R-Funktionen für das ordinale Rasch-Modell

"oRm"

Heinrich Potuschak

Almo Statistik-System
www.almo-statistik.de
heinrich.potuschak@linzag.net

2014

Weitere Almo-Dokumente

Die folgenden Dokumente können alle von der Handbuchseite in http://www.almo-statistik.de/ heruntergeladen werden. Alle Dokumente befinden sich im pdf-Format.

0. Arbeiten_mit_Almo.PDF (1 MB)
1a. Eindimensionale Tabellierung.PDF (1.8 MB)
1b. Zwei- und drei-dimensionale Tabellierung.PDF (1.1 MB)
2. Beliebig-dimensionale Tabellierung.PDF (1.7 MB)
3. Nicht-parametrische Verfahren.PDF (0.9 MB)
4. Kanonische Analysen.PDF (1.8 MB)
   Diskriminanzanalyse.PDF (1.8 MB)
   enthält: Kanonische Korrelation, Diskriminanzanalyse, bivariate Korrespondenzanalyse, optimale Skalierung
5. Korrelation.PDF (1.4 MB)
6. Allgemeine multiple Korrespondenzanalyse.PDF (1.5 MB)
7. Allgemeines ordinales Rasch-Modell.PDF (0.6 MB)
7a. Wie man mit Almo ein Rasch-Modell rechnet.PDF (0.2 MB)
7b. R-Funktionen für das ordinale Rasch-Modell
8. Tests auf Mittelwertsdifferenz, t-Test.PDF (1,6 MB)
9. Logitanalyse.pdf (1,2MB) enthält Logit- und Probitanalyse
10. Koeffizienten der Logitanalyse.PDF (0,06 MB)
11. Daten-Fusion.PDF (1,1 MB)
12. Daten-Imputation.PDF (1,3 MB)
13. ALM Allgemeines Lineares Modell.PDF (2.3 MB)
13a. ALM Allgemeines Lineares Modell II.PDF (2.7 MB)
14. Ereignisanalyse: Sterbetafel-Methode, Kaplan-Meier-Schätzer, Cox-Regression.PDF (1,5 MB)
15. Faktorenanalyse.PDF (1,6 MB)
16. Konfirmatorische Faktorenanalyse.PDF (0,3 MB)
17. Clusteranalyse.PDF (3 MB)
18. Pisa 2012 Almo-Daten und Analyse-Programme.PDF (17 KB)
19. Guttman- und Mokken-Skalierung.PFD (0.8 MB)
20. Latent Structure Analysis.PDF (1 MB)
21. Statistische Algorithmen in C (80 KB)
22. Conjoint-Analyse (PDF 0,8 MB)
23. Ausreisser entdecken (PDF 170 KB)
24. Statistische Datenanalyse Teil I, Data Mining I
25. Statistische Datenanalyse Teil II, Data Mining II
26. Statistische Datenanalyse Teil III, Arbeiten mit Almo-Datenanalyse-System
27. Mehrfachantworten, Tabellierung von Fragen mit Mehrfach-antworten (0.8 MB)

Inhaltsverzeichnis

Heinrich Potuschak

oRm: R-Funktionen für das ordinale Rasch-Modell

Alle R-Funktionen sind (in alphabetischer Reihenfolge) im Dokument „oRM.R“ enthalten. Sie können zusammen in den R-Editor kopiert werden. Im nachfolgenden Punkt 7 wird vorgeführt, wie mit ihnen zu rechnen ist. Eine kurze erläuternde Übersicht über die R-Funktionen steht im nachfolgenden Punkt 6.

Die in "oRM.R" enthaltenen R-Funktionenen sind sehr kompakt programmiert. Auch die Ergebnisse aus diesen Funktionen werden in sehr knapper Form ausgegeben. Die Funktionen sollen dem Benutzer als "Gerüst" für sein eigenes Programm dienen.

Die R-Funktionen wurden von Kurt Holm in die Programmiersprache „C“ übertragen, ausgebaut und in das Statistiksystem **Almo** eingefügt. Dadurch ist ein unter Windows laufendes, leicht zu bedienendes Programm entstanden mit übersichtlicher Ergebnis-Ausgabe und Grafik.

Die R-Funktionen und das C-Programm können im Quellcode frei heruntergeladen werden. Wenn Sie bei

www.almo-statistik.de

das Statistik-System **Almo** herunterladen, dann besitzen Sie damit auch die R-Funktionen "oRM.R" und den Quellcode zum C-Programm.

Sie finden "oRM.R" zusammen mit dieser hier vorliegenden Beschreibung im Almo-Unterordner **Rasch_in_R** und
den C-Quellcode im Almo-Unterordner **Algorithmen_in_C** in den Modulen "a_up_algorith11.c" und "a_Main_Konsole_24.c".

Bezeichnungen

m = Zahl der Items

k_i = Kategorien der Items, d.h. Ausprägungen $x = 0,\ldots,k_i-1$, bzw. je k_i-1 Stufen.

mscs = $\sum_{i=1}^{m}(k_i-1)$ = maximale Scoresumme,

amp = $\prod_{i=1}^{m} k_i$ = Anzahl mögl. Pattern.

np_r = Anzahl der Personen mit Scoresumme r, wobei $0 \le r \le$ mscs und Summe(np_r) = N = Anzahl der Personen.

$nt_{i,x}$ = Anzahl der Personen, die Item i bis zur Stufe x gelöst haben, wobei $1 \le i \le m$, $1 \le x \le k_i-1$ und $nt_{i,0} = N$ – Summe($nt_{i,x}$).

τ = Schwellenparameter. Dimension = Dim(nt), d.h. m Zeilen mit je k_i-1 Spalten. Jede Zeile ist aufsteigend geordnet, die Gesamtsumme ist 0. Def: $\tau_{i,0} = S_{i,0} = 0$.

s = Schwierigkeitsparameter, wobei $s_{i,x} = \sum_{y=1}^{x}\tau_{i,y} \Rightarrow \sum_{i=1}^{m} s(i,k_i-1) = 0$ und

$\tau_{i,x} = S_{i,x} - S_{i,x-1}$. Die sb maximieren die bedingte Likelihood cLL.

```
fw_r = Fähigkeitsparameter einer Person mit Scoresumme r, wobei 0 ≤ r ≤ mscs.
     Die fw maximieren die gewichtete Likelihood wLL.
pat = Ein einzelnes m-dimensionales Pattern („Antwortmuster“)
pah = Vektor der Pattern-Häufigkeiten.
n0  = Zahl der besetzten Pattern.
dez = Dezimalcodes der n0 Pattern. Zusammen mit den pah ist es möglich,
      alle Daten anstelle einer N*m- Matrix sehr kurz mit 2 Vektoren der
      Länge n0 zu speichern.
bas = Basis des Zahlensystems zur Umcodierung von pat nach dez und
      umgekehrt
dezN = Dezimalcodes aller N Pattern (aller N Personen).
d    = ein einzelner Dezimalcode
```

1. Daten

Die Begriffe „Pattern“ (Antwortmuster) und „Dezimalcode“ sollen an einem Beispiel mit 5 Items erklärt werden. Ein Pattern (Antwortmuster) ist beispielsweise

1 0 1 0 0

Das 1. Item wurde von der Person mit 1 beantwortet (gelöst), das 2. Item mit 0 (nicht gelöst) etc. Dieses Pattern kann in einer einzigen Zahl, dem „Dezimalcode“ dieses Patterns, ausgedrückt werden. Sind die Items dichotom, dann ist der Dezimalcode dieses Pattern d = 20. Dieser Dezimalcode ist unverwechselbar; d.h. es gibt kein anderes Pattern, das diesen Wert d besitzt.

oRM erwartet die einzulesenden Daten in der Form der Dezimalkodierung. Das klingt zunächst sehr Benutzer-unfreundlich. Eine Einlesefunktion von Originaldaten aus einer gespeicherten Datei ist nicht vorhanden. Da die Datensituationen der Benutzer sehr unterschiedlich sein können, muss sich der Benutzer eine solche selber programmieren.

