorgestellten MCMC Algorithmus durchgeführt. Anschließend werden auf den gezogenen Netzwerken ERGMs mit denselben Netzwerk Statistiken, die bei der Ziehung des zu schätzenden Netzwerkes verwendet wurden, geschätzt. Die Schätzung eines gezogenen Netzwerkes wird dabei stets mit allen vier in Abschnitt 3.4 vorgestellten Schätzmethoden durchgeführt. Abschließend können nun die Schätzungen der verschiedenen Schätzalgorithmen miteinander verglichen werden. Bei diesem Vergleich wird besonders auf die für eine Schätzung durchschnittlich benötigte Zeit, die Quote, mit der die Schätzalgorithmen konvergieren, sowieso die Qualität der Schätzungen geachtet. Dabei ist unter anderem von Interesse, ob das vom ergm Paket standardmäßig verwendete MCMCMLE Verfahren besser abschneidet, als die anderen drei zur Verfügung stehenden Schätzmethoden. Im Folgenden wird nun zunächst der Aufbau der Simulation und dessen Implementierung in der Softwareumgebung R erläutert, woraufhin im Anschluss die Ergebnisse der durchgeführten Simulationen präsentiert werden.

4.1 Aufbau und Implementierung in R

Der Simulationsaufbau kann in 2 Schritte aufgeteilt werden, welche in Abbildung 10 skizziert sind. Zu Beginn müssen in einem ersten Schritt die interessierenden Netzwerk Statistiken $g(y)$ und die dazugehörigen gewünschten Parameterwerte θ festgelegt werden. Anschließend wird in einem zweiten Schritt ein zufälliges Netzwerk aus dem in Schritt 1 bestimmten ERGM gezogen. Daraufhin wird ein ERGM mit denselben Netzwerk Statistiken, die bei der Ziehung des zufälligen Netzwerkes eingesetzt wurden, auf dem zuvor gezogenen Netzwerk geschätzt. Die Schätzung wird dabei stets mit allen vier Schätzmethoden aus Abschnitt 3.4 durchgeführt. Schritt 2 wird 50 mal wiederholt, so dass am Ende der Simulation für jede Schätzmethode die 50 Schätzungen der Parameterwerte $\hat{\theta}$ aus Schritt 2 (in Abbildung 10 rot umrandet) mit den eingesetzten Parameterwerten θ aus Schritt 1 (in Abbildung 10 schwarz umrandet) verglichen werden können. Eine qualitativ hochwertige Schätzung zeichnet sich vor allem dadurch aus, dass die geschätzten Parameterwerte in den 50 Schätzungen

nah an den bei der Ziehung der zufälligen Netzwerke eingesetzten Parameterwerten sind. Dies wird überprüft, indem mehrere Statistiken der 50 Schätzungen betrachtet werden. Dazu gehören für jeden Modellparameter Mittelwert, Median und Range der 50 pro Schätzmethode durchgeführten Schätzungen sowie die durchschnittliche vom Modell geschätzte Standardabweichung, die beobachtete Standardabweichung und die mittlere quadratische Abweichung (MSE). Darüber hinaus werden auch die von den Schätzalgorithmen für eine Schätzung benötigte Zeit sowie die Quote, mit der die Schätzalgorithmen konvergieren, erfasst und analysiert.

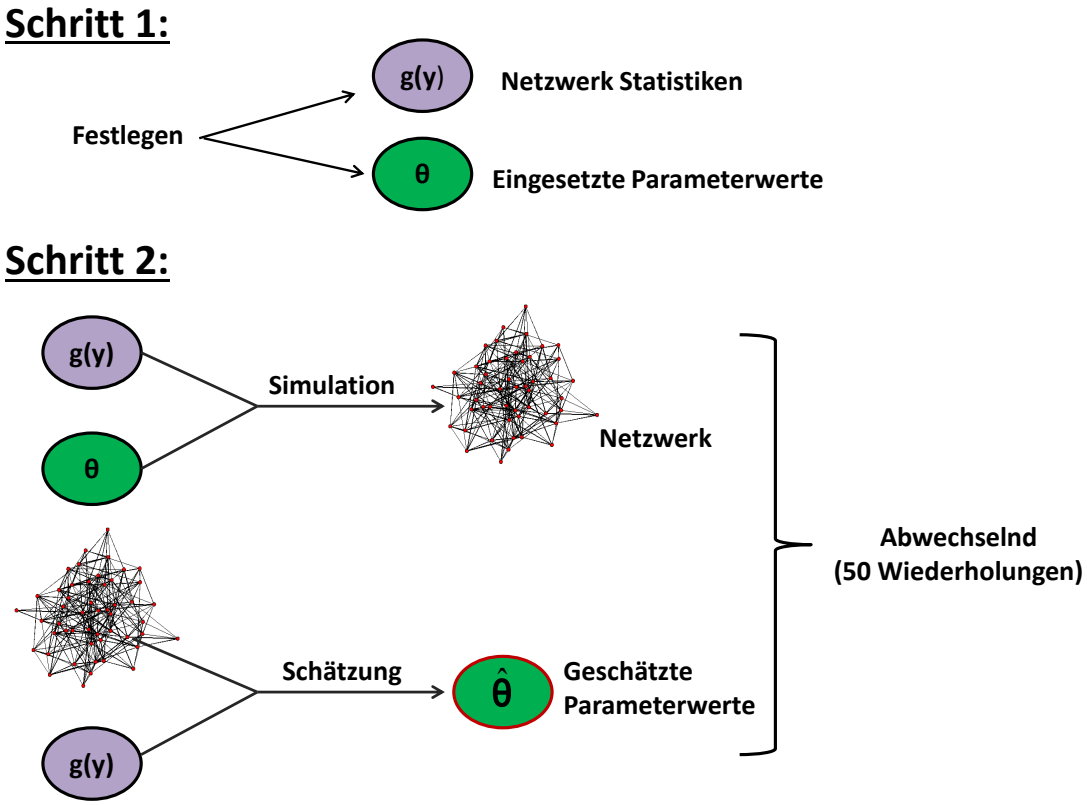

Abbildung 10: Skizzierung des Aufbaus der Simulation.

Durchgeführt wird die Simulation mit Hilfe der Softwareumgebung R (R Development Core Team (2008)). Hierfür ist es notwendig einige Pakete zu implementieren. So liefert das Paket *ergm* (Handcock et al. (2017)) Funktionen, die eine Simulation von Netzwerken aus einem ERGM ermöglichen, wie es in Abschnitt 3.2.3 beschrieben ist. Darüber hinaus enthält dieses Paket auch Funktionen, die es ermöglichen ERGMs zu schätzen. Dabei können alle vier der in Abschnitt 3.4 vorgestellten Schätzmethoden angewandt werden. Zufällige Netzwerke können mit Hilfe des Paketes *sna* (Butts (2014)) gezogen werden. In diesem Teil der Arbeit werden einige zentrale Funktionen und der dazugehörige R-Code erklärt. Der gesamte lauffähige R-Code kann dem elektronischen Anhang entnommen werden.

Zunächst ist es notwendig, die Größe beziehungsweise die Anzahl der Knoten des zu ziehenden Netzwerkes festzulegen. Hierfür wird die Funktion `rgnm()` aus dem sna Paket (Butts (2014)) verwendet, mit deren Hilfe ein zufälliges Startnetzwerk gezogen werden kann, wobei die Anzahl an Knoten des Startnetzwerkes durch das Argument `nv` übergeben wird. Die Anzahl an Kanten wird mit dem Argument `m` bestimmt und ist in diesem Fall nicht von großer Bedeutung, da die Markov-Kette bei ausreichend großem Burn-In unabhängig von ihrem Startpunkt ist. Es muss lediglich darauf geachtet werden, dass die angegebene Anzahl an Kanten `m` nicht die Anzahl möglicher Kanten in einem Netzwerk mit `nv` Knoten überschreitet. Durch die Option `diag=FALSE` werden Schleifen im Netzwerk verhindert. Mit der Option `mode` kann zwischen einem gerichteten (`mode="digraph"`) oder einem ungerichteten Netzwerk (`mode="graph"`) gewählt werden.

```
> Startnetzwerk <- rgnm(n = 1, nv = 300, m = 450, diag=FALSE,
>                                                 mode="graph")
```

Anschließend müssen die gewünschten Netzwerk Statistiken gewählt werden, welche zum Beispiel die Anzahl der Kanten, 2-stars und triangles sein könnten. Dafür wird ein ERGM mit den gewünschten Netzwerk Statistiken definiert.

```
> ERGM <- Startnetzwerk ~ edges + kstar(2) + triangle()
```

