

Algebraische Statistik von Ranking-Modellen

Masterarbeit

betreut von Prof. Dr. Volkmar Welker
am Fachbereich Mathematik und Informatik
der Philipps-Universität Marburg

vorgelegt von

Benjamin Debeerst, geboren am 29. November 1985 in Lingen (Ems)

Marburg, den 21. September 2011

Ich versichere hiermit, dass ich diese Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

BENJAMIN DEBEERST

Marburg, den 21. September 2011

Inhaltsverzeichnis

Einleitung	7
I Algebra von polynomialen Gleichungssystemen	9
§ 1 Grundlagen der Gröbnerbasen-Theorie	9
§ 2 Lösen von polynomialen Gleichungssystemen	12
II Ranking-Modelle	15
§ 3 Einführung	15
§ 3.1 Definition	15
§ 3.2 Eigenschaften	16
§ 3.3 Abstandsmaße auf der symmetrischen Gruppe	17
§ 4 Einige Modelle	19
§ 4.1 Plackett-Luce	20
§ 4.2 Bradley-Terry-Mallows	25
§ 4.3 Inversions-Modell	30
§ 4.4 Weitere Modelle	33
III Maximum Likelihood Schätzer von Ranking-Modellen	37
§ 5 Hilfsmittel	38
§ 6 Vorgehensweise	38
§ 6.1 Genereller Ansatz	38
§ 6.2 Algorithmische Teillösung gegebener Probleme	39
§ 6.3 Beschränkungen und Probleme durch Rechenfehler	42
§ 6.4 Hinreichendes Kriterium für Extremstellen	44
§ 7 Ergebnisse	44
§ 7.1 Plackett-Luce	45
§ 7.2 Bradley-Terry-Mallows	48
§ 7.3 Inversions-Modell	50
§ 8 Diskussion	55
Quellcode der Maple Funktionen	57
A Modellübergreifende Funktionen	57
B Plackett-Luce	65
C Bradley-Terry-Mallows	68
D Inversions-Modell	70

Einleitung

Ranglisten sind allgegenwärtig: die Liste der besten Universitäten beim Hochschulranking, die aktuelle Tabelle in der Fußball Bundesliga oder die Liste der Top-Manager Gehälter im vergangenen Jahr — in fast allen Lebensbereichen werden Dinge bewertet und in Reihenfolgen gebracht. Auch in der Wissenschaft spielen Ranking-Modelle als Spezialfall von allgemeinen Präferenzmodellen eine große Rolle: Eine in Experimenten der Psychologie häufig verwendete Methode ist es, Probanden Auswahlentscheidungen treffen zu lassen, um auf diese Weise Präferenzen im Hinblick auf bestimmte Vorbedingungen oder Einflüsse zu untersuchen. In der Informatik werden diese psychologischen Modelle auf Systeme künstlicher Intelligenz übertragen, etwa um anhand bekannter Entscheidungen eines Benutzers spätere Verhaltensweisen zu prognostizieren. Ein prominentes Beispiel hierfür sind die Produktempfehlungen der Webseite amazon.de, welche die bekannten Kaufentscheidungen ihrer Kunden verwendet um weitere Produkte zu bewerben. Anhand der Überschrift unter der die Produktempfehlungen beworben werden, „Kunden, die diesen Artikel gekauft haben, kauften auch“, lässt sich erkennen dass hier Präferenzmodelle zugrunde gelegt werden um „intelligente“ Empfehlungen geben zu können.

Zur wissenschaftlichen Untersuchung der Zusammenhänge die zu Ranglisten führen, ist die Wahl eines geeigneten statistischen Modells von großer Bedeutung. Für die praktische Seite ist ferner von Belang, ob ein gegebenes Modell überhaupt handelbar ist, das heißt ob in diesem Modell bei gegebenen statistischen Daten die Parameter überhaupt und in angemessener Zeit geschätzt werden können.

Die vorliegende Masterarbeit widmet sich der Bestimmung von Maximum Likelihood Schätzern bei Modellen, dessen Wahrscheinlichkeitsfunktionen rationale Funktionen in ihren Parametern sind, so dass Lösungsmethoden aus der kommutativen Algebra angewendet werden können. Grobe Leitlinie ist dabei zu jedem Modell die Frage nach Bedingungen an den statistischen Daten, die zur Existenz und im besten Falle zur Eindeutigkeit eines Maximum Likelihood Schätzers führen.

Die Methoden zur Lösung polynomialer Gleichungssysteme — das heißt nicht-linearer Gleichungssysteme in mehreren Veränderlichen — werden in Kapitel I vorgestellt. Hierzu wird zunächst in Abschnitt 1 eine kurze Einführung in die Theorie der Gröbnerbasen gegeben, welche aber aufgrund ihres Umfangs nicht vollständig erklärt werden kann. Der interessierte Leser findet ausführlichere Informationen in „An introduction to Gröbner bases“ von William Adams [Ada00] sowie in „Ideals, Varieties, Algorithms“ von David Cox et.al. [CLO00]. Im folgenden Abschnitt 2 wird gezeigt, wie man die Theorie der Gröbnerbasen zur Lösung polynomialer Gleichungssysteme verwenden kann.

Kapitel II widmet sich der Erklärung der zu untersuchenden Objekte, den Ranking-Modellen. Die Theorie der Gröbnerbasen kommt in diesem Kapitel nur am Rande zum Einsatz, vielmehr wird in Abschnitt 3 zunächst der Begriff des Ranking-Modells definiert

und im Folgenden Eigenschaften von Ranking-Modelle besprochen, nämlich die Label-Invarianz und die Umkehrbarkeit. Ferner werden Abstandsmaße auf der symmetrischen Gruppe definiert und ihr Zusammenhang mit Ranking-Modellen aufgezeigt. Definitionen konkreter Ranking-Modelle werden in Abschnitt 4 gegeben und ausführlich besprochen: Zur Sprache kommen das Plackett-Luce-Modell in Abschnitt 4.1, das Bradley-Terry-Mallows-Modell in Abschnitt 4.2 und das Inversionsmodell in Abschnitt 4.3. Es wird außerdem jeweils gezeigt, welche der vorgenannten Eigenschaften die Ranking-Modelle erfüllen. Der Abschnitt 4 schließt ab mit dem Ausblick auf weitere mögliche Ranking-Modelle in, welche im Zusammenhang mit dieser Masterarbeit nicht besprochen werden.

In Kapitel III werden die beiden vorgehenden Kapitel zusammengeführt. Hier wird versucht anhand beispielhafter Untersuchungen Regelmäßigkeiten bei der Existenz von Maximum Likelihood Schätzern festzustellen. Dazu werden in Abschnitt 5 zunächst die Methoden vorgestellt und in Abschnitt 6 die zum Zweck der Untersuchung programmierten Funktionen für Maple beschrieben, deren Quellcodes im Anhang ab Seite 57 zu finden sind. In Abschnitt 7 werden die Ergebnisse für die jeweiligen Modelle präsentiert: Im Plackett-Luce-Modell lassen sich dabei nur Vermutungen formulieren, gestützt durch die Lösungen der betrachteten Fälle. Das Bradley-Terry-Mallows-Modell erweist sich als besonders schwierig, da schon kleine Beispielfälle zu hochgradigen Polynome führen, die nicht exakt gelöst werden können. Im Inversionsmodell hingegen lässt sich ein notwendiges Kriterium für die Existenz von Maximum Likelihood Schätzern formulieren und beweisen. Der Abschnitt 8, in dem die Ergebnisse der Untersuchungen diskutiert und ein Ausblick auf weitere mögliche Untersuchungen gegeben wird, schließt Kapitel III und die Masterarbeit ab.

I Algebra von polynomialen Gleichungssystemen

§ 1 Grundlagen der Gröbnerbasen-Theorie

Die Theorie der Gröbnerbasen liefert Methoden zur Definition und Berechnung von Erzeugendensystemen von Idealen in Polynomringen. Mit deren Hilfe lassen sich eindeutige Erzeugendensysteme für Ideale definieren und auf diese Weise einige Probleme der kommutativen Algebra lösen. Hierzu zählen, bei gegebenen Polynomen f, g im Polynomring $K[x_1, \dots, x_n]$ in n Unbekannten über einem Körper K , und Idealen $I, J \subset K[x_1, \dots, x_n]$, Fragen nach dem Enthaltensein von f in I , der Gleichheit von f und g modulo I oder der Gleichheit der Ideale I und J . Darüber hinaus haben Gröbnerbasen eine Vielzahl schöner theoretischer Eigenschaften, die zudem für praktische Anwendungen ausgenutzt werden können, zum Beispiel für automatische Beweissysteme oder zum Lösen von polynomialen Gleichungssystemen. In diesem Abschnitt wird ein grober Überblick über die nötige Theorie für diese Masterarbeit gegeben, für detaillierte Ausführungen und Beweise siehe [Ada00] oder [CLO00].

Im gesamten Text wird, falls nicht anders definiert, mit S der Polynomring $K[x_1, \dots, x_n]$ in n Unbekannten über einem Körper K bezeichnet. Ein *Monom* im Polynomring S ist ein Produkt $x_{i_1} \cdots x_{i_r}$ von endlich vielen Variablen, geschrieben als $x^\alpha := x_1^{\alpha_1} \cdots x_n^{\alpha_n}$, mit $\alpha = (\alpha_1, \dots, \alpha_n)$ ein Vektor von nicht-negativen ganzen Zahlen. Es bezeichne T die Menge aller Monome in S . Ein Polynom $f \in K[x_1, \dots, x_n]$ kann also gesehen werden als endliche K -Linearkombination von Monomen, $f = \sum_{\alpha \in \mathbb{N}_0^n} c_\alpha x^\alpha$ [CLO00, Kapitel 1.1].

Ein *Ideal* im Polynomring S ist eine additive Untergruppe $I \subset S$ mit der Eigenschaft, dass für alle $f \in I, g \in S$ gilt: $f \cdot g \in I$. Gegeben eine Menge von Polynomen $g_1, \dots, g_r \in S$, so bezeichnet

$$\langle g_1, \dots, g_r \rangle := \{f_1 g_1 + \dots + f_r g_r \mid f_i \in S\}$$

das von den Polynomen g_1, \dots, g_r erzeugte Ideal. Gibt es für ein Ideal I eine Menge von Monomen m_1, \dots, m_r mit $I = \langle m_1, \dots, m_r \rangle$, so heißt I ein *monomiales Ideal* [CLO00, Kapitel 1.4].

Definition 1.1 (Termordnung) Eine Termordnung \prec ist eine totale Ordnung auf der Menge der Monome $T \subset S$ mit:

(i) $1 \preceq m$ für alle $m \in T$.

(ii) Gilt $m_1 \preceq m_2$ und $m_1, m_2, m_3 \in T$, so folgt $m_1 m_3 \preceq m_2 m_3$.

Zwei Standardbeispiele für Termordnungen sind die lexikographische Termordnung und die Grad-lexikographische Termordnung. Beide beziehen sich auf eine Ordnung der Variablen, zum Beispiel $x_1 > \dots > x_n$. Bei ersterer werden zwei Monome geordnet indem sukzessive die Potenzen der Variablen in absteigender Reihenfolge verglichen werden. Die Ordnung wird dann bestimmt durch die Ordnung der ersten Potenz in dem sich die Monome unterscheiden. Bei der grad-lexikographischen Termordnung wird zunächst nach dem Gesamtgrad der Monome geordnet und erst bei gleichem Grad die lexikographische Ordnung verwendet.

Beispiel 1.2 Sei $n = 3$. In lexikographischer Termordnung gilt

$$x_1^2 x_2 \succ x_1 x_2^3 \succ x_1 x_2^2 x_3 \succ x_2 x_3^8.$$

In Grad-lexikographischer Ordnung gilt allerdings

$$x_2 x_3^8 \succ x_1 x_2^3 \succ x_1 x_2^2 x_3 \succ x_1^2 x_2.$$

◇

Weitere wichtige Termordnungen sind zum einen die grad-revers-lexikographische Termordnung (degrevlex) sowie die Klasse der Eliminationsordnungen. Bei degrevlex werden Monome zunächst nach ihrem Grad geordnet und bei gleichem Grad werden die Variablen in aufsteigender Reihenfolge verglichen, wobei das Monom mit größerer Potenz in der ersten sich unterscheidenden Variable in der Termordnung kleiner ist. Es gilt also beispielsweise $x_1^2 x_2 x_3 \prec x_1 x_2^3$. Diese Ordnung ist von besonderem Interesse, da die Berechnung von Gröbnerbasen mit dieser Termordnung oft schneller läuft als mit anderen Termordnungen. Eliminationsordnungen werden zum Lösen von polynomialen Gleichungssystemen gebraucht und im Abschnitt 2 ausführlich besprochen. Weitere Beispiele und ausführliche Erklärung und Motivation können gefunden werden in [CLO00, Kapitel 2.1].

Definition 1.3 (Leitterm, Initialideal, Gröbnerbasis) *Es sei \prec eine beliebige Termordnung auf T .*

(i) *Ist x^β für $f = \sum c_\alpha x^\alpha \in S$ das bezüglich \prec maximale Monom mit $c_\alpha \neq 0$, so heißen $\text{lt}_\prec(f) := c_\beta x^\beta$ der Leitterm (Initialterm) und $\text{lm}_\prec(f) := x^\beta$ das Leitmonom (Initialmonom) von f .*

(ii) *Für ein Ideal $I \subseteq S$ heißt*

$$\text{in}_\prec(I) := \langle \text{lm}_\prec(f) \mid f \in I \rangle$$

das Initialideal von I . Zu einer endlichen Teilmenge $G = \{g_1, \dots, g_r\}$ heißt

$$\text{in}_\prec(G) := \langle \text{lm}_\prec(g_1), \dots, \text{lm}_\prec(g_r) \rangle$$

das Initialerzeugnis von G .

(iii) Für ein Ideal $I \subseteq S$ heißt $G \subseteq I$ eine Gröbnerbasis von I , falls gilt: $\text{in}_{\prec}(I) = \text{in}_{\prec}(G)$.

Ein Erzeugendensystem G eines Ideals I ist also eine Gröbnerbasis, falls in I keine Leitmonome auftauchen, die nicht auch schon von den in G auftretenden Leitmonomen erzeugt werden, oder kürzer, falls das Initialideal des von G erzeugten Ideals und das Initialerzeugnis von G übereinstimmen. Ist die Termordnung klar, so wird diese in der obigen Notation oft weggelassen.

Satz 1.4 Sei $I \subset S$ ein Ideal und sei eine Termordnung festgelegt. Dann existiert eine endliche Menge $G \subset I$, die bezüglich der gegebenen Termordnung eine Gröbnerbasis ist.

Beweis. Mit den bekannten Sätzen der kommutativen Algebra ist dies leicht gezeigt: Nach dem Hilbert'schen Basissatz ist jedes Ideal endlich erzeugt, sowohl das Ideal I als auch das Initialideal $\text{in}(I)$. Sei $G = \{f_1, \dots, f_r\}$ ein Erzeugendensystem von I . Das Initialideal $\text{in}(I)$ ist ein monomiales Ideal und wird, man folge dem Beweis des Basissatzes [CLO00, Kapitel 2.5], von endlich vielen *Monomen* erzeugt. Diese Monome sind Leitmonome von bestimmten Polynomen $g_1, \dots, g_s \in I$, woraus folgt, dass $G \cup \{g_1, \dots, g_s\}$ eine endliche Gröbnerbasis ist, da nun Initialideal und Initialerzeugnis übereinstimmen. \square

Im Gegensatz zum Nachweis der Existenz ist zunächst völlig unklar, wie zu einem gegebenen Ideal eine Gröbnerbasis oder das Initialideal zu bestimmen sind. Insbesondere zeigt das nachfolgende Beispiel, dass im Allgemeinen die Gleichung $\text{in}(\langle f_1, \dots, f_r \rangle) = \langle \text{lm}(f_1), \dots, \text{lm}(f_r) \rangle$ nicht gilt:

Beispiel 1.5 Sei $n = 3$, $f_1 = x_1 - x_2 + x_3$, $f_2 = x_1 + x_2 + x_3$, $I = \langle f_1, f_2 \rangle$. Sei die Termordnung grad-lexikographisch mit $x_1 > x_2 > x_3$.

Nun gilt $\text{lm}(f_1) = \text{lm}(f_2) = x_1$, also $\langle \text{lm}(f_1), \text{lm}(f_2) \rangle = \langle x_1 \rangle$. Aber es gilt $x_2 = \text{lm}(f_1 - f_2)$, also $\langle x_1, x_2 \rangle \subseteq \text{in}(I)$. \diamond

Der Algorithmus von Buchberger löst das Problem, aus einem gegebenen Erzeugendensystem eines Ideals eine Gröbnerbasis zu berechnen. Grundlage hierfür ist die *Reduktion* von Polynomen, eine Verallgemeinerung der Polynomdivision auf n Unbekannte, und das damit formulierbare *Buchberger-Kriterium* für Gröbnerbasen. Die Ausformulierung dieser Theorie würde hier den Rahmen sprengen und wird daher ausgelassen. Der interessierte Leser sei auf [CLO00, Kapitel 1–3] sowie auf [Ada00] verwiesen.

Im Wesentlichen sucht der Algorithmus auf systematische Weise nach einem Polynom im Ideal, dessen Leitmonom noch nicht im Initialerzeugnis des Erzeugendensystems enthalten ist und fügt dieses Polynom gegebenenfalls zum Erzeugendensystem hinzu. Dies wird so lange wiederholt, bis keine solchen Polynome mehr existieren. Nach Konstruktion ist das Resultat eine Gröbnerbasis. Der Algorithmus terminiert, da nach Hilbert'schem Basissatz jedes Ideal — also auch das Initialideal — endlich erzeugt ist und diese Erzeuger nach endlich vielen Schritten vorliegen müssen. Allerdings sind sowohl Laufzeit als auch Platzaufwand von asymptotisch exponentieller Größenordnung, so dass die Berechnung einer Gröbnerbasis mit Hilfe eines Computeralgebrasystems wie Maple oder Singular

selbst bei wenigen Erzeugern schnell mehrere Minuten oder Stunden in Anspruch nehmen kann. Wie schon zuvor angedeutet, spielt die Wahl der Termordnung eine wesentliche Rolle bei der Laufzeit und kann diese drastisch senken [CLO00, BGK86, GMN⁺91], allerdings sind für bestimmte Probleme eben auch bestimmte Termordnungen besonders wichtig, wie im folgenden Abschnitt deutlich wird.

§2 Lösen von polynomialen Gleichungssystemen

Grundlage für das Lösen von polynomialen Gleichungssystemen mit Hilfe von Gröbnerbasen bilden eine geschickte Wahl der Termordnung sowie der Eliminationsatz. In diesem Abschnitt erweisen sich die Abkürzungen $K[X] := K[x_1, \dots, x_n]$, $K[Y] := K[y_1, \dots, y_m]$ und $K[X, Y] := K[x_1, \dots, x_n, y_1, \dots, y_m]$ als hilfreich. Diese Theorie wird ausführlich besprochen in [CLO00, Kapitel 3].

Definition 2.1 (Eliminationsordnung) Sei $S = K[X, Y] := K[x_1, \dots, x_n, y_1, \dots, y_m]$ ein Polynomring über zwei Mengen von Variablen und \prec_x und \prec_y Termordnungen auf den x - beziehungsweise y -Variablen. Die Ordnung auf den Monomen in S , die durch

$$x^{\alpha_1} y^{\beta_1} \prec x^{\alpha_2} y^{\beta_2} :\Leftrightarrow \begin{cases} x^{\alpha_1} \prec_x x^{\alpha_2} \text{ oder} \\ \alpha_1 = \alpha_2 \text{ und } y^{\beta_1} \prec_y y^{\beta_2} \end{cases}$$

definiert wird, heißt Eliminationsordnung bezüglich der y -Variablen.

Bei einer Eliminationsordnung gilt also für alle Monome $x^{\alpha_1} y^{\alpha_2}$ und y^β in $K[X, Y]$ mit $\alpha_1 \neq 0$: $x^{\alpha_1} y^{\alpha_2} \succ y^\beta$. Sie ordnet also die Gesamtheit der x -Variablen „lexikographisch“ größer als die y -Variablen. Insbesondere ist die lexikographische Termordnung in $K[x_1, \dots, x_n]$ mit $x_1 > x_2 > \dots > x_n$ für jedes k eine Eliminationsordnung bezüglich x_k, \dots, x_n . Der folgende Satz formuliert die zentrale Eigenschaft der Eliminationsordnungen:

Satz 2.2 (Eliminationsatz) Seien $I \subset K[X, Y]$ ein Ideal, \prec eine Eliminationsordnung bezüglich der y -Variablen und $G = \{g_1, \dots, g_r\}$ eine Gröbnerbasis von I . Dann ist $\{g_1, \dots, g_r\} \cap K[Y]$ eine Gröbnerbasis von $I_y := I \cap K[Y]$.