Bequemer ist folgende, allerdings nicht R-adäquate Vorgehensweise:
Das Almo-Raschprogramm Prog14m4 bildet aus eingelesenen Originaldaten die Parameter `dez` und `pah` und weitere Parameter sowie sämtliche Aufrufe der benötigten R-Funktionen, die man sich dann ausgeben lassen kann. Siehe weiter unten Punkt 7.

Das Daten-Einleseprogramm des Benutzers muss folgende Struktur besitzen

0. erforderliche Eingabe-Parameter

```
alleVar     = Zahl der Variablen des gesamten Datensatzes der Person
m           = Zahl der Items, die dem Rasch-Verfahren unterworfen werden
analyseVarNr= Vektor, der die Stelle der Items im gesamten eingelesenen
              Datensatz angibt.
k           = Vektor mit den Kategorienzahlen der Items
N           = Gesamtzahl der Personen
```

1. Lese den Datensatz der Person
   von der 1. Variablen bis zur letzten Variablen `alleVar`
2. Selektiere daraus die Items, die dem Rasch-Verfahren unterworfen
   werden sollen, d.h. bilde daraus das "pattern" der Person,
   das in den Vektor `pat` der Länge `m` eingetragen wird.
   Erforderlich dafür ist ein Vektor `analyseVarNr`, der die

Stelle der Items im gesamten eingelesenen Datensatz angibt.

3. Prüfe das pattern auf
   a. fehlende Werte und entscheide was dann getan werden soll, z.B. das gesamte pattern überspringen und zu Punkt 1 zurück und die nächste Person einlesen oder fehlenden Wert auf 0 setzen, d.h. wie "nicht gelöst" betrachten oder einen (noch zu bestimmenden) Ersatzwert einsetzen
   b. darauf ob das Item den Ganzzahlwert 0 bis zur Kategorienzahl unter- oder überschreitet. Erforderlich dafür ist ein Vektor mit den Kategorienzahlen der Items `k`
4. Wandle das pattern `pat` in die Dezimalkodierung.
   Durch den einmaligen Aufruf der R-Funktion `Basis` wird ein Vektor `bas` erzeugt, der während des gesamten Einlese-Vorgangs unverändert bleibt. Aus der Multiplikation

```
d=0;
for(i in (m-1):1) d = d+ bas[i]*pat[i];
```

   entsteht dann der Dezimalcode `d` für die eingelesene Person Das pattern der eingelesenen Person wurde so zu einer einzigen Ganzzahl umgeformt. Diese Ganzzahl wird in den Vektor `dezN` gespeichert. Dieser muss auf die Länge `N` (=Gesamtzahl der Personen) deklariert werden.

5. Zurück zu Punkt 1. Nächste Person einlesen

6. Nachdem alle Personen eingelesen und verarbeitet wurden, liegt der Vektor `dezN` mit der Länge `N` vor.
7. Mit der R-Funktion `Daten.N(dezN)` wird dann aus `dezN` gebildet:

```
n0  = Zahl der besetzten pattern
dez = Vektor der Dezimalkodierung der besetzten pattern
pah = Häufigkeit je besetztes pattern
```

8. Mit der R-Funktion `Daten.n0(n0,dez,pah)` werden dann die Vektoren der Randsummen

```
np  = Vektor der Häufigkeiten je Gesamtpunkt
nt  = Anzahl der Personen je Itemkategorie
```

   gebildet.

   Damit sind alle Größen vorhanden, mit denen das R-Programm gerechnet werden kann

2 Simulation „Rasch-konformer“ Daten

Um Daten modellkonform analysieren zu können, muss der "daten-erzeugende" Mechanismus bekannt sein: Die modellgemäße Wahrscheinlichkeit einer Person mit Fähigkeit f_r, Item i bis zur Stufe x zu lösen, lautet $P(i,x|r) = \exp[x*fw_r - sb_{i,x}] / \sum_{y=0}^{k(i)-1} \exp[y * fw_r - sb_{i,y}]$.

Die function SIMord wird mit selbstgewählten Werten SB,FW,NP aufgerufen; beim Bootstrappen sind es die festen Werte. Für die NP_r Personen mit Fähigkeitsparameter FW_r wird die Matrix P für $1 \le i \le m$ und $0 \le x \le k_i-1$ berechnet und zeilenweise kumuliert. Dann wird je Item

eine Zufallszahl zuf ~ U(0,1) erzeugt und die Stufe pat_i bestimmt, für die $P_{i,x-1} < zuf < P_{i,x}$ gilt. Daraus werden die dezN = pat.bas gebildet, und aus ihnen die Randsummen nt und np berechnet, sowie n0, dez und pah.
Ist man an Pattern interessiert, deren Häufigkeiten exakt (bzw. gerundet, wenn gzz = T) den bedingten Erwartungen $E(\underline{x}|r) = N*P(x|r)$ entsprechen, ist die function SIMexakt(v,gzz) aufzurufen. $P(\underline{x}|r)$ ist das Produkt der P(i,x,|r), und ergibt sich als $\exp(-\sum_{i=1}^{m} sb_{i,x[i]}) / \gamma_r$, wenn alle $P(\underline{x}|r)$ mit gleicher Scoresumme durch ihre Summe geteilt und mit np_r/N multipliziert werden. Die symmetrische Grundfunktion γ_r wird in Kap. 4 behandelt. Je Personenhäufigkeit np_r werden $v*np_r$ Pattern erzeugt, von denen der Dezimalcode (dezE) und die Häufigkeit (pahE) berechnet wird. Im Fall gzz=T summieren sich die pahE zu ca. v∗N, wobei Pattern mit pahE = 0 übergangen werden. Im Fall gzz=F & v=1 entstehen alle (d.h. amp) bedingten Patternwahrscheinlichkeiten. Aus den pahE werden die Randsummen ntE und npE berechnet.
Zur Simulation der SB ist die function SIMsb(ti1,d,rad) aufzurufen. Sie erzeugt die Schwellenparameter $\tau_{i,x}$ = ti1[i]+(x-1)∗d+U(-rad,rad), die aufsteigend sortiert sind, wenn rad<d. Nach Normierung werden durch zeilenweises Kumulieren die sb gebildet.