Aus dem so erhaltenen ERGM wird daraufhin in Schritt 2 mit der ebenfalls im ergm Paket enthaltenen Funktion `simulate` ein Netzwerk gezogen, wobei die dazugehörigen Parameterwerte durch das Argument `coef` übergeben werden. Dabei gilt es zu beachten, dass die Reihenfolge der dazugehörigen übergebenen Parameterwerte mit der Reihenfolge der festgelegten Netzwerk Statistiken übereinstimmt. Am Anfang dieser Funktion muss das in Schritt 1 gewählte ERGM übergeben werden. Mit dem Argument `nsim` kann die Anzahl der zu ziehenden Netzwerke bestimmt werden, welche in diesem Fall gleich eins gesetzt wird. Als Startnetzwerk soll das zuvor mit der Funktion `rgnm()` gezogene Netzwerk gewählt werden. Um dies zu erreichen muss `basis=Startnetzwerk` gesetzt werden. Innerhalb der Option `control`, kann mit der Funktion `control.simulate` der Burn In und das Thinning des verwendeten MCMC Algorithmus angepasst werden. Da in diesem Fall nur ein Netzwerk gezogen wird und nicht mehrere Netzwerke aus derselben Kette gezogen werden sollen, ist die Angabe eines Thinning Wertes hier nicht notwendig.

```
> simul_netzw <- simulate(ERGM, nsim=1,
>           coef=c(-3,-0.1,0.5), basis=Startnetzwerk,
```

```
>          control=control.simulate(MCMC.burnin=200000))
```

Anschließend wird auf dem gezogenen Netzwerk (`simul_netzw`) ein ERGM, mit denselben Netzwerk Statistiken, die für die Ziehung des Netzwerkes verwendet wurden, geschätzt. Dabei wird für jede der vier möglichen Schätzmethoden jeweils eine Schätzung durchgeführt. Die Funktion `ergm()` führt die Schätzung durch, wobei mit Hilfe der Option `main.method` zwischen den Schätzmethoden Markov Chain Monte Carlo Maximum Likelihood Estimation Verfahren (`main.method = "MCMLE"`), Stepping Algorithmus (`main.method = "Stepping"`) und Robbins Monro (`main.method = "Robbins-Monro"`) aus Abschnitt 3.4 gewählt werden kann. Eine Maximum Pseudo Likelihood Schätzung kann mit `estimate = "MPLE"` durchgeführt werden. Wird keine Schätzmethode ausgewählt, so wird automatisch das Markov Chain Monte Carlo Maximum Likelihood Estimation Verfahren für die Schätzung verwendet.

```
> ergm(simul_netzw ~ edges + kstar(2)) + triangle(),
>        control=control.ergm(main.method = "Stepping")
```

Bei den zufällig gezogenen Netzwerken (`simul_netzw`) handelt es sich um Ausprägungen der Zufallsvariable Y. Folglich muss die damit verbundene Unsicherheit in Form von zufälligen Fehlern minimiert werden. Deshalb wird die Ziehung eines Netzwerkes aus einem ERGM und die anschließende Schätzung eines ERGMs mit denselben Netzwerk Statistiken mit Hilfe einer for-Schleife 50 mal wiederholt. Nach dem Gesetz der großen Zahlen, konvergiert das arithmetische Mittel der Statistiken der gezogenen Netzwerke gegen das Erwartete. Somit sollte bei einem unverzerrten Schätzer das arithmetische Mittel der 50 Schätzungen der Parameterwerte bei jedem Modellparameter den bei der Ziehung der zufälligen Netzwerke eingesetzten Parameterwerten nahezu entsprechen. Nachdem die 50 Wiederholungen durchgeführt wurden, werden alle relevanten Informationen in einer Tabelle zusammengetragen, so dass die Performance der Schätzalgorithmen analysiert und miteinander verglichen werden kann. Wie bereits erwähnt gehören zu den relevanten Informationen Mittelwert, Median und Range der 50 pro Schätzmethode durchgeführten Schätzungen sowie die durchschnittliche vom Modell geschätzte Standardabweichung, die beobachtete Standardabweichung und der MSE. Zusätzlich wird auch die durchschnittlich für eine Schätzung benötigte Zeit und die Konvergenzquote erfasst. Da es bei der Schätzung zu Konvergenzproblemen kommen kann, wodurch Fehlermeldungen oder Warnungen ausgegeben werden können, welche ein Durchlaufen der for-Schleife verhindern, wird die Funktion `tryCatch` bei der Schätzung angewandt. Diese setzt die Ergebnisse einer Schätzung auf NA, falls Fehler- oder Warnmeldungen ausgegeben werden und verhindert dadurch einen Abbruch der Schleife.

```
> Schaetzung<-tryCatch(ergm(
>     simul_netzw ~ edges + kstar(2)) + triangle(),
>       control=control.ergm(main.method = "Stepping")),
>         warning=function(w){Schaetzung=NA},
>         error=function(e){Schaetzung=NA})
```

4.2 Ergebnisse

Die Ergebnisse sind in drei Abschnitte unterteilt. Zu Beginn werden Unterschiede der Schätzmethoden hinsichtlich der für eine Schätzung benötigten Zeit analysiert. Anschließend wird die Quote betrachtet, mit der die simulationsbasierten Schätzalgorithmen konvergieren, wobei in diesem Abschnitt teilweise auch schon auf qualitative Unterschiede in den Schätzungen eingegangen wird. Zum Schluss wird in einem dritten Abschnitt die Qualität der Schätzungen genauer untersucht.

4.2.1 Zeit

Zunächst soll auf die Zeit eingegangen werden, welche die vier Schätzmethoden für eine Schätzung benötigen. Tabelle 1 zeigt die für jede der vier Schätzmethoden durchschnittlich für eine Schätzung benötigte Zeit in Abhängigkeit von der Anzahl an Knoten im simulierten Netzwerk. Dafür werden jeweils 50 Netzwerke mit 100, 200, 400, 800, 1600 und 3200 Knoten simuliert. Daraufhin wird auf jedem simulierten Netzwerk für jede der vier Schätzmethoden jeweils eine Schätzung durchgeführt. Jedes Netzwerk wird dabei aus demselben ERGM = edges + kstar(2) + triangle() gezogen, wobei die dazugehörigen Parameterwerte $\theta_{edges} = -2.5$, $\theta_{kstar(2)} = -0.2$ und $\theta_{triangle} = 0.5$ festgelegt werden. Bei der anschließenden Schätzung wird ein ERGM mit identischen Netzwerk Statistiken (θ_{edges}, $\theta_{kstar(2)}$ und $\theta_{triangle}$) geschätzt. Es sei jedoch erwähnt, dass Simulationen mit anderen Netzwerk Statistiken und Modellparameterwerten die Ergebnisse aus Tabelle 1 bestätigen. Demnach steigt die durchschnittlich benötigte Zeit bis zur Konvergenz für alle drei simulationsbasierten Schätzmethoden und auch für die Maximum Pseudo-Likelihood Schätzung (MPLE) mit der Anzahl an Knoten im zu schätzenden Netzwerk deutlich an. In Tabelle 1 fällt bei allen Schätzmethoden auf, dass mit jeder Erhöhung der Anzahl an Knoten im zu schätzenden Netzwerk auch die für eine Schätzung durchschnittlich benötigte Zeit zunimmt. So benötigt zum Beispiel der Stepping Algorithmus für die Netzwerke mit 100 Knoten durchschnittlich nur 2.67 Sekunden, für die Netzwerke mit 200 Knoten bereits 3.50 Sekunden und für die simulierten Netzwerke mit 3200 Knoten benötigt

Anzahl Knoten	Stepping Algorithmus Zeit	MCMCMLE Zeit	Robbins-Monro Zeit	MPLE Zeit
100	2.67	7.16	8.32	0.004
200	3.50	8.97	11.46	0.007
400	5.16	10.54	16.61	0.12
800	11.26	20.80	38.36	0.49
1600	25.24	33.98	71.61	2.09
3200	68.20	78.33	190.44	9.64

Tabelle 1: Durchschnittlich für eine Schätzung benötigte Zeit (in Sekunden) der vier Schätzmethoden in Abhängigeit der Anzahl an Knoten im simulierten Netzwerk. Alle Netzwerke wurden mit dem selben ERGM = edges + kstar(2) + triangle() simuliert und anschließend geschätzt. Für die Simulation wurden die Parameterwerte $\theta_{edges} = -2.5$, $\theta_{kstar(2)} = -0.2$ und $\theta_{triangle} = 0.5$ gewählt.

er im Durchschnitt 68.20 Sekunden.

