Inhaltsverzeichnis

Zusammenfassung

Draxler & Zessin (2015) haben eine Klasse pseudo-exakter oder konditionaler Tests zur Power-Berechnung von Annahmen des Rasch Modells vorgeschlagen. Zum Simulieren der für die Power-Berechnung notwendigen Daten bedarf es Sampling-Algorithmen. Verhelst (2008) hat mit dem *Rasch Sampler* einen relativ schnellen Algorithmus entworfen, der die wahre Verteilung mithilfe von *Markov Chain Monte Carlo* Prozeduren approximiert. Miller & Harrison (2013) haben mit dem *Exact Sampler* einen Algorithmus entwickelt, der die exakte Verteilung abzählen und daraus ziehen kann. Die Genauigkeit der beiden Sampler wird verglichen, indem potentielle Einflüsse der Stichprobengröße, DIF-Parameter und Itemschwierigkeit auf die Genauigkeit der Power-Berechnung untersucht werden. Darüber hinaus werden die Burn-In Phase und der Step-Parameter als Einflussfaktoren auf den *Rasch Sampler* überprüft. Die Genauigkeit der Sampler unterscheidet sich nicht wesentlich. Bei steigender Stichprobengröße steigt die Power an. Auch bei größeren Modellabweichung im positiven wie im negativen kann eine höhere Power beobachtet werden. Bei moderater Itemschwierigkeit ist die Power bei positivem und negativem DIF-Parameter nahezu gleich groß. Bei Modellabweichung eines leichten Items ist die Power bei positiver Abweichung größer als bei negativer. Mit einem schwierigen Item ist mit dem Unterschied, dass die Streeung deutlich höher ausfällt, ein gegensätzlicher Trend zu beobachten. Weder die Burn-In Phase noch der Step-Parameter hat einen Einfluss auf die Genauigkeit des *Rasch Samplers*. Aufgrund von effizienterer Berechnung sollte in jedem Fall der *Rasch Sampler* verwendet werden. Die Ergebnisse bezüglich des Verhaltens der Power unter Variation verschiedener Parameter entsprechen den Beobachtungen von Draxler & Zessin (2015).

Schlüsselwörter: Rasch Modell, Power, Pseudo-exakte Tests, Konditionale Tests, Rasch sampler, Exact sampler

Abstract

Draxler & Zessin (2015) have proposed a class of pseudo-exact or conditional tests for power calculation of assumptions of the Rasch model. Sampling algorithms are required to simulate the data required for power calculation. Verhelst (2008) has designed a relatively fast algorithm called the *Rasch Sampler*, which approximates the true distribution using *Markov Chain Monte Carlo* procedures. Miller & Harrison (2013) have developed an algorithm called the *Exact Sampler*, which can count the exact distribution and draw from it. The accuracy of the two samplers is compared by examining potential influences of sample size, DIF-parameters and item difficulty on the accuracy of the power calculation. Furthermore, the burn-in phase and the step parameter are checked as influencing factors on the *Rasch Sampler*. The accuracy of the samplers does not differ meaningfully. The power increases with higher sample size. Also the power increases with larger positive and negative model deviations. With moderate item difficulty, the power for positive and negative DIF parameters is almost equal. If an easy item deviates from the model, the power is greater if the deviation is positive than if the item is negative. With a difficult item, a contrasting trend can be observed with the difference that the range of the power values is relevantly higher. Neither the burn-in phase nor the step parameter has any influence on the accuracy of the *Rasch Sampler*. Due to more efficient calculation the *Rasch Sampler* should be used in any case. The results concerning the behaviour of the power under variation of different parameters correspond to the observations of Draxler & Zessin (2015).

Keywords: Rasch model, Power, Pseudo-exact tests, Conditional tests, Rasch Sampler, Exact Sampler

1 Einleitung

Die psychologische Testtheorie ist in der Psychologie eine der Grundlagen für neue Erkenntnisse. Anders als zum Beispiel bei der Körpergröße, sind viele interessierende Dimensionen in der Psychologie nicht direkt messbar. Derartige Merkmale nennt man latent. Jedes Mal, wenn ein latentes Merkmal sichtbar gemacht werden soll, muss man einen psychologischen Test anwenden. Dabei können die Anwendungsbereiche von Intelligenzquotienten über Persönlichkeitsmerkmale bis hin zur Erfassung ganzheitlicher Arbeitsbedingungen alle nur erdenklichen Bereiche der Psychologie abdecken. Aber auch in anderen Fachbereichen finden testtheoretische Ansätze Anwendung. Es gibt grundsätzlich zwei Ansätze: die klassische und die probabilistische Testtheorie. Für diese Arbeit ist ausschließlich letztere von Bedeutung. Die probabilistische Testtheorie geht davon aus, dass die Ausprägung des gemessenen Merkmals sowohl von der Personenfähigkeit als auch von der Itemschwierigkeit abhängt. Dabei gibt es je nach Anwendungsbezug im Rahmen der *Item response theory* (IRT) Modelle, mithilfe man die Wahrscheinlichkeit ausrechnen kann, einen bestimmten Wert für ein bestimmtes Item zu erzielen. Bei einer erneuten Testung der Personen würde die Wahrscheinlichkeit für eine Antwort x gleich bleiben, aber das Ergebnis könnte variieren, da der Zusammenhang nicht deterministisch festgelegt ist. Die probabilistische Testtheorie hat in den letzten Jahren vermehrt an Bedeutung gewonnen, was weniger an der kürzlichen Erschließung des theoretischen Fundaments, sondern viel mehr an dem rasanten Fortschritt der Technik liegt, die es benötigt, um Kalkulationen unter der IRT durchzuführen. Ein bekanntes Anwendungsbeispiel sind die PISA-Studien (Prenzel, Walter & Frey, 2007). Da die Lernpläne zwischen den verschiedenen Ländern nicht identisch sind, können nur IRT Modelle einen adäquaten Vergleich schaffen.

Eine zentrale Frage vor und nach der Durchführung eines Tests ist die Anzahl der richtigen und falschen Ergebnisse. Wie oft erkennt der Test zum Beispiel eine pathologische Diagnose, obwohl der Patient gesund ist und wie oft wird der Pa-

tient gesund eingeschätzt, obwohl dieser eine Erkrankung hat? Während ersterer Frage in der Vergangenheit bereits viel Aufmerksamkeit zuteil wurde, hat man die Wichtigkeit letzterer länger unterschätzt. Dabei kann dieser Fehler je nach Kontext nicht nur genauso relevant sein wie der der erste, sondern noch tiefgreifender. Angenommen, ein Test soll Schizophrenie diagnostizieren. Nach der Diagnose erhält der Patient ein starkes Neuroleptikum mit schwerwiegenden Nebenwirkungen. Der erste Fehler beschreibt die Wahrscheinlichkeit, dass ein Patient, der keine Schizophrenie hat, das Neuroleptikum verschrieben bekommt. Das ist sicherlich nicht erwünscht, aber der zweite Fehler beschreibt in diesem Fall die Wahrscheinlichkeit, dass eine schizophrene Person als gesund diagnostiziert wird und unbehandelt nach Hause geschickt wird. Diese letztere Wahrscheinlichkeit kann kontrolliert werden. Man sollte die Power – die Gegenwahrscheinlichkeit besagten Fehlers – eines statistischen Tests, wie Draxler (2010) vorgeschlagen hat, genau wie den Fehler erster Art a priori festlegen und aufgrund dessen die benötigte Stichprobengröße ausrechnen.

Im Anwendungskontext von Modellen unter der IRT kann die Power mithilfe von Simulationen berechnet werden. Für diese Simulationen gibt es zwei verschiedene Ansätze. Einerseits ein relativ schnelles, approximatives Verfahren und andererseits ein langsameres Verfahren, bei dem die exakte Verteilung bekannt ist. Da der exakte Algorithmus extrem rechenintensiv ist, muss die Frage gestellt werden, in welchen Szenarien es genügt, die zugrunde liegende Verteilung zu approximieren und wann oder ob überhaupt die Notwendigkeit der exakten Berechnung besteht. Auch ist es durch die exakte Funktion zum ersten Mal möglich, die Genauigkeit des approximativen Verfahrens im Allgemeinen zu überprüfen, da die genaue Verteilung bekannt ist. Diesen Fragen sowie dem Abklären potentieller Einflüsse verschiedener Parameter auf die Power-Berechnung geht diese Arbeit auf den Grund.

1.1 Theoretischer Hintergrund

Im Folgenden werden die für diese Arbeit notwendigen theoretischen Fundamente vom binären Rasch Modell über die Berechnung der Power pseudo-exakter oder konditionaler Tests von Annahmen des Modells nach Draxler und Zessin (2015) bis hin zum Vergleich der Funktionsweise verschiedener Sampling Algorithmen erklärt.