3. Hilfsfunktionen

Die function Elim(eli) eliminiert die Items mit den Indizes eli und berechnet die neuen Werte von m, k, np, nt, n0, dez und pah. Vor dem Aufruf sind m0, k0 und b0 abzuspeichern.
Die function ItRaSu(m,k,nt,np) berechnet den Index (irs) jenes Items, das eine notwendige Bedingung für die Existenz einer reellen Lösung der sb verletzt. U.A. muss gelten: $0 < nt_{i,x} < N$, $nt_{i,0} > np_0$ und $nt_{i,k[i]-1} > np_{mscs}$ für $1 \le i \le m$ und $0 \le x \le k_i-1$. Im Fall irs>0 ist keine Parameterschätzung möglich, und die function ItemPar bricht mit unso = irs ab.
Die function Tau(m,k,sb) normiert sb, indem zuerst τ berechnet und normiert wird. Stellt sich eine der Zeilen nicht als monoton steigend heraus, wird ihr Index (unso) angegeben. Der Fall unso>0 bedeutet somit eine ungültige oder unmögliche Parameterschätzung.
In einigen Funktionen, bei denen ein Ausdrucken der Matrizen nt oder sb verlangt werden kann, bringt die function Zeile(m,k,M,d) die Matrix in Zeilenform und rundet auf d Dezimalstellen. Diejenigen Elemente $M_{i,x}$, für die $x > k_{i-1}$ gilt, werden übergangen.

4. Parameterschätzung

4.a) Schätzung der bedingten Schwierigkeitsparameter (sb)

Die function ItemPar(m,k,nt,np,sb,imax,eps,pri) maximiert die bedingte Loglikelihood $cLL(sb) = \sum_{j=1}^{n0} pah_j * \ln P(\underline{x}_j \mid r)$ mittels m getrennten (k_i-1)- dimensionalen Newton-Raphson-Verfahren. Übliche Startwerte sind nt∗0; grob geschätzte Startwerte, die man z.B. beim Bootstrappen kennt, nutzen wenig.
Will man sicher gehen, dass die Lösung auch ein globales Maximum bildet, können mittels SIMsb(ti,d) mehrere Startwerte erzeugt werden – dass lokale Maxima äußerst selten vorkommen, sieht man, wenn man (etwa mittels Mischverteilungen) derartige Daten gezielt zu erzeugen versucht..
Getrennt nach den m Zeilen wird sb = sb – verb solange iteriert, bis das Maximum aller absoluten Verbesserungen (vmax) kleiner als eps wird. Ist vmax>10, wird abgebrochen. Ist vmax>1, werden die verb durch vmax dividiert. Beim ersten NR-Schritt und, falls m≤3 bei den ersten 20 Schritten, wird verb halbiert.

Die Verbesserungen werden von der function BedLL(m,k,nt,np,sb) berechnet und lauten für Item i verb = A^{-1}.sb1, wobei $sb1_x$ die erste Ableitung von cLL(sb) nach $sb_{i,x}$ ist. A ist die (k_i-1)- dimensionale Matrix der zweiten Ableitungen nach $sb_{i,x}$ und $sb_{i,y}$ – diejenigen nach $sb_{i,x}$ und $sb_{j,y}$ werden vernachlässigt. Nach jedem NR-Schritt werden die sb mittels Tau(m,k,sb) normiert. Die Ergebnisse der ersten pri Iterationen werden ausgedruckt.
Das Verfahren konvergiert, wenn vmax kleiner eps wird. Eine ungültige Schätzung erkennt man an unso>0, das den Index des nicht monoton steigenden $\tau_{i,.}$ angibt. Eine Schätzung, die wegen ungültiger Randsummen $nt_{i,}$ nicht möglich war, erkennt man ebenso an unso=i. Ist unso=0 und vmax knapp größer als eps, kann es – sofern eine Lösung existiert – erfolgreich sein, andere Startwerte zu versuchen oder imax zu erhöhen.
Die sb werden unter der Bedingung geschätzt, dass bei festgehaltenem r die Summe aller Patternwahrscheinlichkeiten gleich np_r/N wird. Die conditional loglikelihood ist der Logarithmus des Produktes der bedingten Patternwahrscheinlichkeiten $P(\underline{x}|r) = \exp\left[-\sum_{i=1}^{m} sb_{i,x(i)}\right] / \gamma_r$ und lautet $cLL(sb) = -\sum_{i=1}^{m}\sum_{x=1}^{k(i)-1} nt_{i,x} * sb_{i,x} - \sum_{r=1}^{mscs} np_r * \ln \gamma_r$.

γ_r ist symmetrische Grundfunktion (sgf) r- ter Ordnung und gleich der Summe aller Produkte $\exp[-sb_{i,x(i)}]$, bei denen die Summe aller x_i gleich r ist. Die erste Ableitung von γ_r nach $sb_{i,x}$ lautet $-sgf_{r,i,x}$, wobei $sgf_{r,i,x}$ die Summe derjenigen Produkte ist, an denen $sb_{i,x}$ beteiligt ist. Man berechnet $sgf_{r,i,1}$ am einfachsten, indem man (vorübergehend) $\exp[-sb_{i,1}] = 0$ setzt, γ^* berechnet und γ_r^* von γ_r abzieht. Die restlichen berechnen sich als $sgf_{r,i,1} * \exp[-sb_{i,x}+sb_{i,1}]$. Die zweiten Ableitungen von γ_r nach $sb_{i,x}$ lauten $sgf_{r,i,x,x} = +sgf_{r,i,x}$, diejenigen nach $sb_{i,x}$ und $sb_{i,y}$ sind $sgf_{r,i,x,y} = 0$, und diejenigen nach $sb_{i,x}$ und $sb_{j,x}$ (oder $sb_{j,y}$) werden vernachlässigt.

Die erste Ableitung von cLL(sb) nach $sb_{i,x}$ lautet $sb'_{i,x} = -nt_{i,x} + \sum_{r=1}^{mscs} np_r * sgf_{r,i,x} / \gamma_r$.

Die 2. Ableitung nach $sb_{i,x}$ und $sb_{i,y}$ lauten $A_{x,y} = \sum_{r=1}^{mscs} np_r * (-sgf_{r,i,x,y} + sgf_{r,i,x} * sgf_{r,i,y}) / \gamma_r^2$.