In Tabelle 1 liefert die Maximum Pseudo-Likelihood Schätzung, als einzige nicht simulationsbasierte Schätzmethode, die mit Abstand schnellsten Schätzungen. Unter den simulationsbasierten Schätzmethoden schafft der Stepping Algorithmus die schnellsten Schätzungen. Dahinter folgt das MCMCMLE Verfahren. Dabei scheint der Anstieg der durchschnittlich für eine Schätzung benötigten Zeit bei steigender Anzahl an Knoten im simulierten Netzwerk beim Stepping Algorithmus größer auszufallen als beim MCMCMLE Verfahren. Demnach scheinen sich diese beiden Schätzmethoden mit steigender Anzahl an Knoten im Netzwerk zeitlich anzunähern. Für deutlich größere Netzwerke wäre es deshalb denkbar, dass sich die Ergebnisse bezüglich des Zeitfaktors umkehren und das MCMCMLE Verfahren schneller sein könnte als der Stepping Algorithmus. Die langsamste Schätzmethode ist die stochastische Approximation nach Robbins-Monro. Diese Schätzmethode benötigt vor allem bei den größeren Netzwerken mit 1600 beziehungsweise 3200 Knoten deutlich mehr Zeit als die anderen. Nur in sehr wenigen Ausnahmefällen konnte bei den im Rahmen dieser Arbeit durchgeführten Simulationen beobachtet werden, dass eine stochastische Approximation nach Robbins-Monro weniger Zeit benötigte, als eine Schätzung mit dem MCMCMLE Verfahren. Darüber hinaus handelte es sich bei diesen Ausnahmefällen beinahe ausschließlich um Netzwerke mit einer sehr geringen Anzahl an Knoten ($\leqslant 100$).

Wie aus Tabelle 2 hervorgeht, hat auch die Dichte eines Netzwerkes große Auswirkungen auf die von den Schätzalgorithmen benötigte Zeit bis zur Konvergenz. Da für dieses spezielle ERGM die durchschnittlichen Dichten der simulierten Netzwerke mit Erhöhung der Anzahl der Knoten etwas sinken, kann ausgeschlossen werden, dass der beobachtete Zeitanstieg mit steigender Anzahl an Knoten in Tabelle 1 nur auf einen Anstieg der Dichte in den Netzwerken mit einer höheren Anzahl an Knoten

zurückzuführen ist.

In Tabelle 2 ist die durchschnittlich für eine Schätzung benötigte Zeit in Abhängigkeit der Dichte der simulierten Netzwerke zu sehen. Die Dichte eines ungerichteten Netzwerkes kann dabei durch die in Abschnitt 2 vorgestellte Gleichung (1) bestimmt werden. Für Tabelle 2 wird erneut ein ERGM = edges + kstar(2) + triangle() für die Simulation von zufälligen Netzwerken und die anschließenden Schätzungen verwendet. Bei der Ziehung zufälliger Netzwerke aus diesem ERGM bleiben die Parameterwerte $\theta_{kstar(2)} = -0.2$ und $\theta_{triangle} = 0.5$ stets unverändert. Um die durchschnittlich für eine Schätzung benötigte Zeit in Abhängigkeit der Dichte eines Netzwerkes analysieren zu können, werden jedoch unterschiedliche Intercept-Werte θ_{edges} getestet. Für jeden Intercept-Wert θ_{edges} werden 50 Netzwerke mit 400 Knoten simuliert und daraufhin, mit allen vier Schätzmethoden, auf den simulierten Netzwerken eine Schätzung mit identischen Netzwerk Statistiken durchgeführt. Alle simulierten Netzwerke enthalten genau 400 Knoten, um sicherzustellen, dass die gemessenen Zeitunterschiede nur auf Änderungen der durchschnittlichen Dichte der Netzwerke zurückgeführt werden können. Wie Tabelle 2 entnommen werden kann, ergeben sich für größere Intercept-Werte θ_{edges} bei der Simulation Netzwerke mit einer höheren Dichte als für kleinere Intercept-Werte. So ergibt sich für einen Parameterwert $\theta_{edges} = 0$ eine durchschnittliche Dichte der simulierten Netzwerke von 0.0247, wohingegen für $\theta_{edges} = -4$ die simulierten Netzwerke im Durchschnitt eine Dichte von 0.007 aufweisen.

Tabelle 2 zeigt deutlich, dass die für eine Schätzung durchschnittlich benötigte Zeit mit zunehmender Dichte des Netzwerkes ansteigt. So steigt die für eine Schätzung durchschnittlich benötigte Zeit für Netzwerke mit einer höheren durchschnittlichen Dichte bei allen vier Schätzmethoden ausnahmslos an. So benötigt man beispielsweise für eine Schätzung mit dem Stepping Algorithmus bei einem Parameterwert $\theta_{edges} = 0$ und einer sich daraus ergebenden durchschnittlichen Dichte der simulierten Netzwerke von 0.0247 im Durchschnitt 12.12 Sekunden. Wählt man hingegen einen etwas höheren edge Parameter $\theta_{edges} = 1$, wodurch die durchschnittliche Dichte der simulierten Netzwerke leicht auf 0.020 sinkt, benötigt man für die Schätzung eines dieser Netzwerke im Durchschnitt nur noch 10.03 Sekunden. Bei einem Parameterwert $\theta_{edges} = -4$ und einer damit einhergehenden Verringerung der Dichte auf 0.007, benötigt der Stepping Algorithmus durchschnittlich nur noch 4.01 Sekunden für eine Schätzung.

Vergleicht man die Schätzmethoden untereinander, so ergibt sich dasselbe Bild wie in Tabelle 1. Für Netzwerke derselben durchschnittlichen Dichte, liefert die Maximum Pseudo-Likelihood Schätzung die mit Abstand schnellsten Schätzungen. Innerhalb der simulationsbasierten Schätzmethoden erfolgt eine Schätzung mit dem

θ_{edges}	∅ Dichte	Stepping Algorithmus Zeit	MCMCMLE Zeit	Robbins-Monro Zeit	MPLE Zeit
0	0.0247	12.12	26.47	40.06	0.29
-1	0.020	10.03	22.85	38.03	0.27
-2	0.015	6.12	12.27	18.72	0.16
-3	0.010	4.18	7.98	11.86	0.11
-4	0.007	4.01	7.71	10.69	0.10

Tabelle 2: Durchschnittlich für eine Schätzung benötigte Zeit (in Sekunden) der vier Schätzmethoden, in Abhängigeit der Dichte der simulierten Netzwerke. Alle simulierten Netzwerke enthalten 400 Knoten und wurden mit dem selben ERGM = edges + kstar(2) + triangle() simuliert und anschließend geschätzt. Bei der Simulation zufälliger Netzwerke wurden lediglich die Intercept-Werte θ_{edges} variiert, um die durchschnittliche Dichte der simulierten Netzwerke zu verändern.

Stepping Algorithmus am schnellsten. Das MCMCMLE Verfahren benötigt etwas mehr Zeit als der Stepping Algorithmus und die stochastische Approximation nach Robbins-Monro dauert am längsten. Auch hier kann nicht ausgeschlossen werden, dass sich die zeitabhängige Reihenfolge verändern könnte, wenn die simulierten Netzwerke deutlich mehr als 400 Knoten enthielten. In einem solchen Fall wäre es denkbar, dass der Stepping Algorithmus mehr Zeit für eine Schätzung benötigt als das MCMCMLE Verfahren.

4.2.2 Konvergenzquote

Nun soll die Quote betrachtet werden, mit der die Schätzalgorithmen konvergieren. Dabei sollen vor allem die simulationsbasierten Schätzmethoden genauer betrachtet werden, da es bei der Maximum Pseudo-Likelihood Schätzung nicht zu Konvergenzproblemen kommen kann. Trotzdem wird für die MPLE eine Konvergenzquote angegeben. Die Konvergenzquote hängt sehr stark von den ausgesuchten Netzwerk Statistiken und den dazu gewählten Parameterwerten ab.

Werden die Netzwerk Statistiken und ihre Parameterwerte so bestimmt, dass man ein stabiles Modell erhält, führen geringe Änderungen der Parameterwerte nur zu sehr geringen Veränderungen in den Netzwerkstrukturen der aus dem Modell gezogenen Netzwerke. Des Weiteren sind sich die aus einem stabilen Modell zufällig gezogenen Netzwerke sehr ähnlich. In einem solchen Fall gibt es bei keinem der Schätzalgorithmen Probleme bei der Konvergenz.

Ein Beispiel für ein stabiles Modell, bei dem es folglich zu keinerlei Konvergenzproblemen kommt, ist das ERGM = edges + GWESP($\alpha = 0.5$) , mit $\theta_{edges} = -3$ und

	Stepping Algorithmus	MCMC MLE	Robbins-Monro	MPLE
Quote Konvergenz	100 %	100 %	100 %	100 %
Mean ($\hat{\theta}_{edges}$)	-2.99	-2.99	-3.00	-3.00
Mean ($\hat{\theta}_{GWESP}$)	0.29	0.29	0.30	0.30
MSE ($\hat{\theta}_{edges}$)	0.007	0.007	0.008	0.008
MSE ($\hat{\theta}_{GWESP}$)	0.003	0.003	0.004	0.003

Tabelle 3: Ergebnisse der Simulation mit einem ERGM = edges + GWESP($\alpha = 0.5$), mit $\theta_{edges} = -3$ und $\theta_{GWESP} = 0.3$. Diese Tabelle zeigt die Quote, mit der die simulationsbasierten Schätzalgorithmen konvergieren, wobei auch eine Konvergenzquote für die Maximum Pseudo-Likelihood Schätzung angegeben wird. Darüber hinaus werden der Mittelwert und die mittlere quadratische Abweichung (MSE) für jeden Parameter und für jede Schätzmethode gezeigt. Diese werden aus den für jede Schätzmethode jeweils 50 durchgeführten Schätzungen berechnet.