Beweis. Klar ist: $G \cap K[Y] \subset I_y$ und I_y ist ein Ideal. Sei nun ein $f \in I_y$ gegeben. Da $\{g_1, \dots, g_r\}$ Gröbnerbasis von I ist, existiert ein $j \in \{1, \dots, r\}$, so dass $\text{lm}(g_j) \mid \text{lm}(f) = y^\alpha$. Damit gilt schon $g_j \in K[Y]$. Denn angenommen $g_j \in K[X, Y] \setminus K[Y]$, so taucht in g_j ein Monom m mit positiven x_i -Potenzen auf, aber $\text{lm}(g_j)$ enthält keine x_i -Potenzen, das heißt $m \prec \text{lm}(g_j) \preceq y^\alpha$, wobei \prec eine Eliminationsordnung bezüglich den y -Variablen ist, ein Widerspruch. Es folgt $\text{in}(I_y) = \langle \text{lm}(g) \mid g \in G \cap K[Y] \rangle$, das heißt $G \cap K[Y]$ ist Gröbnerbasis von I_y . \square

Dieses Ergebnis kann man nun zum Lösen eines polynomialen Gleichungssystems nutzen, nämlich auf folgende Weise: Sei das Gleichungssystem

$$f_1 = 0, \dots, f_r = 0 \tag{2.1}$$

gegeben, mit $f_1, \dots, f_r \in K[x_1, \dots, x_n]$. Die Varietät $V(\langle f_1, \dots, f_r \rangle)$ ist genau die Lösungsmenge des Gleichungssystems. Nun sucht man mit Hilfe einer entsprechenden Eliminationsordnung nach allen Polynomen, die nur eine Teilmenge der Variablen enthalten. Nach Umbenennung sind dies die Variablen x_1, \dots, x_m und nach dem Eliminationsatz enthält die Gröbnerbasis geschnitten mit dem Ring $K[x_1, \dots, x_n]$ Erzeuger für alle solchen Polynome, etwas g_1, \dots, g_l . Für jede Lösung (a_1, \dots, a_n) des Gleichungssystems (2.1) ist (a_1, \dots, a_m) eine Lösung des Gleichungssystems

$$g_1 = 0, \dots, g_l = 0. \quad (2.2)$$

in K^m . Das heißt man kann zunächst die Lösungen des kleineren Gleichungssystems (2.2) bestimmen, um diese dann (falls möglich) zu Lösungen des ursprünglichen Gleichungssystems zu erweitern. Das folgende Beispiel illustriert dies:

Beispiel 2.3 Betrachte das Ideal $I = \langle xy - 1, xz - 1 \rangle \subset \mathbb{R}[x, y, z]$. Bezüglich der lexicographischen Termordnung mit $x > y > z$ ist die Menge $G = \{z - y, xy - 1\}$ eine Gröbnerbasis¹ von I , also erzeugt $z - y$ das Ideal $I \cap \mathbb{R}[y, z]$. Alle Teillösungen (y_0, z_0) müssen also von der Form $(y_0, z_0) = (a, a)$ sein. Setzt man dies in die Elemente von G ein, so können diese nur gemeinsam 0 werden wenn $a \neq 0$ gilt und $(x_0, y_0, z_0) = (1/a, a, a)$. Somit ist $V(\langle xy - 1, xz - 1 \rangle) = \{(1/a, a, a) \mid a \in \mathbb{R} \setminus \{0\}\}$. \diamond

Dieses Verfahren lässt sich rekursiv anwenden und im besten Falle muss man jeweils nur noch polynomiale Gleichungssysteme in einer oder zwei Variablen lösen. „Im besten Falle“ heißt, das dies nicht immer klappt. Bei ungeschickter Wahl der Variablen x_1, \dots, x_m oder bei entsprechenden Gleichungssystemen kann es vorkommen, dass die mit der Eliminationsordnung bestimmte Gröbnerbasis G keine Polynome aus $K[x_1, \dots, x_m]$ enthält. Daher ist es nahe liegend, stets Gröbnerbasen bezüglich lexicographischer Termordnungen zu berechnen, da man auf diese Weise gleich mehrere mögliche Eliminationsordnungen bearbeitet. Leider sind die lexicographischen Termordnungen aber genau jene, für die der Buchberger-Algorithmus besonders ineffizient ist [CLO00, BGK86, GMN⁺91]. Andererseits kann es dann aber auch genügen, nur diese eine Gröbnerbasis G auszurechnen, wenn zum Beispiel für jedes $k \in \{1, \dots, n\}$ der Schnitt $G \cap K[x_k, \dots, x_n]$ nicht leer ist. Dann ist das Problem reduziert auf eine Kette (beziehungsweise genauer einen Baum) von polynomialen Gleichungssystemen in jeweils nur einer Variablen: Man finde zunächst die Lösungen von $G \cap K[x_n]$, und setze diese jeweils in $G \cap K[x_{n-1}, x_n]$ ein, so dass nur noch die Variable x_{n-1} übrig bleibt. Dann können die Teillösungen für x_{n-1} bestimmt werden und zusammen mit der zuvor verwendeten Lösung von x_n in $G \cap K[x_{n-2}, x_{n-1}, x_n]$ eingesetzt werden usw.

Neben der Tatsache dass die resultierende Gröbnerbasis bei ungünstig gewählter (allgemeiner) Eliminationsordnung manchmal keine wesentliche Vereinfachung mit sich bringt, gibt es auch Ideale, in denen die Gröbnerbasis unabhängig von der Wahl der Eliminationsordnung keine der Variablen eliminiert. Das Ideal $I = \langle x_1 x_2 x_3 - 1 \rangle \subset K[x_1, x_2, x_3]$

¹Der Nachweis hiervon ist nicht ohne Weiteres möglich, es wird dazu mehr von der Theorie benötigt, die vorangegangenen Abschnitt ausgelassen wurde.

ist hierfür ein triviales Beispiel, da unabhängig von der Termordnung gilt:

$$\text{in}(I) = \langle \text{lm}(g) \mid g \in I \rangle = \langle \text{lm}(f \cdot (x_1 x_2 x_3 - 1) \mid f \in S) \rangle = \langle \text{lm}(x_1 x_2 x_3 - 1) \rangle.$$

In solchen Fällen schlägt dieses Verfahren fehl und es muss nach anderen Lösungswegen gesucht werden.

Doch selbst wenn die Elimination bestimmter Variablen funktioniert, lassen sich Teillösungen nicht immer zu Lösungen des ganzen Gleichungssystems erweitern, wie auch schon obiges Beispiel zeigt. Dies ist dem Hinzukommen singulärer Punkte bei der Einschränkung der Varietät $V(f_1, \dots, f_r)$ auf den Unterraum K^m geschuldet. Genauer beschrieben wird dies durch das folgenden Satz, auch zu finden in [CLO00, Kapitel 3.3, Theorem 3]:

Satz 2.4 *Seien K ein algebraisch abgeschlossener Körper, $I = \langle f_1, \dots, f_r \rangle \subset K[X, Y]$ ein Ideal und \prec eine Eliminationsordnung bezüglich der y -Variablen. Dann gilt für die Projektion $\phi: K^n \times K^m \rightarrow K^m, (a, b) \mapsto (b)$:*

$$V(I \cap K[Y]) = \overline{\phi(V(I))}.$$

Dabei bezeichnet $\overline{}$ wie in der algebraischen Geometrie üblich den Abschluss in der Zariski-Topologie [Hul00].

Beweis. Seien $V := V(I)$ und $I_y := I \cap K[Y]$. Sei $b \in \phi(V) \subseteq K^m$. Dann existiert ein $a \in K^n$ so dass $(a, b) \in V$. Für ein $f \in I_y$ gilt $f(a, b) = 0$, da $f \in I$. Da aber f nur y -Variablen enthält, gilt auch $f(b) = 0$. Also gilt $\phi(V) \subseteq V(I_y)$. Da $V(I_y)$ selbst eine Varietät ist, liegt auch der Abschluss $\overline{\phi(V)}$ in $V(I_y)$.

Für die andere Inklusion genügt es nach dem Hilbert'schen Nullstellensatz [Hul00, Kapitel I, Theorem 1.6] die Inklusion $I(V(I_y)) = \sqrt{I_y} \supseteq I(\overline{\phi(V)}) = I(\phi(V))$ zu zeigen. Dazu sei $f \in I(\phi(V))$, das heißt $f(b) = 0$ für alle $b \in \phi(V)$. Betrachtet man f als Polynom in $K[X, Y]$, so gilt auch $f(a, b) = 0$ für alle $(a, b) \in V$. Mit dem Hilbert'schen Nullstellensatz ist damit $f \in \sqrt{I}$, also $f^l \in I$ für ein geeignetes $l \in \mathbb{N}$. Da aber f in $K[Y]$ liegt, gilt auch $f \in \sqrt{I_y}$, was zu zeigen war. \square

Zusammengenommen heißt das, dass Gröbnerbasen zwar ein mächtiges Hilfsmittel zum Lösen von polynomialen Gleichungssystemen sind, jedoch nicht immer ausreichen. Zwar kann man auf eine starke Vereinfachung des Ausgangsgleichungssystems hoffen, doch muss man oft noch andere Techniken anwenden um in Kombination die eigentliche Frage lösen zu können.

II Ranking-Modelle

§ 3 Einführung

§ 3.1 Definition

Es werde eine feste Anzahl von n Objekten $\mathcal{O} = \{O_1, \dots, O_n\}$ durch einen nicht-deterministischen Prozess in eine Rangfolge gebracht. Das heißt, jedem Objekt werde bijektiv eine Rangzahl zwischen 1 und n zugeordnet, wobei das Objekt mit Rangzahl 1 das beste, jenes mit Rangzahl n das schlechteste sei. Zur Vereinfachung der Schreibweise wird die Menge der Objekte mit der Menge $\{1, \dots, n\}$ identifiziert, so dass jede Rangfolge als ein Element der symmetrischen Gruppe S_n dargestellt werden kann. Dabei ist wichtig zu erkennen dass jede Permutation $\pi \in S_n$ auf zwei Arten interpretiert werden kann, und dass die Art der Interpretation von großer Relevanz für die mathematische Modellierung ist. Um die Unterscheidung klar zu halten, soll π abhängig von der Interpretation eine verschiedene Bezeichnung erhalten [Mar95].

Definition 3.1 *Sei die Permutation $\pi \in S_n$ als Rangfolge gegeben. Ist $\pi(i)$ der Rang von Objekt i , so heißt π ein Ranking. Ist $\pi(i)$ das Objekt auf Rang i , so heißt π eine Ordnung.*

Permutationen aus S_n werden hier meist in Wortschreibweise angegeben, das heißt $\pi = 312$ bedeutet $\pi(1) = 3, \pi(2) = 1, \pi(3) = 2$. Ebenfalls zur Anwendung kommt die Zykelschreibweise, jedoch nur für Transpositionen. Hier ist $t = (i, j)$ diejenige Permutation die i auf j und j auf i abbildet und in allen anderen Fällen die Identität ist. Es gilt somit auch $(i, j) = (j, i)$. Die Multiplikation von Permutationen, notiert mit \circ , wird als Hintereinanderausführung von rechts nach links definiert, das heißt für Permutationen π und σ ist $(\pi \circ \sigma)(i) = \pi(\sigma(i))$.

In der Modellierung von Prozessen die in irgendeiner Weise die von Zufall bestimmt ist, erscheint die Gleichverteilung, bei der alle $n!$ möglichen Permutationen gleiche Wahrscheinlichkeit zugewiesen bekommen oft unpassend, da hierbei die Unterschiede zwischen den Objekten, wie beispielsweise die Stärke der Mannschaften beim Fussball, in keiner Weise modelliert werden können. Üblicherweise wird ein Modell also durch eine Menge von Parametern bestimmt.

Definition 3.2 *Eine Menge \mathcal{P} von Wahrscheinlichkeitsverteilungen P_r auf der symmetrischen Gruppe S_n heißt Ranking-Modell. Dabei seien die Verteilungen P_r parametrisiert über einen Parameterraum \mathcal{R} , das heißt es sei $\mathcal{P} = \{P_r \mid r \in \mathcal{R}\}$.*

Trotz der in der Literatur üblichen Bezeichnung des *Ranking*-Modells ist hier noch nicht festgelegt, ob die Permutationen als Rankings oder als Ordnungen interpretiert werden. Die Definition ist abstrakt auf der Permutationsgruppe zu verstehen und die Interpretation ihrer Elementen muss zusätzlich angegeben werden wenn man über die Eigenschaften des Modells sprechen will.

§ 3.2 Eigenschaften

Hat man nun ein Ranking-Modell gewählt, so stellt sich die Frage inwiefern das Modell „sinnvoll“ ist, das heißt ob es bestimmte Eigenschaften erfüllt, die man auf „natürliche Weise“ erwarten würde. So sollte zum Beispiel die willkürliche Nummerierung der Objekte $\{O_1, \dots, O_n\}$ mit den Zahlen $\{1, \dots, n\}$ keine Rolle spielen.

Damit die folgenden Begriffe sinnvoll sind, das heißt tatsächlich das definieren was sie bezeichnen, muss klar sein, ob die algebraisch auf S_n definierte Wahrscheinlichkeitsverteilungen Rankings oder Ordnungen als Eingabe erwarten. Durch die inverse Beziehung von Ordnungen und Rankings kann jede auf Ordnungen definierte Funktion auch als Modell auf Rankings definiert werden (und umgekehrt), indem man in der Funktionsdefinition π mit π^{-1} ersetzt. Ohne die Änderung der Funktionsdefinition entsteht im Allgemeinen ein neues, nicht-äquivalentes Modell, und es kann zum Beispiel die Label-Invarianz verloren gehen, wie auch das Plackett-Luce-Modell in Abschnitt 4.1 zeigt. Oftmals erscheint die Definition auf Ordnungen intuitiver, daher wird in den Folgenden Definitionen von Ordnungen ausgegangen. Hat man statt Ordnungen aber Rankings gegeben, so müssen in den beiden nachfolgenden Definitionen die Multiplikationsreihenfolgen von σ und π beziehungsweise γ und π umgedreht werden.

Definition 3.3 (Label-Invarianz) *Ein Ranking-Modell \mathcal{P} heißt label-invariant, falls für jeden Parameter $r \in \mathcal{R}$ und jede Umnummerierung der Objekte $\sigma \in S_n$ ein Parameter $s \in \mathcal{R}$ existiert, so dass für jede Ordnung $\pi \in S_n$ gilt:*

$$P_r^\sigma(\pi) := P_r(\sigma \circ \pi) = P_s(\pi).$$

Label-Invarianz entspricht genau der Unabhängigkeit von der willkürlichen Nummerierung der Objekte. Betrachtet man für einen Parameter $r \in \mathcal{R}$ die Verteilung die entsteht indem man vor der Berechnung der Wahrscheinlichkeit immer eine Umnummerierung der Objekte vornimmt, dann fordert die Label-Invarianz die Existenz eines Parameters $s \in \mathcal{R}$, der genau diese Verteilung als P_s hervorbringt.

Definition 3.4 (Umkehrbarkeit) *Sei $\gamma \in S_n$ die „umkehrende“ Permutation, das heißt $\gamma(j) := n + 1 - j$ für $j = 1, \dots, n$. Ein Ranking-Modell \mathcal{P} heißt umkehrbar, falls für jeden Parameter $r \in \mathcal{R}$ ein Parameter $s \in \mathcal{R}$ existiert, so dass für jede Ordnung $\pi \in S_n$ gilt:*

$$P_r^{-1}(\pi) := P_r(\pi \circ \gamma) = P_s(\pi).$$

Umkehrbarkeit bedeutet, dass es unerheblich ist, ob man bei einer Rangfolge von Objekten diese von „gut“ nach „schlecht“ oder umgekehrt angeordnet hat, das heißt ob die Begriffe „gut“ und „schlecht“ austauschbar sind.

§ 3.3 Abstandsmaße auf der symmetrischen Gruppe

Bei der Untersuchung von statistischen Samples auf Eigenschaften die zur Existenz von eindeutigen Maximum Likelihood Schätzern in dem jeweiligen Modell führen (siehe Kapitel III), ist es nützlich, Maße des Unterschieds zwischen auftretenden Permutationen zu haben. Es gibt eine Menge verschiedene mögliche Maße, Critchlow, Flinger und Verducci listen in ihrem Übersichtsartikel „Probability Models on Rankings“ [CFV91] eine Reihe davon auf. Die beiden in dieser Masterarbeit verwendeten Metriken sind die *Cayley-Metrik* und *Kendalls τ -Metrik*:

Definition 3.5 Seien $\pi, \sigma \in S_n$. Die Abbildung

$$C(\pi, \sigma) := \begin{array}{l} \text{minimale Anzahl von nötigen Transpositionen} \\ t_1, \dots, t_r, \text{ so dass } \pi \circ t_1 \circ \dots \circ t_r = \sigma \end{array}$$

heißt *Cayley-Metrik*. Die Abbildung

$$\tau(\pi, \sigma) := \begin{array}{l} \text{min. Anzahl nötiger Transpositionen } t_1, \dots, t_r \\ \text{der Form } (i, i + 1), \text{ so dass } \pi \circ t_1 \circ \dots \circ t_r = \sigma \end{array}$$

heißt *Kendalls τ -Metrik*.

Beide Metriken können als Abstände in einem *Cayley-Graphen* der Gruppe S_n gesehen werden. Dieser Graph ist, ganz allgemein zu einer Gruppe G und Erzeugendensystem F von G , ein Graph mit der Knotenmenge $V = G$. Dabei existiert eine Kante von g nach h genau dann, wenn für ein Element f aus der Erzeugermenge gilt $gf = h$ gilt. Je nach Anwendungsfall kann man hier den gerichteten oder ungerichteten Graphen betrachten. Gilt jedoch für jedes $f \in F$ dass $f^{-1} \in F$, dann spielt dies keine Rolle da dann eine Kante von g nach h existiert genau dann, wenn es eine Kante von h nach g gibt.

Die Menge der Transpositionen in S_n , $F = \{(i, j) \in S_n \mid 1 \leq i < j \leq n\}$, ist ein kanonisches Erzeugendensystem für S_n . Der Abstand im Cayley-Graphen von S_n bezüglich F ist der Abstand in der Cayley-Metrik. Auch die Menge $H = \{(i, i + 1) \in S_n \mid 1 \leq i < n\}$ ist ein Erzeugendensystem für S_n , der Abstand im Cayley-Graphen von S_n bezüglich dieser Erzeugermenge ist der Abstand in Kendalls τ -Metrik. Beide Erzeugendensysteme sind abgeschlossen unter Bildung von Inversen, das heißt dass jede Kante in dem entsprechenden Cayley-Graphen in beiden möglichen Richtungen vorliegt.

Durch die Definition der Abbildungen C und τ als Abstand in einem (quasi ungerichteten) Graphen, sind die Metrik-Eigenschaften unmittelbar klar:

Lemma 3.6 *Kendalls τ und Cayley sind Metriken, das heißt für beliebige $\pi, \sigma \in S_n$ und $d \in \{C, \tau\}$ gilt:*

- (i) $d(\pi, \sigma) = 0 \Rightarrow \pi = \sigma$ (Definitheit),
- (ii) $d(\pi, \sigma) = d(\sigma, \pi)$ (Symmetrie),
- (iii) für jedes $\rho \in S_n$ gilt: $d(\pi, \sigma) \leq d(\pi, \rho) + d(\rho, \sigma)$ (Dreiecks-Ungleichung).

Analog zur Label-Invarianz der Wahrscheinlichkeitsmodelle sollten auch die Abstandsmaße invariant sein unter Umnummerierung der Objekte. Auch hier hängt wieder alles davon ab, ob Rankings oder Ordnungen betrachtet werden. Im Fall von Ordnungen sollte eine Metrik d linksinvariant sein, das heißt für $\pi, \sigma, \rho \in S_n$ sollte $d(\rho \circ \pi, \rho \circ \sigma) = d(\pi, \sigma)$ gelten. Analog muss im Fall von Rankings Rechtsinvarianz gefordert werden, das heißt für $\pi, \sigma, \rho \in S_n$ gilt $d(\pi \circ \rho, \sigma \circ \rho) = d(\pi, \sigma)$.

Ferner kann für die Umkehrbarkeit die entsprechende Invarianz der Metrik unter der Permutation $\gamma = n \cdots 1$ gefordert werden

Lemma 3.7 *Die Cayley-Metrik ist sowohl rechts- als auch linksinvariant. Kendalls τ -Metrik ist linksinvariant und im Allgemeinen nicht rechtsinvariant. Bezüglich γ ist Kendalls τ auch rechtsinvariant.*

Beweis. Die Linksinvarianz beider Metriken ist leicht einzusehen durch die Äquivalenz

$$\begin{aligned} \pi &= \sigma \circ t_1 \circ \cdots \circ t_r \\ \Leftrightarrow \rho \circ \pi &= \rho \circ \sigma \circ t_1 \circ \cdots \circ t_r. \end{aligned}$$

Zur Rechtsinvarianz der Cayley-Metrik betrachte eine Transposition (i, j) und eine Permutation ρ . Es gilt $(i, j) \circ \rho = \rho \circ (\rho^{-1}(i), \rho^{-1}(j))$. Es gilt also allgemein

$$\begin{aligned} \pi &= \sigma \circ t_1 \circ \cdots \circ t_r \\ \Leftrightarrow \pi \circ \rho &= \sigma \circ t_1 \circ \cdots \circ t_r \circ \rho \\ \Leftrightarrow \pi \circ \rho &= \sigma \circ \rho \circ t'_1 \circ \cdots \circ t'_r. \end{aligned}$$

Dabei sind t'_1, \dots, t'_r die entsprechend transformierten Transpositionen und dies liefert die Rechtsinvarianz. Diesen Beweis kann man für Kendalls τ -Metrik so nicht führen, da $(\rho^{-1}(i), \rho^{-1}(i+1))$ im Allgemeinen nicht die Form $(j, j+1)$ hat. Für $\rho = \gamma = n \cdots 1$ ist dies allerdings wahr, wodurch die obige Äquivalenz der Beweis für die Rechtsinvarianz von Kendalls τ bezüglich γ ist. Als Gegenbeispiel für die Rechtsinvarianz im Allgemeinen betrachte für $n = 3$ die Permutationen $\pi = 312, \sigma = 213$, dann gilt $312 \circ (1, 2)(2, 3)(1, 2) = 213$ und die Transformation kann nicht mit weniger Transpositionen der Form $(i, i+1)$ durchgeführt werden, das heißt $\tau(\pi, \sigma) = 3$. Für $\rho = 213$ ist aber $\pi \circ \rho = 132$ sowie $\sigma \circ \rho = 123$ und es gilt $132 \circ (2, 3) = 123$, das heißt $\tau(\pi \circ \rho, \sigma \circ \rho) = 1$. \square

Dadurch dass Kendalls τ -Metrik nicht rechtsinvariant ist, kann man diese Metrik für Untersuchungen bezüglich Rankings zunächst nicht gebrauchen, sondern nur für Ordnungen. Dieses Problem kann man allerdings leicht beheben, indem man statt der obigen Definition die minimale Anzahl von nötigen Transpositionen von π^{-1} nach σ^{-1} zählt. Dann dreht sich alles um: Linksinvarianz gilt im Allgemeinen nicht mehr, Rechtsinvarianz schon [Dia88].

Eine andere, leichter zu implementierende Definition für Kendalls τ -Metrik findet man in [CFV91], dort wird definiert:

$$\tau(\pi, \sigma) := \sum_{1 \leq i < j \leq n} I\{(\pi(i) - \pi(j)) \cdot (\sigma(i) - \sigma(j)) < 0\}. \quad (3.1)$$

Dabei ist I die Indikatorfunktion, welche 1 ist, falls die Ungleichung erfüllt ist und sonst 0.

Lemma 3.8 *Die Definition der Funktion τ in Formel (3.1) ist äquivalent zur Definition 3.5 der Kendalls τ -Metrik mit den jeweils inversen Permutationen.*

Beweis. Setzt man in der Definition 3.5 die Inversen Permutationen ein, so ist der Abstand die minimale Anzahl von Transpositionen der Form $(i, i + 1)$, die π^{-1} in σ^{-1} überführen. Gilt also $\pi^{-1} \circ t_1 \circ \dots \circ t_r = \sigma^{-1}$, dann gilt $\sigma \circ \pi^{-1} \circ t_1 \circ \dots \circ t_r = \text{id}$. Somit ist der Wert von τ in der rechtsinvarianten Version genau die Anzahl der Inversionen von $\sigma \circ \pi^{-1}$. Für jeden Beitrag der Indikatorfunktion in (3.1) gilt genau einer der folgenden beiden Fälle: 1. Fall: $\pi(i) < \pi(j)$, $\sigma(i) > \sigma(j)$. Dann gilt

$$\sigma(\pi^{-1}(\pi(i))) = \sigma \circ \pi^{-1}(\pi(i)) > \sigma \circ \pi^{-1}(\pi(j)) = \sigma(\pi^{-1}(\pi(j))),$$

also ist $(\pi(i), \pi(j))$ eine Inversion von $\sigma \circ \pi^{-1}$. 2. Fall: $\pi(i) > \pi(j)$, $\sigma(i) < \sigma(j)$. Dann gilt

$$\sigma(\pi^{-1}(\pi(i))) = \sigma \circ \pi^{-1}(\pi(i)) < \sigma \circ \pi^{-1}(\pi(j)) = \sigma(\pi^{-1}(\pi(j))),$$

also ist $(\pi(j), \pi(i))$ eine Inversion von $\sigma \circ \pi^{-1}$.