Die symmetrischen Grundfunktionen γ

Am einfachsten und genauesten ist die Berechnung, wenn mit Item 1 und $G_{1,.}$ begonnen und sukzessive ein weiteres Item dazu genommen wird, um $G_{i,.}$ zu berechnen. Die Länge von $G_{i,}$ ist die Summe der k_j-1 für j = 1 bis i. Am Ende ergeben sich die mscs Elemente von $\gamma = G_{m,}$, wobei das letzte zur Kontrolle $\gamma_{mscs} = e^0 = 1$ sein muss. Da der Algorithmus verbal nur umständlich zu beschreiben ist, folgt ein Beispiel:
Die drei Items haben je 3, 2 und 4 Kategorien. $G_{1,}$ besteht aus den zwei Elementen 11 und 12, wobei z.B. $\exp[-sb_{1,2}]$ mit 12 abgekürzt ist.
Die Aufnahme von Item 2 ergibt $G_{2,1} = G_{1,1} + 21$, $G_{2,2} = G_{1,2} + G_{1,1}*21$ und $G_{2,3} = G_{1,2}*21$.
Mit Item 3 ergibt sich $G_{3,1} = G_{2,1} + 31$, $G_{3,2} = G_{2,2} + 32 + G_{2,1}*31$, $G_{3,3} = G_{2,3} + 33 + G_{2,2}*31 + G_{2,1}*32$, $G_{3,4} = G_{2,3}*31 + G_{2,2}*32 + G_{2,1}*33$, $G_{3,5} = G_{2,3}*32 + G_{2,2}*33$ und $G_{3,6} = G_{2,3}*33$.
In der function SGF(m,k,sb,i) wird $G_{i,r}$ nur einfach indiziert, und nach jeder Aufnahme $\gamma = G$ gesetzt.

4.b) Schätzung der gewichteten Personenparameter (fw)

Die Modellgleichung des partial credit model lautet $P(i,x_i|r) = Zl_{r,i} / Ne_{r,i}$ mit dem Zähler $Zl_{r,i} = \exp[x_i * fw_r - sb_{i,x(i)}]$ und dem Nenner $Ne_{r,i} = \sum_{y=0}^{k(i)-1} \exp[y * fw_r - sb_{i,y}]$.

Die ungewichtete Likelihood einer Person mit Scoresumme r lautet uL(f_r) = $\prod_{i=1}^{m} P(i, x_i \mid r)$.

Nach Warm gewichtet lautet sie wL(fw_r) = $\left(\prod_{i=1}^{m} P(i, x_i \mid r) \right) * \sqrt{-f_r''}$,

wobei f_r'' die 2. Ableitung der uL(f) nach fw_r ist, d.h. nach Einsetzen des fw_r in die uL. f_r'' wird im Verlauf des NR- Verfahrens $fw_r -= fw_r' / fw_r''$ als Nenner verwendet. Nach Konvergenz des Verfahrens ist $1/\sqrt{-f_r''}$ der geschätzte Standardfehler se(fw_r).

Nur für dichotome Items gilt $f_r'' = -\sum_{i=1}^{m} P(i, x_i \mid r) * (1 - P(i, x_i \mid r))$.

Die gewichtete Loglikelihood lautet wLL(f_r) = $\sum_{i=1}^{m} \ln P(i, x_i \mid r) + \ln(-f_r'')/2$.

Die Summe wird zu $\sum_{i=1}^{m} \left(x_i * fw(r) - sb_{i,x(i)} - \ln Ne_{r,i} \right) = r * f_r - \sum_{i=1}^{m} \ln Ne_{r,i} + c$,

da die Summe der x_i gleich r ist, und $sb_{i,x}$ nicht von r abhängt. Mit L_r= $\sum_{i=1}^{m} \ln Ne_{r,i}$ ergibt sich

wLL(fw_r) = r∗fw_r – L_r + $\ln(-f_r'')/2$.

Die Likelihood aller Personen mit Scoresumme r ist die np_r- te Potenz von wL(fw_r). Die gewichtete Loglikelihood aller Beobachtungen lautet

wLL(fw) = $\sum_{r=0}^{mscs} np_r * \left(r * fw_r - L_r + \ln(-f_r'')/2 \right)$.

Die 1. Ableitung von wLL(fw_r) nach fw_r lautet $fw_r' = r - L_r' - f_r''' / f_r'' / 2$,

wobei $L_r' = \sum_{i=1}^{m} \frac{Ne_{r,i}'}{Ne_{r,i}}$ und $Ne_{r,i}^{(\ell)} = \sum_{y=1}^{k(i)-1} y^{\ell} * \exp[y * fw_r - sb_{i,y}]$ gilt.

$L_r'' = \sum_{i=1}^{m} \left(\frac{Ne_{r,i}''}{Ne_{r,i}} - \left(\frac{Ne_{r,i}'}{Ne_{r,i}} \right)^2 \right) = -f_r''$, wobei $1/\sqrt{-f_r''}$ der geschätzte Standardfehler ist, und $-f_r''$ als Schätzer von fw_r'' verwendet wird (mit einem etwas kleineren Absolutbetrag). Seine Ableitung lautet $f_r''' = \sum_{i=1}^{m} \left(Ne_{r,i}''' - 3 * Ne_{r,i}'' * Ne_{r,i}' / Ne_{r,i} + 2 * Ne_{r,i}'^{3} / Ne_{r,i}^2 \right) / Ne_{r,i}$.

Der Iterationsschritt des NR- Verfahrens lautet somit fw– = fw'/fw''.
Die function PersPar(np,sb,imax,eps,pri) berechnet sb mit mscs+1 getrennten eindimensionalen NR-Schritten, wobei für die Verbesserungen, wLL und f'' die function GewLL(np,sb,fw) aufgerufen wird. Die ersten pri Schritte können ausgedruckt werden.
Als Startwerte eignen sich mscs+1 Nullen oder beliebige aufsteigend geordnete und summennormierte Zahlen. Das Verfahren konvergiert, wenn Max(|verb|) < ε, und benötigt meistens knapp 10 Schritte. Für die Standardfehler gilt se(fw_r) = $\sqrt{-1/f2_r}$. Der Mittelwert Mf weicht stets knapp von 0 ab. Berechnet man wLL mit den summennormierten fw-Mf, ergibt sich eine geringfügig kleinere loglikelihood.

4.c) Standardfehler der Schwierigkeitsparameter sb

Setzt man sb und fw in die uLL ein, und bildet man die 2. Ableitung nach $sb_{i,x}$, ergibt sich $sb''_{i,x} = -\sum_{r=0}^{mscs} np_r * P(i,x|r) * (1-P(i,x|r))$ und der Standardfehler $se(sb_{i,x}) = \sqrt{-1/s''_{i,x}}$. Dies berechnet die function StdFsb(np,sb,fw).

4.d) Andrichs Reliabilität:

Die function AndRel() berechnet ARel = 1 – mev/gVf

mittels der marginal error variance: mev = $\sum_{r=0}^{mscs} np_r * sefw_r^2 / N$,

dem gewichteten Mittelwert der fw: gMf = $\sum_{r=0}^{mscs} np_r * fw_r / N$

und der gewichteten Varianz der fw: gVf = $\sum_{r=0}^{mscs} np_r * (fw_r - gMf)^2 / N$.

5. Modell-Tests

5.a) Globale Modell-Tests

Die function AnpTest3(np,n0,dez,pah,sb,cLL,pri) berechnet Pearsons und Cressie-Reads χ^2 (PC und CR) sowie die log-ratio (LR). Ist ein Beitrag (pc) zur Prüfgröße PC größer als pri, wird er zusammen mit dem Pattern, pah und E(pah) ausgedruckt.