$\theta_{GWESP} = 0.3$. Tabelle 3 zeigt die Konvergenzquoten der vier Schätzmethoden, die sich bei einer wie in Abschnitt 4.1 beschriebenen Simulation mit diesem ERGM ergeben. Zusätzlich zeigt Tabelle 3 ausgewählte Statistiken der Parameterschätzungen. Die Anzahl an Knoten in den gezogenen Netzwerken wurde bei dieser Simulation auf 100 festgesetzt. Wie bereits angekündigt, kann man Tabelle 3 entnehmen, dass alle vier Schätzmethoden eine Konvergenzquote von 100% erzielen. Darüber hinaus ist auch die Qualität der Schätzungen sehr gut. Die Mittelwerte der Schätzungen beider Parameter entsprechen bei allen vier Schätzmethoden den ursprünglich eingesetzten Parameterwerten. Auch die mittlere quadratische Abweichung (MSE), welche angibt, wie stark ein Punktschätzer um den zu schätzenden Wert streut, ist bei den Schätzungen beider Modellparameter gering. Ausführlichere Erläuterungen bezüglich der Qualität der Schätzungen finden sich in Abschnitt 4.2.3 .

Wählt man dagegen Netzwerk Statistiken und dazugehörige Parameterwerte die zusammen ein eher instabiles Modell ergeben, kann es zu großen Unterschieden bei der Quote kommen, mit der die verschiedenen Schätzalgorithmen konvergieren. Bei instabilen Modellen können schon geringe Änderungen der Parameterwerte zu großen Veränderungen in den Netzwerkstrukturen der aus dem Modell gezogenen Netzwerke führen. Darüber hinaus können sich die aus einem instabilen Modell zufällig gezogenen Netzwerke stark voneinander unterscheiden. Bei Netzwerken, die aus einem eher instabilen ERGM gezogen wurden, ist es das MCMCMLE Verfahren, bei dem es als erstes zu Problemen bei der Konvergenz kommt. In den Simulationen zeigte sich, dass die Konvergenzquote des MCMCMLE Verfahrens immer kleiner oder gleich der Konvergenzquoten der anderen Schätzmethoden ist.

Ein Beispiel für ein eher instabiles Modell, bei dem es zu Konvergenzproblemen

kommt, ist das ERGM = edges + k-star(2), mit $\theta_{edges} = -4.1$ und $\theta_{k-star(2)} = 0.1$. Tabelle 4 zeigt die Konvergenzquoten der vier Schätzmethoden, die sich bei einer wie in Abschnitt 4.1 beschriebenen Simulation mit diesem ERGM ergeben, sowie den Mittelwert und die mittlere quadratische Abweichung der Schätzungen für jeden Modellparameter. Die Anzahl an Knoten in den gezogenen Netzwerken wurde bei dieser Simulation ebenfalls auf 100 festgesetzt. Zunächst fällt bei der Betrachtung von Tabelle 4 auf, dass keine der Schätzmethoden eine Konvergenzqote von 100% erreicht. Dies ist auf den ersten Blick vor allem bei der nicht simulationsbasierten Maximum Pseudo-Likelihood Schätzung überraschend, bei der es nur in 90% der Schätzungen zu einem Ergebnis kommt. Dies ist der Tatsache geschuldet, dass bei der Ziehung eines zufälligen Netzwerkes, aus diesem eher instabilen ERGM, in manchen Fällen das volle Netzwerk gezogen wird. Genauer gesagt, wurde bei dieser Simulation bei genau 10% der Ziehungen eines zufälligen Netzwerkes das volle Netzwerk gezogen. Für ein solches volles Netzwerk, kommt auch eine Maximum Pseudo-Likelihood Schätzung zu keinem Ergebnis. Darüber hinaus ist in Tabelle 4 zu sehen, dass das MCMCMLE Verfahren mit 54% die mit Abstand niedrigste Konvergenzquote aufweist. Der Stepping Algorithmus und die stochastische Approximation haben mit 78% und 70% eine deutlich höhere Konvergenzquote.

Eine Auffälligkeit, die der Tabelle 4 nicht entnommen werden kann, ist, dass die stochastische Approximation nach Robbins-Monro eher selten Fehler- oder Warnmeldungen produziert. Stattdessen werden in Fällen, in denen der Algorithmus nach Robbins-Monro nicht konvergiert, des Öfteren unplausible Parameterwerte bei der Schätzung als Ergebnis ausgegeben, die weit vom wahren Parameterwert entfernt sind. Für die Erstellung von Tabelle 4 wurden alle Schätzungen der stochastischen Approximation nach Robbins-Monro, die zu unplausiblen Parameterwerten führten, nachträglich als nicht konvergiert gewertet und blieben damit auch bei der Berechnung der Mittelwerte und der mittleren quadratischen Abweichungen für Tabelle 4 unberücksichtigt.

Wie sich Tabelle 4 entnehmen lässt, ist die Qualität der übrig gebliebenen Schätzungen jedoch ziemlich gut. So entsprechen die Mittelwerte der Schätzungen für beide Parameter bei allen vier Schätzmethoden in etwa den ursprünglich eingesetzten Parameterwerten. Auch die mittleren quadratischen Abweichungen der Schätzungen beider Modellparameter bestätigen die eher gute Qualität der Schätzungen. Diese liegen für den edge Parameter, je nach verwendeter Schätzmethode, zwischen 0.07 und 0.11, was einer geringen mittleren quadratischen Abweichung entspricht. Für den 2-star Parameter ergibt sich für alle verwendeten Schätzmethoden, mit Werten des MSE zwischen 0.001 und 0.003, ebenfalls eine geringe mittlere quadratische Abweichung.

	Stepping Algorithmus	MCMC MLE	Robbins-Monro	MPLE
Quote Konvergenz	78 %	54 %	70 %	90 %
Mean ($\hat{\theta}_{edges}$)	-3.95	-3.87	-3.96	-4.00
Mean ($\hat{\theta}_{k-star(2)}$)	0.08	0.06	0.08	0.06
MSE ($\hat{\theta}_{edges}$)	0.07	0.10	0.11	0.07
MSE ($\hat{\theta}_{k-star(2)}$)	0.001	0.002	0.003	0.001

Tabelle 4: Ergebnisse der Simulation mit einem ERGM = edges + k-star(2), mit $\theta_{edges} = -4.1$ und $\theta_{k-star(2)} = 0.1$. Diese Tabelle zeigt die Quote, mit der die simulationsbasierten Schätzalgorithmen konvergieren, wobei auch eine Konvergenzquote für die Maximum Pseudo-Likelihood Schätzung angegeben wird. Darüber hinaus werden der Mittelwert und die mittlere quadratischer Abweichung (MSE) für jeden Parameter und für jede Schätzmethode gezeigt. Diese werden aus den für jede Schätzmethode jeweils 50 durchgeführten Schätzungen berechnet.

Eine weitere Auffälligkeit im Zusammenhang mit unplausiblen Werten bei der Parameterschätzung zeigt sich im Rahmen einer Simulation, die sich in ihrem Aufbau etwas von der in Abschnitt 4.1 vorgestellten Simulation unterscheidet. Bei dieser Simulation wird nur ein einziges zufälliges Netzwerk aus einem ERGM gezogen. Anschließend wird 50 mal ein ERGM mit denselben Netzwerk Statistiken, die bei der Ziehung des zufälligen Netzwerkes verwendet wurden, auf diesem einen Netzwerk geschätzt. Zunächst soll das Netzwerk, welches in Abbildung 11 zu sehen ist, 50 mal mit allen vier Schätzmethoden geschätzt werden. Dieses Netzwerk wurde aus einem eher instabilen ERGM = edges + kstar(2) + triangle() , mit $\theta_{edges} = 2.3$, $\theta_{kstar(2)} = -0.6$ und $\theta_{triangle} = 1.3$ gezogen.

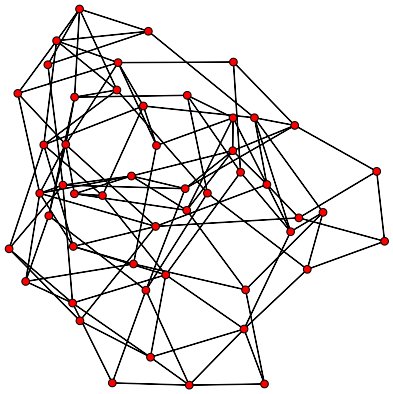

Abbildung 11: Das Netzwerk, welches aus einem ERGM = edges + kstar(2) + triangle() , mit $\theta_{edges} = 2.3$, $\theta_{kstar(2)} = -0.6$ und $\theta_{triangle} = 1.3$ gezogen wurde.