1.1.1 Das binäre Rasch Modell

Eines der am weitesten verbreiteten probabilistischen Modelle ist das dichotome beziehungsweise binäre Rasch Modell. Das Rasch Modell gibt die Wahrscheinlichkeit für eine richtige oder falsche Antwort gegeben einer Itemschwierigkeit β und Personenfähigkeit θ an (Rasch, 1960). Die Gleichung ist gegeben als

$$P(X_{vi} = x_{vi}|\theta_v, \beta_i) = \frac{\exp\big[x_{vi}(\theta_v - \beta_i)\big]}{1 + \exp(\theta_v - \beta_i)}, \tag{1}$$

mit θ_v als Personenparameter für v = 1,...,n Personen, β_i als Itemschwierigkeit für i = 1,...,k Items, X_{vi} als Variable für eine Antwort einer Person v auf ein Item i und x_{vi} als konkreter Wert, den die Variable X_{vi} annehmen kann, mit $x_{vi} \in \{0, 1\}$. Die Wahrscheinlichkeitsverteilung aller Antworten von *n* Personen auf *k* Items ist dabei gegeben als

$$\prod_{v=1}^{n} \prod_{i=1}^{k} P(X_{vi} = x_{vi}), \tag{2}$$

wodurch die Multiplikation sämtlicher gegebener Einzelwahrscheinlichkeiten ausgedrückt wird. Das binäre Rasch Modell hat fünf Haupteigenschaften, die als Voraussetzung zur Gültigkeit gegeben sein müssen: eine uniforme Verteilung, eine streng monoton steigende, logistisch verlaufende Itemcharakteristikkurve, Eindimensionalität, und lokale stochastische Unabhängigkeit. Aus diesen Modellannahmen resultiert statistische Suffizienz. Von Suffizienz spricht man, wenn bestimmte Parameter ohne Informationsverlust durch andere Parameter ersetzt werden können. Im Falle des

Rasch Modells bedeutet es, dass die Zeilen- und Spaltenrandsummen die selben Informationen enthalten, wie die Personen- respektive Itemparameter. Daraus resultiert beispielsweise ein gleicher Personenparameter, für jene Personen mit gleichem Summenscore (Anzahl richtig beantworteter Items). Die Matrizen unterliegen also gegeben der Zeilen- und Spaltenrandsummen einer bedingten Gleichverteilung. Die durch das Rasch Modell gegebene suffiziente Statistik ermöglicht Kalkulation ohne alle Parameter des Modells zu kennen. Wenn die wahren Personen- und Itemparameter unbekannt sind, genügen beispielsweise die Randsummen, da letztere die selben Informationen beinhalten. Diese Eigenschaften haben sich Draxler & Zessin (2015) bei pseudo-exakten oder konditionalen Tests zur Powerberechnung zu Nutze gemacht.

1.1.2 Pseudo-exakte oder konditionale Tests

Um die Power nach Draxler & Zessin (2015) auszurechnen, benötigt es neben der Zeilen- und Spaltenrandsummen ebenfalls ein a priori festgelegtes α-Niveau. Dabei ist α die Wahrscheinlichkeit des Fehlers 1. Art. Die Power wird mit $1 - \beta$ bezeichnet. Die bereits angesprochene Wahrscheinlichkeit des Fehlers 2. Art ist β. Die Power drückt somit die Wahrscheinlichkeit aus, die Nullhypothese korrekterweise zu verwerfen. Es handelt sich um konditionale oder pseudo-exakte Tests, weil die exakte bedingte Verteilung der Matrizen – gegeben der Zeilen- und Spaltenrandsummen – aufgrund der Vielzahl an Möglichkeiten und rechen bedingter Limitationen nicht bestimmt werden kann (Draxler & Zessin, 2015). In der Psychologie ist häufig der Vergleich zweier oder mehreren Gruppen gefragt wie zum Beispiel zwischen Männern und Frauen oder Gesunden und Kranken. Die Gruppe sei bezeichnet als $t = 1, ..., u$ mit $u \leq n$. Die Anzahl aller Gruppen t ist somit u. Zur Diskriminierung zwischen der Gruppenzugehörigkeit verschiedener Personen wird der Vektor $a'_v = (a_{v1}, ..., a_{vu})$ mit Person v in u Gruppen verwendet. Wenn Person v zur Gruppe t zugeordnet ist, ergibt sich $a_{vt} = 1$, ansonsten $a_{vt} = 0$. Wenn mehrere Gruppen mit $u \geq 2$ gegeben sind, entsteht eine Power genau dann, wenn mindestens ein Item i innerhalb einer

Gruppe t abweicht. Diese Modellabweichung nennt man DIF-Parameter und wird als δ_{ti} bezeichnet. Damit die Lösungswahrscheinlichkeit unter Berücksichtigung des Gruppenvektors a_{vt} sowie des DIF-Parameters δ_{ti} möglich ist, wird eine Konfiguration von (1) benötigt. Die Gleichung der Lösungswahrscheinlichkeit eines Items ist somit gegeben als

$$P_1(X_{vi} = x_{vi}) = \frac{\exp\left[x_{vi}\left(\theta_v + \beta_i + \sum_{t=1}^{u} a_{vt}\delta_{ti}\right)\right]}{1 + \exp\left(\theta_v + \beta_i + \sum_{t=1}^{u} a_{vt}\delta_{ti}\right)}. \tag{3}$$

Es sei für den Fall keiner abweichenden Verteilung mit $\delta_t = 0$ für $t = 1, ..., u$ erwähnt, dass die Gleichung aus (3) der aus (1) entspricht. Die Lösungswahrscheinlichkeit der Personen für alle Items gegeben der Zeilenrandsummen $r_1, ..., r_n$ und der Spaltenrandsummen $s_1, ..., s_k$, erhält man aufgrund der suffizienten Statistik der Zeilen- und Spaltenrandsummen für θ_v und β_i als

$$P(X'_1 = x'_1, ..., X'_{n-1} = x'_{n-1} | r_1, ..., r_n, s_1, ..., s_k) = \frac{\exp\left(\sum_{v=1}^{n}\sum_{i=1}^{k}\sum_{t=1}^{u} x_{vi}a_{vt}\delta_{ti}\right)}{\sum_{\Omega}\exp\left(\sum_{v=1}^{n}\sum_{i=1}^{k}\sum_{t=1}^{u} x_{vi}a_{vt}\delta_{ti}\right)}, \tag{4}$$

wobei Ω den Stichprobenraum beschreibt. $X'_v = (X_{v1}, ..., X_{vk-1})$ ist der Vektor $k-1$ freier Antworten von Person v, mit $x'_v = (x_{v1}, ..., x_{vk-1})$ als die zugehörigen, beobachteten Ausprägungen dessen. Es existieren $k-1$ Freiheitsgrade, weil zu Beginn ein Item als Normierungsparameter festgelegt werden muss, welches aufgrund von Identifizierbarkeit immer eine Abweichung von $\delta_{it} = 0$ haben muss. Alle weiteren Itemparameter dürfen beliebig abweichen. Der kritische Bereich der Größe α innerhalb von Ω sei C genannt. Die Power für verschiedene DIF-Parameter $\delta_1, ..., \delta_u$ erhält man schlussendlich durch das Summieren von (4) über C mit

$$\beta(\delta_1, ..., \delta_u) = \sum_{C} P(X'_1 = x'_1, ..., X'_{n-1} = x'_{n-1} | r_1, ..., r_n, s_1, ..., s_k). \tag{5}$$

Um im Rahmen dieser Arbeit in verschiedenen Szenarien die Power ausrechnen zu können, werden Daten benötigt. Diese werden entweder aus gegeben Zeilenrandsummen $r_1, ..., r_n$ und Spaltenrandsummen $s_1, ..., s_k$ oder aus den Personen- und Itemparametern simuliert. Bei kleinen Matrizen ($n \leq 150$ und $k \leq 4$) müssen diese manuell ausgewählt, da bei zufälliger Auswahl beträchtliche Differenzen zwischen informativer und totaler Stichprobengröße entstehen können und durch die Auswahl besagte Differenz als Ursache für Schwankungen der Power ausgeschlossen werden kann. Nicht informative Zeilen- und Spaltenrandsummen sind jene, die einen Score von 0 oder k haben. Aufgrund rechenbedingter Limitationen enthält Ω in dieser Arbeit nicht alle möglichen Matrizen, sondern nur eine zufällig gezogene Auswahl. Den Vorgang des Ziehens bezeichnet man als Sampling.