Ein allgemeines Abstandsmaß ist $C(\lambda) = \frac{2}{\lambda(\lambda+1)} * \sum_{j=1}^{kl} B_j * \left(\left(\frac{B_j}{E_j} \right)^{\lambda} - 1 \right)$, wobei kl die Zahl der Klassen ist, und B_j und E_j die beobachteten und erwarteten Häufigkeiten sind.

Für $\lambda = 1$ ergibt sich Pearsons χ^2 : $PC = \sum_{j=1}^{amp} \frac{(pah_j - E(pat_j)^2}{E(pat_j)}$.

Für $\lambda = 2/3$ ergibt sich Cressie-Reads χ^2 : $CR = 1.8 * \sum_{j=1}^{amp} pah_j * \left(\left(\frac{pah_j}{E(pat_j)} \right)^{2/3} - 1 \right)$.

Beide Prüfgrößen sind asymptotisch χ^2- verteilt mit df Freiheitsgraden. Deren Anzahl ist df = amp–2∗mscs, da mscs Parameter sb geschätzt worden sind, von denen einer wegen der Summennormierung entfällt. Weiters entfallen mscs+1 Freiheitsgrade wegen des Zusammenhangs $\sum_{scs(j)=r} E(pat_j) = np_r = \sum_{scs(j)=r} pah_j$.

Die erwarteten bedingten Häufigkeiten berechnen sich als $E(pat_j) = np_r * \exp\left[-\sum_{i=1}^{m} sb_{i,x(i)} \right] / \gamma_r$,

wobei $r = \sum_{i=1}^{m} pat_{j,i}$ die Scoresumme ist. Da amp extrem groß werden kann, wird bei beiden Prüfgrößen nur über j = 1 bis n0 summiert. Da der Beitrag eines unbeobachteten Patterns zu PC bzw. CR gleich $E(pat_j)$ bzw. 0 ist, muss zu PC noch die Summe derjeniger Erwartungen addiert werden, bei denen $pah_j = 0$ gilt; sie lautet $N - \sum_{pah(j)>0} E(pat_j)$.

Der Likelihood-Quotienten-Test:
Die log-ratio $LR = -2*(LL_{mod} - LL_{sat})$ ist asymptotisch χ^2- verteilt mit df Freiheitsgraden.

$LL_{mod} = cLL(\underline{sb}) + \sum_{r=0}^{mscs} np_r * \ln(np_r / N)$ für $np_r > 0$.

$LL_{sat} = \sum_{j=1}^{n0} pah_j * \ln(pah_j / N)$, wobei n0 die Anzahl der besetzten Pattern ist.

Sind mehrere Erwartungen kleiner als 5, sind die drei Prüfgrößen nicht χ^2- verteilt, und p-values können nur mittels Bootstrappen geschätzt werden.

5.b) Itemfit-Tests nach Wright&Masters (MESA Press 1982):

Out- und infit- Maße dienen zur Überprüfung der Modellannahme, dass die Trennschärfen der Items gleich sind, und verhalten sich unterschiedlich sensibel gegenüber bestimmten Modellverletzungen. Ihre t- Werte sollten standardnormalverteilte Prüfgrößen sein.

Der t- Wert des outfit- Maßes von Item i berechnet sich als $tout_i = \left(outfit_i^{1/3} - 1\right) * c + 1/c$, wobei $c = 3/\sqrt{q2out_i / N^2 - 1/N}$ und in Multira der Quotient 1/N nicht subtrahiert wird.

$outfit_i = \sum_{j=1}^{n0} pah_j * (pat_{j,i} - erw_i)^2 / var_i / N$. Da nicht die Trennschärfen der Items berechnet werden, sondern nur deren vom Modell geforderte Gleichheit untersucht wird, sind die m outfit-Maße derart zu korrigieren, dass ihr Mittelwert gleich 1 wird: $outfit_i+ = 1 - mean(outfit)$. Ist $tout_i$ signifikant zu groß oder zu klein (z.B. größer 2 bzw. kleiner -2), schließt man mit ca. 95%- iger Sicherheit auf underfit bzw. overfit – allerdings nur, wenn $tout_i \sim N(0,1)$ nachgewiesen ist.

Zu beachten ist, dass die m Prüfgrößen der modellgemäßen Gleichheit der Trennschärfen selbst bei perfekten Daten ($pah_j = E(pat_j)$) nicht gleich groß, und ihre t-Werte nicht Null sind. Die exakten Erwartungswerte können als pahE mittels SIMexakt(1,F) berechnet werden. Die mit diesen Daten berechneten out- und infit- Maße sind gute Orientierungshilfen. Verwendet man gzz=F, entsteht n0=amp, was eine extrem große Zahl sein kann; in diesem Fall ist gzz=T & (v=1 oderv=10) zu übergeben.

$q2out_i = \sum_{j=1}^{n0} pah_j * X4_i / var_i^2$,

wobei $X4_i = \sum_{x=0}^{k(i)-1} (x - erw_i)^4 * p_x$, $var_i = \sum_{x=0}^{k(i)-1} (x - erw_i)^2 * p_x$, $erw_i = \sum_{x=0}^{k(i)-1} x * p_x$,

und p_x die Modellwahrscheinlichkeit $P(i,x_i|r)$ ist, und r die Scoresumme von pat_j.

Die t- Werte der infit- Maße berechnen sich als $tin_i = \left(infit_i^{1/3} - 1\right) * c + 1/c$,

wobei $c = 3/\sqrt{q2in_i}$ und $q2in_i = \sum_{j=1}^{n0} pah_j * (X4_i - var_i^2) / \left(\sum_{j=1}^{n0} pah_j * var_i\right)^2$.

Die m Maße $infit_i = \sum_{j=1}^{n0} pah_j * (pat_{j,i} - erw_i)^2 / \sum_{j=1}^{n0} pah_j * var_i$

sind noch wie vorhin zu korrigieren: $infit_i + = 1 - mean(infit)$.

<u>Man beachte</u>: Die t- Werte der outfit- und infit-Maße sind nicht standardnormal verteilt, wie folgende drei jederzeit wiederholbare Versuchsanordnungen mit dichotomen Items zeigen:

a) 1000 Mal wird wiederholt: Simulation von m normierten Schwierigkeitsparametern SB ~ U(-2,2), Simulation von m+1 summennormierten und sortierten Fähigkeitsparametern FW ~ N(0,2), NP_i =50, Simulation der Daten, Schätzung der Modellparameter, Berechnung der tout- und tin- Werte und Abspeicherung in 2 Matrizen der Dimensionen 1000∗m.