Abbildung 12 zeigt zwei Boxplots, welche die Ergebnisse der 50 Schätzungen des Netzwerkes aus Abbildung 11 mit dem Stepping Algorithmus und der stochastischen Approximation nach Robbins-Monro für den edge Parameter θ_{edge} visualisieren. Beide Schätzmethoden liefern bei allen 50 Schätzungen Ergebnisse, ohne dass es zu Warn- oder Fehlermeldungen kommt. Bei diesem Simulationsaufbau ist es jedoch nicht sinnvoll, die eingesetzten Parameterwerte mit den geschätzten zu vergleichen, da es sich lediglich um ein einzelnes zufällig gezogenes Netzwerk handelt und man folglich die wahren Parameterwerte nicht genau kennt. Es wäre allerdings zu erwarten, dass Schätzungen desselben Netzwerkes zu sehr ähnlichen Ergebnissen führen und nur sehr kleine Unterschiede in den Schätzungen, aufgrund der durch die Simulation verursachten Unsicherheit, zu beobachten sind. So zu sehen auch in dem linken Boxplot aus Abbildung 12, der die 50 Schätzungen des edge Parameters mit dem Stepping Algorithmus visualisiert und aufgrund der beinahe identischen Schätzungen, in diesem Maßstab, kaum mehr als Boxplot zu erkennen ist. Alle 50 Schätzungen des edge Parameters mit dem Stepping Algorithmus liegen zwischen 3.45 und 3.66 und sind sich demnach, wie erwartet, sehr ähnlich. Die Schätzungen, welche mit der stochastischen Approximation nach Robbins-Monro durchgeführt wurden, ergeben jedoch ein anderes Bild. Der rechte Boxplot visualisiert die 50 Schätzungen des edge Parameters mit der stochastischen Approximation nach Robbins-Monro. Hier schwanken die Schätzungen des edge Parameters für dasselbe Netzwerk sehr stark. Dies ist vor allem im Vergleich mit dem linken Boxplot, also den Schätzungen des edge Parameters mit dem Stepping Algorithmus, deutlich zu erkennen. Bei der stochastischen Approximation nach Robbins-Monro liegen 50% der Schätzungen des edge Parameters zwischen 2.23 und 3.98. Darüber hinaus sind auch extreme Ausreißer zu beobachten. So gibt es eine Schätzung, die einen edge Parameter von nur -4.25 ergibt, wohingegen eine andere Schätzung den edge Parameter auf 6.18 schätzt. Abbildung 12 zeigt demnach, dass Schätzungen desselben Netzwerkes mit der stochastischen Approximation nach Robbins-Monro unter Umständen stark voneinander abweichen können.

Auf einen Boxplot für die Ergebnisse des MCMCMLE Verfahrens wurde in Abbildung 12 verzichtet, da es bei der Schätzung dieses Netzwerkes mit dem MCMCMLE Verfahren zu Problemen bei der Konvergenz des Schätzalgorithmus kommt. Es handelt sich demnach bei den Schätzungen des in Abbildung 11 gezeigten Netzwerkes um ein weiteres Beispiel, welches vorschlägt, dass das MCMCMLE Verfahren am ehesten zu Konvergenzproblemen führt. Nur 8 der 50 durchgeführten Schätzungen führten zu einem Ergebnis, welche denen des Stepping Algorithmus sehr ähnlich waren. Ein Boxplot dieser 8 Schätzungen würde somit dem linken der beiden Boxplots aus Abbildung 12 in etwa entsprechen. Auf einen Boxplot der 50 Schätzungen mit der nicht simulationsbasierten Maximum Pseudo-Likelihood Methode wurde ebenfalls

verzichtet, da mehrere Schätzungen desselben Netzwerkes mit der Maximum Pseudo-Likelihood Methode, immer zu demselben Ergebnis führen.

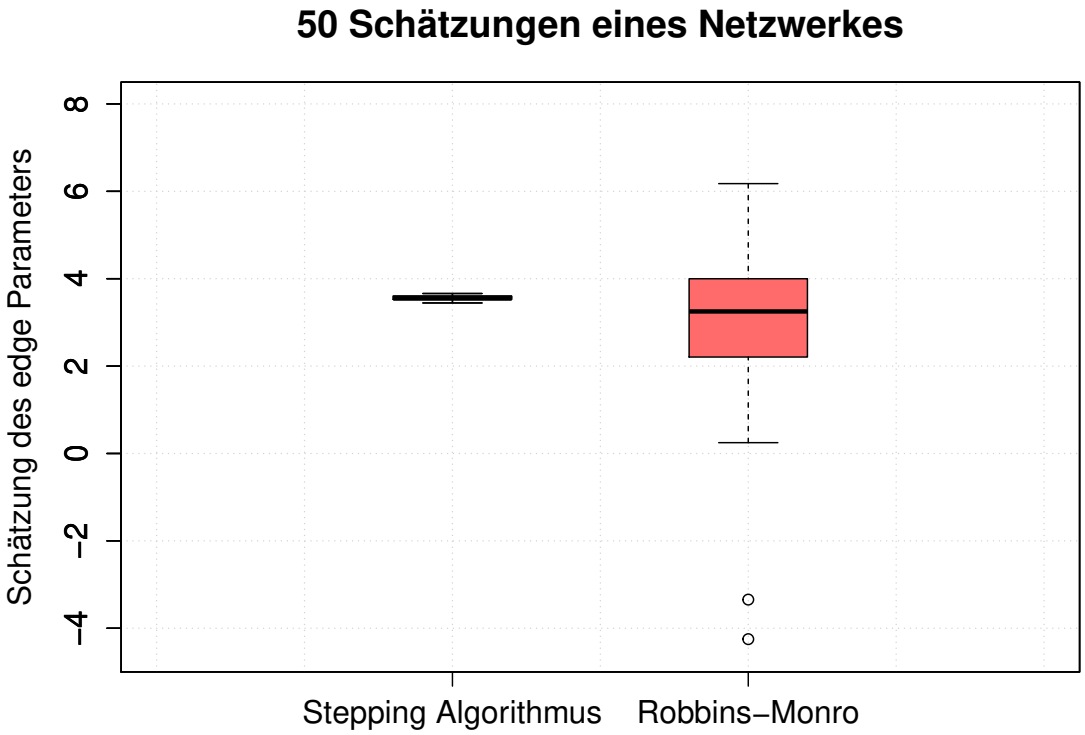

Abbildung 12: Boxplots der Ergebnisse der 50 Schätzungen des edge Parameters θ_{edge} des Netzwerkes aus Abbildung 11. Links ist der Boxplot der Ergebnisse der Schätzungen mit dem Stepping Algorithmus zu sehen, rechts ist der Boxplot der Ergebnisse der Schätzungen, welche mit der stochastischen Approximation nach Robbins-Monro durchgeführt wurden, abgebildet.

Bei weiteren Simulationen konnten jedoch Beispiele gefunden werden, bei denen sich auch die Schätzungen eines Netzwerkes mit dem Stepping Algorithmus stark voneinander unterscheiden. Zu diesen Beispielen gehört ein Netzwerk, welches aus einem eher instabilen ERGM = edges + kstar(2) + triangle() , mit $\theta_{edges} = 2.3$, $\theta_{kstar(2)} = -0.6$ und $\theta_{triangle} = 1.59$ gezogen wurde. Auf diesem Netzwerk wird nun erneut 50 mal mit allen drei simulationsbasierten Schätzmethoden und denselben Netzwerk Statistiken, die bei der Ziehung dieses zufälligen Netzwerkes verwendet wurden, ein ERGM geschätzt. Abbildung 13 veranschaulicht die Ergebnisse der jeweils 50 Schätzungen dieses Netzwerkes für den Stepping Algorithmus und für die stochastische Approximation nach Robbins-Monro. Der linke, blau gefärbte Boxplot visualisiert die Ergebnisse der Schätzungen des edge Parameters mit dem Stepping Algorithmus. Der rechte, rote Boxplot zeigt die Ergebnisse der Schätzungen des edge Parameters, die sich mit der stochastischen Approximation nach Robbins-Monro

ergeben. Auf einen Boxplot der Ergebnisse des MCMCMLE Verfahrens wird hier erneut verzichtet, da nur eine einzige der 50 Schätzungen mit dieser Schätzmethode zu einem Ergebnis führte. Die 50 Schätzungen mit dem Stepping Algorithmus und der stochastischen Approximation nach Robbins-Monro führten hingegen jedes Mal zu einem Ergebnis, ohne dass es zu Warn- oder Fehlermeldungen kam. Abbildung 13 zeigt, dass sich die Schätzungen desselben Netzwerkes diesmal bei beiden Schätzmethoden stark voneinander unterscheiden. So liegen 50% der Schätzungen des edge Parameters mit dem Stepping Algorithmus zwischen 1.25 und 2.23. Der kleinste mit dem Stepping Algorithmus geschätzte Wert für den edge Parameter ist mit -15.90 in Abbildung 13, aus Gründen der Übersicht, nicht zu sehen. Der größte geschätzte Wert für den edge Parameter ist mit 2.62 dagegen schon in Abbildung 13 enthalten. Die Schätzungen, die mit der stochastischen Approximation nach Robbins-Monro durchgeführt wurden, unterscheiden sich ebenfalls ziemlich stark. So gibt es eine Schätzung, die einen edge Parameter von -0.16 vorschlägt, und eine andere, bei der der edge Parameter auf 4.13 geschätzt wird.