1.1.3 Sampling

Bevor die Funktionalität der beiden in dieser Arbeit verglichenen Sampling Algorithmen verglichen werden kann, muss zuerst erklärt werden, was Sampling eigentlich ist. Sampling ist das zufällige Ziehen aus einer Verteilung. Wenn sich beispielsweise fünf rote und fünf blaue Kugeln in einer Urne befinden und man zufällig eine Kugel zieht, liegt die Wahrscheinlichkeit, eine rote Kugel zu erhalten, bei 50 Prozent. Allerdings kann es auch passieren, dass man in fünf Versuchen mit Zurücklegen jedes mal eine blaue Kugel erhält. Die Wahrscheinlichkeit dafür beträgt zwar lediglich 3.1 Prozent, aber der zentrale Punkt bei der Arbeit mit Wahrscheinlichkeiten ist der Zufall. Man kann nie wissen, ob man bei zehnmaligem Ziehen aus einer Binomialverteilung mit einer Erfolgswahrscheinlichkeit von .5 am Ende tatsächlich fünf rote Kugeln erhält. Die *Large Sample Theory* besagt jedoch, dass sich bei einer hohen Anzahl an Ziehungen die beobachtete Verteilung an die zugrundeliegende Verteilung annähert (Anscombe, 1952). Das heißt, dass man bei 1 000 Versuchen annäherungsweise 500 rote Kugeln erhalten würde. Aber wie geht man vor, wenn zwar die Art der Verteilung, aber nicht die genaue Population bekannt ist? Also in diesem Beispiel zwar zu wissen, dass eine Binomialverteilung zugrunde liegt, aber weder

die genaue Anzahl der Kugeln in der Urne, noch das Verhältnis zwischen roten und blauen Kugeln zu kennen.

1.1.4 Ziehen aus approximativer Verteilung

Über die Jahre wurden verschiedene Algorithmen entwickelt, um dem Problem der unbekannten Verteilung entgegenzuwirken. Da man bei Gültigkeit des Rasch Modells von einer uniformen Verteilung ausgeht, hat folglich jede mögliche Matrix unter gegeben Zeilen- und Spaltenrandsummen dieselbe Auftrittswahrscheinlichkeit von n^{-1}, mit n als Anzahl an Möglichkeiten. Der Grund für die Schwierigkeit des genauen Berechnens der zugrunde liegenden Verteilung bei unter dem Rasch Modell erstellten Matrizen ist die Masse an Möglichkeiten. Wenn man sich das Antwortmuster einer Person bei zehn Items unter einem Summenscore von 6 ansieht, gibt es bereits $\binom{10}{6}$ beziehungsweise 210 verschiedene Möglichkeiten, wie dieser Antwortvektor aussehen kann. Wenn 10 Personen 10 Items beantworten, mit den Spaltenrandsummen von beispielsweise (4, 6, 5, 5, 5, 4, 2, 4, 4, 4) und den Zeilenrandsummen von (1, 3, 8, 7, 5, 9, 4, 2, 2, 2), existieren hingegen bereits 884 321 373 036 verschiedene Möglichkeiten, wie die Matrix konfiguriert sein kann. Es ist leicht vorstellbar, wie viel mehr Möglichkeiten bei praxisrelevanten Szenarien wie zum Beispiel mit 150 Personen und 25 Items existieren. Anstelle einer genauen Berechnung kann man die Verteilung jedoch approximieren. Dafür verwendet man *Markov Chain Monte Carlo* (MCMC) Prozeduren. Es resultiert eine stationäre Verteilung, die unabhängig vom Startpunkt der Markov Kette ist. Vergangene Ziehungen haben also keinen Einfluss auf weitere Schritte der Markov Kette. Man könnte folglich keine Markov Kette zur Simulation der stationären Verteilung verwenden, wenn aus dem Beispiel unter 1.1.3 die Kugeln nach jeder Ziehung nicht wieder in die Urne zurückgelegt, sondern entfernt werden.

Zur Bewertung der Approximation verschiedener Sampler schlägt Verhelst (2008) die Betrachtung aus zwei Perspektiven vor: der *statistischen Effizienz* und der *rechnerischen Effizienz*. Mit der *statistischen Effizienz* ist die Genauigkeit und somit Rich-

tigkeit des Samplers gemeint, die mithilfe der Standardabweichung gemessen werden kann. Die *rechnerische Effizienz* wird als Zeit gemessen, die der Sampler benötigt, um eine bestimmte (gewünschte) Standardabweichung zur erhalten. Eine besondere Art von MCMC ist das so genannte *Sequential Importance Sampling* (SIS). Snijders (1991) hat als erstes einen SIS Algorithmus implementiert. Da dieser allerdings nur 7 x 7 Matrizen und in Abhängigkeit der Randsummen auch ausgewählte größere Szenarien ausrechnen kann, haben Chen & Small (2005) einen neuen SIS basierten Algorithmus entwickelt. Dieser kann jedoch ebenfalls aufgrund von begrenzten Rechenkapazitäten lediglich Szenarien von 100 x 100 Matrizen mit allen Zeilen- und Spaltenrandsummen gleich 2 (Chen, Diaconis, Holmes & Liu, 2005) und 200 x 30 große Matrizen mit konstanten Randsummen abdecken (Chen & Small, 2005). Laut Verhelst (2008) besitzt der Algorithmus von Chen & Small (2005) bei moderaten oder großen realistischeren Szenarien lediglich eine geringe *statistische Effizienz*. Die zwei bedeutendsten MCMC-Algorithmen zur Simulation von Matrizen unter dem Rasch Modell ohne SIS haben Ponocny (2001) und Verhelst, Hatzinger & Mair (2007) entwickelt. Der Algorithmus von Verhelst et al. (2007) basiert im Grunde genommen auf einer ähnlichen Idee wie der von Ponocny (2001), allerdings ist ersterer bedeutend effizienter geschrieben. Durch die effiziente Berechnung können Szenarien mit bis zu 4 096 Zeilen und 128 Spalten berechnet werden (Mair, Hatzinger & Maier, 2016). Resultierend aus der Bewertung der vorgestellten Algorithmen ist zu diesem Zeitpunkt der *Rasch Sampler* von Verhelst hinsichtlich der *statistischen* sowie *rechnerische Effizienz* für das Ziehen eines Rasch Modells aus einer approximierten Verteilung die beste Wahl (Verhelst et al., 2007).

Bei einer MCMC-Prozedur wird eine sogenannte Burn-In Phase benötigt, um die Markov Kette konvergieren zu lassen. Diese Phase bezeichnet also die Anzahl notwendiger Matrizen, die für eine angemessene Approximation der tatsächlichen Verteilung benötigt wird. Anders als beispielsweise bei bayesschen MCMC-Prozeduren muss die Burn-In Phase nicht mit mehreren Tausend gewählt werden, sondern es genügen nach Aussage von Verhelst (2008) wenige Hundert. Dies wird in dieser Ar-

beit überprüft. Der *Rasch Sampler* ist in R implementiert und hat drei Parameter, die bei der Simulation der Matrizen von Bedeutung sind. *n_eff* beschreibt die Anzahl der zu ziehenden Matrizen. *burn_in* legt die zuvor definierte Burn-In Phase fest. *step* beschreibt die Anzahl der Matrizen, die bei der Ziehung nicht berücksichtigt werden. Bei einem voreingestellten Step-Parameter von 16 würde die Funktion also nur jede 16. Matrix, die berechnet wurde, zum Konvergieren der Markov Kette verwenden.

1.1.5 Ziehen aus exakter Verteilung

Miller & Harrison (2013) haben einen Algorithmus entwickelt, um die diskrete Verteilung bei gegebenen Spalten- und Zeilenrandsummen zu berechnen. Man beachte, dass trotz des Wissens über die genaue Verteilung eine zufällige Komponente involviert bleibt, da nicht alle Matrizen aus der Verteilung, sondern nur eine Auswahl in Abhängigkeit der gewünschten Genauigkeit, zur Power-Berechnung herangezogen werden. Im Gegensatz zum *Rasch Sampler* von Verhelst et al. (2007) kann man die Verteilung im *Exact Sampler* für größere Item- und Personenzahlen nur unter stunden- oder gar tagelanger Berechnung bestimmen. Der neue Algorithmus ist dennoch schneller als bisherige Versuche, die Verteilung mithilfe naiver Rekursion exakt zu berechnen, weil Symmetrien durch wiederholt auftretende Spaltensummen ausgenutzt werden (Miller & Harrison, 2013). Bei naiver Rekursion hingegen würde der Speicherplatzverbrauch linear mit der Rekursionstiefe steigen, wodurch eine bedeutend schlechtere Performance im Vergleich zu der neuen Methode von Miller & Harrison (2013) zu erwarten wäre. Miller & Harrison (2013) empfehlen trotzdem lediglich eine maximale Zeilen- und Spaltenanzahl von $m + n \leq 100$. Dies ist aus praktischer Sicht eine nicht unwesentliche Einschränkung, da in der Psychologie in der Regel deutlich größere Szenarien benötigt werden. Diese Limitation ist jedoch ausschließlich rechenbedingt. Es ist grundsätzlich also durchaus möglich, Matrizen mit größeren Dimensionen ausrechnen zu lassen. In dieser Arbeit werden auch größere Szenarien (maximal $m + n = 154$) durchgerechnet. Es sei an dieser Stelle anzumerken, dass diese Limitation bei fortschreitender Technik in einigen Jahren

bereits obsolet sein kann.