b) Wie Methode a), jedoch werden die Parameter SB und FW als arithmetische Folgen zwischen -2 und 2 bzw. -2.5 und 2.5 festgehalten, d.h. nicht mehr simuliert.
Bei Betrachtung der 2∗m Spalten, die Stichproben aus N(0,1)- Verteilungen sein sollten, stellt sich (von zufallsbedingten geringen Abweichungen abgesehen) Folgendes heraus:

a) m = 5: $tout_i \sim N(0.1, 0.35^2)$, $tin_i \sim N(0.1, 0.75^2)$
m = 10: $tout_i \sim N(0.05, 0.5^2)$, $tin_i \sim N(0.1, 0.85^2)$

b) Die Mittelwerte und Varianzen hängen von den Positionen der SB ab. Beide sind bei den Items 1 und m minimal und steigen zum mittelschweren Item hin an. Dies ist in den folgenden Tabellen mittels „min→max“ beschrieben:
m = 5: $tout_i \sim N(\ 0.10 \rightarrow 0.20,\ 0.30^2 \rightarrow 0.40^2)$
$tin_i \sim N(-0.25 \rightarrow 0.35,\ 0.40^2 \rightarrow 0.65^2)$
m =10: $tout_i \sim N(\ 0.00 \rightarrow 0.10,\ 0.50^2 \rightarrow 0.60^2)$
$tin_i \sim N(-0.20 \rightarrow 0.10,\ 0.60^2 \rightarrow 0.85^2)$

c) Die Parameter SB werden als arithmetische Folgen zwischen -2 und 2 festgelegt, und es werden 30000 bzw. 55000 exakte Daten erzeugt – d.h. die Patternhäufigkeiten (pah) stimmen bis auf Rundungsfehler mit ihren Erwartungen überein:
m = 5: $outfit_i = 0.97 \rightarrow 1.03$, $infit_i = 0.97 \rightarrow 1.02$
$tout_i = -0.93 \rightarrow 1.94$, $tin_i = -2.77 \rightarrow 3.38$
m = 10: $outfit_i = 0.97 \rightarrow 1.02$, $infit_i = 0.98 \rightarrow 1.01$
$tout_i = -1.27 \rightarrow 1.80$, $tin_i = -2.10 \rightarrow 1.54$

Aus diesem Grund können die p- values der Itemfitmaße nur mit Bootstrappen geschätzt werden.

5.c) Der Q-Index (nach Rost & von Davier, Appl. Psy. M. 1994, 171-182)

Die Q-Indizes sind Maße dafür, wie weit die beobachteten von den optimalen Pattern abweichen, und sollten modellgemäß gleich groß sein. Je Item werden 3 Vektoren der Länge N erzeugt: im ersten stehen die Scoresummen r, im zweiten die Beobachtungen $x = pat_i$. Diese beiden werden gemeinsam aufsteigend sortiert, d.h. zuerst nach r, und bei Gleichheit nach x. Im dritten Vektor stehen die Werte g und entsprechen den Guttmann-Pattern: für sie gilt $g_n = 0$ für $1 \le n \le nt_{i,0}$, $g_n = 1$ für $nt_{i,0} < n \le nt_{i,1}$ usw. bis $g_n = k_i\text{-}1$ für die letzten $nt[i,k_i\text{-}1]$ Werte. Die Anti-Guttman-Werte ergeben sich als $ag_n = g_{N+1-n}$ und werden zur Normierung gebraucht.

Nun berechnet sich der Q-Index des Items i als

$$Q_i = \sum_{n=1}^{N} (x_n - g_n) * fw_{r[n]+1} \,/\, \sum_{n=1}^{N} (x_n - ag_n) * fw_{r[n]+1},$$

wobei r, x und g die Elemente der drei Vektoren sind.
Im oben angeführten Artikel werden außerdem noch $z(Q_i)$ hergeleitet, die standardnormalverteilte Prüfgrößen sein sollten. Das kann nicht zutreffen, da dabei Erwartungswerte und Varianzen von x∗fw unter Unabhängigkeitsannahme berechnet werden – deren Korrelationen sind signifikant größer Null und können als Maße für die Itemtrennschärfen betrachtet werden. Aus diesem Grund, und weil die Q-Indizes wie die Itemfitmaße positionsabhängig sind, können p- values nur mit Bootstrappen geschätzt werden. Da nur die Gleichheit der Trennschärfen (deren Prüfgrößen aber auch bei exakten Daten nie gleich bzw. Null sind) getestet wird, sind dazu die summennormierten Q_i – mean(Q) zu verwenden. Gute Orientierungshilfen sind die Q-Indizes perfekter Daten, die mittels SIMexakt(10,T) erzeugt werden können.

5.d) Bootstrappen

Die function Boot33m(B,Chi3,QiOI) bootstrappt gleichzeitig die p-values der 3 globalen (PC, CR, LR) und der 3∗m Itemfit-Prüfgrößen (Qi, outfit, infit), sofern die Längen von Chi3 und QiOI gleich 3 bzw. 3m sind. Die 3+3m festgehaltenen Prüfgrößen werden als Chi3 und QiOI übergeben. Es folgt die Beschreibung, wie mit einer einzelnen Prüfgröße verfahren wird:
1. Die zu prüfende Statistik wird als X0, die Parameter sb, fw und np werden als SB, FW und NP festgehalten.
2. Dann wird eine Zahl B gewählt und B Mal werden neue Daten mittels SIMord() parametrisch simuliert – im Unterschied zum originalen Verfahren, wo B Mal mit Zurücklegen aus den Daten (dezN) gezogen wird. In jeder der B Stichproben werden
2a) sb und (für die Itemfitmaße) fw geschätzt,
2b) die Statistik XB berechnet,
2c) und wird mitgezählt, wie oft XB > X0 ist – diese Anzahl ist $0 \le n \le Bk$, wobei $Bk \le B$ die Zahl der gültigen Fälle ist (ItemPar konvergiert).
3. Die relative Häufigkeit pv = n/Bk ist nun die bootsgetrappte Überschreitungswahrscheinlichkeit, dass der wahre Wert X unter H_0 (Gültigkeit der Modellannahmen) größer als das beobachtete X0 ist. Ist p der wahre p-value, dann ist pv Bin(B,p)- verteilt. Ist B∗pv∗(1-pv) hinreichend groß, dann ist pv annähernd N(pv, B∗pv∗(1-pv))- verteilt, womit ein Konfidenzintervall angegeben werden kann.
Die p-values sind die Wahrscheinlichkeiten von Fehlern 1. Art, d.h. H_0 irrtümlich abzulehnen. Eine notwendige Bedingung, dass sie richtig geschätzt worden sind, ist die, dass sie der Stichprobe einer U(0,1)- Verteilung entsprechen müssen (Hinw.: Die Transformation $Y = F_x(X)$ ist gleichverteilt). Dies kann man wie folgt nachprüfen:
Man nimmt m und k an, sowie die Verteilungen von SB0,FW0,NP, und wiederholt Wh Mal: 1. Simulation von SB0,FW0, 2. Schätzung der sb,fw und Festhalten als SB,FW, 3. Berechnung der Prüfgröße und Festhalten als X0, 4. B Mal Bootstrappen und Speichern des p-values. Das ergibt Wh Beobachtungen pv, deren Gleichverteilung mittels Histogramm graphisch und mittels Verteilungstest statistisch geprüft werden kann – man wird, insbesonders in den kritischen Bereichen 0<pv<0.1 und 0.9<p<1, keine systematischen Abweichungen finden.
Die Wahrscheinlichkeit, bei modellkonformen Daten mindestens ein signifikantes (pv < 0.025 oder pv > 0.975) Itemfitmaß (Qi, Out- oder Infit) zu beobachten, wird mit steigender Itemzahl naturgemäß immer größer; sie beträgt bei 5 Items ca. 54%, bei 10 Items ca. 79%, und bei 20 Items ca. 95%.