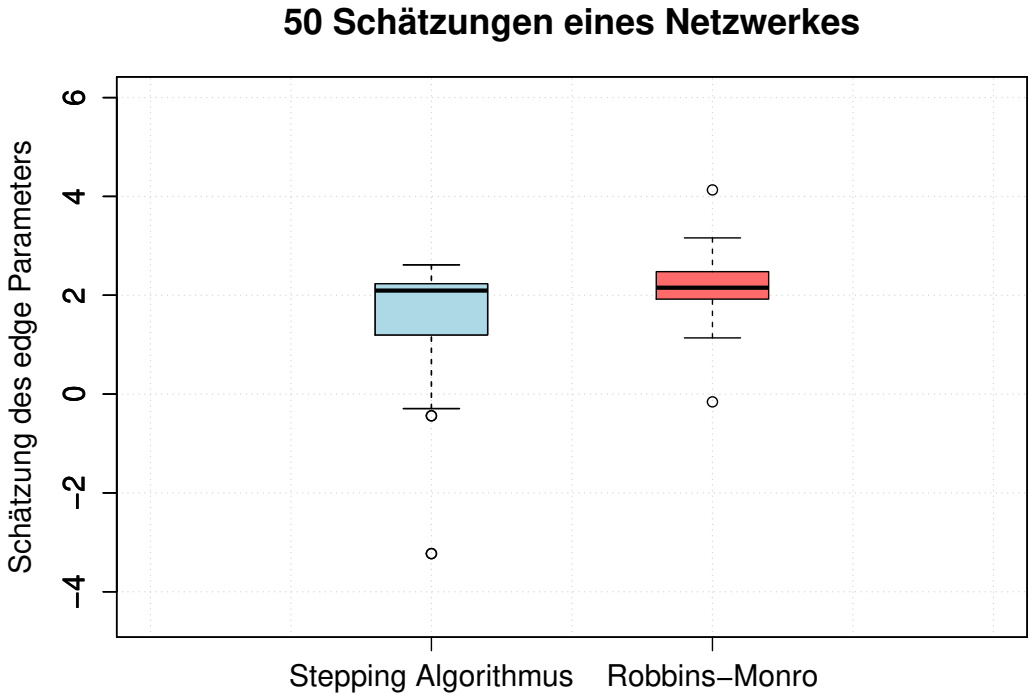

Abbildung 13: Boxplots der Ergebnisse der Schätzungen eines Netzwerkes, welches aus einem ERGM = edges + kstar(2) + triangle() , mit $\theta_{edges} = 2.3$, $\theta_{kstar(2)} = -0.6$ und $\theta_{triangle} = 1.59$ gezogen wurde. Links ist der Boxplot der Ergebnisse der Schätzungen mit dem Stepping Algorithmus zu sehen, rechts ist der Boxplot der Ergebnisse der Schätzungen, welche mit der stochastischen Approximation nach Robbins-Monro durchgeführt wurden, abgebildet

Tatsächlich ergaben die im Rahmen dieser Arbeit durchgeführten Simulationen, dass es bei allen drei simulationsbasierten Schätzmethoden passieren kann, dass sich die geschätzten Parameterwerte bei einer zweiten Schätzung desselben Netzwerkes durchaus stark von den geschätzten Parameterwerten einer ersten Schätzung unterscheiden. Es sei jedoch angemerkt, dass dies nur bei Netzwerken beobachtet werden konnte, die aus eher instabilen ERGMs gezogen wurden. Bei Netzwerken die aus stabilen ERGMs gezogen und anschließend mehrmals geschätzt wurden, unterscheiden sich die einzelnen Schätzungen desselben Netzwerkes kaum voneinander. Allerdings konnte auch hier beobachtet werden, dass die Schätzungen eines Netzwerkes mit der stochastischen Approximation nach Robbins-Monro etwas stärker schwanken, als die Schätzungen desselben Netzwerkes mit dem Stepping Algorithmus oder dem MCMCMLE Verfahren.

Bei Netzwerken die aus eher instabilen ERGMs gezogen und anschließend mehrmals geschätzt wurden, war es die stochastische Approximation nach Robbins-Monro, bei der starke Abweichungen zwischen den einzelnen Schätzungen am häufigsten beobachtet werden konnten. So gab es, wie in Abbildung 13 zu sehen, Beispiele, bei denen sich mehrere Schätzungen eines Netzwerkes sowohl mit der stochastischen Approximation nach Robbins-Monro als auch mit dem Stepping Algorithmus deutlich voneinander unterschieden. Jedoch konnten keine Beispiele gefunden werden, bei denen mehrere Schätzungen eines Netzwerkes mit der stochastischen Approximation nach Robbins-Monro zu sehr ähnlichen Parameterschätzungen führen, aber sich mehrere Schätzungen desselben Netzwerkes mit dem Stepping Algorithmus stark voneinander unterscheiden. Im Gegensatz dazu kommt es, wie in Abbildung 12 zu sehen, allerdings schon vor, dass mehrere Schätzungen eines Netzwerkes mit dem Robbins-Monro Algorithmus zu sehr unterschiedlichen Parameterschätzungen führen, während mehrere Schätzungen desselben Netzwerkes mit dem Stepping Algorithmus zu sehr ähnlichen Ergebnissen führen.

Unabhängig davon, welcher simulationsbasierte Schätzalgorithmus für eine Schätzung verwendet werden soll, könnte es sinnvoll sein, mehrere Schätzungen des zu analysierenden Netzwerkes durchzuführen. Falls diese Schätzungen voneinander abweichen, wäre dies ein Anzeichen dafür, dass die Parameterschätzungen nicht verlässlich sind, und es bei der Schätzung wohl zu Konvergenzproblemen kommt. In einem solchen Fall könnte es darüber hinaus sinnvoll sein, die Schätzungen mit anderen Startwerten durchzuführen.

4.2.3 Qualität der Schätzungen

Wie Tabelle 3 und Tabelle 4 bereits andeuten, führen die verschiedenen Schätzmethoden in den durchgeführten Simulationen zu guten Schätzungen der Parameterwerte. Die größten Unterschiede zwischen den Schätzmethoden konnten bei der Quote, mit der die Schätzalgorithmen konvergieren, und bei der für eine Schätzung benötigten Zeit ausgemacht werden. Kommt es bei einer Schätzung nicht zu Konvergenzproblemen, sind sich die Schätzungen der Parameterwerte aller Schätzmethoden sehr ähnlich. Besonders genau sind die Schätzungen bei Netzwerken, die aus stabilen ERGMs gezogen wurden. Tabelle 5 zeigt die ausführliche Auswertung einer, wie in Abschnitt 4.1 beschriebenen, Simulation mit einem stabilen ERGM = edges + triangle, mit $\theta_{edges} = -1$ und $\theta_{triangle} = -1$. Die Anzahl der Knoten in den zufällig gezogenen Netzwerken wurde für diese Simulation auf 100 festgelegt. Zu sehen ist, dass bei allen Schätzmethoden sowohl Mittelwert als auch Median der geschätzten Parameterwerte für beide Modellparameter beinahe identisch mit den bei der Ziehung zufälliger Netzwerke eingesetzten Parameterwerten sind. Darüber hinaus ist die mittlere quadratische Abweichung (MSE) der Parameterschätzungen beider Modellparameter sehr gering. Dies bedeutet zum einen, dass der Bias klein ist und zum anderen, dass die geschätzten Parameterwerte wenig streuen. Die geringe Varianz der geschätzten Parameterwerte kann auch an der niedrigen beobachteten Standardabweichung festgemacht werden. Selbst das Minimum und das Maximum aller 50 Schätzungen, welche zusammen den Range ergeben, liegen für jeden Modellparameter bei jeder Schätzmethode nicht weit auseinander. Eine Auffälligkeit ergibt sich allerdings bei Betrachtung der durchschnittlichen vom Modell geschätzten Standardabweichung und der tatsächlich beobachteten Standardabweichung. So sind die geschätzten Standardabweichungen der Maximum Pseudo-Likelihood Schätzung niedriger als die der simulationsbasierten Schätzmethoden. Die beobachtete Standardabweichung in den Parameterschätzungen der Maximum Pseudo-Likelihood Schätzung ist allerdings nicht niedriger als die der anderen Schätzmethoden. Tatsächlich sind die vom Modell geschätzten Standardabweichungen bei der Maximum Pseudo-Likelihood Schätzung zu niedrig. In beinahe allen Simulationen konnte festgestellt werden, dass die beobachtete Standardabweichung der Schätzungen der Maximum Pseudo-Likelihood Schätzung deutlich höher war als die vom Modell geschätzte Standardabweichung. Dies liegt daran, dass bei der Maximum Pseudo-Likelihood Schätzung fälschlicherweise angenommen wird, dass die Beobachtungen unabhängig voneinander sind. Eine weitere Auffälligkeit in diesem Kontext, welche in Tabelle 5 allerdings nicht zu sehen ist, konnte bei Betrachtung weiterer Simulationen im Hinblick auf die stochastische Approximation nach Robbins-Monro beobachtet werden. Hier tendiert die beobachtete Standardabweichung der Parameterschätzungen etwas höher zu sein als die der anderen Schätzmethoden.