Es gibt für diesen Algorithmus einen R Wrapper (Miller & Harrison, 2013). Dieser besteht aus zwei Funktionen: eine zum Zählen aller möglichen Matrizen und eine zum tatsächlichen Ziehen einer gewünschten Anzahl von Matrizen aus der bekannten Gesamtverteilung in Abhängigkeit der Spalten- und Zeilenrandsummen. Es sei darauf verwiesen, dass der R Wrapper auf zwei Executables (.exe) zurückgreift, sodass die Berechnung mit dem *Exact Sampler* zum Zeitpunkt dieser Arbeit auf UNIX(-like) Systemen lediglich mit einer in einer Virtual Machine installierten Windows Umgebung möglich ist.

1.2 Fragestellung

Die Frage ist nun, ob der *Exact Sampler* die Power genauer als der approximative *Rasch Sampler* berechnen kann, der deutlich weniger Zeit zum Ziehen der Matrizen benötigt. Außerdem wird die durch die Zufallsziehung resultierende Varianz der Power bei Variation verschiedener Parameter untersucht. Die Fragestellung der Arbeit lässt sich in fünf Teile (A - E) gliedern. In Fragestellung A wird der *Rasch Sampler* in kleinen Szenarien direkt mit dem *Exact Sampler* verglichen. Alle Weiteren Fragestellungen beschäftigen sich ausschließlich mit dem *Rasch Sampler*. Fragestellung B untersucht die Power pseudo-exakter Tests bei steigender Personen- und gleichbleibender Itemanzahl. In Fragestellung C wird der Einfluss des DIF-Parameters überprüft. In der darauf folgenden Fragestellung wird der Einfluss der Itemschwierigkeit untersucht. Die Fragestellung E überprüft Genauigkeitsschwankungen des *Rasch Samplers* bei Veränderung der Burn-In Phase und des Step-Parameters, die einen potentiellen Einfluss auf das Konvergieren der Markov Kette haben. Die Zielsetzungen sind konkrete Empfehlungen für die Anwendung in der Praxis bezüglich der Wahl der Sampling-Prozedur, der zu erwartenden Schwankung der Power sowie der zu berücksichtigen Einflussfaktoren.

2 Methodik

Im Folgenden wird das genaue Studiendesign zur Beantwortung der sechs Fragestellungen und die für die Datenanalyse verwendete Hardware und Software vorgestellt.

2.1 Studiendesign

Für sämtliche Berechnungen wird α mit 5 Prozent gewählt. Die Anzahl der gezogenen Matrizen wird bei Fragestellung A auf 3 000 und bei Fragestellung B bis E auf 8 000 festgesetzt, da diese als genügend groß angenommen werden kann, um die gesamte Verteilung abzubilden (Draxler & Zessin, 2015). Auf Basis der jeweils gezogenen Matrizen wird jeweils ein Power-Wert berechnet. Dieses Verfahren wird in Fragestellung A 1 000 und bei den anderen Fragestellungen 3 000 Mal wiederholt, sodass man pro Szenario 1 000 respektive 3 000 Power-Werte und somit eine Wahrscheinlichkeitsverteilung der Power erhält. Der Datensatz wird in der Mitte halbiert, um zwei Gruppen zu erhalten. Welches Merkmal genau diese Gruppen unterscheidet, ist für die Zielsetzung dieser Arbeit nicht näher von Bedeutung. Für eine

Tabelle 1

Spaltenrandsummen aller Items für verwendete Personenzahlen (Fragestellung A).

Item	Personenanzahl					
	10	30	60	90	120	150
1	8	23	60	73	95	117
2	6	17	28	51	69	89
3	3	12	16	34	41	53
4	2	7	12	15	19	24

bessere Verständlichkeit sei die Gruppe nach Geschlecht unterteilt. Die erste Hälfte der Personen ist mit $a_{vt} = 1$ männlich und die andere mit $a_{vt} = 0$ weiblich. Innerhalb der männlichen Stichprobe wird ein abweichendes Item i ausgewählt. Die Burn-In Phase und der Step-Parameter für die MCMC-Prozedur im *Rasch Sampler* werden,

absehen von Fragestellung E, auf 300 und 16 festgelegt. Für alle Szenarien, außer denen in Fragestellung D, wird Item 13 für die Modellabweichung ausgewählt, da dieses Item sich aufgrund der Spaltenrundsumme von 56 genau in der Mitte befindet und somit eine moderate Schwierigkeit aufweist. Für die Fragestellungen A, B und E

Tabelle 2

Absolute Häufigkeit der Summenscores (Zeilenrandsummen) für verwendete Personenzahlen (Fragestellung A).

Summenscore	Personenanzahl					
	10	30	60	90	120	150
1	3	8	17	26	38	46
2	5	15	30	45	60	75
3	2	7	13	19	22	29

wird ein DIF-Parameter von 0.6 gewählt. Die Personenzahl wird bei Fragestellung A bei vier Items zwischen 10, 30, 60, 90, 120 und 150 variiert. Die gewählten Zeilen- und Spaltenrandsummen können Tabelle 1 respektive Tabelle 2 entnommen werden. Als normierendes Item wird das vierte Item und als differierendes das zweite festgelegt. Für Fragestellung B wird eine Itemanzahl von 25 für 10, 30, 90, 150, 250, 350 und 500 Personen gewählt. Die Personenparameter für Fragestellung B und folgende Fragestellungen werden aus einer Normalverteilung mit einem Mittelwert von 0 und einer Standardabweichung von 2 bei einem Seed von 123 gezogen. Die 25 Itemparameter für Fragestellungen B bis E werden arbiträr als (-3, -2.5, -2, -1.5, -1, -0.875, -0.75, -0.625, -0.5, -0.375, -0.25, -0.125, 0, 0.125, 0.25, 0.375, 0.5, 0.625, 0.75, 0.875, 1, 1.5, 2, 2.5, 3) gewählt. Bei Fragestellung C wird ein Szenario mit 100 Personen und 25 Items verwendet. Dabei wird die Power für 15 DIF-Parameter von -1.75 bis 1.75 mit einem Abstand zwischen den Parametern von 0.25 (ausgenommen 0) ausgerechnet. Das abweichende Item wird hier ebenfalls als Item 13 gewählt. In Fragestellung D werden die Berechnungen aus C wiederholt, mit dem Unterschied die Modellabweichung einmal bei Item 24 mit 19 als Minimum der Spaltenrandsummen und somit einem besonders schwierigem Item festzulegen und einmal bei Item 1, welches mit 93 dem Maximum der Spaltenrandsummen entspricht und somit ein

besonders leichtes Item darstellt. Für Fragestellung E werden die gleichen Parameter wie in C gewählt. Variiert wird die Burn-In Phase zwischen 300, 4 000 und 8 000 sowie die Step-Parameter von 16, 32 und 50.

2.2 Datenanalyse

Sämtliche Berechnung werden mit R 3.4.3 durchgeführt (R Core Team, 2017). Die geschriebenen Funktionen befinden sich im Anhang. Der vollständige Code zur Berechnung, Auswertung sowie Erstellung der Graphen kann auf GitHub unter `https://github.com/j3ypi/Bachelorarbeit` heruntergeladen werden. Der Code ist so geschrieben, dass die Möglichkeit besteht, nach der Installation der notwendigen Packages den Code unter Windows in dieser Form auszuführen. Auf UNIX(-like) Systemen können aus unter 1.1.5 beschriebenen Gründen lediglich die Fragestellungen B bis E nachgerechnet werden. Zur Berechnung und Ziehung des Rasch Modells wird auf das *eRm* Package zurückgegriffen (Mair et al., 2016). Darüber hinaus werden Funktionen aus den Packages *here* (Müller, 2017), *rio* (Chan, Chan, Leeper & Becker, 2018) und dem *tidyverse* (Wickham, 2017) verwendet. Um die verschiedenen Fragestellungen zu beantworten, werden deskriptiv alle Szenarien ausgewertet (siehe A.6). Außerdem werden Box Plots erstellt, die jeweils die Wahrscheinlichkeitsverteilung der Power-Werte basierend auf 1 000 Replikation bei Fragestellung A und 3 000 Replikation bei den anderen Fragestellungen erstellt.