5.e) Personenhomogenität

Sie wird mit der function PersHom(split,pri) getestet. Nach Teilung der Personen in die Gruppen r ≤ split und r > split werden in beiden Gruppen die Itemparamer sb1 und sb2 geschätzt und die loglikelihoods cLL1 und cLL2 berechnet. Andersens Prüfgröße -2(cLL-cLL1-cLL2) ist asymptotisch χ^2- verteilt mit mscs-1 Freiheitsgraden.
Um festzustellen, welche Items eine eventuelle Inhomogenität verursachen, werden die Wald-Statistiken $z_{i,x} = (sb1_{i,x} - sb2_{i,x}) / \sqrt{se(sb1)^2_{i,x} + se(sb2)^2_{i,x}}$ berechnet. Sie sind nur bei dichotomen Items N(0,1)- verteilt. Die Summe $\sum_{x=1}^{k(i)-1} z^2_{i,x}$ bei polytomen Items ist nicht χ^2- verteilt.
Ist pri>0, werden für beide Gruppen die Ergebnisse der Itemparameterschätzung ausgedruckt, ist pri>1, zusätzlich die Randsummen und Personenparameter. Können in einer der Gruppen die sb nicht geschätzt werden, endet die function mit df = -1 bzw. -2; an Chi2 = -unso erkennt man den Index des Items, das eine ungültige Schätzung verursacht hat .

5.f) Itemhomogenität

Für den Martin-Löf-Test wird die function MarLoef(ind1,cLL,pri) aufgerufen. Die Items werden entsprechend der Indexmenge ind1 in zwei Gruppen geteilt. Die Maximalscores beider Gruppen lauten msc1 und msc2. Es werden die Itemparameter sb1 und sb2 geschätzt und die loglikelihoods cLL_1 und cLL_2 berechnet. Andersens Prüfgröße $-2*(LL_{mod} - LL_{12})$ ist asymptotisch χ^2- verteilt mit msc1∗msc2-1 Freiheitsgraden.

$$LL_{mod} = cLL(sb) + \sum_{r=0}^{mscs} np_r * \ln(np_r / N).$$

$$LL_{12} = cLL(sb1) + cLL2(sb2) + \sum_{r1=0}^{msc1}\sum_{r2=0}^{msc2} n_{r1,r2} * \ln(n_{r1,r2} / N),$$

wobei $n_{r1,r2}$ die Häufigkeit der Scoresummen r1 und r2 in den beiden Gruppen ist. Sind mehrere n12 kleiner als 5, muss X2 bootsgetrappt werden. Ist pri>0, werden die sb beider Gruppen ausgedruckt; ist pri=2, die Randsummen; ist pri>2, zusätzlich noch n12. Mit m- fachen Aufruf der function MarLöf(i,0) für i = 1:m findet man das Item, das eine eventuelle Inhomogenität verursacht.
Zu beachten ist: Entsteht beim Teilen eine Gruppe mit wenigen Items und kleinen Zahlen np, kann möglicherweise keine Lösung existieren – drei Existenzbedingungen werden in der function ItRaSu abgefragt. Im Fall von zwei dichotomen Items existiert eine explizite Lösung. Sie lautet $\exp[2*sb_2] = (-nt_1+nt_2-np_1) / (nt_1-nt_2-np_1)$ und kann kleiner gleich Null sein. Endet die function mit df = -1 oder -2, konnte in einer der beiden Gruppen keine oder nur eine ungültige Lösung gefunden werden. Den Index des betreffenden Items erkennt man am Wert von Chi2 = -unso.

6. Alphabetische Übersicht aller R-Funktionen

AndRel(np,fw): Andrichs Reliabilität.
AnpTest3(np,n0,dez,pah,sb,cLL,pri): Berechnet die Prüfgrößen (PC, CR, LR) von 3 globalen Modelltests. Ergebnis: list(Chi3,df).
Basis(): Berechnet die Basis (bas).
BedLL(m,k,nt,np,sb): Berechnet die Verbesserungen und cLL(sb) für ItemPar. Ergebnis: list(verb, cLL).
Boot33m(B,Chi3,QiOI): Schätzt die 3 p-values der Anpassungs- und die 3∗m p-values der Itemfit-Tests. Ergebnis: list(pv3, pvQi, pvOu, pvIn).
Daten.N(dezN): Ergebnis: list(nt, np, n0, dez, pah). Aufruf von Daten.n0.
Daten.n0(n0,dez,pah): Ergebnis: list(nt, np).
DezPat(d): Rechnet eine Dezimalzahl zum Pattern zurück. Wird von AnpTest3, Daten.n0, Itemfit, PersHom, Q.Index und SIMexakt aufgerufen. Ergebnis: pat.
Elim(eli): Eliminiert die Items mit Indizes eli. Ergebnis: list(m, k, nt, np, n0, dez, pah).
GewLL(np,sb,fw): Berechnet die Verbesserungen, wLL(fw) und se(fw) für PersPar. Ergebnis: list(wLL, verb, f2).
Itemfit(nt,n0,dez,pah,sb,fw): Berechnet die out- und infit- Maße und deren Prüfgrößen. Ergebnis: c(outfit, infit, t.out, t.in).
ItemPar(m,k,nt,np,sb,imax,eps,pri): Schätzung der Itemparameter und der cLL. Aufruf von ItRaSu, Tau und BedLL. Ergebnis: list(itmax, vmax, cLL, sb, tau, unso).
ItRaSu(m,k,nt,np): Abfrage, ob die Itemrandsummen gültig sind. Ergebnis: irs.
MarLöf(ind1,pri): Berechnet die Prüfgröße des Martin-Löf-Tests. Aufruf von ItemPar. Ergebnis: list(Chi2, df).
PersHom(split,pri): Prüfgrößen des Andersen- und der Wald-Tests (wenn dichotom).

Aufruf von ItemPar, PersPar und StdFsb. Ergebnis: list(Chi2, df, zWald).
PersPar(np,sb,imax,eps,pri): Schätzung der Personenparameter samt se(fw). Aufruf von GewLL. Ergebnis: list(itmax, cmax, wLL, fw, se.fw).
Q.Index(nt,np,n0,dez,pah,sb,fw): Ergebnis: Die Q-Indizes Qi.
SGF(m,k,sb,i0): Ergebnis: Die symmetrischen Grundfunktionen sgf (ohne sb[i0,1]).
SIMexakt(v,gzz): Erzeugt exakte pah. Ergebnis: list(ntE, npE, n0E, dezE, pahE).
SIMord(): Simuliert die Dezimalcodes aller N Pattern. Ergebnis: dezN.
SIMsb(ti1,d,rad): Simuliert Startwerte für ItemPar. Ergebnis: sb.
StdFsb(np,sb,fw): Schätzt die Standardfehler der sb. Ergebnis: se.sb.
Tau(m,k,sb): Normiert sb und prüft deren Gültigkeit. Ergebnis: list(sb, tau, unso).
Zeile(m,k,M,d): Vektor der Länge mscs für Zwischenausdrucke von Matrizen in Zeilenform.