Gilt andererseits $\pi(i) < \pi(j)$ und $(\pi(i), \pi(j))$ ist eine Inversion von $\sigma \circ \pi^{-1}$, dann gilt $\sigma(i) = \sigma \circ \pi^{-1}(\pi(i)) > \sigma \circ \pi^{-1}(\pi(j)) = \sigma(j)$, also liefert die Indikatorfunktion bei diesem Paar (i, j) einen Beitrag. Analog liefert die Indikatorfunktion auch einen Beitrag falls $\pi(j) < \pi(i)$ und $(\pi(j), \pi(i))$ eine Inversion von $\sigma \circ \pi^{-1}$ ist.

Also ist die Anzahl der Inversionen von $\sigma \circ \pi^{-1}$ genau die Anzahl der Beiträge der Indikatorfunktion in der Definition (3.1). \square

In Anwendungen in Kapitel III wird stets mit Ordnungen gerechnet, das heißt die linksinvariante Version von Kendalls τ -Metrik wird dort benötigt. Für die Implementierung wurde folglich die Formel (3.1) mit den jeweils Inversen Permutationen verwendet, siehe Anhang auf Seite 60.

§ 4 Einige Modelle

In diesem Abschnitt werden einige Ranking-Modelle vorgestellt und bezüglich der vorgeannten Eigenschaften untersucht. Konkret eingeführt werden das Plackett-Luce-Modell, das Bradley-Terry-Mallows-Modell, welches eine Spezialisierung des Babington-Smith-Modells darstellt, sowie das Inversions-Modell. Kriterium für die Auswahl der Modelle war vor allem die Eigenschaft, dass die Wahrscheinlichkeiten der Permutationen rationale

Ausdrücke in den Modellparametern sind, damit sie in Kapitel III mit Hilfe von Gröbnerbasen untersucht werden können. Im letzten Teil diesen Abschnitts wird noch ein Ausblick auf weitere mögliche und relevante Ranking-Modelle gegeben, dessen Wahrscheinlichkeiten im Allgemeinen keine rationalen Ausdrücke sind und daher im Zusammenhang mit dieser Masterarbeit keine wesentliche Rolle spielen.

§ 4.1 Plackett-Luce

Definition 4.1 Für den Parameterraum $\mathcal{R}_{PL} := \{x = (x_1, \dots, x_n) \in \mathbb{R}_+^n\}$ der Vektoren von n positiven reellen Zahlen heißt die Menge der Abbildungen $P_x : S_n \rightarrow [0, 1]$ mit

$$P_x(\pi) := \prod_{i=1}^n \frac{x_{\pi(i)}}{x_{\pi(i)} + x_{\pi(i+1)} + \dots + x_{\pi(n)}}$$

Plackett-Luce-Modell.

Lemma 4.2 Für jedes $x \in \mathcal{R}_{PL}$ ist P_x eine Wahrscheinlichkeitsverteilung auf S_n . Das Plackett-Luce-Modell $\mathcal{P}_{PL} := \{P_x \mid x = (x_1, \dots, x_n) \in \mathbb{R}_+^n\}$ ist label-invariant und nicht umkehrbar.

Beweis. Da die Parameter x_i positive reelle Zahlen sind, sind im Produkt der Funktionsdefinition alle Faktoren positiv und alle Zähler kleiner oder gleich dem Nenner. Daher ist P_x wohldefiniert. Der Beweis dass die Summe der Einzelwahrscheinlichkeiten 1 ist geht per Induktion über n . Der Induktionsanfang für $n = 1$ ist trivial. Für den Induktionsschritt definiere $S_{n,k} := \{\pi \in S_n \mid \pi(1) = k\}$. Dann gilt für jedes $x \in \mathcal{R}_{PL}$:

$$\begin{aligned} \sum_{\pi \in S_{n+1}} P_x(\pi) &= \sum_{\pi \in S_{n+1}} \frac{x_{\pi(1)} \cdots x_{\pi(n+1)}}{\prod_{i=1}^{n+1} (x_{\pi(i)} + \dots + x_{\pi(n+1)})} \\ &= \sum_{k=1}^{n+1} \sum_{\pi \in S_{n+1,k}} \frac{x_1 \cdots x_{n+1}}{\prod_{i=1}^{n+1} (x_{\pi(i)} + \dots + x_{\pi(n+1)})} \\ &= \sum_{k=1}^{n+1} \left(\frac{x_k}{x_1 + \dots + x_{n+1}} \sum_{\pi \in S_{n+1,k}} \frac{x_1 \cdots x_{k-1} \cdot x_{k+1} \cdots x_{n+1}}{\prod_{i=2}^{n+1} (x_{\pi(i)} + \dots + x_{\pi(n+1)})} \right). \end{aligned} \quad (4.2)$$

In der ersten Umformung wird ausgenutzt das in der Menge der Indizes $\pi(1), \dots, \pi(n+1)$ jeder Index genau einmal auftaucht, da π eine Permutation ist. Ferner wird die Summe mit Hilfe der zuvor definierten Menge $S_{n,k}$ aufgeteilt. Im zweiten Schritt wird ausgeklammert, so dass in der inneren Summe des letzten Ausdrucks x_k nach Definition von $S_{n,k}$ nicht auftaucht. Ersetzt man in diesem Ausdruck x_i mit x_{i-1} für $i > k$, so ist er per Induktion 1. Damit ist auch schon die äussere Summe 1 und P_x eine Wahrscheinlichkeitsverteilung.

Zur Label-Invarianz betrachte eine Umbenennung $\sigma \in S_n$. Betrachte den Parameter $y := (y_1, \dots, y_n)$, wobei $y_i := x_{\sigma(i)}$, $1 \leq i \leq n$. Dann gilt für jedes Ranking $\pi \in S_n$:

$$\begin{aligned} P_x^\sigma(\pi) &= P_x(\sigma \circ \pi) = \prod_{i=1}^n \frac{x_{\sigma(\pi(i))}}{x_{\sigma(\pi(i))} + \dots + x_{\sigma(\pi(n))}} \\ &= \prod_{i=1}^n \frac{y_{\pi(i)}}{y_{\pi(i)} + \dots + y_{\pi(n)}} = P_y(\pi), \end{aligned}$$

also ist das Plackett-Luce-Modell label-invariant.

Bezüglich der Umkehrbarkeit betrachte das folgende Gegenbeispiel: Sei $n = 3$ und $x = (2, 3, 5)$, dann wird ein $y = (y_1, y_2, y_3)$ gesucht so dass für alle $\pi \in S_n$ gilt:

$$P_x(\pi \circ \gamma) = P_y(\pi) = \prod_{i=1}^n \frac{y_{\pi(i)}}{y_{\pi(i)} + \dots + y_{\pi(n)}}. \quad (4.3)$$

Dies liefert, bei Einsetzen der 6 möglichen Permutationen und Berechnung der Wahrscheinlichkeiten der linken Seite der Gleichung (4.3), das folgende Gleichungssystem:

$$\begin{aligned} 3(y_1 + y_2 + y_3)(y_1 + y_2) - 35y_1y_3 &= 0, \\ 3(y_1 + y_2 + y_3)(y_1 + y_2) - 40y_2y_3 &= 0, \\ (y_1 + y_2 + y_3)(y_1 + y_3) - 5y_1y_2 &= 0, \\ (y_1 + y_2 + y_3)(y_1 + y_3) - 8y_2y_3 &= 0, \\ 3(y_1 + y_2 + y_3)(y_2 + y_3) - 10y_1y_2 &= 0, \\ 3(y_1 + y_2 + y_3)(y_2 + y_3) - 14y_1y_3 &= 0. \end{aligned}$$

Dies kann man lösen mit Hilfe von Gröbnerbasen, siehe Abschnitt 2. Mit lexikographischer Termordnung bezüglich $y_1 > y_2 > y_3$ ist die Menge

$$G = \{y_3^3, y_3(15y_2 + 7y_3), 225y_2^2 + 287y_3^3, y_3(15y_1 + 8y_3), 75y_1y_2 + 56y_3^2, 75y_1^2 + 184y_3^2\}$$

eine Gröbnerbasis des Ideals I , das von den Polynomen des obigen Gleichungssystems erzeugt wird. Hier kann man sukzessive $y_3 = y_2 = y_1 = 0$ ablesen, welches kein zulässiger Parameter ist. \square

Die Parameter x_i im Plackett-Luce-Modell können nach Belieben skaliert werden, ohne dass sich die Wahrscheinlichkeiten für die Elemente in S_n ändern: für jede reelle Zahl $r \neq 0$ gilt:

$$\begin{aligned} P_{r \cdot x}(\sigma) &= \prod_{i=1}^n \frac{r \cdot x_{\pi^{-1}(i)}}{r \cdot x_{\pi^{-1}(i)} + \dots + r \cdot x_{\pi^{-1}(n)}} \\ &= \prod_{i=1}^n \frac{x_{\pi^{-1}(i)}}{x_{\pi^{-1}(i)} + \dots + x_{\pi^{-1}(n)}} \end{aligned}$$

Dies erlaubt verschiedene Interpretationen der Zahlen x_i und ist daher für die Herleitung von Motivationen nützlich. Hierzu sind im Wesentlichen zwei zu nennen: Plackett-Luce als mehrstufiges Auswahlexperiment nach Plackett oder die Entwicklung aus einem allgemeineren Modell mit bestimmten Axiomen nach Luce.

Bevor diese Herleitungen angeführt werden soll allerdings anhand eines Beispiels verdeutlicht werden, wie die Parameter x_i als Bewertung für das i -te Objekt verstanden werden können:

Beispiel 4.3 Betrachte als Beispiel drei Mannschaften der Fußball-Bundesliga, die jeweils 2 Spiele gegeneinander gespielt haben, wovon keines unentschieden ausgegangen ist. Es habe Mannschaft 1, Bayern München, drei der vier Spiele gewonnen, Mannschaft 2, Wolfsburg, zwei der vier Spiele und Mannschaft 3, der 1. FC Köln, nur eines von vier Spielen. Das Gewinnverhältnis jeder Mannschaft kann als Bewertung verwendet werden und liefert für das Plackett-Luce-Modell nach Normierung den Parametervektor $x = (x_1, x_2, x_3) = (3, 2, 1)$. Damit ergeben sich in diesem Modell die folgenden Wahrscheinlichkeiten:

$$\begin{aligned} P_x(123) &= 1/3, & P_x(132) &= 1/6, & P_x(213) &= 1/4, \\ P_x(231) &= 1/12, & P_x(312) &= 1/10, & P_x(321) &= 1/15 \end{aligned}$$

Erwartungsgemäß verhalten sich die Wahrscheinlichkeiten für die mögliche Ranglisten am Ende eines Turniers: Die Wahrscheinlichkeit das Bayern München das Turnier gewinnt, Wolfsburg Zweiter und der 1. FC Köln dritter wird, ist mit $P_x(123) \approx 0.334$ am höchsten, während die genau umgekehrte Rangfolge nur eine geringe Wahrscheinlichkeit mit $P_x(321) \approx 0.067$ hat. \diamond

Plackett-Luce als mehrstufiges Auswahlexperiment

Plackett motivierte das Modell über Pferderennen [Pla75]. Dabei seien die Parameter p_i jeweils die Wahrscheinlichkeit, dass Pferd i den ersten Platz erreicht. Ferner nahm er an, dass wenn ein Pferd, beispielsweise Pferd 1, bereits ausgeschieden ist — sei es weil es schon im Ziel ist oder aus anderen Gründen — die Gewinnwahrscheinlichkeiten der verbliebenen Pferde mit $\frac{p_2}{1-p_1}, \dots, \frac{p_n}{1-p_1}$ gegeben sind. Diese genügen, um stufenweise die Wahrscheinlichkeit eines Rankings zu bestimmen, und zwar auf folgende Weise: Die Wahrscheinlichkeit, dass Pferd i den ersten Platz erreicht ist $p_i = \frac{p_i}{p_1 + \dots + p_n}$. Nun muss das beste Pferd unter den verbliebenen bestimmt werden, die Wahrscheinlichkeit das dies Pferd j ist, ist $\frac{p_j}{\sum_{k \neq i} p_k}$. Setzt man dies sukzessive fort, so erhält man genau die Formel aus Definition 4.1.

Eine alternative Sichtweise auf das mehrstufige Auswahlexperiment von Plackett-Luce ist die Folgende: Es sei eine Vase mit unendlich vielen Kugeln gegeben, wobei jede Kugel mit genau einem der n Objekte assoziiert wird und die Verhältnisse der Kugeln entsprechend den Verhältnissen der Parameter p_i sind. Eine Ordnung werde nun in n Stufen erstellt: Zunächst wird eine Kugel aus der Vase gezogen und das zugehörige Objekt bekommt den ersten Platz zugewiesen. Im zweiten Schritt wird erneut eine Kugel gezogen,

dessen Objekt den zweiten Platz erhält. Sollte in diesem zweiten Schritt eine Kugel zugehörig zum ersten Objekt gezogen werden, so wird dies schlicht ignoriert und es wird erneut gezogen, so lange bis eine Kugel verschieden vom ersten Objekt gezogen wird. Auf diese Weise werden alle n Stufen bearbeitet und das Ergebnis ist eine vollständige Ordnung der n Objekte wobei die Wahrscheinlichkeit dieser Ordnung genau mit der Formel in Definition 4.1 übereinstimmt.

Plackett-Luce als Äquivalent zu Lucas Auswahlaxiom

Luce kommt auf andere Weise zum selben Modell [Mar95, Luc59]. Er untersuchte die mathematische Modellierung von Auswahlverhalten im Hinblick auf die Psychologie und ging dabei von einem allgemeineren Modell aus: Gegeben eine Menge von Objekten $\mathcal{O} = \{1, \dots, n\}$, dann enthält eine Menge von *Auswahlwahrscheinlichkeiten* Wahrscheinlichkeitsverteilungen $P_{\mathcal{A}}$ für alle Teilmengen $\mathcal{A} \subset \mathcal{O}$, wobei für $a \in \mathcal{A}$ der Wert $P_{\mathcal{A}}(a)$ interpretiert wird als die Wahrscheinlichkeit, dass a als das präferierte Objekt der Menge \mathcal{A} ausgewählt wird. Ganz ohne Beschränkungen wären hier nicht zusammen passende Werte möglich, wenn zum Beispiel ein Element für $\mathcal{A} = \mathcal{O}$ klar bevorzugt wird, aber im direkten Vergleich mit einzelnen anderen, also für $\#\mathcal{A} = 2$, selten oder mit Wahrscheinlichkeit 0 ausgewählt wird. Damit die Wahrscheinlichkeiten auf den verschiedenen Mengen zusammenpassen formulierte Luce zwei Axiome, in der Literatur bekannt als *Lucas Auswahlaxiom* [Mal57, §5.13.1], [Luc59, Kapitel 1.C.1]:

Definition 4.4 *Eine Menge von Auswahlwahrscheinlichkeiten erfüllt Lucas Auswahlaxiom, falls für alle Teilmengen $\mathcal{A} \subset \mathcal{O}$ gilt:*

- (i) *Falls $P_{\{a,b\}}(a) > 0$ für alle $a, b \in \mathcal{A}$, dann gilt für jedes $c \in \mathcal{B} \subset \mathcal{A}$: $P_{\mathcal{A}}(c) = P_{\mathcal{B}}(c)P_{\mathcal{A}}(\mathcal{B})$.*
- (ii) *Sind $a, b \in \mathcal{A}$ mit der Eigenschaft, dass $P_{\{a,b\}}(a) = 0$ gilt, dann gilt auch schon für jedes $\mathcal{B} \subset \mathcal{O}$: $P_{\mathcal{A}}(\mathcal{B}) = P_{\mathcal{A} \setminus \{a\}}(\mathcal{B} \setminus \{a\})$.*

Die Idee des ersten Teils ist, dass die Wahrscheinlichkeit, dass c der Favorit in der Menge \mathcal{A} ist, gleich dem Produkt der Wahrscheinlichkeiten sein soll, dass der Favorit in $\mathcal{B} \subset \mathcal{A}$ liegt und dass c der Favorit in \mathcal{B} ist. Der zweite Teil sagt aus, dass ein Element a , welches mit Wahrscheinlichkeit 0 gegenüber einem anderen Element b präferiert wird, für den Auswahlprozess in einer Menge, in der b enthalten ist, keine Relevanz hat. Letzteres hat eine unmittelbare Konsequenz [Luc59, Kapitel 1.C.1]:

Lemma 4.5 *Eine Menge von Auswahlwahrscheinlichkeiten $P_{\mathcal{A}}$ erfülle das Auswahlaxiom. Existiert zu $a \in \mathcal{O}$ ein $b \in \mathcal{O}$ mit $P_{\{a,b\}}(a) = 0$, dann gilt schon $P_{\mathcal{O}}(a) = 0$.*

Beweis. Durch einfache Anwendung des zweiten Teils des Auswahlaxioms gilt:

$$P_{\mathcal{O}}(a) = P_{\mathcal{O}}(\{a\}) = P_{\mathcal{O} \setminus \{a\}}(\emptyset) = 0$$

□

Wird also ein Element a mit Wahrscheinlichkeit 0 gegenüber einem anderen Element b bevorzugt, so wird a auch mit Wahrscheinlichkeit 0 der Favorit jeder Menge sein, in der b enthalten ist. Wird nun wie bei Plackett ein Ranking in mehreren Stufen erstellt, so kann ein solches Objekt a für den Auswahlprozess vernachlässigt werden, so lange b noch nicht ausgewählt wurde. In einem solchen Rankingprozess kann man sich also immer auf eine Menge von Objekten beschränken bei denen keine solchen paarweisen eindeutigen Favoriten existieren.

Der folgende Satz stellt den Zusammenhang zwischen Luces Auswahlaxiom und dem Plackett-Luce-Modell her [Luc59]. Die zusätzliche Bedingung nicht-existenter perfekter Favoriten in Teil (i) kann allerdings nach obiger Argumentation durch wiederholte Anwendung von Lemma 4.5 vernachlässigt werden.

Satz 4.6 *Für eine Menge von Auswahlwahrscheinlichkeiten $P_{\mathcal{A}}$ sind äquivalent:*

- (i) *Es gilt $P_{\{a,b\}}(a) > 0$ für alle $a, b \in \mathcal{O}$ und die Auswahlwahrscheinlichkeiten erfüllen Luces Auswahlaxiom.*
- (ii) *Es existieren positive reelle Zahlen $p_a, a \in \mathcal{O}$, so dass für jede Teilmenge $\mathcal{A} \subset \mathcal{O}$ gilt: $P_{\mathcal{A}}(a) = p_a / \sum_{b \in \mathcal{A}} p_b$.*

Beweis. (i) \Rightarrow (ii): Umstellen der ersten Gleichung des Auswahlaxioms für $a \in \mathcal{A} \subset \mathcal{O}$ liefert $\frac{P_{\mathcal{O}}(a)}{P_{\mathcal{O}}(\mathcal{A})} = P_{\mathcal{A}}(a)$. Für alle $a \in \mathcal{O}$ sei $p_a := P_{\mathcal{O}}(a)$, dann gilt nun

$$P_{\mathcal{A}}(a) = \frac{P_{\mathcal{O}}(a)}{P_{\mathcal{O}}(\mathcal{A})} = \frac{p_a}{\sum_{b \in \mathcal{A}} p_b}.$$

(ii) \Rightarrow (i): Da die Zahlen p_a positiv sind, gilt dies auch für jede Wahrscheinlichkeit $P_{\{a,b\}}(a) = \frac{p_a}{p_a + p_b}$. Es ist also nur der erste Teil des Auswahlaxioms zu zeigen. Sei $c \in \mathcal{B} \subset \mathcal{A}$. Dann muss gelten

$$\begin{aligned} P_{\mathcal{A}}(c) &= P_{\mathcal{B}}(c)P_{\mathcal{A}}(\mathcal{B}) \\ \Leftrightarrow \frac{p_c}{\sum_{x \in \mathcal{A}} p_x} &= \frac{p_c}{\sum_{x \in \mathcal{B}} p_x} \frac{\sum_{x \in \mathcal{B}} p_x}{\sum_{x \in \mathcal{A}} p_x}. \end{aligned}$$

Kürzen auf der rechten Seite der Gleichung vervollständigt den Beweis. □

False Plackett-Luce

Wie bereits zu Anfang in Abschnitt 3.2 angedeutet, kann man ein neues, nicht-äquivalentes Ranking-Modell erhalten, indem man Rankings statt Ordnungen als Eingabe wählt oder in der Definition π mit π^{-1} ersetzt. Auf den ersten Blick sieht dies nicht viel anders aus, doch die zentrale Eigenschaft der Label-Invarianz geht verloren. Dafür ist das neue Modell allerdings umkehrbar und die Beweise hierzu sind genau die obigen Beweise für die jeweils

andere Eigenschaft im „richtigen“ Plackett-Luce-Modell, siehe Lemma 4.2. Aus praktischer Sicht erscheint dieses Modell aufgrund der fehlenden Label-Invarianz allerdings ungeeignet, da dadurch die Existenz passender Parameter von einer willkürlichen Nummerierung der Objekte abhängig ist. Die schöne Eigenschaft des Plackett-Luce-Modells, dass die Parameter x_i Bewertungen für die Objekte repräsentieren geht hier ebenfalls verloren, vielmehr ist der Parameter x_i nun eine Bewertung des Ranges i , wofür eine sinnvolle Interpretation schwierig erscheint.

Es ist wichtig in jedem Zusammenhang zu wissen, ob gerade von Rankings oder Ordnungen die Rede ist. Die verschiedenen Autoren haben verschiedene Präferenzen und daher findet man oft verschiedene Definitionen für das gleiche Modell oder vermeintliche Fehler bei der Verwendung der Begriffe. Selbiges gilt auch für andere Modelle oder beispielsweise die in Abschnitt 3.3 definierten Abstandsmaße.

§ 4.2 Bradley-Terry-Mallows

Ein alternativer Ansatz zur Untersuchung von Auswahlpräferenzen betrachtet statt ganzer Rankings nur paarweise Vergleiche der zur Auswahl stehenden Objekte. Ein Datensatz besteht dann aus der Angabe des jeweils präferierten Objekts zu jedem der $\binom{n}{2}$ möglichen Paare. Ist ein Ranking gegeben, so kann man diese paarweisen Präferenzen aus dem Ranking ableiten, der umgekehrte Weg ist allerdings nicht immer möglich. Sind beispielsweise für $\mathcal{O} = \{A, B, C\}$ die Präferenzen A gegenüber B , B gegenüber C und C gegenüber A gegeben, so existiert kein Ranking dass diese drei Präferenzen abbildet.

Eine Reihe von Artikeln beschäftigen sich mit Modellen paarweiser Präferenzen („paired comparison models“), wobei diese Präferenzen in der Regel keinen Beschränkungen unterworfen sind, das heißt zirkuläre Präferenzen wie im obigen Beispiel sind erlaubt [BT52, Dav88]. Eine Sichtweise um aus paarweise Präferenzen ein Ranking-Modell zu erhalten ist die, dass eine Menge von paarweisen Präferenzen, die zirkuläre Präferenzen enthält, schlicht verworfen wird und das Experiment so lange wiederholt wird, bis es eine konsistente Menge von Präferenzen hervorbringt. Diese Sichtweise wird *Babington-Smith*-Modell genannt [CFV91, Mar95].