	Stepping Algorithmus	MCMC MLE	Robbins-Monro	MPLE
$\hat{\theta}_{\mathbf{edges}}$:				
Mean / Median	-1.00 / -0.98	-1.00 / -0.98	-1.00 / -0.98	-1.00 / -0.99
Range	[-1.14 ; -0.86]	[-1.14 ; -0.86]	[-1.21 ; -0.86]	[-1.16 ; -0.84]
Standardabweichung geschätzt/beobachtet	0.09 / 0.07	0.09 / 0.07	0.11 / 0.09	0.06 / 0.08
MSE	0.005	0.005	0.008	0.006
$\hat{\theta}_{\mathbf{triangle}}$:				
Mean / Median	-1.02 / -1.01	-1.02 / -1.01	-1.02 / -1.02	-1.02 / -1.01
Range	[-1.16 ; -0.84]	[-1.17 ; -0.84]	[-1.18 ; -0.86]	[-1.16 ; -0.82]
Standardabweichung geschätzt/beobachtet	0.10 / 0.08	0.10 / 0.08	0.10 / 0.08	0.06 / 0.08
MSE	0.007	0.007	0.007	0.007

Tabelle 5: Ergebnisse der Simulation mit einem ERGM = edges + triangle , mit $\theta_{edges} = -1$ und $\theta_{triangle} = -1$. Insgesamt wurden 50 zufällige Netzwerke aus diesem ERGM gezogen. Diese Tabelle zeigt die Auswertung der auf diesen gezogenen Netzwerken durchgeführten Schätzungen für jede Schätzmethode.

Abbildung 14 veranschaulicht die Simulationsergebnisse aus Tabelle 5. Die linke Grafik in Abbildung 14 zeigt für jede Schätzmethode einen Boxplot der 50 durchgeführten Schätzungen für den edge Parameter θ_{edges}. In der rechten Grafik sind die Boxplots für die Schätzungen des triangle Parameters $\theta_{triangle}$ zu sehen. Die rot gestrichelte Linie zeigt in beiden Grafiken den Parameterwert an, der bei der Ziehung der zufälligen Netzwerke eingesetzt wurde ($\theta_{edges} = \theta_{triangle} = -1$). Abbildung 14 zeigt deutlich, dass für jede Schätzmethode die Schätzungen der Parameterwerte der simulierten Netzwerke sehr nah an den eingesetzten Parameterwerten liegen. So schneidet die rot gestrichelte Linie die Boxplots in etwa in der Mitte. Vergleicht man die Boxplots untereinander, so sind kaum Unterschiede in den Schätzungen zu erkennen. Alle Schätzmethoden führen demnach zu sehr guten Schätzungen. Anders als in van Duijn et al. (2009), konnte in diesem Simulationsaufbau somit nicht gezeigt werden, dass die Maximum Pseudo-Likelihood Schätzung den simulationsbasierten Schätzmethoden unterlegen ist, wenn Abhängigkeitsstrukturen im Netzwerk vorliegen.

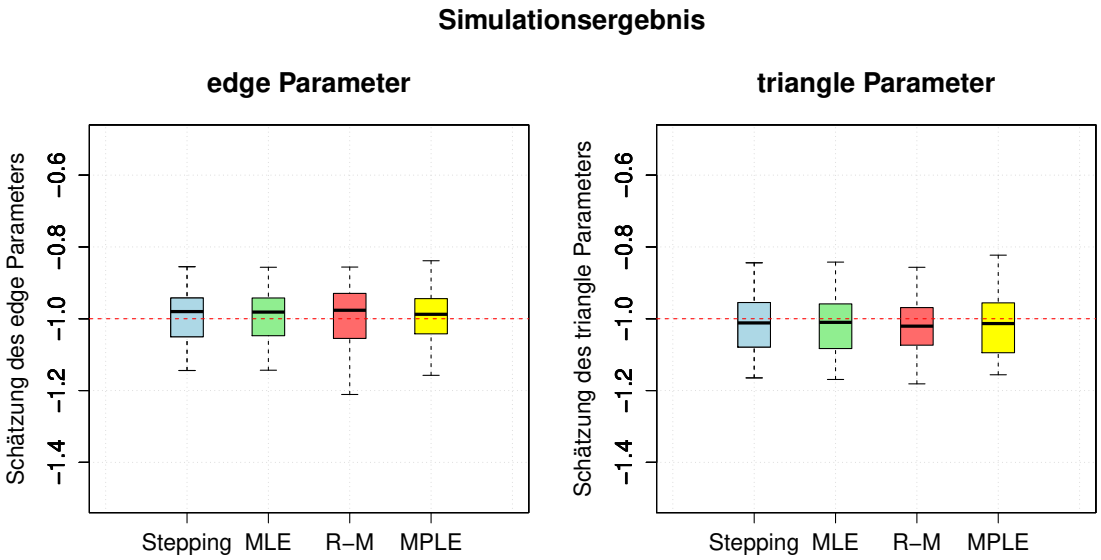

Abbildung 14: Veranschaulichung der Simulationsergebnisse aus Tabelle 5. Die Ziehung zufälliger Netzwerke erfolgte aus einem ERGM = edges + triangle(), mit $\theta_{edges} = 1$ und $\theta_{triangle} = 1$. Die linke Grafik zeigt für jede Schätzmethode einen Boxplot der 50 durchgeführten Schätzungen für den edge Parameter θ_{edges}. Der blaue Boxplot visualisiert die Schätzungen des Stepping Algorithmus, der grüne die des MCMCMLE Verfahrens, der rote die der stochastischen Approximation nach Robbins-Monro und der gelbe die der Maximum Pseudo-Likelihood Schätzung. Die rechte Grafik zeigt für jede Schätzmethode einen Boxplot der 50 durchgeführten Schätzungen für den triangle Parameter $\theta_{triangle}$, wobei die Reihenfolge und die Farbauswahl der Boxplots für die verschiedenen Schätzmethoden dieselben bleiben.

Wie Abschnitt 4.2.2 bereits gezeigt hat, kommt es bei instabilen ERGMs zu Problemen bei der Konvergenz und teilweise auch zu unplausiblen Schätzungen der Parameterwerte. Vergleicht man die übrig gebliebenen Schätzungen der Schätzmethoden untereinander, also in den Fällen, in denen der Schätzalgorithmus konvergiert, führen alle Schätzmethoden trotzdem zu ziemlich guten und sehr ähnlichen Ergebnissen. Unter den simulationsbasierten Schätzmethoden könnte demnach der Stepping Algorithmus grundsätzlich eine gute Wahl sein, da es mit ihm, im Vergleich zu den anderen simulationsbasierten Schätzmethoden, seltener zu Konvergenzproblemen kommt und nur in Ausnahmefällen unplausible Parameterwerte geschätzt werden.

5 Fazit

Aufgrund der vorherrschenden komplexen Abhängigkeitsstruktur in vielen realen Netzwerken erweist sich die Modellierung von Netzwerkdaten als besonders schwer. Exponential Random Graph Modelle (ERGM) ermöglichen es, Netzwerke zu modellieren. Für die Schätzung solcher ERGMs mussten allerdings spezielle simulationsbasierte Schätzalgorithmen entwickelt werden. Ohne diese ist die Maximierung der Log-Likelihood Funktion aufgrund einer darin enthaltenen, grundsätzlich unberechenbaren Normalisierungskonstante nicht möglich.