3 Ergebnisse

In Abbildung 1 lässt sich kein erkennbarer Unterschied zwischen der Wahrscheinlichkeitsverteilungen der Power Werte zwischen dem *Rasch Sampler* und dem *Exact Sampler* erkennen. Auffällig ist die bedeutend höhere Standardabweichung bei 150 Personen und 4 Items. Diese ist mit $SD = 0.063$ beim Rasch Sampler und $SD = 0.064$ beim Exact Sampler mit der Ausnahme von 60 Personen, da dort einige Ausreißer

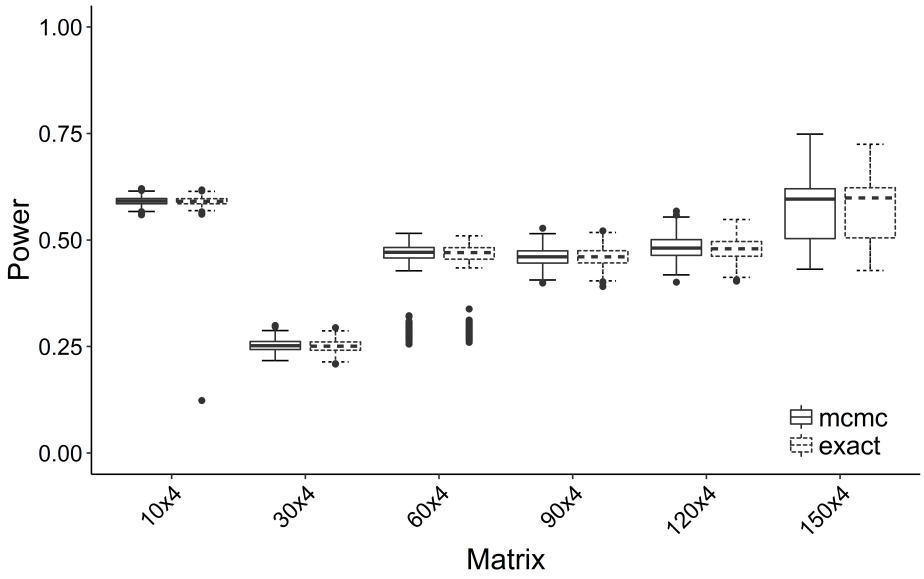

Abbildung 1. Box Plots der Wahrscheinlichkeitsverteilung der Power mit Vergleich zwischen dem *Rasch Sampler* (mcmc) und dem *Exact Sampler* (exact)

zu beobachten sind, bedeutend höher als bei anderen Personenkonstellationen die in einem Range von 0.009 und 0.026 liegen. Des Weiteren ist die Power bei beiden Samplern bei der 30 x 4 Matrix mit M = .25 und SD = 0.014 respektive M = .25 und SD = 0.013 beim Exact Sampler deutlich niedriger als die der anderen Matrizen. An diesen Beispielen kann man exemplarisch auch erkennen, wie gering die Unterschiede zwischen den Samplern sind. Die Matrix mit zehn Personen weist mit M = .59 die größte mittlere Power auf. Auffällig ist außerdem die fehlende Kontinuität beider Sampler. Bei steigender Personenzahl ist kein linearer Zusammenhang in Form von steigender Power beobachtbar. In Abbildung 2 wird der in Fragestellung B untersuchte Zusammenhang zwischen der Power und der Stichprobengröße illustriert. Dabei kann man mit der Ausnahme von 150 x 25 einen stetigen Trend steigender Power bei ansteigender Stichprobengröße beobachten. Die Ausnahme des Szenarios mit 150 Personen hat lediglich eine um 0.01 geringere mittlere Power im Vergleich zur Matrix mit 90 Personen. Wie man Tabelle 3 entnehmen kann, ist die Standardabweichung der Wahrscheinlichkeitsverteilung der Power-Werte auch bei bedeutend größeren Personen- und Itemanzahlen gering. Die Anzahl der Ausreißer steigt bei

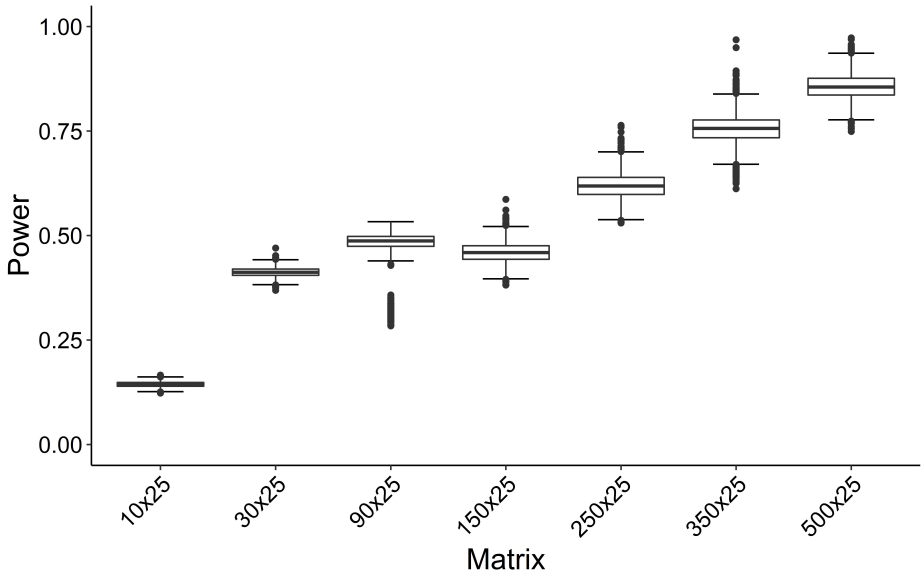

Abbildung 2. Box Plots der Wahrscheinlichkeitsverteilung der Power verschiedener Stichprobengrößen bei gleichbleibender Itemanzahl

steigender Personenanzahl, was man an den geringfügigen Differenzen zwischen Minimum, dem 2.5% Quantil und dem 25% Quantil erkennen kann. In Abbildung

Tabelle 3

Deskriptive Statistik zu Abbildung 2 mit Minimum (Min), 2.5% Quantil (Q.025), 25% Quantile (Q.25), Median, Mittelwert (Mean), 75% Quantil (Q.75), 97.5% Quantil (Q.975), Maximum (Max) und Standardabweichung (SD).

Matrix	Min	Q.025	Q.25	Median	Mean	Q.75	Q.975	Max	SD
10x25	0.12	0.13	0.14	0.14	0.14	0.15	0.16	0.17	0.007
30x25	0.37	0.39	0.40	0.41	0.41	0.42	0.43	0.47	0.011
90x25	0.28	0.31	0.47	0.49	0.47	0.50	0.51	0.53	0.051
150x25	0.38	0.42	0.44	0.46	0.46	0.48	0.51	0.59	0.024
250x25	0.53	0.56	0.60	0.62	0.62	0.64	0.68	0.76	0.031
350x25	0.61	0.68	0.73	0.76	0.76	0.78	0.82	0.97	0.035
500x25	0.75	0.80	0.84	0.86	0.86	0.88	0.92	0.97	0.031

3 erkennt man eine gleichmäßige Verteilung der Power zwischen jeweils gleichen negativen und positiven DIF-Paramtern. Durch die Wahl eines moderat schwierigen Items ist die Power also, unabhängig ob die Modellabweichung positiv oder negativ ist, gleich groß. Erkennbar ist darüber hinaus der große Einfluss des DIF-Parameters

auf die Höhe der Power. Ab einer Abweichung von -1.5 respektive 1.5 nähert sich die Power bereits 1. In Abbildung 2 wurde ein ähnlicher Einfluss durch die Stichprobengröße beobachtet. Anders als in Abbildung 3 illustriert, ist die Verteilung der Boxplots bei der Auswahl des leichtesten Items wie in Abbildung 4 ersichtlich ins negative nach links verschoben. Bei positiver Modellabweichung ist die Power demnach höher. Je negativer der DIF-Parameter gewählt wird, desto größer wird auch die Standardabweichung, sodass diese bei einer Modellabweichung von -0.75 mit *SD* = 0.11 eine mehr als sechs mal höhere Standardabweichung als das positive Äquivalent von 0.75 mit *SD* = 0.018 vorzeigt. Auffällig ist zudem die erhöhte Anzahl an Ausreißern bei negativer Modellabweichung.