In diesem Modell gibt es für $i < j$ Wahrscheinlichkeiten p_{ij} , dass das Objekt i gegenüber dem Objekt j präferiert wird. Für $i > j$ gilt dann $p_{ij} = 1 - p_{ji}$. Bringen die $\binom{n}{2}$ Realisierungen dieser zufälligen Präferenzen ein eindeutiges Ranking hervor, so ist dies das Ergebnis des Rankingprozesses. Sind die realisierten paarweisen Präferenzen inkonsistent so werden alle paarweise Vergleiche wiederholt bis ein eindeutiges Ranking feststeht.

Wie bei den anderen Modellen zuvor sei ein Element der symmetrischen Gruppe π interpretiert als Ordnung, das heißt $\pi(i)$ sei das Objekt auf Rang i . Bei festgelegten Wahrscheinlichkeiten p_{ij} ist die Wahrscheinlichkeit, dass eine Menge von paarweisen Präferenzen die Ordnung σ hervorbringt gegeben durch

$$\prod_{i < j} p_{\sigma(i)\sigma(j)}.$$

Das heißt, die Wahrscheinlichkeit dass die paarweisen Präferenzen irgendein Ranking eindeutig hervorbringen, also konsistent sind, ist gegeben durch

$$c(p) := \sum_{\sigma \in S_n} \prod_{i < j} p_{\sigma(i)\sigma(j)}.$$

Die Wahrscheinlichkeit für eine Ordnung $\pi \in S_n$ unter der Bedingung dass die Präferenzen konsistent sind ist dann nach der Formel für die bedingte Wahrscheinlichkeit

$$P_p(\pi \mid \text{Konsistenz}) = \frac{P_p(\pi \text{ und Konsistenz})}{P_p(\text{Konsistenz})} = \frac{\prod_{i < j} p_{\pi(i)\pi(j)}}{c(p)}.$$

Bringen die paarweisen Präferenzen π hervor so sind sie auch implizit konsistent, daher ist der Wert des Zählers schlicht die Wahrscheinlichkeit dafür dass die Präferenzen π hervorbringen.

Durch die vielen Parameter ist dieses Modell recht unhandlich, weshalb Mallows vorschlug [Mal57], die Parameter des Bradley-Terry-Modells für paarweise Vergleiche einzusetzen. Diese gehen von positiven Parametern v_1, \dots, v_n aus, wobei die Zahl v_i mit dem Objekt i assoziiert wird und setzen $p_{ij} := \frac{v_i}{v_i + v_j}$. Dies reduziert die Zahl der Parameter von $\binom{n}{2}$ auf n und vereinfacht das Modell wesentlich:

$$\begin{aligned} P_p(\pi \mid \text{Konsistenz}) &= \frac{1}{c(p)} \prod_{i < j} p_{\pi(i)\pi(j)} \\ &= \frac{1}{\sum_{\sigma \in S_n} \prod_{i < j} \frac{v_{\sigma(i)}}{v_{\sigma(i)} + v_{\sigma(j)}}} \prod_{i < j} \frac{v_{\pi(i)}}{v_{\pi(i)} + v_{\pi(j)}} \\ &= \frac{1}{\sum_{\sigma \in S_n} \prod_{i < j} v_{\sigma(i)}} \prod_{i < j} v_{\pi(i)} \\ &= \frac{1}{\sum_{\sigma \in S_n} \prod_{i=1}^n v_{\sigma(i)}^{n-i}} \underbrace{\prod_{i=1}^n v_{\pi(i)}^{n-i}}_{=: f(\pi, v)} \end{aligned} \tag{4.4}$$

Die erste Umformung ist schlicht das Einsetzen der Parameter $p_{ij} = \frac{v_i}{v_i + v_j}$. Im nächsten Schritt kann für alle Parameter $i < j$ der Bruch $\frac{1}{v_i + v_j}$ aus den beiden Produkten gekürzt werden: Für jede Permutation σ taucht jede Kombination von Indizes i, j in den Nennern des Produktes $\prod_{i < j} \frac{1}{v_{\sigma(i)} + v_{\sigma(j)}}$ genau einmal auf, das heißt es gilt:

$$\prod_{i < j} \frac{1}{v_{\sigma(i)} + v_{\sigma(j)}} = \prod_{i < j} \frac{1}{v_i + v_j}.$$

Ferner gilt $\prod_{i < j} v_{\sigma(i)} = \prod_{i=1}^n v_{\sigma(i)}^{n-i}$, da jedes v_i genau so oft im Produkt auftaucht wie Objekt i besser platziert wird als ein anderes Objekt. Das Objekt auf Platz eins ist

besser platziert alle Objekte auf den Plätzen 2 bis n , das Objekt auf Platz zwei ist besser platziert als alle Objekte auf den Plätzen 3 bis n usw.

Definition 4.7 Sei für $v \in \mathcal{R}_{BTM} := \mathbb{R}_+^n$ die Abbildung $P_v: S_n \rightarrow [0, 1]$ definiert durch

$$P_v(\pi) := \frac{1}{c^*(v)} f(\pi, v),$$

mit f definiert wie in der Formel (4.4) sowie $c^*(v) := \sum_{\sigma \in S_n} f(\sigma, v)$, so heißt die Menge $\mathcal{P}_{BTM} := \{P_v \mid v \in \mathbb{R}_+^n\}$ Bradley-Terry-Mallows-Modell.

Analog zum Plackett-Luce-Modell ändert sich die Wahrscheinlichkeit der Rankings nicht, wenn man alle Parameter um den gleichen Faktor skaliert. Daher ist dieses Modell durch $n - 1$ Parameter bestimmt. Mallows schlug noch weitere Vereinfachungen vor, indem die p_{ij} eine durch nur zwei Parameter beschriebene Form annehmen [Mar95, Mal57]. Diese sind dann jedoch keine rationalen Funktionen in den Parametern mehr, weshalb die Betrachtung dieser Spezialisierung in diesem Zusammenhang keinen Sinn macht.

Genau wie beim Plackett-Luce-Modell sind die Parameter v_i jeweils die Bewertung des Objektes i .

Beispiel 4.8 Man betrachte erneut die Situation der Fussball Bundesliga aus Beispiel 4.3 und nehme den gleichen Parameter $v = (v_1, v_2, v_3) = (3, 2, 1)$ für das Bradley-Terry-Mallows-Modell. Nun sind die Wahrscheinlichkeiten

$$\begin{aligned} P_v(123) &= 3/8, & P_v(132) &= 3/16, & P_v(213) &= 1/4, \\ P_v(231) &= 1/12, & P_v(312) &= 1/16, & P_v(321) &= 1/24. \end{aligned}$$

Wieder hat die Ordnung 123 die höchste Wahrscheinlichkeit mit $P_v(123) = 0.375$, die hier allerdings etwas höher ist als beim Plackett-Luce-Modell, die Ordnung 321 hat mit $P_v(321) \approx 0.0142$ die geringste Wahrscheinlichkeit, die kleiner ist als beim Plackett-Luce-Modell. Den selben Unterschieden zwischen den Parametern wird also im Bradley-Terry-Mallows-Modell mehr Bedeutung zugemessen als beim Plackett-Luce-Modell. Allerdings geben die beiden Modelle den Permutationen im Allgemeinen nicht die gleiche Rangfolge: Im Bradley-Terry-Mallows-Modell gilt $P_v(231) = 1/12 > 1/16 = P_v(321)$, während im Plackett-Luce-Modell gilt: $P_x(231) = 1/12 < 1/10 = P_x(321)$. \diamond

Lemma 4.9 Das Bradley-Terry-Mallows-Modell ist label-invariant und umkehrbar.

Beweis. Es sei $v = (v_1, \dots, v_n) \in \mathbb{R}_+^n$ gegeben, sowie $\sigma \in S_n$. Definiere für den Nachweis der Label-Invarianz $w_i := v_{\sigma(i)}$. Dann gilt für jedes $\pi \in S_n$:

$$P_v(\sigma \circ \pi) = \frac{1}{c^*(v)} \prod_{i=1}^n v_{\sigma \circ \pi(i)}^{n-i} = \frac{1}{c^*(w)} \prod_{i=1}^n w_{\pi(i)}^{n-i} = P_w(\pi).$$

Dabei geht ein, dass $c^*(v) = c^*(w)$ gilt, da eine Permutation der Parameter v_i nur einem Indexshift in der Summe über alle Permutationen entspricht. Daher ist das Bradley-Terry-Mallows-Modell label-invariant.

Für die Umkehrbarkeit ist zu zeigen dass ein Parameter $w \in \mathcal{R}_{BTM}$ existiert, so dass für alle $\pi \in S_n$ und $\gamma = n \cdots 1$ die Gleichung

$$P_v(\pi \circ \gamma) = P_w(\pi) \quad (4.5)$$

gilt. Betrachte den Parameter w mit $w_i := 1/v_i$. Einsetzen der Definitionen und Multiplikation der Gleichung mit den Nennern liefert:

$$\begin{aligned} (4.5) &\iff \frac{1}{c^*(v)} \prod_{i=1}^n v_{\pi \circ \gamma(i)}^{n-i} = \frac{1}{c^*(w)} \prod_{i=1}^n w_{\pi(i)}^{n-i} \\ &\iff \frac{1}{c^*(v)} \prod_{i=1}^n v_{\pi \circ \gamma(i)}^{n-i} = \frac{1}{c^*(w)} \prod_{i=1}^n \frac{1}{v_{\pi(i)}^{n-i}} \\ &\iff \prod_{i=1}^n v_{\pi \circ \gamma(i)}^{n-i} \prod_{i=1}^n v_{\pi(i)}^{n-i} c^*(w) = c^*(v). \end{aligned} \quad (4.6)$$

Es gilt $\gamma(i) = n - i + 1$, daher gilt

$$(4.6) \iff \prod_{i=1}^n v_{\pi(i)}^{i-1} \prod_{i=1}^n v_{\pi(i)}^{n-i} c^*(w) = c^*(v). \quad (4.7)$$

Zusammenfassen der beiden Produkte auf der linken Seite der Gleichung und Einsetzen der Werte für $c^*(w)$ und $c^*(v)$ liefert

$$(4.7) \iff \prod_{i=1}^n v_i^{n-1} \sum_{\sigma \in S_n} \prod_{i=1}^n \frac{1}{v_{\sigma(i)}^{n-i}} = \sum_{\sigma \in S_n} \prod_{i=1}^n v_{\sigma(i)}^{n-i}. \quad (4.8)$$

Durch Kommutativität des Produktes gilt für jedes $\sigma \in S_n$ gilt die Gleichung

$$\prod_{i=1}^n \frac{v_i^{n-1}}{v_{\sigma(i)}^{n-i}} = \prod_{i=1}^n v_{\sigma(i)}^{i-1}.$$

Damit kann man auf der linken Seite der Gleichung (4.8) ausmultiplizieren und man erhält die Äquivalenz

$$\begin{aligned} (4.8) &\iff \sum_{\sigma \in S_n} \prod_{i=1}^n v_{\sigma(i)}^{i-1} = \sum_{\sigma \in S_n} \prod_{i=1}^n v_{\sigma(i)}^{n-i} \\ &\iff \sum_{\sigma \in S_n} \prod_{i=1}^n v_{\sigma \circ \gamma(i)}^{n-i} = \sum_{\sigma \in S_n} \prod_{i=1}^n v_{\sigma(i)}^{n-i} \end{aligned}$$

Da die Summen in der letzten Gleichung über die ganze symmetrische Gruppe laufen

steht nun auf der linken Seite nur eine Umordnung der rechten Seite, daher ist das Bradley-Terry-Mallows-Modell umkehrbar. \square

Bradley-Terry-Mallows und Plackett-Luce im Vergleich

Sowohl im Plackett-Luce-Modell als auch bei Bradley-Terry-Mallows werden Parameter x_i mit den Objekten assoziiert und ein höherer Wert diese Parameters sorgt dafür das Rankings an denen diese Objekt einen guten Platz einnimmt eine größere Wahrscheinlichkeit erhalten. Es stellt sich nun die Frage, wie sich diese beiden Modelle im Vergleich verhalten.

Die ähnliche Verwendung der Parameter ist kein Zufall. Wie bereits zu Anfang dieses Abschnitts erwähnt, können paarweise Präferenzen aus einem Ranking abgeleitet werden. Wahrscheinlichkeiten für diese Vergleiche können also durch Marginalisierungen eines Ranking-Modells erstellt werden. Dies angewendet auf das Plackett-Luce-Modell liefert genau das Bradley-Terry-Modell für paarweise Vergleiche: Es sei ein Paar (i, j) , $1 \leq i, j \leq n$, $i \neq j$, gegeben. Ohne Einschränkung der Allgemeinheit gilt nach Umbenennung $i = 1$, $j = 2$. Für $n = 2$ gibt es nur eine Permutation in der 1 besser platziert wird als 2, nämlich $\pi = 12$, welche die Wahrscheinlichkeit $\frac{x_1}{x_1+x_2}$ hat.

Ist auf dem ersten Platz das Objekt 1, dann ist die Position der anderen Elemente irrelevant da dann Objekt 2 mit Sicherheit einen schlechteren Platz als Objekt 1 erhält. Ist auf dem ersten Platz ein Objekt verschieden von 1, so ist dies eines der Objekte 3 bis n , und in der restlichen Permutation muss noch Objekt 1 einen besseren Platz als 2 erhalten. Somit gilt:

$$\sum_{\substack{\sigma \in S_n \\ \sigma^{-1}(1) < \sigma^{-1}(2)}} P_{PL}(\sigma) = \sum_{\substack{\sigma \in S_n \\ \sigma(1)=1}} P_{PL}(\sigma) + \sum_{k=3}^n \sum_{\substack{\sigma \in S_n \\ \sigma(1)=k \\ \sigma^{-1}(1) < \sigma^{-1}(2)}} P_{PL}(\sigma). \quad (4.9)$$

Die Wahrscheinlichkeit im Plackett-Luce-Modell dafür, das Objekt k erstplatziert ist und die restlichen Objekte irgendwie, ist gegeben durch $\frac{x_k}{x_1+\dots+x_n}$, wie etwa im Beweis von Lemma 4.2 (Eigenschaften des Plackett-Luce-Modells) gezeigt oder in der Motivation des Plackett-Luce-Modells als mehrstufiges Auswahlexperiment verwendet (S. 22). Damit kann obige Gleichung fortgeführt werden und mit Hilfe des Induktionsprinzips gilt:

$$\begin{aligned} (4.9) &= \frac{x_1}{x_1 + \dots + x_n} + \sum_{k=3}^n \frac{x_k}{x_1 + \dots + x_n} P_{PL} \left(\begin{array}{c} 1 \text{ wird besser platziert als } 2 \\ \text{in einem Modell mit } n-1 \text{ Objekten} \end{array} \right) \\ &= \frac{x_1}{x_1 + \dots + x_n} + \frac{x_3 + \dots + x_n}{x_1 + \dots + x_n} \cdot \frac{x_1}{x_1 + x_2} \\ &= \frac{x_1}{x_1 + x_2}. \end{aligned}$$

Das Bradley-Terry-Modell ist also die Marginalisierung des Plackett-Luce-Modells auf paarweise Vergleiche. Führt man diese Marginalisierung auf „Babington-Smith’sche“ Wei-

se auf Rankings zurück, erhält man genau das Bradley-Terry-Mallows-Modell. Wie man an den Beispielen 4.3 und 4.8 erkennen kann, ist das Ergebnis eine stärkere Bewertung der Unterschiede zwischen den Parametern im Bradley-Terry-Modell, welches Ordnungen, die näher an der Rangfolge der Parameter liegen, eine größere Wahrscheinlichkeit gibt als das Plackett-Luce-Modell und den Ordnungen, die weiter weg liegen von der Ordnung durch die Parameter, entsprechend geringere Wahrscheinlichkeiten gibt.

Interessant ist, dass durch diesen Prozess die Rangfolge der Wahrscheinlichkeiten aus dem Plackett-Luce-Modell im Allgemeinen nicht beibehalten wird, siehe Beispiel 4.8. Dies ist ein Hinweis darauf, dass der Konditionale Ansatz des Babington-Smith-Modells ein unnatürlicher ist. Das Auftauchen von inkonsistenten paarweisen Vergleichen, die in diesem Modell schlicht „weg konditioniert“ werden, ist nämlich im Allgemeinen zu erwarten, etwa durch natürliche Variationen im Bewertungsprozess oder wenn zum Beispiel die zu bewertenden Objekt gar nicht auf einer linearen Skala angeordnet werden können, man denke zum Beispiel an das Vergleichen von Bewerbern auf eine Arbeitsstelle. Die Qualifikationen der Bewerber sind in vielen verschiedenen Dimensionen zu messen und sind daher kaum auf einer linearen Skala darstellbar. Dies stellten selbst Kendall und Babington Smith schon zu Beginn ihrer Betrachtung von Modellen zu paarweisen Präferenzen fest [KBS40].

§ 4.3 Inversions-Modell

Definition 4.10 Sei zu Parametern $r \in \mathcal{R}_{Inv} := \{(u_{ij}, v_{ij})_{1 \leq i < j \leq n} \in \mathbb{R}_+^{n(n-1)}\}$ die Abbildung $P_r : S_n \rightarrow \mathbb{R}_+$ definiert durch

$$P_r(\pi) := \prod_{\substack{i < j \\ \pi^{-1}(i) < \pi^{-1}(j)}} u_{ij} \cdot \prod_{\substack{i < j \\ \pi^{-1}(i) > \pi^{-1}(j)}} v_{ij}$$

Das Inversions-Modell ist die Menge $\mathcal{P}_{Inv} := \{P_r \mid r \in \mathcal{R}_{Inv}, \sum_{\pi \in S_n} P_r(\pi) = 1\}$ [SW11].

Es ist an manchen Stellen nützlich, die Parameter in Form einer Matrix aufzuschreiben. Die Diagonaleinträge einer $n \times n$ -Matrix A haben keine Bedeutung und für $i < j$ sei $a_{ij} = u_{ij}$ sowie $a_{ji} = v_{ij}$. Ein Parameter $r = (u_{ij}, v_{ij})$ kann dann für $n = 3$ geschrieben werden als

$$r = \begin{pmatrix} \cdot & u_{12} & u_{13} \\ v_{12} & \cdot & u_{23} \\ v_{13} & v_{23} & \cdot \end{pmatrix}.$$

Anders als bei Plackett-Luce und Bradley-Terry-Mallows können die Parameter nicht völlig frei gewählt werden, da die Summe der Einzelwahrscheinlichkeiten 1 sein muss. Dies ist allerdings keine schwere Voraussetzung, denn summieren sich die Werte $P_r(\pi)$ nicht zu 1, so können sie als Wahrscheinlichkeitsverhältnisse interpretiert werden und es existiert ein anderer Parameter der durch Skalierung genau diese Verhältnisse als Wahrscheinlichkeiten hervorbringt:

Lemma 4.11 Sei $r = (u_{ij}, v_{ij}) \in \mathcal{R}_{Inv}$ ein beliebiger Parameter und $c := \sum_{\pi \in S_n} P_r(\pi)$.

Dann existiert ein Parameter $s = (u'_{ij}, v'_{ij}) \in \mathcal{R}_{Inv}$ so dass für alle $\pi \in S_n$ gilt: $P_s(\pi) = \frac{1}{c} P_r(\pi)$.

Beweis. Wähle für $\{i, j\} \neq \{1, 2\}$ $u'_{ij} = u_{ij}$ und $v'_{ij} = v_{ij}$ und setze $u'_{12} = \frac{u_{12}}{c}$ sowie $v'_{12} = \frac{v_{12}}{c}$. Da für jedes $\pi \in S_n$ in $P_s(\pi)$ genau einer der Faktoren u'_{12} und v'_{12} ein mal auftaucht (und der andere nicht), taucht der Faktor $\frac{1}{c}$ in jeder Einzelwahrscheinlichkeit genau einmal auf, also gilt:

$$\sum_{\pi \in S_n} P_s(\pi) = \frac{1}{c} \sum_{\pi \in S_n} P_r(\pi) = 1$$

□

Die Wahl von $(i, j) = (1, 2)$ ist in diesem Beweis völlig willkürlich, ebenso hätte jedes andere Paar von Indizes (i, j) gewählt werden können. Bei genauerer Betrachtung sieht man hier sofort eine Menge Redundanz die in dem Modell steckt: Skaliert man ein beliebiges Paar (u_{ij}, v_{ij}) um den Faktor $c > 0$ und ein beliebiges (anderes) Paar (u_{kl}, v_{kl}) um den Faktor $\frac{1}{c}$, so ändern sich die Einzelwahrscheinlichkeiten der Permutationen nicht, da sich die beiden Faktoren aufheben. Man kann also alle Paare bis auf (zum Beispiel) (u_{12}, v_{12}) so skalieren, dass $v_{ij} = 1$ gilt für alle $(i, j) \neq (1, 2)$. Die entsprechenden Faktoren können jeweils in dem Paar (u_{12}, v_{12}) kompensiert werden. Auf diese Weise reduziert sich das Modell von $2 \binom{n}{2}$ auf $\binom{n}{2} + 1$ Parameter.

Die obige Reduktion der Parameter deckt auch eine sinnvolle Interpretation für das Modell auf, und zwar mit den Parametern u_{ij} als „Wetteinsätze“ für „i schlägt j im Ranking“, $v_{ij} = 1$, wobei als zusätzlicher Parameter noch der Normierungsfaktor eingebracht werden muss, indem eben ein Parameterpaar entsprechen skaliert wird oder die Skalierung über alle Parameterpaare verteilt wird.

Das Inversions-Modell ist eine formale Verallgemeinerung des Babington-Smith-Modells, indem „i schlägt j“ mit u_{ij} und „j schlägt i“ mit v_{ij} bewertet wird. Die Nebenbedingung des Inversions-Modells kodiert dabei den normalisierenden Faktor des Babington-Smith-Modells. Im Bradley-Terry-Mallows-Modell, der besprochenen Spezialisierung des Babington-Smith-Modells, ist dieser Faktor mit $c(v)$ gegeben. Mit Berücksichtigung eines normalisierenden Faktors und der Möglichkeit der Skalierung der Parameter v_{ij} zu 1 wird deutlich, dass diese formale Verallgemeinerung des Babington-Smith-Modells keine wirkliche Verallgemeinerung ist, sondern dem Modell nur Redundanzen hinzufügt. Wenn in der Literatur also vom Inversions-Modell [SW11] oder vom Babington-Smith-Modell [Mar95] die Rede ist, so werden hier die gleichen Modelle betrachtet.