Unter Verwendung des ergm Pakets können Netzwerke mit der Softwareumgebung R analysiert und simuliert werden. Dabei stehen für die Schätzung von ERGMs drei simulationsbasierte Schätzmethoden zur Auswahl: das Markov Chain Monte Carlo Maximum Likelihood Estimation Verfahren, der Stepping Algorithmus und die stochastische Approximation nach Robbins-Monro. Zusätzlich kann auch eine vierte, nicht simulationsbasierte Maximum Pseudo-Likelihood Schätzung verwendet werden, wobei diese die Abhängigkeitsstruktur der Daten ignoriert. Diese Masterarbeit untersucht und vergleicht die Performance dieser vier Schätzmethoden an simulierten Netzwerken. Dafür werden aus einem ERGM mehrere zufällige Netzwerke simuliert, um anschließend ERGMs mit identischen Netzwerk Statistiken auf den zuvor simulierten Netzwerken zu schätzen. Dabei werden mehrere Modelle mit unterschiedlichen Netzwerk Statistiken und verschiedenen Parameterwerten getestet.

Bei der Betrachtung der Simulationsergebnisse fällt auf, dass die Performance der Schätzmethoden davon abhängt, ob für die Simulation stabile oder eher instabile ERGMs verwendet wurden. Bei der Schätzung von Netzwerken, die aus stabilen ERGMs simuliert wurden, kommt es bei keiner der simulationsbasierten Schätzmethoden zu Konvergenzproblemen. Darüber hinaus sind die Schätzungen aller Schätzmethoden sehr gut und unterscheiden sich kaum voneinander. Bei den Simulationsergebnissen eher instabiler ERGMs konnten allerdings deutliche Unterschiede bei der Quote, mit der die Schätzalgorithmen konvergieren, festgestellt werden. Hier ist es das vom ergm Paket standardmäßig verwendete Markov Chain Monte Carlo Maximum Likelihood Estimation Verfahren, bei dem es am ehesten zu Konvergenzproblemen kommt. Weniger häufig kommt es bei den anderen simulationsbasierten Schätzmethoden zu Konvergenzproblemen, wobei es tendenziell der Stepping Algorithmus ist, bei dem am seltensten Konvergenzprobleme beobachtet werden konnten.

Zusätzliche Simulationen wurden durchgeführt, um zu überprüfen, wie stark wiederholte Schätzungen desselben Netzwerkes voneinander abweichen. Dabei konnte

festgestellt werden, dass es bei Netzwerken, die aus instabilen ERGMs simuliert wurden, vorkommen kann, dass sich wiederholte Schätzungen desselben Netzwerkes stark voneinander unterscheiden. Dies wurde bei allen simulationsbasierten Schätzmethoden festgestellt. Am häufigsten konnten diese starken Abweichungen bei wiederholten Schätzungen desselben Netzwerkes jedoch bei den Schätzungen mit der stochastischen Approximation nach Robbins-Monro beobachtet werden. Es könnte demnach allgemein sinnvoll sein, mehrere Schätzungen eines Netzwerkes durchzuführen, um zu überprüfen, ob die Schätzungen der Parameterwerte nah beieinander liegen. Voneinander stark abweichende Schätzungen würden auf unverlässliche Parameterschätzungen hinweisen. In einem solchen Fall könnte es sinnvoll sein, die Schätzungen mit anderen Startwerten zu wiederholen.

Ein geeignetes Vorgehen bei der Auswahl anderer Startwerte und eine möglicherweise daraus resultierende Veränderung der Simulationsergebnisse wurden im Rahmen dieser Arbeit nicht untersucht. Darüber hinaus erheben die präsentierten Ergebnisse keinen Anspruch auf Vollständigkeit. Bei den Simulationen wurde eine Auswahl von vielen möglichen Kombinationen verschiedener Netzwerk Statistiken betrachtet. Auch bei der Auswahl der dazugehörigen Parameterwerte konnte nur stichprobenartig vorgegangen werden.

Literaturverzeichnis

Literatur

Barndorff-Nielsen, O. (2014). *Information and Exponential Families in Statistical Theory*. JOHN WILEY & SONS INC.

Brown, L. D. (1987). *Fundamentals of Statistical Exponential Families (Ims Lecture Notes-Monograph Ser.: Vol.9)*. Inst of Mathematical Statistic.

Butts, C. T. (2014). *sna: Tools for Social Network Analysis*. R package version 2.3-2.

Caimo, A. und N. Friel (2011). Bayesian inference for exponential random graph models. *Social Networks 33*(1), 41–55.

Csardi, G. und E. D. Kolaczyk (2014). *Statistical Analysis of Network Data with R*. Springer.

Geyer, C. J. und E. A. Thompson (1992). Constrained monte carlo maximum likelihood for dependent data. *Journal of the Royal Statistical Society. Series B (Methodological) 54*(3), 657 – 699.

Handcock, M. S. (2003). Assessing degeneracy in statistical models of social networks. *Technical report, Center for Statistics and Social Sciences, University of Washington*.

Handcock, M. S., D. R. Hunter, C. T. Butts, S. M. Goodreau, P. N. Krivitsky, und M. Morris (2017). *ergm: Fit, Simulate and Diagnose Exponential-Family Models for Networks*. The Statnet Project (http://www.statnet.org). R package version 3.7.1.

Hastings, W. K. (1970). Monte carlo sampling methods using markov chains and their applications. *Biometrika 57*(1), 97 – 109.

Hummel, R. M., D. R. Hunter, und M. S. Handcock (2012). Improving simulation-based algorithms for fitting ergms. *Journal of Computational and Graphical Statistics 21*(4), 920 – 939.

Hunter, D. R. (2007). Curved exponential family models for social networks. *Social networks 29*(2), 216 – 230.

Hunter, D. R. und M. S. Handcock (2006). Inference in curved exponential family models for networks. *Journal of Computational and Graphical Statistics 15*(3), 565–583.

Kolaczyk, E. D. (2009). *Statistical Analysis of Network Data*. Springer New York.

Lusher, D., J. Koskinen, und G. Robins (2013). *Exponential Random Graph Models for Social Networks*. Cambridge University Press.

Metropolis, N., A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, und E. Teller (1953). Equation of state calculations by fast computing machines. *The Journal of Chemical Physics 21*(6), 1087 – 1092.

Morris, M., M. S. Handcock, und D. R. Hunter (2008). Specification of exponential-family random graph models: Terms and computational aspects. *Journal of Statistical Software 24*(4).

R Development Core Team (2008). *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing. ISBN 3-900051-07-0.

Robbins, H. und S. Monro (1951). A stochastic approximation method. *The Annals of Mathematical Statistics 22*(3), 400–407.

Robins, G., P. Pattison, Y. Kalish, und D. Lusher (2007). An introduction to exponential random graph (p*) models for social networks. *Social Networks 29*(2), 192 – 215.

Robins, G., P. Pattison, und P. Wang (2009). Closure, connectivity and degree distributions: Exponential random graph (p*) models for directed social networks. *Social Networks 31*, 105–117.

Schweinberger, M. (2011). Instability, sensitivity, and degeneracy of discrete exponential families. *Journal of the American Statistical Association 106*(496), 1361 – 1370.

Snijders, T. A. (2002). Markov chain monte carlo estimation of exponential random graph models. *Journal of Social Structure 3*(2), 1–40.

Snijders, T. A., P. E. Pattison, G. L. Robins, und M. S. Handcock (2006). New specifications for exponential random graph models. *Sociological Methodology 36*(1), 99 – 153.

Strauss, D. und M. Ikeda (1990). Pseudolikelihood estimation for social networks. *Journal of the American Statistical Association 85*(409), 204–212.

van Duijn, M. A., K. J. Gile, und M. S. Handcock (2009). A framework for the comparison of maximum pseudo-likelihood and maximum likelihood estimation of exponential family random graph models. *Social Networks 31*(1), 53 – 62.

Abbildungsverzeichnis

Tabellenverzeichnis

Anhang

Alle Schätzungen und Simulationen wurden mit der Softwareumgebung R durchgeführt. Auf der beiliegenden CD sind in dem Ordner „R-Codes“ alle verwendeten R-Codes zu finden. Zusätzlich ist in diesem Ordner ein PDF „Anleitung“ zu finden, in dem die Verwendung der verschiedenen R-Code Dateien erklärt wird. Einige Simulationsergebnisse wurden in Tabellen festgehalten, welche in dem Ordner „Ergebnisse“ als Power Point Präsentation unter „Tabellen“ zu finden sind. In dem Ordner „Masterarbeit“ ist die vorliegende Masterarbeit in elektronischer Form zu finden.

Eidesstattliche Erklärung

Hiermit erkläre ich, Ben Winter, die vorliegende Masterarbeit selbstständig verfasst und dabei ausschließlich die angegebenen Quellen und Hilfsmittel verwendet zu haben. Ich habe diese Arbeit weder einer anderen Prüfungsbehörde vorgelegt noch habe ich sie veröffentlicht.

München, den 24. Mai 2017

. . . . . .

(Ben Stefan Winter)