7. Demonstrationsbeispiel

Anmerkung Holm:
Die Daten, auf die sich das folgende Beispiel bezieht, sind im Almo-Ordner unter dem Namen „sim200x3.fre“ enthalten. Diese Datei umfasst 200 Personen (Zeilen) und 3 Items (Spalten). Das 1. Item besitzt 2 Ausprägungen, das 2. und 3. Item 3 Ausprägungen. Die Daten wurden als „Rasch-perfekte“ Daten simuliert. Wird in Almo das Rasch-Programm Prog14m4 gerechnet und dabei die Option „R-Programm erzeugen“ in der Optionsbox „Verschiedene Optionen“ aktiviert, dann wird nachfolgend beschriebene Folge von Funktionsaufrufen (und auch die relevanten Daten, z.B. `dez` und `pah`) erzeugt.

Bekannt seien die Anzahl der Items (m=3), deren Kategorienzahl (k=c(2,3,3)) und die Anzahl der Personen (N=200). Zuerst sind die maximale Scoresumme, die Anzahl möglicher Pattern und die Basis zu berechnen:

```
mscs = sum(k-1); amp = prod(k) # 5 und 18
bas  = Basis()                 # 9, 3, 1
```

Die N Dezimalcodes stammen entweder a) vom Einlesen von N Pattern und der Rechnung $dezN_j$ = Summe($pat_{j,i}$∗bas_i) oder b) aus einer Simulation mittels dezN = SIMord(), die mit selbst gewählten SB, FW und NP (Summe = N) rechnet.

```
U   = Daten.N(dezN) # berechnet n0, dez, pah, nt und np.
nt  = U$nt   # Ergebnis: 17
dez = U$dez  # die Zahlen 0:7 und 9:17
pah = U$pah  # 13, 4, 1, 7, 10, 5, 7, 6, 9, 10, 2, 23, 29, 26, 7, 22, 19
np  = U$np   # 13, 20, 51, 49, 48, 19
nt  = U$nt; Zeile(m,k,nt,0) # 147,100,61,81,53. Anstelle einer 3∗2- Matrix
             # in einer Zeile angegeben. Die Indizes lauten 1,1 2,1 2,2 3,1
             # 3,2. 1,2 ist ausgelassen.
```

Itemparameter:

```
U  = ItemPar(m,k,nt,np,nt∗0,50,0.0001,0) # Schrittweise Ausgabe wenn pri>0.
sb = U$sb; Zeile(m,k,sb,3) # -0.910, -1.058. -0.007, -0.162, 0.917
cLL = U$cLL # -188.495, gebraucht von für AnpTest3, PersHom und Marlöf
cat(U$itmax, U$vmax, U$unso, fill=T) # 21, 0.00003, 0
```

Globale Modelltests:

```
U    = AnpTest3(np,n0,dez,pah,sb,cLL,0) # Ausgabe von n0 Zeilen, wenn pri<0
Chi3 = U$Chi3     # 10.62, 10.92, 12.88= PC, CR, LR. Gebraucht von Boot33m.
df   = U$df       # 8 Freiheitsgrade
1-pchisq(Chi3,df) # 0.22, 0.21, 0.12. Die p-values, wenn ~ χ²(8).
```

Personenparameter:

```
U  = PersPar(np,sb,30,0.0001,0)  # Schrittweise Ausgabe wenn pri>0.
fw = U$fw                        # -2.698,-1.274,-0.373,0.419,1.268,2.633
cat(U$itmax,U$vmax,U$wLL,fill=T) # 8, 0.00006, -583.931
```

Standardfehler:

```
U$se.fw              # 1.702, 1.100, 0.968, 0.960, 1.079, 1.668
StdFsb(np,sb,fw)     # 0.179 0.148 0.177 0.152 0.186
```

Personenhomogenität:

```
U  = PersHom(2,0) # Trennung in Personen mit Scores 0:2 und 3:5.
       # pri>0: Ausgabe beider sb. pri>1: Ausgabe beider np, nt und fw.
Chi2    = U$Chi2   # 5.33   … Andersens Prüfgröße
df      = U$df       # 4
zWald   = U$zWald # -0.308, NA, NA. Nur für das dichotome Item 1.
```

Itemhomogenität:

```
for(i in 1:m){ U = MarLöf(i,0); cat(i, U$Chi2, U$df, fill=T) }
      # Ergibt 3 Zeilen     1, 8.32, 3     2, 6.50, 5     3, 1.35, 5.
      # Die jeweils erste Itemgruppe besteht aus Item i. Zusätzlicher
      # Ausdruck bei pri>0.
```

Itemfit-Tests:

```
Qi  = Q.Index(nt,np,n0,dez,pah,sb,fw) # 0.104, 0.097, 0.049
U   = Itemfit(nt,n0,dez,pah,sb,fw) # outfit, infit und t-Werte
out = U[1:m]      # 1.026, 1.041, 0.934
inf = U[1:m+m]    # 1.112, 1.000, 0.889
U[(2*m+1):(4*m)]  # 0.200,0.374,-0.464   1.242,0.026,-1.300  … je 3 t-Werte
                  # die t-Werte der out- und infit-Maße sind keine N(0,1)-
                  # verteilten Prüfgrößen
```

Bootstrappen:

```
U = Boot33m(200, Chi3, c(Qi,out,inf) )
U$pv3    # 0.250, 0.230, 0.145   p-values von PC, CR und LR
U$pvQi   # 0.720, 0.045, 0.815   p-values der Q-Indizes
U$pvOu   # 0.785, 0.135, 0.505   p-values der outfit-Maße
U$pvIn   # 0.580, 0.180, 0.760   p-values der infit-Maße
```

Itemfitmaße perfekter Daten:

```
U = SIMexakt(1,F) # Alle amp erwarteten pah. Ist amp extrem groß, setzt man
                  # gzz=T.
pah = U$pahE      # Sie stimmen mit denen von AnpTest3 überein, wo aber
                  # E(pah_j) von unbesetzten Pattern übergangen werden
n0   = amp; dez = 1:amp-1 # nt, np, sb und fw bleiben unverändert
U    = Itemfit(nt,n0,dez,pah,sb,fw)
U[1:m)   # 1.077, 0.990, 0.933   … outfit-Maße von perfekten Daten
U[1:m+m] # 1.119, 0.965, 0.917   … infit-Maße von perfekten Daten
U = SIMexakt(10,T) # ganzzahlige erwartete pah für die Q-Indizes
nt= U$ntE; np = U$npE; N = sum(np) # sind ca. die 10- fachen Originalwerte
                  # sb und fw könnten neu geschätzt werden, unterscheiden
                  # sich aber kaum
n0 = U$n0E; dez =U$dezE; pah = U$pahE
Q.Index(nt,np,n0,dez,pah,sb,fw)    # 0.114, 0.087, 0.054   … Q-Indizes
                                   # perfekter Daten
```