Beispiel 4.12 Betrachte erneut die Situation der Fußballbundesliga aus den Beispielen 4.3 und 4.8. Unter der vereinfachenden Annahme, dass die Anzahl der Tore und der Heimvorteil keine Rolle spielen, könnte man anhand dieser Ergebnisse Wetteinsätze für eine erneute Begegnung von je zwei Mannschaften kalkulieren, wobei man bei geringen Chancen einer Mannschaft natürlich eher weniger wetten wollen würde. Also berechnet man einen Multiplikator für einen festen Wetteinsatz von beispielsweise 10 Euro, indem die Gewinnverhältnisse dieser Mannschaften ins Verhältnis gesetzt werden. Das heißt

$u_{13} = \frac{3/4}{1/4} = 3$, $v_{13} = \frac{1/4}{3/4} = 1/3$, $u_{12} = \frac{3/4}{2/4} = 3/2$ usw. Spielt dann Bayern München gegen den ersten FC Köln, sind die Chancen für Bayern München eher groß, also kann man ruhig den Wetteinsatz für Bayern erhöhen, etwa auf $3 \cdot 10$ Euro, den Wetteinsatz für den Gewinn des 1. FC Köln würde man wohl eher klein halten, etwa bei $0.\bar{3} \cdot 10$ Euro.

Diese Parameter kann man nun für das Inversions-Modell verwenden, um Wahrscheinlichkeiten für die Rangliste am Ende eines vollständigen Turniers dieser Mannschaften zu berechnen. Es ergibt sich als Parameter in der oben angedeuteten Matrix-Schreibweise:

$$r = \begin{pmatrix} \cdot & u_{12} & u_{13} \\ v_{12} & \cdot & u_{23} \\ v_{13} & v_{23} & \cdot \end{pmatrix} = \begin{pmatrix} \cdot & 3/2 & 3 \\ 2/3 & \cdot & 2 \\ 1/3 & 1/2 & \cdot \end{pmatrix}$$

Verwendet man diese, um die „Wahrscheinlichkeiten“ für die Permutationen in S_3 , zu berechnen¹, so erhält man:

$$P_r(123) = 9, \quad P_r(132) = 9/4, \quad P_r(213) = 4, \\ P_r(231) = 4/9, \quad P_r(312) = 1/4, \quad P_r(321) = 1/9.$$

Dies sind natürlich keine Wahrscheinlichkeiten, da ihre Summe nicht 1 ist, sondern $289/18$, dies ist der Skalierungsfaktor c aus dem obigen Beweis. Man kann nun die Parameter u_{12} und v_{12} durch diesen Wert teilen und erhält

$$r' = \begin{pmatrix} \cdot & 27/289 & 3 \\ 12/289 & \cdot & 2 \\ 1/3 & 1/2 & \cdot \end{pmatrix}.$$

Dieser Parameter bringt nun tatsächlich eine Wahrscheinlichkeitsverteilung für S_3 hervor, nämlich

$$P_{r'}(123) = 324/578, \quad P_{r'}(132) = 81/578, \quad P_{r'}(213) = 144/578, \\ P_{r'}(231) = 16/578, \quad P_{r'}(312) = 9/578, \quad P_{r'}(321) = 4/578.$$

Diese Werte summieren sich zu eins und produzieren auch erwartungsgemäße Wahrscheinlichkeiten für die Möglichen Ranglisten. Die Wahrscheinlichkeit, dass Bayern München den ersten, Wolfsburg den zweiten und der 1. FC Köln den dritten Platz holt, ist sehr hoch, $P_{r'}(123) = 324/578 \approx 0.561$, während die genau umgekehrte Rangfolge am Ende des Turniers nur eine sehr geringe Wahrscheinlichkeit hat, $P_{r'}(321) = 4/578 \approx 0.007$.

In der Reihenfolge ihrer Wahrscheinlichkeiten sind die Permutationen nun wie im Bradley-Terry-Mallows-Modell angeordnet (siehe Beispiel 4.8), während die spezifischen Einzelwahrscheinlichkeiten wiederum ein Stück extremer sind als dort, beispielsweise liegt die Wahrscheinlichkeit der Permutation 123 nun bei über 0.5. \diamond

¹Dies geht sehr einfach mit den in Maple für Kapitel III geschriebenen Funktionen, siehe Anhang. Diese Zahlen erhält man beim Aufruf von `params := [[0,3/2,3],[2/3,0,2],[1/3,1/2,0]];`, `map(prob_inv,S(3),params);` sowie `add(i,i in %);`. Mehr Erläuterung zu den Funktionen und was sie tun findet man in Kapitel III sowie im Anhang ab Seite 57.

Lemma 4.13 *Das Inversions-Modell ist label-invariant und umkehrbar.*

Beweis. Um die Label-Invarianz zu zeigen, erweist sich die nach der Definition 4.10 des Modells beschriebene Matrix-Schreibweise als sinnvoll. Betrachte einen Parameter $r = (u_{ij})_{i \neq j}$, wobei in der Funktionsdefinition des Inversions-Modells für $i < j$ der Parameter v_{ij} mit u_{ji} ersetzt wird. Betrachte ferner eine Relabelling-Permutation $\sigma \in S_n$ und definiere den Parameter $r' = (x_{ij})_{i \neq j}$ mit $x_{ij} := u_{\sigma(i)\sigma(j)}$. Es gilt nun für jede Ordnung π , $\pi' := \pi^{-1}$:

$$P_r^\sigma(\pi) = P_r(\sigma \circ \pi) = \prod_{\substack{i < j \\ \pi^{-1} \circ \sigma^{-1}(i) < \pi^{-1} \circ \sigma^{-1}(j)}} u_{ij} \quad \prod_{\substack{i < j \\ \pi^{-1} \circ \sigma^{-1}(i) > \pi^{-1} \circ \sigma^{-1}(j)}} u_{ji} \quad (4.10)$$

$$\begin{aligned} &= \prod_{\substack{i < j \\ \pi^{-1}(i) < \pi^{-1}(j)}} u_{\sigma(i)\sigma(j)} \quad \prod_{\substack{i < j \\ \pi^{-1}(i) > \pi^{-1}(j)}} u_{\sigma(j)\sigma(i)} \quad (4.11) \\ &= P_{r'}(\pi) \end{aligned}$$

Die Gleichheit der Produkte (4.10) und (4.11) kann wie folgt eingesehen werden: Es sei $i < j$ und $\pi^{-1} \circ \sigma^{-1}(i) < \pi^{-1} \circ \sigma^{-1}(j)$. Dann taucht u_{ij} im linken Produkt in (4.10) auf. Es seien $a := \sigma^{-1}(i)$ und $b := \sigma^{-1}(j)$. Ist $a < b$, dann gilt $\pi^{-1}(a) < \pi^{-1}(b)$, also taucht $u_{\sigma(a)\sigma(b)} = u_{ij}$ auf der linken Seite in (4.11) auf. Ist $b < a$ dann gilt $\pi^{-1}(b) > \pi^{-1}(a)$ und es taucht $u_{\sigma(a)\sigma(b)} = u_{ij}$ auf der rechten Seite in (4.11) auf. Ist andererseits $i < j$ aber $\pi^{-1} \circ \sigma^{-1}(i) > \pi^{-1} \circ \sigma^{-1}(j)$, dann taucht u_{ji} im rechten Produkt in (4.10) auf und kann analog zum vorherigen Fall auch in (4.11) gefunden werden.

Zur Umkehrbarkeit betrachte die umkehrende Permutation $\gamma = n \dots 1$. Es gilt $\gamma = \gamma^{-1}$ und für $i < j$ gilt $\gamma(i) > \gamma(j)$. Betrachte nun einen Parameter $r = (u_{ij}, v_{ij})_{1 \leq i < j \leq n}$, dann schreibt sich die Wahrscheinlichkeit einer Ordnung π im umgekehrten Modell als

$$\begin{aligned} P_r(\pi \circ \gamma) &= \prod_{\substack{i < j \\ \gamma^{-1} \circ \pi^{-1}(i) < \gamma^{-1} \circ \pi^{-1}(j)}} u_{ij} \cdot \prod_{\substack{i < j \\ \gamma^{-1} \circ \pi^{-1}(i) > \gamma^{-1} \circ \pi^{-1}(j)}} v_{ij} \\ &= \prod_{\substack{i < j \\ \pi^{-1}(i) < \pi^{-1}(j)}} v_{ij} \cdot \prod_{\substack{i < j \\ \pi^{-1}(i) > \pi^{-1}(j)}} u_{ij} \end{aligned}$$

In der zweiten Zeile geht ein, dass γ genau die Größenverhältnisse jedes Paares umdreht: gilt $\pi^{-1}(i) < \pi^{-1}(j)$ dann gilt $\gamma^{-1} \circ \pi^{-1}(i) > \gamma^{-1} \circ \pi^{-1}(j)$ und umgekehrt. Für $r' = (v_{ij}, u_{ij})_{1 \leq i < j \leq n}$ gilt also $P_r^{-1} = P_{r'}$. \square

§ 4.4 Weitere Modelle

Es gibt noch eine ganze Menge weiterer Modelle, die im Zusammenhang mit Rankings von Bedeutung sind. Das Buch „Analyzing and Modelling Rank Data“ von John Marden [Mar95] und der Artikel „Probability Models on Rankings“ von Douglas E. Critchlow, Michael Flinger und Joseph Verducci [CFV91], auf denen auch Teile dieser Arbeit gestützt

sind, geben darüber einen guten Überblick. Zwei dieser Modelle sollen hier zumindest Erwähnung finden, auch wenn sie im Zusammenhang mit Maximum Likelihood Schätzern, siehe Kapitel III, nicht mit den hier vorgestellten Methoden bearbeitet werden können, da die Wahrscheinlichkeiten der Rankings keine rationale Funktionen in ihren Parametern sind. Weitergehende und wesentlich ausführlichere Betrachtungen sind in den jeweils angegebenen Referenzen zu finden.

Distanzbasierte Modelle

In Situationen, in denen eine objektiv „richtige“ Ordnung der Objekte in \mathcal{O} existiert und Datenpunkte in einer Beobachtung dieser zentralen Ordnung wahrscheinlich ähneln, machen distanzbasierte Modelle Sinn. Ein Anwendungsbeispiel hierfür könnte ein Experiment sein, in dem Probanden verschiedene akustische Stimuli nach Lautstärke sortieren sollen. Das objektive Maß der Lautstärke liefert eine zentrale Ordnung; in einem passenden Modell sollten Ordnungen geringere Wahrscheinlichkeit haben, je „verschiedener“ sie von dieser zentralen Ordnung sind.

Das folgende 1-parametrische Exponentialmodell ist üblich: zu gegebener zentraler Ordnung π_0 und Parameter $\theta \in \mathbb{R}$ sei die Wahrscheinlichkeit einer Ordnung $\pi \in S_n$ gegeben durch

$$P_\theta(\pi, \pi_0) := \frac{1}{c(\theta)} \exp(\theta d(\pi_0, \pi)),$$

wobei d eine Metrik auf der symmetrischen Gruppe und $c(\theta)$ eine normalisierende Konstante ist. Verschiedene Autoren verwenden hier verschiedene Abstandsmaße. Das vielbesprochene ϕ -Modell von Mallows verwendet Kendalls τ -Metrik [Mal57], Critchlow, Flinger und Verducci besprechen dieses und andere Modelle unter Verwendung verschiedener Metriken, unter anderem auch der Cayley-Metrik [CFV91]. Die besprochenen Invarianzen der Metriken (siehe Abschnitt 3.3) liefern automatisch die Label-Invarianz beziehungsweise Umkehrbarkeit in diesen Modellen (hier im Fall von Ordnungen):

$$P_{\theta, \sigma}(\pi, \pi_0) = P_\theta(\sigma \circ \pi, \pi_0) = \frac{1}{c(\theta)} \exp(\theta d(\pi_0, \sigma \circ \pi)) = \frac{1}{c(\theta)} \exp(\theta d(\sigma^{-1} \circ \pi_0, \pi)).$$

Im letzten Schritt geht die vorauszusetzende Linksinvarianz der Metrik d ein. Analog gilt mit Rechtsinvarianz bezüglich γ :

$$P_\theta^{-1}(\pi, \pi_0) = P_\theta(\pi \circ \gamma, \pi_0) = \frac{1}{c(\theta)} \exp(\theta d(\pi_0, \pi \circ \gamma)) = \frac{1}{c(\theta)} \exp(\theta d(\pi_0 \circ \gamma, \pi)).$$

Thurstones Ordnungsstatistik-Modelle

Die Situation des Probanden, der akustische Stimuli nach Lautstärke sortieren soll, gab Thurstone den Anlass, ein Modell zu betrachten, bei dem jedem Objekt O_1, \dots, O_n eine reelle Zufallsvariable X_i zugeordnet wird [Thu27]. Dieser betrachtete dabei allerdings nur paarweise Vergleiche und erst Daniels verallgemeinerte dies auf ein Modell für Rankings, in dem die Wahrscheinlichkeit der Ordnung π definiert wird als $P(\pi) := P(X_{\pi(1)} < \dots <$

$X_{\pi(n)}$ [Dan50].

Die Zufallsvariablen X_i können dabei beliebig und verschieden verteilt sein und zunächst sind auch beliebige Abhängigkeiten zwischen den Zufallsvariablen erlaubt. Man kann zeigen, dass jede Verteilung auf der Menge der Permutationen als ein solches Modell ausgedrückt werden kann. Fordert man die Unabhängigkeit der Zufallsvariablen X_i , so geht dies im Allgemeinen nicht mehr. Die Label-Invarianz lässt sich an der Definition des Modells ablesen indem eben die Zufallsvariablen umbenannt werden. Umkehrbarkeit gilt in dieser Klasse im Allgemeinen nicht und hängt von den gewählten Verteilungen ab [CFV91].

Verschiedene Spezialisierungen dieses Modells beschränken die Art der zulässigen Zufallsvariablen, indem beispielsweise von Normalverteilten Zufallsvariablen mit gleicher Varianz und unterschiedlichen Erwartungswerten ausgegangen wird. Die Unterklasse mit unabhängigen Zufallsvariablen X_i wird als Klasse der ordnungsstatistischen Modelle (*order statistics models*) bezeichnet.

III Existenz und Eindeutigkeit von Maximum Likelihood Schätzern in Ranking-Modellen

Hat man statistische Daten, das heißt absolute Häufigkeiten für das Auftreten aller Permutationen in einem bestimmten Beobachtungszeitraum, so ist eine Möglichkeit die Parameter eines angenommenen Modells zu schätzen die, die Wahrscheinlichkeit für das Auftreten genau dieser Daten über den Parameterraum zu maximieren. Eine solche Maximalstelle heißt, sofern existent, *Maximum Likelihood Schätzer*. Bezeichnet man mit a_π die Häufigkeit des Auftretens von π , so ist die *Likelihood-Funktion*

$$L(r) := \prod_{\pi \in S_n} P_r(\pi)^{a_\pi},$$

wobei P_r die Wahrscheinlichkeitsverteilung mit Parameter r in dem gewählten Modell ist. Die Likelihood-Funktion beschreibt also die Wahrscheinlichkeit für das Auftreten dieses Samples und die Methode der Schätzung ist es, diese Wahrscheinlichkeit zu maximieren.

Sind die Wahrscheinlichkeiten der einzelnen Permutationen rationale Ausdrücke in den Parametern, dann ist die Likelihood-Funktion selbst eine rationale Funktion. Somit ist die Bestimmung der gemeinsamen Nullstellen der Richtungsableitungen ein algebraisches Problem, das mit Hilfe von Gröbnerbasen bearbeitet werden kann.

Direkt aus der Definition der Likelihood-Funktion wird allerdings ersichtlich, dass ihr Grad in der Regel sehr groß sein wird, wenn nicht die Größe des Samples außerordentlich klein ist. Dies ist bei der praktischen Anwendung der Theorie der Gröbnerbasen problematisch, da hohe Grade in den Polynomen lange Laufzeiten des Buchberger-Algorithmus begünstigen. Bei großen Samples erweist sich die Berechnung einer Gröbnerbasis als unpraktikabel, weil sie schlicht zu lange dauert. Daher stellt sich die Frage, inwiefern schon anhand der statistischen Daten, ohne explizite Berechnung eines Maximum Likelihood Schätzers, Aussagen über Existenz und Eindeutigkeit eines Maximum Likelihood Schätzers getroffen werden können. Dies soll hier zunächst anhand von kleinen beispielhaften Fällen untersucht werden, um daraus gegebenenfalls allgemeine Aussagen ableiten und beweisen zu können.

Dieses Kapitel widmet sich diesen Untersuchungen, die man als „systematisches ausprobieren“ bezeichnen kann. Dazu werden zunächst Hilfsmittel und Vorgehensweisen in den Abschnitten 5 und 6 erklärt. Ausgangspunkt ist hier das Computeralgebrasystem Maple sowie eine Reihe von Funktionen, die speziell zum Zweck dieser Untersuchungen geschrieben wurden, dessen Funktionsweisen jeweils im Kontext erklärt werden und deren Quellcodes im Anhang ab Seite 57 zu finden sind. Im Abschnitt 7 werden

die Resultate zu den jeweiligen Modellen vorgestellt, Probleme und Grenzen der Untersuchungen erklärt und Hinweise zu möglichen weiteren Untersuchungen gegeben. Im Abschnitt 8 werden diese Ergebnisse zusammengefasst und diskutiert.

§5 Hilfsmittel

Um überhaupt sinnvolle Aussagen im Bezug auf die Eindeutigkeit von Maximum Likelihood Schätzern treffen zu können habe ich mit Hilfe von Maple eine ganze Reihe von Beispielen durchgerechnet, um rein heuristisch Vermutungen über das Verhalten der Maximalstellen aufstellen zu können. Dazu habe ich zunächst eine Reihe von Hilfsfunktionen für das Rechnen mit Permutationen in Wortschreibweise geschrieben: Produkt und Inverse von Permutationen, sowie Kendalls τ - und die Cayley-Metrik. Den Programmcode aller selbst geschriebenen Funktionen ist im Anhang abgedruckt, kann aber auch heruntergeladen werden und ist an den Rechnern des Fachbereichs 12 der Philipps-Universität Marburg direkt zugänglich, für mehr Information siehe Anhang ab Seite 57.

In weiteren Schritten habe ich Funktionen geschrieben, die für die verschiedenen Modelle die Wahrscheinlichkeiten einzelner Permutationen in Abhängigkeit bestimmter Parameter, die Likelihood-Funktion in Abhängigkeit statistischer Daten sowie zufällige oder allgemeine Parameterlisten berechnen. Auf dieser Basis war es mir zunächst möglich die Label-Invarianz sowie die Umkehrbarkeit der Modelle auf heuristischer Basis zu prüfen und anschließend allgemein zu beweisen. Die Ergebnisse dieser Arbeit wurden in dem vorangegangenen Abschnitt 4 vorgestellt.

In weiteren Schritten habe ich dann einfache Fälle von statistischen Daten erzeugt — zum Teil auf systematische Weise alle Möglichkeiten einer bestimmten Art, zum Teil willkürliche Beispiele — und versucht die zugehörigen Maximum Likelihood Schätzer zu bestimmen. Hierbei kann man sich die Label-Invarianz der vorgestellten Modelle zunutze machen: Durch entsprechendes Relabelling kann man eine beliebige Permutation in die identische Permutation überführen, während die anderen Permutationen entsprechend transformiert werden. Auf diese Weise wird die Anzahl der zu betrachtenden Fälle stark reduziert. Um nicht alle Fälle von Hand ausrechnen zu müssen, habe ich die Funktion `get_solutions` geschrieben, welche bei gegebenem Gleichungssystem weite Teile der Bestimmung der gültigen Nullstellen automatisiert übernimmt. Ausführlichere Information dazu gibt es im nachfolgenden Abschnitt, der Programmcode hiervon ist ebenfalls im Anhang zu finden.

§6 Vorgehensweise

§6.1 Genereller Ansatz

Sämtliche beispielhafte Berechnungen wurden hier nur für kleine n durchgeführt, das heißt $n < 7$. Für größere n werden einerseits die Berechnungen der Gröbnerbasen langwieriger, gleichzeitig werden die Gleichungssysteme und die Anzahl verschiedener zu betrachtender Fälle größer, wodurch diese nicht mehr sinnvoll, beziehungsweise vollständig

betrachtet werden können. Die Fälle $n = 1, 2$ dagegen sind trivial: für $n = 1$ gibt es nur eine Wahrscheinlichkeitsverteilung und für $n = 2$ ist sie durch eine einzige rationale Zahl vollständig bestimmt. Für $n = 3, 4$ wird es dann allerdings interessanter, da nun theoretisch nicht-eindeutige Maximalstellen auftreten und die Maximalstellen auch noch in angemessener Zeit effektiv bestimmt werden können.

Die Vorgehensweise ist dabei immer wie folgt: Ausgangspunkt ist ein möglicher statistischer Datensatz, das heißt ein Element $a \in \mathbb{N}_0^{n!}$. Die nicht-negative ganze Zahl a_i gibt an, wie oft die i -te Permutation in S_n beobachtet wurde, dabei wird von einer lexikographischen Sortierung der Permutationen in Wortschreibweise ausgegangen. Im Fall $n = 3$ gilt also $S_3 = \{123, 132, 213, 231, 312, 321\}$ und die Permutationen werden zum Teil auch in dieser Reihenfolge referenziert, zum Beispiel mit $\pi_3 = 213$ und $\pi_6 = 321$.

Die n partiellen Ableitungen der Likelihood-Funktion werden zu einem Ideal zusammengefasst und es wird eine Gröbnerbasis dieses Ideals ausgerechnet. Termordnung ist die lexikographische, wobei die Variablen des Modells jeweils selbst lexikographisch geordnet werden und die kleinste lexikographische Variable die größte Variable der Termordnung ist, zum Beispiel $x_1 > x_2 > \dots > x_n$.¹ Maple gibt die Polynome der Gröbnerbasis aufsteigend nach Leitmonom sortiert aus. Das erste Polynom genügt dann oft schon um vieles abzulesen, da in den besprochenen Modellen ausschließlich positive Parameter auftauchen. Ein Polynom der Form $x_1 x_2 x_3^2 (x_2 + 1)(x_3 + 1)$ zum Beispiel hat keine Nullstellen in \mathbb{R}_+^3 . Diesen Ansatz weiter verfolgend kann man dann versuchen die für die positiv parametrisierten Modelle relevanten Lösungen des Gleichungssystems zu berechnen, was aber nicht immer klappen muss und an verschiedenen Stellen scheitern kann, siehe Abschnitt 6.3. Die im folgenden Abschnitt beschriebene Prozedur `get_solutions` automatisiert den größten Teil dieser Arbeit.

§ 6.2 Algorithmische Teillösung gegebener Probleme

Da der Polynomring in endlich vielen Variablen faktoriell ist, können die Polynome der Gröbnerbasis in eindeutiger Weise faktorisiert werden und die einzelnen Faktoren der Polynome jeweils getrennt betrachtet werden. Durch Rechnung mit lexikographischer Termordnung beinhaltet das erste Polynom eine eher kleine Anzahl von Variablen, wodurch hier mögliche erste Teillösungen schon abgelesen werden können. Diese können eingesetzt werden in die anderen Polynome und auf diese Weise können sukzessive alle relevanten Lösungen des Gleichungssystems bestimmt werden, wie bereits in Abschnitt 2 besprochen. Bei der Art der auftretenden Faktoren sind 4 Fälle zu unterscheiden:

1. Fall: Der Faktor ist ein Polynom in nur einer Variablen vom Grad kleiner als 4. Dann

¹Diese Wahl hat rein praktische Gründe: hat man eine Liste von Variablen, so sortiert Maple diese mit der Funktion `sort()` lexikographisch. Gibt man allerdings eine Liste von Variablen in die Funktion `plex()` um eine lexikographische Termordnung zu generieren, dann werden die Variablen der Liste in dieser Reihenfolge als absteigend sortiert angenommen. Natürlich kann es sein (und kommt sogar oft vor), dass eine andere Ordnung der Variablen ein besseres oder schnelleres Ergebnis erzielt, doch ist dies einerseits schwer vorherzusagen und andererseits hat man hierfür wiederum $n!$ Möglichkeiten, welche gesammelt auszuprobieren nur wenig Sinn macht.

können die Lösungen für diese Variable exakt bestimmt und in die restlichen Polynome eingesetzt werden.