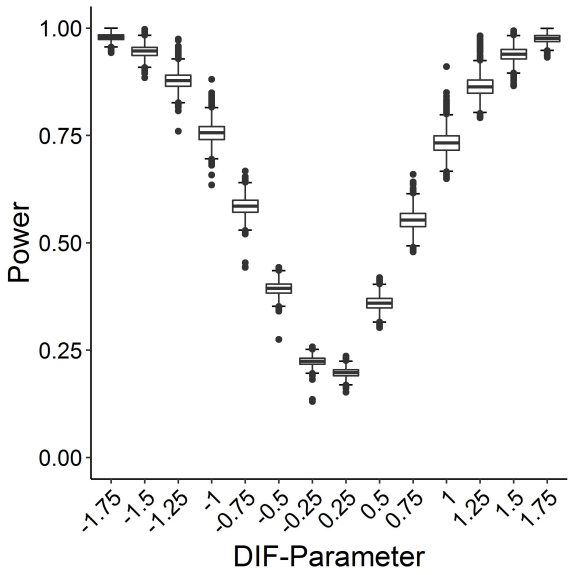

Abbildung 3. Box Plots der Wahrscheinlichkeitsverteilung der Power unter Auswahl verschiedener Modellabweichungen bei Abweichung des Items 13 mit moderater Schwierigkeit

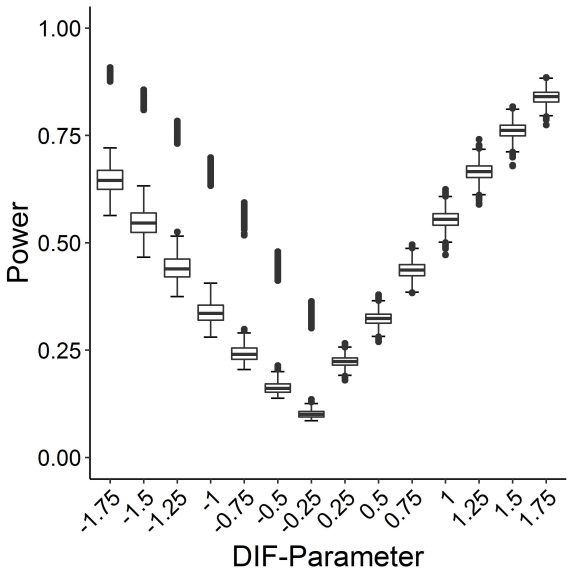

Abbildung 4. Box Plots der Wahrscheinlichkeitsverteilung der Power unter Auswahl verschiedener Modellabweichungen bei Abweichung des leichtesten Items 93 (Maximum der Spaltenrandsumme)

Diese hat allerdings kaum Einfluss auf die Wahrscheinlichkeitsverteilung der Power, da die Differenz zwischen Mittelwert und Median bei negativer Modellabweichung nicht größer als 0.039 wird. In Abbildung 5 sieht man bei Auswahl des schwierigsten Items als Modellabweichung erneut ein große Unterschiede der Verteilungen zu der

gleichverteilten Referenz in Abbildung 3. Die Streuung bei negativen DIF-Parameter ist sichtbar höher als zuvor. Die Mediane unterscheiden sich hingegen kaum vom positiven Äquivalent. Auffällig sind bei positiven DIF-Parametern die erhöhte Zahl an Ausreißern, die sich auch in einer im Vergleich zu Abbildung 4 in einer erhöhten Standardabweichung wiederspiegelt. Während in Abbildung 4 bei einer Modellabweichung von 1 mit *SD* = 0.02 eine niedrige Standardabweichung zu beobachten ist, beträgt diese in Abbildung 5 0.058.

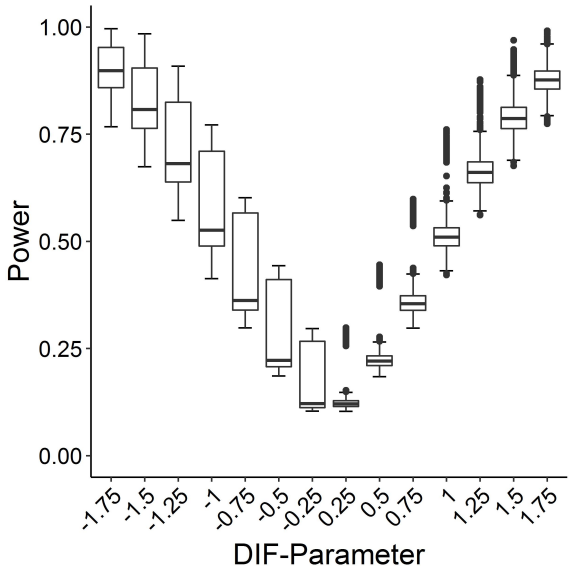

Abbildung 5. Boxplots der Wahrscheinlichkeitsverteilung der Power unter Auswahl verschiedener Modellabweichungen bei Abweichung des schwierigsten Items 24 (Minimum der Spaltenrandsumme)

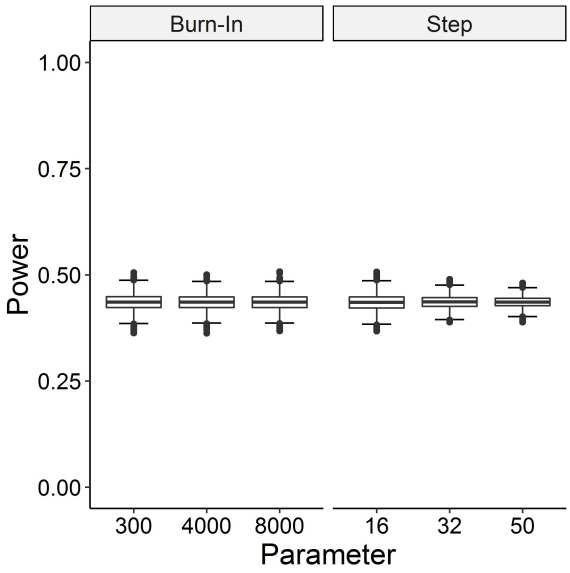

Abbildung 6. Boxplots der Wahrscheinlichkeitsverteilung der Power unter Variation der Burn-In Phase und des Step-Parameters des *Rasch Samplers*

In Abbildung 6 erkennt man, dass die im Rahmen von Fragestellung E überprüften potentiellen Einflüsse der Burn-In Phase und des Step-Parameters auf die Genauigkeit der Power-Berechnung des *Rasch Samplers* nicht vorhanden sind. Auch auf die Standardabweichung scheint weder die Variation der Burn-In Phase noch das Verändern des Step-Parameters einen Einfluss zu haben.

4 Diskussion

Draxler & Zessin (2015) schlagen eine pseudo-exakte beziehungsweise konditionale Methode zur Power-Berechnung von Annahmen des Rasch Modells vor. Aufgrund der suffizienten Statistik der Zeilen- und Spaltenrandsummen für die Personen- und Itemparameter, wird immer nur eine der beiden Informationen gebraucht. Um das Verhalten des pseudo-exakten Testens zu überprüfen, werden zufällig generierte Daten benötigt. Diese können auf zwei Arten simuliert werden. Verhelst et al. (2007) haben mit dem *Rasch Sampler* ein approximatives Sampling Verfahren entwickelt, welches mit *Markov Chain Monte Carlo* Methoden zufällig Matrizen unter gegebenen Item- und Personenparametern generiert. Die genaue Verteilung ist hierbei nicht bekannt. Miller & Harrison (2013) haben mit dem *Exact Sampler* einen Algorithmus vorgestellt, der die genaue Verteilung bei gegebenen Zeilen- und Spaltenrandsummen abzählen und daraus ziehen kann. In dieser Arbeit wurden sowohl der *Rasch Sampler* sowie der *Exact Sampler* bezüglich der Genauigkeit verglichen, als auch das Verhalten der Klasse pseudo-exakter Tests oder konditionaler Tests unter Variation verschiedene Parameter überprüft.