2. **Fall:** Der Faktor ist ein Polynom in nur einer Variablen vom Grad 4 oder höher. Dann ist eine numerische Lösung effizient möglich. Um die Fortsetzung des Fehlers durch die numerische Näherung dieser Variable so gering wie möglich zu halten ist es sinnvoll zunächst jedes auftauchen des hiermit gelösten Faktors in den restlichen Polynomen mit 0 zu ersetzen, bevor in den verbleibenden Polynomen der ungenaue Wert der betrachteten Variable eingesetzt wird.
3. **Fall:** Der Faktor enthält 2 oder mehr Variablen, hat aber in einer der Variablen einen Grad kleiner oder gleich 3. Dann kann das Polynom auf exakte Weise nach dieser Variable aufgelöst werden und diese Lösungen können in die anderen Polynome eingesetzt werden. Hierbei ist Vorsicht geboten: Die resultierenden Ausdrücke sind nämlich im Allgemeinen keine Polynome mehr, denn sie können durch die Lösung von quadratischen oder kubischen Gleichungen Wurzeln enthalten.
4. **Fall:** Der Faktor enthält 2 oder mehr Variablen und hat in jedem dieser Variablen einen Grad von 4 oder höher. In diesem Fall kann im Allgemeinen nicht exakt weitergerechnet werden und eine multivariate numerische Approximation wäre mühselig. Eine weitere automatische Rechnung ist in diesem Fall zunächst nicht möglich.

In den Fällen 1 bis 3 ist auf diese Weise die Menge der auftretenden Variablen um (mindestens) 1 reduziert worden und das reduzierte System kann nun weiter gelöst werden. Dazu wird nur erneut faktorisiert, denn die Struktur der lexikographischen Termordnung garantiert, dass auch das reduzierte System noch von tendenziell einfacher Form ist. Schritt für Schritt kann man so zumindest einige der Lösungen bestimmen, wie auch schon in Abschnitt 2 beschrieben.

Tauchen in diesem Prozess Werte für die Variablen auf, die 0 oder negativ sind, so kann man diese Teillösungen für die hier betrachteten Modelle verwerfen, da nur positive Variablenwerte relevant sind. Im 3. Fall kann man diese Gültigkeitsprüfung nicht allgemein machen. Zwar wird ein Faktor der Form $(x_4 + x_3)$ nie mit gültigen Parametern 0 werden, bei einem Faktor der Form $(x_4 + x_3 - x_3^2)$ ist das weniger klar. Es könnte beispielsweise sein das nach Einsetzen von $x_4 = x_3^2 - x_3$ in einem der folgenden Schritte $x_3 < 1$ bestimmt wird, wodurch $x_4 < 0$ gelten würde. Sogar komplexe Werte könnten so entstehen, beispielsweise wenn $x_4 = \sqrt{1 - x_3^2}$ ist. Das Auftreten solcher Fälle kann nur schwer vorausgesehen werden, weshalb in der Implementierung dieses Lösungsverfahrens im Fall 3 ohne Vorzeichenprüfung fortgefahren wird.

Die Prozedur `get_solutions` ist eine rekursive Implementierung dieses Verfahrens. Sie erwartet 3 Parameter² (in dieser Reihenfolge): Eine Liste von Polynomen, der Index der

²In der eigentlichen Implementierung sind 6 Parameter möglich, wovon die letzten drei optional sind.

Der vierte Parameter `sup_sols` gibt die bisher gefundenen Lösungen in die rekursiven Funktionsaufrufe mit, damit sie bei fehlschlagenden Rechnungen abgespeichert werden können, der Standardwert von `sup_sols` ist also die leere Liste `[]`. Der fünfte Parameter `make_coeffs_rational` ist ein boolescher Wert der bestimmt ob nach dem Einsetzen gefundener Lösungen die Koeffizienten der

Variable, die im aktuellen Schritt bestimmt werden soll, sowie eine Liste von Variablennamen. Dabei wird davon ausgegangen dass die Liste der Polynome eine Gröbnerbasis ist oder dass sie das Ergebnis des Einsetzens von Teillösungen in eine solche Liste ist. Es wird von einer lexikographischen Termordnung ausgegangen, die die angegebene Liste der Variablen absteigend sortiert, es müssen die Polynome aufsteigend nach Leitmonom sortiert vorliegen und es wird ferner davon ausgegangen dass alle Polynome dieser Liste faktorisiert sind. Rückgabewert ist jeweils eine Liste der (mit dem Algorithmus berechenbaren) Teillösungen.

Die Vorgehensweise bei der Berechnung der Lösung im aktuellen Schritt ist jeweils wie folgt: Es wird zunächst nach dem ersten Polynom in der Liste gesucht welches nicht 0 ist. Ein Faktor dieses Polynoms kann nur mit gültigen Parametern 0 werden, wenn mindestens ein „-“ Zeichen darin vorkommt. Danach wird gesucht bevor überhaupt versucht wird Lösungen zu bestimmen. Kann ein exakter oder approximierter, nicht-negativer Wert für die Lösung der angegebenen Variable in diesem Faktors bestimmt werden (obige Fälle 1 und 2), so wird dieser Faktor in der gesamten Liste der Polynome mit 0 ersetzt und erst dann der berechnete Wert der Variable eingesetzt. Die Ersetzung des gelösten Faktors durch 0 ist bei exakter Rechnung überflüssig, bei approximierter Rechnung erhöht dies allerdings die Genauigkeit, da Polynome die nun „eigentlich 0 sein müssten“ auch wirklich exakt 0 sind und die weitere Rechnung nicht unnötig mit Fehlern belasten.

Im obigen Fall 3 werden alle exakten Lösungen in die Liste der Polynome eingesetzt, da wie bereits erwähnt keine Gültigkeitsprüfung durchgeführt werden kann. Im obigen Fall 4 kann nicht ohne weiteres weitergerechnet werden. Die Berechnung einer Gröbnerbasis mit einer anderen lexikographischen Termordnung könnte gegebenenfalls ein einfacheres System von Polynomen hervorbringen. Daher wird hier nur eine Meldung ausgegeben und das bisher berechnete Ergebnis sowie der unlösbaren Faktor gespeichert. Dies erlaubt es dem Benutzer zu einem späteren Zeitpunkt an dieser Stelle weiter zu rechnen.

Für alle in den Fällen 1 bis 3 berechneten Lösungen werden die resultierenden Polynome faktorisiert und mit Hilfe eines rekursiven Aufrufs die weiteren möglichen Variablenwerte bestimmt. Alle auf diese Weise resultierenden Teillösungen werden mit der entsprechenden Lösung für die aktuelle Variable verknüpft und die hieraus resultierende Liste von (Teil-) Lösungen ist der Rückgabewert der Funktion.

Beispiel 6.1 Es sei $n = 4$ und g_1, g_2, g_3, g_4 seien die partiellen Ableitungen der Likelihood-Funktion, welche eine rationale Funktion in den Parametern x_1, x_2, x_3, x_4 sei. Die Nullstellen eines gekürzten rationalen Ausdrucks sind genau die Nullstellen des Zählers, also genügt es diese zu betrachten. Dadurch bleiben nur Polynome übrig und es kann eine Gröbnerbasis des von den verbleibenden Polynomen erzeugten Ideals ausgerechnet werden. Wählt man die lexikographische Termordnung bezüglich $x_1 > x_2 > x_3 > x_4$, so

verbleibenden Polynome in rationale Ausdrücke umgewandelt werden sollen. Dies führt dazu, dass gegebenenfalls mehr Lösungen vollständig berechnet werden können, wobei allerdings auch mehr Rechenfehler gemacht werden, siehe Abschnitt 6.3. Der sechste und letzte Parameter, v , ist nur für das Inversionsmodell von Belang und hat den Standardwert 1. Hier wird die Substitution für v in der Likelihood-Funktion mitgegeben. Es wird jeweils geprüft, ob eine gefundene Teillösung diesen Parameter nicht schon verschwinden lässt und nur andernfalls weitergerechnet. Siehe hierzu auch Abschnitt 7.3.

kann dies gemacht werden mit Hilfe des Befehls

$$\text{Basis}(\text{PolyId}(\text{map}(\text{numer}, [g_1, g_2, g_3, g_4])), \text{plex}(x_1, x_2, x_3, x_4))^3.$$

Ist das Ergebnis die Liste $[f_1, \dots, f_r]$ so ist dies eine aufsteigend nach Leitmonomen sortierte Gröbnerbasis bezüglich lexikographischer Termordnung mit $x_1 > x_2 > x_3 > x_4$. Der Funktionsaufruf

$$\text{get_solutions}(\text{factor}([f_1, \dots, f_r]), 4, [x_1, x_2, x_3, x_4])^4$$

versucht dann alle Teillösungen im gültigen Parameterraum zu bestimmen und gibt Meldungen aus für die Fälle die nicht ausgerechnet werden können.

Sind Freiheitsgrade in den Parametern im Vorhinein bekannt, wie etwa beim Plackett-Luce-Modell die beliebige Skalierung, so ist es sinnvoll diese vor Berechnung der Gröbnerbasis einzubringen um den Rechenaufwand zu minimieren. Beim Plackett-Luce-Modell kann zum Beispiel der Wert $x_1 = 1$ a priori festgelegt werden, das heißt statt

$$\text{Basis}(\text{PolyId}(\text{map}(\text{numer}, [g_1, g_2, g_3, g_4])), \text{plex}(x_1, x_2, x_3, x_4)).$$

würde man

$$\text{Basis}(\text{PolyId}(\text{subs}(x_1=1, \text{map}(\text{numer}, [g_2, g_3, g_4])), \text{plex}(x_2, x_3, x_4)))^5.$$

aufrufen. ◇

§ 6.3 Beschränkungen und Probleme durch Rechenfehler

Neben der bereits besprochenen Tatsache dass die Berechnung von Gröbnerbasen zum Teil exorbitant lange dauern kann, gibt es eine Reihe weiterer Probleme, die bei der im Abschnitt 6.2 besprochenen algorithmischen Vorgehensweise eine Rolle spielen. Zunächst können Faktoren auftreten, die, wie im obigen Fall 4 besprochen, mehr als eine Variable enthalten und von zu hohem Grad sind. Dann können ohne weiteres schlicht keine Lösungen berechnet werden, das heißt die Lösung des Gleichungssystems scheitert an dieser Stelle.

Ein weiteres Problem ist die Faktorisierung der Polynome sobald ein approximativ bestimmter Wert eingesetzt wurde. Dann kann Maple die Polynome in mehr als einer

³Die Funktion `Basis` ruft den Algorithmus zur Berechnung von Gröbnerbasen auf, `PolyId` ist eine Abkürzung für die Funktion `PolynomialIdeal` im Paket `PolynomialIdeals`, welche aus einer Liste von Polynomen eine Datenstruktur für Polynom-Ideale erstellt. Diese Datenstruktur kann die Laufzeit des in Maple implementierten Buchberger-Algorithmus positiv beeinflussen. `map` ist eine Funktion um eine andere Funktion auf alle Elemente einer Liste anzuwenden, `numer` holt den Nenner aus einem rationalen Ausdruck und `plex` ist das Kommando zur Erzeugung einer lexikographischen Termordnung. Vergleiche hierzu die Dokumentation von Maple [MGH⁺05].

⁴Die Funktion `factor` faktorisiert, falls möglich, alle Elemente der übergebenen Liste [MGH⁺05].

⁵Der Befehl `subs(g1, g2, ..., gn, arg)` substituiert in `arg` sukzessive die linken Seiten der Gleichungen `g1, ..., gn` mit den jeweils rechten Seiten [MGH⁺05].

Variablen mit Fließkommakoeffizienten nicht faktorisieren. Eine solche Faktorisierung ist im Allgemeinen schwierig, da Polynome durch Rechenfehler irreduzibel werden können, die eigentlich reduzibel sind. Unter Einbeziehung einer gewissen Ungenauigkeit der Koeffizienten ist jedes Polynom in mehreren Veränderlichen irreduzibel, da der reduzible Fall der singuläre Fall ist. Um solche Polynome trotzdem zu faktorisieren ist etwas mehr Mühe nötig, wie etwa beschrieben in [GKM⁺04], doch ein solcher Algorithmus ist in Maple offenbar nicht implementiert, auch wenn man dazu keinen Hinweis in der Dokumentation von Maple findet [MGH⁺05]. Das Problem kann umschifft werden indem die Fließkommateile der Koeffizienten vor der Faktorisierung in rationale Ausdrücke umgewandelt mit Hilfe der Funktion `convert(.,rational)`, was die fehlerbehafteten Werte auf künstliche Weise „exakt“ macht, die existenten Rechenfehler aber möglicherweise vervielfältigt.

So lange mit exakten Werten und Lösungen gerechnet wird, hat die Konversion in rationale Ausdrücke keinerlei Einfluss auf die Rechnung, da der Befehl `convert(.,rational)` exakte (irrationale) Ausdrücke unberührt lässt, das heißt Werte wie $0.5 \cdot \sqrt{2}$ in $1/2 \cdot \sqrt{2}$ umgewandelt, und nicht etwa in $47321/66922$, was eine Rationalisierung der Approximation des gesamten Ausdrucks wäre. Bei approximativer Rechnung können sich allerdings die Fehler durch die Rationalisierung fortsetzen und gegebenenfalls verstärken, wodurch die Lösungen auch stark fehlerbehaftet sein können.

Die schwierige und gegebenenfalls approximierete Faktorisierung ist nicht nur ein Problem bei der vorgestellten Methode der Berechnung von Nullstellen. Auch die Ersetzung von Faktoren „die jetzt 0 sein müssen“ durch 0 wird im Allgemeinen fehlschlagen da Faktoren durch Rechenfehler nicht ganz gleich aussehen können, obwohl sie bei exakter Rechnung möglicherweise gleich wären. Auf diese Weise vervielfältigen sich die Fehler der Näherung mehr als nötig. Zudem werden auf diese Weise gegebenenfalls Faktoren in die Rechnung einbezogen die bei exakter Rechnung eigentlich schon längst hätten verschwinden müssen und nun durch ihre Existenz eigentlich korrekte Lösungen des Gesamtsystems verschwinden lassen:

Beispiel 6.2 Es bestehe eine Gröbnerbasis mit lexikographischer Termordnung bezüglich $x_3 > x_2 > x_1$ aus 4 Polynomen, $G = \{g_1, g_2, g_3, g_4\} \subset \mathbb{R}[x_1, x_2, x_3]$ und es gelte $G \cap \mathbb{R}[x_1] = \{g_1\}$, $G \cap \mathbb{R}[x_1, x_2] = \{g_1, g_2\}$. Sei (a_1, a_2) eine gemeinsame exakte Nullstelle von g_1 und g_2 und es gelte $g_3(a_1, a_2, x_3) = 0$, für jedes $x_3 \in \mathbb{R}$. Jede Lösung a_3 des Polynoms $g_4(a_1, a_2, x_3)$ würde hier eine Lösung (a_1, a_2, a_3) hervorbringen. Wird aber (a_1, a_2) nur näherungsweise bestimmt, etwa mit der Approximation $(\tilde{a}_1, \tilde{a}_2)$, dann gilt mit großer Wahrscheinlichkeit $g_3(\tilde{a}_1, \tilde{a}_2, x_3) \neq 0$, so sind nur die *gemeinsamen* Nullstellen von $g_3(\tilde{a}_1, \tilde{a}_2, x_3)$ und $g_4(\tilde{a}_1, \tilde{a}_2, x_3)$ gültige Erweiterungen. Taucht aber kein gemeinsamer Faktor in $g_3(\tilde{a}_1, \tilde{a}_2, x_3)$ und $g_4(\tilde{a}_1, \tilde{a}_2, x_3)$ auf, was durch approximative Rechnung zu erwarten ist, so gibt es keine Erweiterung von $(\tilde{a}_1, \tilde{a}_2)$ und es ist offenbar (mindestens) eine Lösung verloren gegangen. \diamond

Das dies kein rein theoretisches Beispiel ist, sondern tatsächlich passiert, kann anhand des Bradley-Terry-Mallows-Modells eingesehen werden. Am Ende des Abschnitts 7.2 wird hierauf noch einmal eingegangen.

Zusammenfassend ist festzustellen, dass näherungsweise berechnete Lösungen stark fehlerbehaftet sind und leider nicht einmal darüber Aufschluss geben können *ob* Lösungen

existieren. Gleichzeitig ist die approximative Rechnung kaum zu umgehen: Bei den im nachfolgenden Abschnitt besprochenen Fällen der einzelnen Modelle ist näherungsweise Rechnung bei Plackett-Luce für $n = 5$, bei Bradley-Terry-Mallows bereits für $n = 3$ nötig.

Ein weiteres Problem kann sogar bei exakter Rechnung auftreten, und zwar falls im Laufe der Rechnung ein Faktor in mehreren Unbekannten vom Grad 2 oder 3 gelöst werden muss. Betrachte zum Beispiel den Faktor $x^3 + 7xy^2 - 32x^2y - 12y^3$, der homogen vom Grad 3 in den Variablen x und y ist. Löst man diesen Ausdruck mit Hilfe von Maple nach x auf, erhält man 3 sehr komplizierte Ausdrücke für die Lösungen, wovon zwei echt komplexe Werte sind, falls y nicht 0 ist. Diese beiden Lösungen kann man eigentlich fallen lassen, doch lässt sich im Allgemeinen die Schlussfolgerung „Es ist die imaginäre Einheit im Ausdruck enthalten, also ist dies auf jeden Fall eine ungültige Lösung“ gar nicht ziehen. Es könnte beispielsweise die Lösung von einer so speziellen Form sein, dass die imaginären Teile auch für positiv Werte von y verschwinden. Da man diesen Spezialfall nicht einfach ausser Acht lassen kann, muss man für jede der drei Lösungen alle möglichen Erweiterungen rekursiv berechnen obwohl in vielen Fällen bei genauerer Betrachtung zwei der drei Lösungen verworfen werden könnten. Dies belastet die automatische Berechnung der Beispielfälle mit unnötig viel Rechenarbeit, wodurch diese möglicherweise wesentlich länger dauert als nötig. Für größere n reproduziert sich dieses Problem natürlich durch die rekursive Struktur der Lösungsmethode.

§ 6.4 Hinreichendes Kriterium für Extremstellen

In der gesamten Diskussion der Berechnung der Maxima wird die Betrachtung des hinreichenden Kriteriums für mögliche Maximalstellen ausgelassen. Dies hat zwei Gründe: Zum Einen ist der erste Teil, das Finden von kritischen Punkten, mit Sicherheit der relevantere, da in der praktischen Anwendung die Art dieser Punkte schneller experimentell überprüft werden kann. Zum Anderen ist die Berechnung der Eigenwerte der Hesse-Matrix im Allgemeinen ein sehr schwieriges Problem, erst recht für große n . Gleichzeitig zeigt die Untersuchung der im Plackett-Luce-Modell betrachteten Beispiele für $n = 3$ und $n = 4$, dass alle kritischen Punkte in diesem Modell tatsächlich Maximalstellen sind. Dies legt die Vermutung nahe, dass die kritischen Punkte immer Maximalstellen sind (oder zumindest oft), was als Heuristik für die Anwendung zunächst genügen kann. Eine weitergehende Untersuchung wäre zwar wünschenswert, kann aber aufgrund ihrer Komplexität in diesem Kontext nicht ausführlich besprochen werden. Analoges gilt für die anderen Modelle.

§ 7 Ergebnisse

Im folgenden werden die in Abschnitt 4 vorgestellten Modelle näher untersucht. Die Rede ist dabei stets abstrakt von Permutationen, die aber in diesem Kontext stets als Ordnungen und nicht als Rankings interpretiert werden, analog zur Definition der jeweiligen Modelle.

§ 7.1 Plackett-Luce

Die einfachsten Fälle statistischer Daten sind solche bei denen genau eine oder genau zwei verschiedene Permutationen genau einmal auftauchen. Mit dieser geringen Samplegröße bleiben die Polynome in den zu lösenden Gleichungssystemen relativ klein, wodurch die kritischen Punkte der Likelihood-Funktion auch wirklich berechnet werden können. Durch Label-Invarianz ist jeweils nur ein Fall beziehungsweise nur $n! - 1$ Fälle zu betrachten. Ferner kann durch Skalierung der Parameter angenommen werden, dass jeweils $x_1 = 1$ gilt.

Tritt nur eine einzelne Permutation im Sample auf, so ist diese ohne Einschränkung der Allgemeinheit $\pi = 1 \dots n$. Für $n = 3$ ist die Likelihood-Funktion $L = \frac{x_1 x_2}{(x_1 + x_2 + x_3)(x_2 + x_3)}$ und die Zähler der partiellen Ableitungen von L sind

$$\{x_2(x_2 + x_3), x_1(-x_2^2 + x_1 x_3 + x_3^2), -x_1 x_2(2x_2 + 2x_3 + x_1)\}.$$

Ohne dass man hier überhaupt eine Gröbnerbasis ausrechnen muss, kann man ablesen dass keine Lösungen mit ausschließlich positiven Parametern existieren, da das Polynom $x_2(x_2 + x_3)$ nur 0 sein kann falls $x_2 = 0$ oder $x_2 = -x_3$ gilt. Dieses Ergebnis lässt sich auch intuitiv erklären: Durch festhalten des Parameters x_2 und gleichzeitiger Vergrößerung von x_1 oder Verkleinerung von x_3 steigt die Wahrscheinlichkeit des Rankings 123. Somit kann man den Wert der Likelihood-Funktion beliebig vergrößern während die degenerierte Wahrscheinlichkeitsverteilung mit $P(123) = 1$ und $P(\pi) = 0$ für $\pi \neq 123$ keine Verteilung im Plackett-Luce-Modell ist. Die Fälle $n > 3$ verhalten sich hier völlig analog. In diesem Fall existiert also kein Maximum Likelihood Schätzer.

Auch im Fall in dem genau zwei Permutationen auftreten ist ohne Einschränkung eine der beiden auftauchenden Permutationen $1 \dots n$. Das einmalige Auftauchen genau einer anderen Permutation resultiert in $n! - 1$ Fälle, die sich für $n = 3$ und $n = 4$ in genau zwei Möglichkeiten aufteilen: Entweder existiert kein Maximum Likelihood Schätzer, oder er ist eindeutig (siehe Tabellen 7.1 und 7.2). Dabei ist durch Skalierung der Parameter ohne Einschränkung $x_1 = 1$, wodurch die Rechnungen kürzer werden.

Die Prozedur `testrun_pl` automatisiert diese Rechnungen weitestgehend mit Hilfe der im Abschnitt 6.2 beschriebenen Prozedur `get_solutions`, in dem sie alle Fälle einzeln betrachtet, die Likelihood-Funktion und ihre Ableitungen berechnet und die Lösungen bestimmt. Für $n = 3$ sind die Lösungen in Tabelle 7.1, für $n = 4$ in Tabelle 7.2 dargestellt. Alle Rechnungen können in diesen Fällen exakt durchgeführt werden, auch wenn in der Tabelle 7.2 die meisten Werte für eine kompaktere Darstellung approximiert wurden. Ferner sind einige der exakt und kompakt darstellbaren Lösungen skaliert, so dass etwa gemeinsame Nenner eliminiert werden. In den Tabellen sind zudem die Abstände zwischen den beiden Permutationen bezüglich der Cayley-Metrik und Kendalls τ -Metrik angegeben.

Die Idee hinter der Betrachtung des Abstandes der beiden auftauchenden Permutationen ist, dass für den Maximum Likelihood Schätzer „genügend verschiedene Information“ existieren muss, damit die n Objekte hinreichend gut unterschieden werden können. Dies scheint sich am Beispiel $n = 3$ zu bestätigen: Im ersten Fall sind die Objekte 2 und 3

π_1	π_2	Lösungen	Cayley	Kendalls τ
123	132	keine	1	1
123	213	keine	1	1
123	231	$x_1 = 4, x_2 = 7 + \sqrt{17}, x_3 = -1 + \sqrt{17}$	2	2
123	312	$x_1 = 8, x_2 = -6 + 2\sqrt{17}, x_3 = 7 - \sqrt{17}$	2	2
123	321	$x_1 = 1, x_2 = \sqrt{2}, x_3 = 1$	1	3

Tabelle 7.1: Lösungen im Plackett-Luce-Modell für den Fall $n = 3$ bei einmaligem Auftreten von jeweils genau zwei Permutationen.

ununterscheidbar, im zweiten Fall die Objekte 1 und 2, während kein Maximum gefunden werden kann. In den anderen drei Fällen sind die Abstände größer und die Maximum Likelihood Schätzer existieren. Die gefundenen Parameter sind in ihren Größenverhältnissen auch nachvollziehbar, zum Beispiel im Fall $\pi_2 = 231$: Objekt 2 muss mit Sicherheit am besten bewertet werden, da es die besten Ränge erzielt. Objekt 1 ist zwar nicht immer besser als Objekt 3, aber erzielt einmal den 1. und einmal den 3. Platz, während Objekt 3 nur den 2. und 3. Platz bekommt. Daher muss der Parameter von Objekt 1 größer sein als der von Objekt 3.

Im letzten Fall allerdings sind die Objekte 1 und 3 wiederum ununterscheidbar — dennoch wird ein Maximum Likelihood Schätzer gefunden. Hier ist aber anders als in den ersten beiden Fällen der Abstand in Kendalls τ -Metrik verschieden von 1.

Insgesamt kann die Existenz von Lösungen in Zusammenhang gebracht werden mit der Summe der beiden verwendeten Metriken: In den Fällen in denen keine Lösung existiert ist die Summe der Abstände kleiner oder gleich zwei, in den anderen Fällen ist die Summe größer oder gleich 4.

Der Zusammenhang zwischen Existenz von Lösungen und Abstand der Permutationen bestätigt sich für $n = 4$. Die Ergebnisse für diesen Fall sind in Tabelle 7.2 angegeben. Alle Lösungen werden durch `get_solutions` exakt bestimmt, die meisten sind hier nur näherungsweise angegeben um die Darstellung zu vereinfachen. Die Fälle, in denen Lösungen existieren sind immer solche in denen die Abstände eher groß sind. Sind die Abstände eher klein, existiert keine Lösung. Der relative Begriff „klein“ hat für größere n natürlich eine andere Bedeutung. Während für $n = 3$ im letzten Fall mit den Abständen 1 und 3 eine Lösung existiert, gibt es für $n = 4$ in den Fällen mit diesen Abständen keine Lösung. Ist der Abstand in der Cayley-Metrik 1, so muss hier offenbar Kendalls τ -Metrik einen Wert größer als 3 haben damit eine Lösung existiert, wie etwa im Fall $\pi_2 = 4231$.

Auch hier funktioniert die Unterscheidung anhand der Summe beider Maße: Existiert keine Lösung ist die Summe kleiner oder gleich 4, sonst größer oder gleich 6.

Der Fall $n = 5$ ist mit `testrun_p1(5)` zu aufwändig und ist im Laufe der Arbeit an einem Rechner des Fachbereichs nie fertig geworden. Genauer schien die Berechnung der Lösungen im Fall $\pi_2 = 24315$ besonders ausufernd zu sein. Interessanterweise scheint dies nicht etwa an der Berechnung der Gröbnerbasen, sondern an der Rechnung mit

π_1	π_2	Lösungen	Cayley	Kendalls τ
1234	1243	keine	1	1
1234	1324	keine	1	1
1234	1342	keine	2	2
1234	1423	keine	2	2
1234	1432	keine	1	3
1234	2134	keine	1	1
1234	2143	keine	2	2
1234	2314	keine	2	2
1234	2341	$x_1 = 2, x_2 = 8 + 5\sqrt{2}, x_3 = 2 + 2\sqrt{2}, x_4 = \sqrt{2}$	3	3
1234	2413	$x_1 = 1, x_2 \approx 2.2338, x_3 \approx 0.2338, x_4 \approx 0.2884$	3	3
1234	2431	$x_1 = 1, x_2 \approx 5.5901, x_3 \approx 1.2449, x_4 \approx 0.8614$	2	4
1234	3124	keine	2	2
1234	3142	$x_1 = 1, x_2 \approx 0.1291, x_3 \approx 0.4477, x_4 \approx 0.1046$	3	3
1234	3214	keine	1	3
1234	3241	$x_1 = 1, x_2 \approx 3.6287, x_3 \approx 3.2450, x_4 \approx 0.7227$	2	4
1234	3412	$x_1 = x_3 = 8, x_2 = x_4 = -1 + \sqrt{17}$	2	4
1234	3421	$x_1 = 1, x_2 \approx 1.6590, x_3 \approx 2.3295, x_4 \approx 0.8761$	3	5
1234	4123	$x_1 = 7, x_2 = 3\sqrt{2} - 2, x_3 = -5 + 4\sqrt{2}, x_4 = 8 - 5\sqrt{2}$	3	3
1234	4132	$x_1 = 1, x_2 \approx 0.1569, x_3 \approx 0.2268, x_4 \approx 0.1821$	2	4
1234	4213	$x_1 = 1, x_2 \approx 1.1183, x_3 \approx 0.2227, x_4 \approx 0.3082$	2	4
1234	4231	$x_1 = 1, x_2 \approx 2.6590, x_3 \approx 1.2056, x_4 = 1$	1	5
1234	4312	$x_1 = 1, x_2 \approx 0.3761, x_3 \approx 0.7120, x_4 \approx 0.4293$	3	5
1234	4321	$x_1 = x_4 = 2, x_2 = x_3 = 1 + \sqrt{5}$	2	6

Tabelle 7.2: Lösungen im Plackett-Luce-Modell für $n = 4$ bei einmaligem Auftreten von jeweils genau zwei Permutationen. Alle Lösungen wurden exakt bestimmt und zum Teil nur für die kompakte Darstellung approximiert.

approximierten Lösungen zu liegen. Die Prozedur `test_first_polynomial_pl(n)` schaut nämlich vereinfachend nur in das erste Polynom der Gröbnerbasis und kann auf diese Weise eine Menge Fälle erkennen in denen sicher keine gültige Lösung existiert: Taucht kein einziges „-“ Zeichen in einem Polynom auf, so kann es nur 0 werden wenn mindestens ein Parameter 0 oder negativ ist. Diese Prozedur terminierte am Rechner `monrovia` des Fachbereichs 12 der Philipps-Universität Marburg nach etwa 120 Minuten. Mit diesen Ergebnissen bestätigt sich der zuvor beobachtete Trend: Die Paare von Permutationen, in denen an dem ersten Polynom erkennbar ist, dass keine Lösungen existieren, sind auch für $n = 5$ solche, in denen die Abstände eher klein sind. Genauer tritt der Fall in dem die nicht-Existenz von Lösungen sofort erkennbar ist nur auf, wenn die Summe der beiden Abstände kleiner oder gleich 8 ist. Aufgrund der Länge der Ergebnislisten wird an dieser Stelle auf ihre Darstellung verzichtet, sie kann jedoch mit Hilfe der genannten Prozedur generiert werden.

Die genaue Regelmäßigkeit die hinter diesen Beobachtungen steckt, konnte noch nicht formalisiert werden, aber es ist zu erwarten dass für den Fall des Auftretens von genau zwei Permutationen eine Aussage der Form „Wenn die Abstände kleiner x sind, dann existiert kein Maximum Likelihood Schätzer, sind die Abstände größer als y , dann existiert ein eindeutiger Maximum Likelihood Schätzer“ möglich ist.

Frage 7.1 *Welcher Zusammenhang besteht zwischen Abständen der in einem gegebenen Sample auftauchenden Permutationen und der Existenz eines zugehörigen Maximum Likelihood Schätzers im Plackett-Luce-Modell?*

Ferner wäre interessant inwiefern dies auch für größere Samples verallgemeinert werden kann, beispielsweise mit dem von Kendall vorgeschlagenen Übereinstimmungskoeffizienten innerhalb eines Samples [Ken70]. Hierbei wird ein Maß der Diversität für eine Menge von auftauchenden Permutationen definiert, welches 1 ist, falls nur eine einzige Permutation (gegebenenfalls mehrfach) auftaucht und sonst einen Wert zwischen 0 und 1 annimmt der den „Grad der Übereinstimmung“ im Sample misst. Eine dahingehende Untersuchung würde den Rahmen der Masterarbeit sprengen und wäre zudem vermutlich schwierig, da mit steigenden Samplegrößen das Berechnen der Gröbnerbasen wieder schwieriger beziehungsweise langwieriger wird. Und selbst wenn dies gelingt, ist zu erwarten dass die Grade erneut zu groß sind um exakte Lösungen zu bestimmen, während die approximative Lösungen weiter problembehaftet sind, wie in Abschnitt 6.3 beschrieben.

§ 7.2 Bradley-Terry-Mallows

Analog zum Plackett-Luce-Modell kann in diesem Modell angenommen werden, dass einer der Parameter 1 ist, um die Rechnungen zu vereinfachen. Die Verwandtschaft mit dem Plackett-Luce-Modell lässt vermuten dass es sich bei der Suche nach Maximum Likelihood Schätzern ähnlich verhält, jedoch sind selbst die einfachen Fälle schon schwieriger als bei Plackett-Luce.

Es sei zunächst $n = 3$. Im dem Fall in dem nur eine Permutation genau ein mal auftaucht ist dies die Permutation $\pi = 123$, die Likelihood-Funktion ist dann $v_1^2 v_2 / (v_1^2 v_2 + v_1^2 v_3 + v_2^2 v_1 + v_2^2 v_3 + v_3^2 v_1 + v_3^2 v_2)$. Der Gradient dieser Funktion hat keine Nullstellen in

π_1	π_2	Lösungen	Cayley	Kendalls τ
123	132	keine	1	1
123	213	keine	1	1
123	231	$v_1 \approx 1, v_2 \approx 1.7844, v_3 \approx 0.5604$	2	2
123	312	$v_1 \approx 1, v_2 \approx 0.3141, v_3 \approx 0.5604$	2	2
123	321	$v_1 = v_2 = v_3 = 1$	1	3

Tabelle 7.3: Lösungen im Fall $n = 3$ bei einmaligem Auftreten von genau zwei Permutationen im Bradley-Terry-Mallows-Modell.

\mathbb{R}_+^3 , das heißt es existieren keine Maxima im zulässigen Parameterraum. Wie auch beim Plackett-Luce-Modell ist ein solches Maximum auch gar nicht zu erwarten.

Die Fälle in denen genau zwei Permutationen genau ein mal auftreten können wie zuvor mit Hilfe der Prozedur `testrun_btm` gelöst werden, welche alle $n! - 1$ möglichen Fälle durchgeht und mit Hilfe von `get_solutions` versucht das jeweils zugehörige Gleichungssystem zu lösen. Für $n = 3$ ergeben sich die Lösungen aus Tabelle 7.3. Genau wie beim Plackett-Luce-Modell gibt es in den ersten beiden Fällen keine Lösung und in den anderen Fällen genau eine Lösung, weshalb man möglicherweise zunächst ein ähnliches Verhalten der beiden Modelle vermuten könnte. Für den Fall $n = 4$ kann die Prozedur `get_solutions` die meisten Fälle nicht lösen, in 19 der 23 Fälle bleiben Faktoren von zu hohem Grad mit mehr als einer Variablen ungelöst. Einer der lösbaren Fälle widerlegt aber die obige Vermutung: für $\pi_1 = 1234, \pi_2 = 4312$ existiert keine Lösung, während in diesem Fall beim Plackett-Luce-Modell eine Lösung existiert (siehe Tabelle 7.2 auf Seite 47).

Das in Beispiel 6.2 dargestellte Problem von verschwindenden Lösungen ist nicht nur von theoretischer Natur, sondern passiert tatsächlich und kann anhand des Bradley-Terry-Mallows-Modells im Fall $n = 4$ nachgewiesen werden. Der Übergang zu rationalen Ausdrücken mit Hilfe der Funktion `convert(·, rational)` kann, wie im Abschnitt 6.3 dargelegt, fehlerverstärkend wirken, wodurch hier gegebenenfalls neue Fehler entstehen und noch mehr Lösungen verschwinden. Das Umwandeln in rationale Ausdrücke ist per default deaktiviert, lässt sich aber aktivieren, in dem man beim Aufruf von `testrun_btm` als zweiten Parameter `true` mitgibt (beziehungsweise bei `get_solutions` als letzten Parameter). Mit dem Übergang zu rationalen Ausdrücken kann mit Hilfe von `testrun_btm(4, true)` nur ein einziges Maximum bestimmt werden, und zwar für $\pi_2 = 4321$. Dabei können alle Fälle vollständig ausgerechnet werden, zumindest approximativ. Der Aufruf `testrun_btm(4, false)` kann dagegen zwei Maxima konkret ausrechnen, für $\pi_2 = 4231$ und für $\pi_2 = 4321$. Dafür bleiben aber etwa 50 Fälle⁶ unvollständig bearbeitet. Dies zeigt, dass der Nichtexistenz von Lösungen in den anderen 21 Fällen im Resultat von `testrun_btm(4, false)` sowie der Nichtexistenz der Lösungen im Resultat

⁶Die Anzahl nicht abgeschlossener Fälle schwankt bei mehrfachem Aufruf etwas, ich habe bei meinen Rechnungen zwischen 47 und 59 unfertiger Fälle beobachtet. Diese Schwankungen sind vermutlich auf nicht-deterministische Verfahrensweisen bei der approximativen Bestimmung von Nullstellen mit Hilfe von `fsolve` zurück zu führen.

von `testrun_btm(4,true)` nicht getraut werden kann.

Eine vereinfachende Untersuchung der ersten Polynome in den Gröbnerbasen, wie sie bei Plackett-Luce gemacht wurde, lässt hier leider keine weiteren Schlüsse zu, da in jedem Fall relevante Faktoren gefunden werden. Die analog zum Plackett-Luce-Modell funktionierende Prozedur `test_first_polynomial_btm` liefert keine neue Information.

Insgesamt stellt sich die Frage nach der Relevanz des betrachteten Modells: Einerseits ist es so kompliziert das selbst einfache Fälle nur schwierig zu berechnen sind. Die approximativen Lösungen müssen hierbei jedenfalls bezweifelt werden. Ferner wirkt der konditionale Ansatz des Babington-Smith-Modells sehr künstlich, siehe auch Ende Abschnitt 4.2, und es erscheint sinnvoller gleich Modelle zu betrachten die nicht mit Rankings sondern nur mit paarweisen Vergleichen arbeiten.

§ 7.3 Inversions-Modell

Im Inversionsmodell muss ein wenig anders vorgegangen werden, denn hier muss ein Extremum unter einer Nebenbedingung bestimmt werden: Die Summe der Einzelwahrscheinlichkeiten muss 1 sein. Wie auch schon beim Plackett-Luce-Modell werden im Vorhinein einige Parameter 1 gesetzt, so dass bei der Berechnung genau eine Lösung herauskommen muss, wenn die Wahrscheinlichkeitsverteilung eindeutig ist. Dies vereinfacht die Berechnung der Gröbnerbasis wesentlich. Wie im Abschnitt 4.3 gezeigt, kann ohne Einschränkung angenommen werden, dass $v_{ij} = 1$ gilt für $(i, j) \neq (1, 2)$. Zu maximieren ist noch stets die Likelihood-Funktion $L((u_{ij}, v_{ij}))$, nun unter der Nebenbedingung

$$N((u_{ij}, v_{ij})) := \sum_{\pi \in S_n} (P_{(u_{ij}, v_{ij})}(\pi)) - 1 = 0.$$

Da N in jeder Variablen den Grad 1 hat, kann man die Nebenbedingung nach v_{12} auflösen und erhält nach Einsetzen in L eine neue Funktion L^* , die ohne Nebenbedingungen auf Nullstellen untersucht werden kann und nur noch von den Variablen u_{ij} abhängt. L^* ist eine rationale Funktion, doch da für die Nullstellen nur die Zähler relevant sind, kann hier wieder mit Gröbnerbasen gearbeitet werden.

Beispiel 7.2 Sei $n = 3$ und sei $a = (1, 3, 0, 2, 1, 0)$ ein Vektor von statistischen Daten. Die Likelihood-Funktion L ist

$$L(u_{ij}, v_{ij}) = u_{12}^5 u_{13}^4 u_{23}^3 v_{23}^4 v_{12}^2 v_{13}^3$$

Einsetzen von $v_{13} = v_{23} = 1$ und Auflösen der Nebenbedingung

$$u_{12} u_{13} u_{23} + u_{12} u_{13} v_{23} + v_{12} u_{13} u_{23} + v_{12} v_{13} u_{23} + u_{12} v_{13} v_{23} + v_{12} v_{13} v_{23} - 1 = 0$$

nach v_{12} liefert

$$v_{12} = -\frac{u_{12}(1 + u_{13} + u_{13}u_{23})}{1 + u_{23} + u_{13}u_{23}}$$

und somit

$$L^*(u_{12}, u_{13}, u_{23}) = \frac{u_{12}^7 u_{13}^4 u_{23}^3 (1 + u_{13} + u_{13}u_{23})^2}{(1 + u_{23} + u_{13}u_{23})^2}.$$

Es kann nun die Funktion L^* maximiert werden über \mathbb{R}_+^3 , wobei das Ideal der Nenner der 3 partiellen Ableitungen von L^* betrachtet wird. Die sonstige Vorgehensweise ist wie im Plackett-Luce-Modell: Berechnen der Gröbnerbasis mit lexikographischer Termordnung, Faktorisieren der Polynome und Suche nach Faktoren die mit gültigen Parametern 0 werden können:

Die Gröbnerbasis mit lexikographischer Termordnung bezüglich $u_{12} > u_{13} > u_{23}$ enthält als erstes Polynom

$$u_{12}^4 u_{13}^4 u_{23}^3 (u_{23} - 1)(3u_{23} + 4)(4u_{23} + 3)(u_{12}u_{13}u_{23} + u_{12}u_{13} + u_{12} - 1).$$

Hier tauchen zwei relevante Faktoren auf: Ist der letzte Faktor, $(u_{12}u_{13}u_{23} + u_{12}u_{13} + u_{12} - 1)$, 0, so gilt selbiges für die Variable v_{12} , denn dieser Faktor ist genau der Zähler von v_{12} . Es muss demzufolge, durch den einzig gebliebenen relevanten Faktor $(u_{23} - 1)$, $u_{23} = 1$ gelten. Eingesetzt in die Polynome der Gröbnerbasis lässt dies die ersten beiden Polynome verschwinden. Das dritte Polynom ist nun in faktorisierter Form:

$$16u_{12}^4 u_{13}^4 (3u_{13} + 4)(u_{13} - 1)(2u_{12}u_{13} - 1 + u_{12})$$

weshalb entweder $u_{13} = 1$ oder $u_{13} = -\frac{1}{2} \frac{u_{12}-1}{u_{12}}$ gelten muss. Letzteres liefert $v_{12} = 0$, eine ungültige Lösung. Ersteres eingesetzt in die verbleibenden Polynome liefert eine Reihe von Polynomen in u_{12} , deren größter gemeinsamer Teiler $(2u_{12}^4(21u_{12} - 5)(3u_{12} - 1))$ ist. Ist $u_{12} = \frac{1}{3}$, so ist $v_{12} = 0$, für $u_{12} = \frac{5}{21}$ ergibt sich allerdings eine eindeutige gültige Lösung, nämlich $(u_{12}, u_{13}, u_{23}, v_{12}, v_{13}, v_{23}) = (\frac{5}{21}, 1, 1, \frac{2}{21}, 1, 1)$. Bei dieser Stelle handelt es sich tatsächlich um ein Maximum, da alle Eigenwerte der Hesse-Matrix von L^* an dieser Stelle negativ sind. \diamond

Versucht man strukturelle Eigenheiten zu untersuchen wie beim Plackett-Luce-Modell, so fällt zunächst auf, dass die Wahl der Labels einen Einfluss auf den Rechenaufwand hat. Dies liegt an der Parametrisierung durch Auflösen der Nebenbedingung. Die Likelihood-Funktion enthält per Definition nur Potenzen der u_{ij} sowie eine Potenz von v_{12} . Da v_{12} nach Auflösen der Nebenbedingung eine rationale Funktion in den Variablen u_{ij} ist, ist auch L^* eine rationale Funktion in den Variablen u_{ij} und ein Monom, wenn v_{12} in der Likelihood-Funktion die Potenz 0 hat. Alle partiellen Ableitungen von Monomen sind skalare Vielfache von Monomen, welche nur 0 werden wenn eine der Variablen 0 ist. Das heißt, wenn v_{12} in L nicht auftaucht, so existiert sicher kein Maximum Likelihood Schätzer. Und v_{12} taucht genau dann auf, wenn im Sample eine Permutation existiert, in der $(1, 2)$ eine Inversion ist. Doch die Wahl von $(i, j) = (1, 2)$ ist willkürlich, das heißt mit der Möglichkeit der Umparametrisierung erhält man das folgende Ergebnis:

Satz 7.3 *Sei ein statistisches Sample in $\mathbb{N}_0^{n!}$ gegeben und seien $1 \leq i < j \leq n \in \mathbb{N}$ so, dass entweder alle auftauchenden Permutationen i und j invertieren oder so dass alle*

auftauchenden Permutationen i und j nicht invertieren. Dann existiert kein Maximum Likelihood Schätzer.