Wie die Ergebnisse im Rahmen von Fragestellung A suggerieren, gibt es keinen nennbaren Unterschied zwischen den beiden Samplern. Nicht erklärbar sind die Schwankungen in der Höhe der Power zwischen den verschieden großen Matrizen. Es wäre mit einem kontinuierlichen Anstieg der Power bei steigender Personenzahl zu rechnen gewesen. Da die Zeilen- und Spaltenrandsummen manuell so gewählt wurde, dass die informative Stichprobengröße der gesamten Stichprobengröße entspricht, kann auch dies als mögliche Ursache ausgeschlossen werden. Auch wenn durch Miller & Harrison (2013) nun die Möglichkeit der exakten Berechnung der zugrunde liegenden Verteilung für kleinere Matrizen gegeben ist, wird dies in der Praxis keine Verwendung finden, weil die vom *Rasch Sampler* approximierte Verteilung nah genug an der wahren Verteilung liegt. Die Berechnungsdauer des *Exact Samplers* hat bei der Matrix von 150 x 4 beispielsweise über 20 Stunden in Anspruch ge-

nommen. Durch den Algorithmus von Verhelst et al. (2007) kann man also in der Praxis eine nicht unerhebliche Anzahl an Zeit sparen. Das bereits in Fragestellung A erwartete Verhalten bezüglich der Höhe der Power, wird erst durch Fragestellung B gezeigt. Abgesehen von einer Ausnahme, die eine leichte Abweichung vom stetigen Anstieg der Wahrscheinlichkeitsverteilungen der Power darstellt, ist hier ein klarer Trend zu beobachten. In diesem Fall wurden die Personenparameter simuliert, wodurch Unterschiede in der informativen Stichprobengröße als mögliche Ursache für die Abweichung in Frage kommen könnte. In Fragestellung C wurde mit dem DIF-Parameter neben der Stichprobengröße ein weitere Einflussfaktor auf die Power geprüft. Hierbei zeigen sich die gleichen Ergebnisse wie von Draxler & Zessin (2015) beschrieben. Interessant ist hier, dass die Ergebnisse von Draxler & Zessin (2015) auf deutlich weniger Replikation und Anzahl der Matrizen basieren und dennoch Ergebnisse herauskommen, die sich nicht stark voneinander unterscheiden. Während in dieser Arbeit die Power-Berechnung 3 000 repliziert wurde, beruht die Wahrscheinlichkeitsverteilung in dem Paper lediglich auf 100 Replikationen (Draxler & Zessin, 2015). Dies legt Nahe, dass die Anzahl der Replikationen in der Praxis im Vergleich zu dieser Arbeit ohne nennenswerten Informationsverlust deutlich reduziert werden kann, was wiederum zu einer weiteren Reduktion der Berechnungsdauer führt. Im Rahmen von Fragestellung D wurde der Einfluss der Itemschwierigkeit auf die Berechnung pseudo-exakter Tests geprüft. Bei Auswahl des leichtesten Items ist in dieser Arbeit das Gegenteil der Ergebnisse von Draxler & Zessin (2015) zu beobachten, da hier die Itemschwierigkeit und nicht Einfachheit als Basis der Interpretation der Itemparameter festgelegt wird. Die verschobene Verteilung kann durch die Abhängigkeit der Power von den gegebenen Zeilen- und Spaltenrandsummen erklärt werden. Der Einfluss einer Modellabweichung wird größer, je weiter das differierende Item von der Mitte des möglichen Parameterraumes der Spaltenrandsummen entfernt liegt. Bei Auswahl des schwierigsten Items als Modellabweichung zeichnet sich jedoch ein etwas anderes Bild ab als von Draxler & Zessin (2015) erwartet. Es lässt sich zwar eine höhere Power bei negativen Modellabweichungen

beobachten, allerdings ist diese nicht so klar ausgeprägt wie bei Modellabweichung eines Items mit relativ hoher Spaltenrandsumme. Diese Schwankungen der Power können möglicherweise auf den geringeren Informationsgehalt zurückgeführt werden, welcher aus der Nähe zur Grenze des Parameterraumes resultiert. Es sei an dieser Stelle darauf hingewiesen, dass die Differenzen zwischen den Ergebnissen dieser Arbeit bezüglich dieser Fragestellung und denen beschrieben in Draxler & Zessin (2015) auch deshalb resultieren können, weil hier das Item mit der niedrigsten Spaltenrandsumme eine Randsumme von 19 und nicht 4 aufweist. Somit lösen in dieser Arbeit selbst das schwierigste Item noch 19 Prozent, wohingegen im beschriebenen Paper lediglich 4 Prozent der Personen das schwierigste Item lösen. In Fragestellung E wurde der Eindruck von Verhelst (2008) bestätigt, dass weder die Burn-In Phase noch der Step-Parameter erhöht werden müssen, da keine Steigerung der Genauigkeit zu erwarten wäre. Während andere MCMC-Prozeduren mehrere tausend Durchläufe benötigen, um die Verteilung zu approximieren, kommt der *Rasch Sampler* demnach mit 300, wahrscheinlich sogar mit weniger aus.

Generell können pseudo-exakte oder konditionale Tests als Tests mit viel Power beschrieben werden. Schon ab einer Größe von 500 x 25 geht die Power gegen 1. Daraus resultiert ein Anwendungskontext dieser Klasse von Tests vor allem bei kleineren Stichproben. Bei größeren Stichproben kann dann wie bisher beispielsweise auf asymptotische χ^2-Tests zurückgegriffen werden, die weniger Power haben. Offen bleibt, weshalb in Fragestellung A kein kontinuierlicher Anstieg beobachtet werden konnte. Es wäre daher von Interesse in zukünftigen Arbeiten diesen durch verschiedene Randsummen oder Item- beziehungsweise Personenparametern induzierten Schwankungen genauer zu betrachten. Auch die Anwendung pseudo-exakter Tests auf reale Datensätze würde die Ergebnisse dieser Arbeit ergänzen. Eine weitere wichtige Fragestellung ist das Verhalten der Power unter gleichzeitiger Abweichung von mehr als einem Item, wie von Draxler & Zessin (2015) bereits in einigen Beispielen illustriert. Abschließend kann festgehalten werden, dass für das Ziehen einer arbiträren Anzahl an Matrizen der *Rasch Sampler* nach wie vor das beste Mittel ist.

Die Klasse pseudo-exakter oder konditionaler Tests sollte vor allem bei kleineren Stichproben angewandt werden. Als Haupteinflussfaktoren auf die Höhe der Power fungieren sowohl die Stichprobengröße, die Modellabweichung als auch die Schwierigkeit der Items. Parameter des *Rasch Sampler* wie die Burn-In Phase und der Step-Parameter müssen hingegen nicht weiter verändert werden.

5 Literaturverzeichnis

Anscombe, F. J. (1952). Large-sample theory of sequential estimation. *Mathematical Proceedings of the Cambridge Philosophical Society*, *48* (4), 600–607.

Chan, C.-H., Chan, G. C., Leeper, T. J. & Becker, J. (2018). rio: A swiss-army knife for data file i/o [Software-Handbuch]. (R package version 0.5.9)

Chen, Y., Diaconis, P., Holmes, S. P. & Liu, J. S. (2005). Sequential monte carlo methods for statistical analysis of tables. *Journal of the American Statistical Association*, *100* (469), 109–120.

Chen, Y. & Small, D. (2005). Exact tests for the rasch model via sequential importance sampling. *Psychometrika*, *70* (1), 11–30.

Draxler, C. (2010). Sample size determination for rasch model tests. *Psychometrika*, *75* (4), 708–724.

Draxler, C. & Zessin, J. (2015). The power function of conditional tests of the rasch model. *AStA Advances in Statistical Analysis*, *99* (3), 367–378.

Mair, P., Hatzinger, R. & Maier, M. J. (2016). eRm: Extended Rasch Modeling [Software-Handbuch]. Zugriff auf `http://erm.r-forge.r-project.org/` (0.15-7)

Miller, J. W. & Harrison, M. T. (2013). Exact sampling and counting for fixed-margin matrices. *The Annals of Statistics*, *41* (3), 1569–1592.

Müller, K. (2017). here: A simpler way to find your files [Software-Handbuch]. Zugriff auf `https://CRAN.R-project.org/package=here` (R package version 0.1)

Ponocny, I. (2001). Nonparametric goodness-of-fit tests for the rasch model. *Psychometrika*, *66* (3), 437–459.

Prenzel, M., Walter, O. & Frey, A. (2007). Pisa misst kompetenzen. *Psychologische rundschau*, *58* (2), 128–136.

R Core Team. (2017). R: A language and environment for statistical computing [Software-Handbuch]. Vienna, Austria. Zugriff auf `https://www.R-project.org/`

Rasch, G. (1960). *Probabilistic models for some intelligence and attainment tests.* Kopenhagen: Danish Institute for Educational Research.

Snijders, T. A. (1991). Enumeration and simulation methods for 0–1 matrices with given marginals. *Psychometrika*, *56* (3), 397–417.

Verhelst, N. D. (2008). An efficient mcmc algorithm to sample binary matrices with fixed marginals. *Psychometrika*, *73* (4), 705–728.

Verhelst, N. D., Hatzinger, R. & Mair, P. (2007). The rasch sampler. *Journal of Statistical Software*, *20* (4), 1–14.

Wickham, H. (2017). tidyverse: Easily install and load 'tidyverse' packages [Software-Handbuch]. Zugriff auf `https://CRAN.R-project.org/package=tidyverse` (R package version 1.1.1)

A R Funktionen

A.1 Powerfunktion für Fragestellung A

```r
pwr_A <- function(rows_sums, cols_sums,
                  n_repeats = 1000, n_matrices = 3000,
                  alpha = .05, dev = .6,
                  item_pos = 2, burnIn = 300,
                  step = 16, folder = ""){
  ##################################################################
  # INPUTS:
  # rows_sums: Zeilenrandsummen
  # cols_sums: Spaltenrandsummen
  # n_repeats: Anzahl an Power-Werte pro Szenario
  # n_matrices: Anzahl an Matrizen pro Power-Wert
  # alpha: Fehler 1. Art
  # dev: DIF-Parameter
  # item_pos: Items mit Modellabweichung
  # burnIn: Burn-In Phase fuer Rasch Sampler
  # step: Step-Parameter fuer Rasch Sampler
  # folder: Speicherort
  ##################################################################
  model <- sample(rows_sums, cols_sums, 1)
  half_length <- length(cols_sums) / 2
  groups <- c(rep(1, half_length), rep(0, half_length))
  dif <- rep(0, length(cols_sums))
  dif[item_pos] <- dev

  path <- paste0(folder, "/",
                 as.character(length(rows_sums)), "x",
```

```
                as.character(length(cols_sums)), ".csv")

  mcmc <- exact <- vector("numeric", n_repeats)

  count(rows_sums, cols_sums)

  mcmc <- replicate(n_repeats,
                    pwr_mcmc(mat = model,
                             group = groups,
                             dif = dif,
                             repetitions = n_matrices,
                             alpha = alpha,
                             burn = burnIn,
                             steps = step))
  exact <- replicate(n_repeats,
                     pwr_exact(rows = rows_sums,
                               cols = cols_sums,
                               group = groups,
                               dif = dif,
                               repetitions = n_matrices,
                               alpha = alpha))

  rio::export(data.frame(power = c(mcmc, exact),
                         method = rep(c("mcmc", "exact"),
                                      each = n_repeats)), path)

}
```

A.2 Powerfunktion für Fragestellung B, C, D und E

```r
pwr_BCDE <- function(itempars, n_repeats = 3000,
                     n_matrices = 8000, alpha = .05,
                     n_pers = 100, sd_pers = 2,
                     dev = .6, burnIn = 300, difficulty = "
                        moderat",
                     step = 16, folder = ""){
  ##################################################################
  # INPUTS:
  # itempars: Itemparameter
  # n_repeats: Anzahl an Power-Werte pro Szenario
  # n_matrices: Anzahl an Matrizen pro Power-Wert
  # alpha: Fehler 1. Art
  # n_pers: Anzahl Personen
  # n_items: Anzahl Items
  # sd_pers: Standardabweichung der Personenparameter
  # dev: DIF-Parameter
  # burnIn: Burn-In Phase fuer Rasch Sampler
  # difficulty: Itemschwierigkeit ("leicht", "moderat", "schwer")
  # step: Step-Parameter fuer Rasch Sampler
  # folder: Speicherort
  ##################################################################
  set.seed(123)
  personenpars <- rnorm(n = n_pers, mean = 0, sd = sd_pers)
  half_length <- length(personenpars) / 2
  groups <- c(rep(1, half_length), rep(0, half_length))
  model <- sim.rasch(persons = personenpars,
                     items = itempars,
                     seed = 123)
  cols_sums <- colSums(model)
  rows_sums <- rowSums(model)
```

```
30    dif <- vector("numeric", length(cols_sums))
31    mcmc <- vector("numeric", n_repeats)
32    path <- paste0(folder, "/",
33                   as.character(length(rows_sums)), "x",
34                   as.character(length(cols_sums)), "_",
35                   as.character(dev), ".csv")
36    switch(
37      difficulty,
38      "leicht" = dif[which(cols_sums == max(cols_sums[-length(
          itempars)]))[1]] <- dev,
39      "moderat" = dif[which(cols_sums == getMiddle(cols_sums[-
          length(itempars)]))[1]] <- dev,
40      "schwer" =  dif[which(cols_sums == min(cols_sums[-length(
          itempars)]))[1]] <- dev
41    )
42    mcmc <- replicate(n_repeats,
43                      pwr_mcmc(mat = model,
44                               group = groups,
45                               dif = dif,
46                               repetitions = n_matrices,
47                               alpha = alpha,
48                               burn = burnIn,
49                               steps = step))
50    rio::export(data.frame(power = mcmc,
51                           method = rep(c("mcmc"), n_repeats)),
                               path)
52  }
```

A.3 Unterfunktion zur Power-Berechnung mit dem *Exact Sampler*

```
pwr_exact <- function(rows, cols, group,
                      dif, repetitions, alpha) {
  ################################################
  # INPUTS:
  # rows: Zeilenrandsummen
  # cols: Spaltenrandsummen
  # group: Gruppeneinteilung
  # dif: DIF-Parameter
  # repetitions: Anzahl an Matrizen pro Power-Wert
  # alpha: Fehler 1. Art
  ################################################
  s <- sample(a = rows, b = cols, k = repetitions)
  t <- colSums(s * group)
  e <- exp(colSums(t * dif))
  pwr <- sum(e[e >= quantile(e, 1 - alpha)]) / sum(e)

  return(pwr)
}
```

A.4 Unterfunktion zur Power-Berechnung mit dem *Rasch Sampler*

```
pwr_mcmc <- function(mat, group, dif, repetitions,
                     burn, steps, alpha) {
  ################################################
  # INPUTS:
  # mat: Rasch Modell
```

```
  # group: Gruppenaufteilung
  # dif: DIF-Parameter
  # repetitions: Anzahl an Matrizen pro Power-Wert
  # burn: Burn-In Phase
  # steps: Step-Parameter
  # alpha: Fehler 1. Art
  ##################################################
  s <- rsampler(mat, controls = rsctrl(n_eff = (repetitions - 1),
                                       burn_in = burn,
                                       step = steps))
  t <- rstats(s, function(x) colSums(x * group))
  e <- exp(colSums(matrix(unlist(t), ncol = s$n_tot) * dif))
  pwr <- sum(e[e >= quantile(e, 1 - alpha)]) / sum(e)

  return(pwr)
}
```

A.5 Summary im *tidy* Format

```
auswertung_summary <- function(df, col1, col2){
  ############################################################
  # Auswertung mit Summary im tidy Format
  # INPUTS:
  # df: Datensatz als data.frame
  # col1: Gruppierende Spalte als quosure
  # col2: Auszuwertende Spalte als quosure
  # z.B.: auswertung_summary(daten, quo(method), quo(power))
  ############################################################
  df %>%
    group_by(!!col1) %>%
```

```
    summarise(min = min(!!col2, na.rm = T),
              Q.025 = quantile(!!col2, .025, na.rm = T),
              Q.25 = quantile(!!col2, .25, na.rm = T),
              median = median(!!col2, na.rm = T),
              mean = mean(!!col2, na.rm = T),
              Q.75 = quantile(!!col2, .75, na.rm = T),
              Q.975 = quantile(!!col2, .975, na.rm = T),
              max = max(!!col2, na.rm = T),
              sd = sd(!!col2, na.rm = T))
}
```

A.6 Allgemeine Auswertungsfunktion

```
auswertung <- function(df, col1, col2){
  ########################################
  # Allgemeine Auswertungsfunktion
  # INPUTS:
  # df: Datensaetze als Liste
  # col1: Gruppierende Spalte
  # col2: Auszuwertende Spalte
  # z.B.: auswertung(daten, method, power)
  ########################################
  col1 <- enquo(col1)
  col2 <- enquo(col2)

  # Allgemeine Auswertung mit Summary im tidy Format
  a <- df %>%
    map_df(~ auswertung_summary(df = .x, col1 = col1, col2 = col2
       )) %>%
    mutate(szenario = rep(names(df), each = 2))
```

```
17   # Vergleich der Standardabweichungen
18   b <- a %>%
19     select(!!col1, sd, szenario) %>%
20     spread(!!col1, sd) %>%
21     mutate(mcmc_smaller = mcmc < exact)
22
23   return(list(Allgemeine_Auswertung = a,
24               Vergleich_Standardabweichungen = b))
25 }
```

A.7 Mittlere Zahl zurückgeben

```
1  getMiddle <- function(x){
2    ############################
3    # INPUTS:
4    # x: Numerischer Vektor
5    ############################
6    # Mittlere Zahl zurueckgeben
7    # bei ungeraden Itemanzahlen rundet R ab
8    # (z.B. bei 10.5 nimmt es den 10. Index)
9    sorted <- sort(x)
10   middle <- sorted[length(x) / 2]
11   return(middle)
12 }
```