Beweis. Betrachte zur Erinnerung die Formel für die Wahrscheinlichkeit einer Permutation aus Definition 4.10

$$P_{(u_{ij}, v_{ij})}(\pi) := \prod_{\substack{i < j \\ \pi^{-1}(i) < \pi^{-1}(j)}} u_{ij} \cdot \prod_{\substack{i < j \\ \pi^{-1}(i) > \pi^{-1}(j)}} v_{ij}. \quad (7.1)$$

Nach einer Umbenennung der Objekte gilt $(i, j) = (1, 2)$. Dies geht ohne Einschränkung, da das Inversions-Modell label-invariant ist, siehe Lemma 4.13. Es gelte zunächst für alle Permutationen π die im Sample auftauchen, dass π die Zahlen 1 und 2 nicht invertiert, das heißt $\pi^{-1}(1) < \pi^{-1}(2)$. Dann ist die Potenz von v_{12} in der zugehörigen Likelihood-Funktion L schon 0, da für jede auftauchende Permutation in (7.1) u_{12} in das Produkt genommen wird und nicht v_{12} . Die Nebenbedingung $N((u_{ij}, v_{ij})) = \sum_{\pi \in S_n} P_{(u_{ij}, v_{ij})}(\pi) - 1$ ist ein Polynom die in jeder Variable den Grad 1 hat. Daher kann v_{12} als rationale Funktion in den anderen Variablen ausgedrückt werden und die Bestimmung der Maximum Likelihood Schätzer ist dann die Bestimmung der Maxima der Funktion L^* , in der v_{12} durch diesen rationalen Ausdruck ersetzt wurde. Da aber in L die Variable v_{12} nicht auftaucht, taucht dieser rationale Ausdruck in L^* nicht auf, das heißt L^* ist ein Monom. Somit sind alle partiellen Ableitungen von L^* skalare Vielfache von Monomen, welche keine Nullstellen in $\mathbb{R}_+^{n(n-1)-1}$ haben. Der Fall in dem alle im Sample auftauchenden Permutationen die Zahlen 1 und 2 invertieren verläuft völlig analog. Man löse hier stattdessen die Nebenbedingung nach u_{12} auf und erhält das Ergebnis auf die selbe Weise. \square

Auch dieses Ergebnis lässt sich intuitiv erklären: wie bereits im Abschnitt 4.3 angedeutet, können die Werte u_{ij} als Wette für „ i wird besser platziert als j “ interpretiert werden, die Werte v_{ij} als Wette für „ j wird besser platziert als i “. Wird aber i nie besser als j platziert, so würde man hier eher gar nichts wetten, das heißt $u_{ij} = 0$ setzen wollen, doch der Wert 0 ist nicht zulässig. Stattdessen wählt man das Verhältnis u_{ij}/v_{ij} sehr klein, und je kleiner das Verhältnis, desto wahrscheinlicher wird das beobachtete Sample. Hier kann auf natürliche Weise kein Maximum existieren.

Ein notwendiges Kriterium für die Existenz eines Maximums ist mit Satz 7.3 gegeben. Ist es auch hinreichend? Das folgende kanonische Beispiel kann dies zumindest nicht widerlegen:

Beispiel 7.4 Sei $n = 3$ und Sei $(1, 0, 0, 0, 0, 1) \in \mathbb{N}_0^6$ das gegebene Sample, das heißt die Permutationen $\pi_1 = 123$ und $\pi_6 = 321$ tauchen jeweils genau ein mal auf. Für dieses Sample gilt, das jedes Paar genau einmal invertiert vorkommt (in π_6), und einmal nicht invertiert (in π_1), das heißt die Bedingung von Satz 7.3 gilt nicht. Wie zuvor kann $v_{13} = v_{23} = 1$ angenommen werden und v_{12} nach Auflösen der Nebenbedingung wie in Beispiel 7.2 als rationale Funktion der Variablen u_{ij} dargestellt werden. Das erste Polynom in der Gröbnerbasis des Ideals der Zähler der 3 partiellen Ableitungen von

L^* ist $u_{13} u_{23} (u_{23} - 1) (u_{23} + 1)$, also muss $u_{23} = 1$ gelten. Das zweite Polynom in der Gröbnerbasis ist $u_{13} u_{23} (u_{13} - 1) (1 + u_{13})$, also gilt $u_{13} = 1$. Dies substituiert im dritten Polynom der Gröbnerbasis liefert das Polynom $6 u_{12} - 1$, daher gilt $u_{12} = \frac{1}{6}$, womit alle Elemente der Gröbnerbasis 0 werden und $v_{12} = \frac{1}{6}$ gilt. Diese Stelle ist tatsächlich ein Maximum, da alle Eigenwerte der Hesse-Matrix negativ sind, wie die numerische Prüfung zeigt. \diamond

Um weitere Fälle zu untersuchen um gegebenenfalls ein Gegenbeispiel zu finden für die Vermutung, dass das Kriterium aus Satz 7.3 hinreichend ist, habe ich auch hier einige Funktionen zur automatischen Bearbeitung der Einzelfälle geschrieben, die aufgrund der Struktur des Modells ein wenig anders aussehen als beim Plackett-Luce- und Bradley-Terry-Mallows-Modell. Zunächst habe ich die Funktion `test_criterion_inv` geschrieben, welche als Argument die Problemgröße n und einen Samplevektor entgegen nimmt und prüft, ob die notwendige Bedingung aus Satz 7.3 erfüllt ist. Rückgabewert ist `true`, falls das Sample die Bedingung erfüllt und `false` sonst. Die Funktion `gls_inv` ist eine Hilfe um das zu lösende Gleichungssystem zu erstellen. Diese Funktion übernimmt den etwas umständlichen Teil der Umparametrisierung der Nebenbedingung und Substitutionen der entsprechenden v_{ij} mit 1. Der Rückgabewert ist eine Liste, die in dieser Reihenfolge die partiellen Ableitungen von L^* in den Richtungen u_{ij} , L^* selbst, den durch u_{ij} parametrisierten Wert von v_{12} sowie eine Liste der Parameter u_{ij} enthält.

Die Funktion `solve_case_inv` fasst diese Hilfsfunktionen zusammen. Die Argumente dieser Funktion sind Problemgröße und ein Samplevektor. Hier wird erst die notwendige Bedingung geprüft und bei positivem Ergebnis mit Hilfe der Funktion `get_solutions` versucht, das Gleichungssystem zu lösen.

Das notwendige Kriterium für die Existenz von Maximum Likelihood Schätzern im Inversions-Modell ist insofern interessant, als das nur eines der beim Plackett-Luce-Modell betrachteten Fälle überhaupt die notwendige Bedingung erfüllt, nämlich das aus dem obigen Beispiel. Denn tauchen genau zwei Permutationen im Sample auf, so ist ohne Einschränkung nach Umbenennung der Objekte eine davon $\pi = 1 \cdots n$, welche genau alle Paare (i, j) mit $i < j$ nicht invertiert. In der Gesamtheit der anderen auftretenden Permutationen muss also jedes Paar (i, j) mit $i < j$ mindestens einmal invertiert werden, doch es gibt im Sample nur eine einzige andere Permutation, welche demzufolge $\gamma = n \cdots 1$ sein muss. Die Betrachtung der Fälle analog zum Plackett-Luce-Modell kann also keine wesentlichen Erkenntnisse über die Existenz von Maximum Likelihood Schätzer hervorbringen, da die notwendigen Bedingungen nur in einem der $n! - 1$ Fälle erfüllt wird.

Daher habe ich im Folgenden, anders als bei den anderen beiden Modellen, Fälle von statistischen Daten betrachtet, in denen drei Permutationen je genau ein mal auftauchen. Wieder kann man die Benennung der Objekte so wählen, dass die identische Permutation $1 \cdots n$ immer auftaucht. Dann bleiben 10 solcher Fälle für $n = 3$, wovon 3 das notwendige Kriterium nicht erfüllen. In einem ersten Versuch dauerten zwei davon besonders lange (mehr als 24 Stunden), weshalb ich noch eine kleine Verbesserung in die Funktion `get_solutions` eingebaut habe: Diese Funktion nimmt als sechstes Argument `v` entgegen, dass hier die Substitution sein soll, die zur Auflösung der Nebenbedingung in der

Permutationen im Sample	Lösungen in der Form $(u_{12}, u_{13}, u_{23}, v_{12}, v_{13}, v_{23})$
123, 132, 213	notw. Bedingung nicht erfüllt
123, 132, 231	$(3 - \sqrt{3}, 9, 9 + 9\sqrt{3}, 3(3 + 2\sqrt{3})^{-1}, 9, 9)$
123, 132, 312	notw. Bedingung nicht erfüllt
123, 132, 321	$(6 - 2\sqrt{3}, 18 + 18\sqrt{3}, -9 + 9\sqrt{3}, 12(3 + \sqrt{3})^{-1}, 18, 18)$
123, 213, 231	notw. Bedingung nicht erfüllt
123, 213, 312	$(3 - \sqrt{3}, 9, 9 + 9\sqrt{3}, 3(3 + 2\sqrt{3})^{-1}, 9, 9)$
123, 213, 321	$(-3 + 2\sqrt{3}, 9 + 9\sqrt{3}, 9, 6(3 + \sqrt{3})^{-1}, 9, 9)$
123, 231, 312	notw. Bedingung erfüllt, aber keine Lösungen
123, 231, 321	$(6 - 2\sqrt{3}, -9 + 9\sqrt{3}, 18 + 18\sqrt{3}, 12(3 + \sqrt{3})^{-1}, 18, 18)$
123, 312, 321	$(4\sqrt{3}, -9 + 9\sqrt{3}, 18, 12(3 + \sqrt{3})^{-1}, 18, 18)$

Tabelle 7.4: Lösungen im Inversions-Modell mit $n = 3$ für den Fall des Auftretens von genau 3 Permutationen jeweils 1 mal.

Funktion L eingesetzt wurde, in den bisherigen Beispielen war das der in den Variablen u_{ij} ausgedrückte Wert von v_{12} . Bevor ein rekursiver Schritt ausgeführt wird, wird nun die aktuelle Lösung in diesem Ausdruck substituiert. Wird dieser dadurch 0, so wäre diese Lösung in jedem Fall ungültig und kann daher verworfen werden. Dies passiert in Beispiel 7.2 bei der Teillösung $u_{23} = 1$, $u_{13} = -\frac{1}{2} \frac{u_{12}-1}{u_{12}}$.

Diese Spezialisierung spart die Rechenarbeit bei irrelevanten Fällen, die hier tatsächlich Ursache für lange Rechenzeit waren: Jetzt können für $n = 3$ alle Fälle innerhalb weniger Sekunden ausgerechnet werden. Die Lösungen dieser Fälle sind in Tabelle 7.4 angegeben.

Für $n = 4$ muss allerdings in allen 179 Fällen, in denen das notwendige Kriterium erfüllt ist, numerisch gerechnet werden und keine einzige Lösung kann bestimmt werden. Nach einer Rechnung, die am Rechner **sakania** am Fachbereich 12 der Philipps-Universität Marburg etwa 50 Minuten dauerte, blieben 64 Teillösungen ungelöst. Wenn die Koeffizienten nach der numerischen Lösung in rationale Ausdrücke umgewandelt werden, wie schon beim Bradley-Terry-Mallows-Modell versucht, dauert die Rechnung wieder sehr lange: Auf dem Rechner **sakania** waren nach über 72 Stunden Rechenzeit noch nicht alle Fälle bearbeitet, konkret belief sich der wesentliche Teil dieser Rechenzeit auf die Bearbeitung des Falles mit dem Sample $\{1234, 2143, 4321\}$.

Entfernt man die 4 der 179 Fälle, die verantwortlich sind für die lange Rechenzeit⁷, können alle anderen Fälle in jeweils weniger als 5 Minuten gelöst werden, wenn auch in allen Fällen noch stets mit Approximationen gerechnet wird. Nach einer gesamten Rechenzeit von etwa 35 Minuten am selben Rechner waren alle anderen 175 Fälle vollständig durchgerechnet und es wurden in keinem einzigen Fall eine Lösung gefunden. Wie bereits beim Bradley-Terry-Mallows-Modell festgestellt und in der Diskussion der Rechenfehler in Abschnitt 6.3 bemerkt, kann diesen Lösungen aber nicht getraut werden.

Ferner ist es so, dass der Anteil der Fälle in denen bei genau drei Permutationen das notwendige Kriterium überhaupt erfüllt ist, mit steigendem n sinkt. Für $n = 3$ sind es 7 von 10 Fällen, für $n = 4$ sind es 179 von 253 Fällen, für $n = 5$ aber nur noch 948 von 6073⁸. Man müsste also für die weitergehende Untersuchungen nicht nur ein Maß der Diversität in einem statistischen Sample definieren (wie bereits im Abschnitt 7.1 im Zusammenhang mit dem Plackett-Luce-Modell angedeutet), sondern auch eine wesentlich größere Klasse von möglichen statistischen Daten, in denen auch mehr als drei Permutationen vorkommen, untersuchen. Dabei ist unklar, ob diese bei den dann auftretenden, wesentlich größeren Fallzahlen überhaupt sinnvoll, das heißt in angemessener Zeit, vollständig behandelt werden können. Dies würde den Rahmen der Masterarbeit leider sprengen, weshalb diese Verallgemeinerung hier nicht weiter diskutiert wird.

Denkt man aber an reale Situationen, in denen $n > 10$ Objekte in einem Ranking geordnet werden, so wird der Fall, das ein bestimmtes Paar nie (oder immer) invertiert vorkommt, recht häufig eintreten. Oftmals gibt es in konkreten Anwendungsfällen klare Favoriten und klare Außenseiter, in dessen Rangfolge sich dann mit großer Wahrscheinlichkeit alle Rankings im Sample einig sind. Passiert dies auch nur in einem einzigen Fall, so ist das notwendige Kriterium aus Satz 7.3 schon nicht mehr erfüllt. Es stellt sich somit die Frage, inwiefern die Maximum Likelihood Schätzung in diesem Modell auf reale Situationen anwendbar ist, wenn nicht gerade besonders große Diversität im Sample zu erwarten ist.

§ 8 Diskussion

Die im vorangegangenen Abschnitt präsentierten Untersuchungen bestimmter Spezialfälle sind nur ein Anfang und von sehr theoretischer Natur. In Anwendungsfällen wird in der Regel eine wesentlich größere Anzahl von Permutationen im Sample vorhanden sein und auch die Anzahl zu ordnender Elemente größer sein. Dennoch ist fest zu stellen, dass

⁷Die 4 Samples $\{1234, 2143, 4321\}$, $\{1234, 2314, 4321\}$, $\{1234, 2413, 4321\}$ und $\{1234, 2431, 4321\}$ waren jene, die bei der Bestimmung der Lösung nach Berechnung der Gröbnerbasis für Rechenzeiten von mehr als 15 Minuten sorgten und wurden daher zugunsten der anderen Lösungen übersprungen. Aufgrund der Rechenfehler, die in allen Lösungen gemacht werden können die Lösungen dieser wenigen Einzelfälle ohnehin vernachlässigt werden. Dieses überspringen lässt sich reproduzieren, wenn man in der Datei `inversions` in der Prozedur `testrun_inv` nach Zeile 13 (siehe Seite 71) eine weitere if-Verzweigung einbaut, etwa von der Form `if n=4 and ((i=8 and j=24) or (i=9 and j=24) or (i=11 and j=24) or (i=12 and j=24))`. Diese Indizes sind genau die der benannten Samples.

⁸Die identische Permutation tritt nach Umbenennung der Objekte immer auf, es verbleiben $\binom{n-1}{2}$ Möglichkeiten, die beiden anderen auftretenden Permutationen zu wählen

selbst die betrachteten ausserordentlich kleinen Spezialfälle schon schwer zu lösen sind, und es ist nicht zu erwarten dass diese nur mit den vorgestellten Methoden überhaupt gelöst werden können. Es wäre daher interessant zu untersuchen, inwiefern spezielle Verfahren mit den besonderen Strukturen der Probleme effizienter umgehen können — beispielsweise könnte man ausnutzen, dass an vielen Stellen nicht beliebige, sondern homogene Polynome auftauchen und so spezielle Auswahlstrategien für den Buchberger-Algorithmus verwenden [GMN⁺91].

Ein schönes Ergebnis ist, dass die intuitive Idee, dass die Diversität unter den betrachteten Permutationen groß genug sein muss um zu Maximum Likelihood Schätzern zu kommen, im Plackett-Luce-Modell experimentell bestätigt wird und im Inversionsmodell durch das notwendige Kriterium in Satz 7.3 bewiesen werden kann.

In allen betrachteten Fällen waren die Maxima, falls existent, eindeutig, daher drängt sich die Frage auf, ob dies im Allgemeinen auch so ist. Es ist allerdings gut möglich, dass die betrachteten Fälle schlicht zu klein waren um mehrfache Maxima hervorzubringen und dass bei systematischer Konstruktion für größere n und größere Samples mehrdeutige Fälle gefunden werden können — wenn es gelingt sie zu lösen.

Zusammenfassend bleibt zu sagen, dass die Bestimmung von Maximum Likelihood Schätzern ein schwieriges Problem bleibt, bei dem die hier praktizierte naive Herangehensweise an die Lösung der Gleichungssysteme vielleicht zu wenig filigran ist. Ausführlichere Untersuchungen, die etwa die auftauchenden Ideale mit Methoden der kommutativen Algebra genauer untersuchen um ihre Struktur zur Lösung der Probleme auszunutzen, würden möglicherweise weitere Ergebnisse liefern, können jedoch im Rahmen dieser Masterarbeit nicht weiter verfolgt werden.

Quellcode der Maple Funktionen

Im Folgenden sind die Funktionen die zum Zweck der Untersuchung der Modelle in Kapitel III geschrieben wurden aufgelistet und dokumentiert. Die Funktionsweise der einzelnen Prozeduren ist hier in Kommentaren knapp beschrieben, für ausführlichere Information siehe die jeweiligen Abschnitte der Modelle in Kapitel III. Die Kenntnis grundlegender Funktionalitäten von Maple wird dabei vorausgesetzt, diese können in der Dokumentation von Maple [MGH⁺05] oder in einem entsprechenden Buch zur Einführung in die Arbeit mit Maple wie beispielsweise [Hec03] nachgelesen werden

Die Funktionen sind in mehreren „Paketen“⁹ organisiert, die in Maple mit dem Befehl `read <Dateiname>` eingelesen werden können um die Funktionen zur Verfügung zu stellen. Für jedes der drei Module wird das Paket `algstat` und die durch `init` geladenen Pakete benötigt. Falls nicht aus besonderen Gründen beabsichtigt, sollten also stets alle Pakete mit Hilfe von `init` geladen werden. Die Dateien sind an den Rechnern des Fachbereichs 12 der Universität Marburg verfügbar und liegen dort im Verzeichnis `/app/home/debeerst/maple/`. Ferner können sie auch von meiner Homepage am Fachbereich 12 heruntergeladen werden¹⁰, wobei dann jedoch die Verzeichnisse in den Dateien `init` und `algstat` angepasst werden müssen.

Das Verzeichnis enthält 6 Pakete: `init` ist die Initialisierungsdatei, die die anderen Dateien lädt, `permutations` enthält die Funktionen zur Manipulation von Permutationen in Wortschreibweise (oder genauer: Listenschreibweise), `algstat` enthält die Modellübergreifende Funktion `get_solutions` und die Dateien `plackett`, `btm` und `inversions` enthalten jeweils die Funktionen zu den Modellen Plackett-Luce, Bradley-Terry-Mallows sowie dem Inversionsmodell.

A Modellübergreifende Funktionen

Die Datei `init` sorgt für das initiale Laden der benötigten Pakete, definiert Abkürzungen und lädt die eigentlichen Funktionen. Liegen die Dateien in einem anderen Verzeichnis als `/app/home/debeerst/maple`, so muss dies in den Zeilen 3 bis 8 angepasst werden.

Paket `init`

```
1 # Dateipfade abspeichern damit man bei Aenderungen leicht re-initialisieren  
   kann ohne bisherige Ergebnisse verwerfen zu muessen
```

⁹Diese Pakete sind eigentlich sind es nur einfache Textdateien, die die Funktionsdefinitionen enthalten. Es sind keine Pakete im Sinne der Pakete in Maple, daher ist der Begriff hier in Anführungszeichen gesetzt. Im Folgenden ist weiterhin von Paketen die Rede, dabei immer in dem hier gemeinten einfachen Sinne von Dateien die eine Sammlung von Funktionen enthalten.

¹⁰<http://www.mathematik.uni-marburg.de/debeerst/masterarbeit>

```

2 init := "/app/home/debeerst/maple/init":
3 permutations := "/app/home/debeerst/maple/permutations":
4 algstat := "/app/home/debeerst/maple/algstat":
5 plackett := "/app/home/debeerst/maple/plackett":
6 btm := "/app/home/debeerst/maple/btm":
7 inversions := "/app/home/debeerst/maple/inversions":
8
9 # Noetige Pakete laden
10 with(combinat):
11 with(Groebner):
12
13 # Abkuerzungen definieren
14 grad := VectorCalculus[Gradient]:
15 hess := VectorCalculus[Hessian]:
16 PolyId := PolynomialIdeals[PolynomialIdeal]:
17
18 # Funktionen laden
19 read permutations:
20 read algstat:
21 read plackett:
22 read btm:
23 read inversions:

```

Die Datei `permutations` enthält alle nötigen Funktionen zur Manipulation von Permutationen: Eine Abkürzung für die Generierung der symmetrischen Gruppe (Zeile 2), eine Funktion zur Berechnung der Inversionen einer Permutation (Zeile 8), Produkt (Zeile 28) und Inverse (Zeile 39) von Permutationen sowie Kendalls τ -Metrik (Zeile 71) und Cayley-Metrik (Zeile 50).

Paket `permutations`

```

1 # Kurzschreibweise fuer die Permutationsgruppe
2 S := permute;
3
4 # Berechnet die Inversionen und die Nicht-Inversionen der Permutation a
5 # Ein Tupel (i,j) mit i<j ist eine Inversion falls a(-1)(i) > a(-1)(j) und
   eine Nicht-Inversion sonst
6 # Output:
7 # - Liste von zwei Mengen. Die erste Menge enhaelt alle Tupel (i,j) die
   Inversionen sind, die zweite Menge enthaelt alle anderen Tupel.
8 get_inversions := proc(a)
9   local b,i,j,invset:={},noninvset:={};
10  b := inverse(a);
11  for i from 1 to nops(a) do:
12    for j from i+1 to nops(a) do:
13      if b[i] > b[j] then
14        invset := invset union {[i,j]};
15  else
16    noninvset := noninvset union {[i,j]};
17    fi;

```

```

18     od;
19     od;
20     return [invset,noninvset];
21 end proc;
22
23 # Funktion zur Berechnung des Produkts von Permuatationen.
24 # Input:
25 # - Permutationen a und b
26 # Output
27 # - Permuatation a*b, wobei von Hintereinanderausfuerung von rechts
    ausgegangen wird, d.h. (a*b)(i) = a(b(i)).
28 pp := proc(a,b)
29     local n,c,i;
30     n := nops(a);
31     c := [0$n];
32     for i from 1 to n do:
33         c[i] := a[b[i]];
34     od;
35     return c;
36 end proc;
37
38 # Funktion zur Berechnung der Inversen einer Permutation
39 inverse := proc(a)
40     local b,n,i;
41     n := nops(a);
42     b := [0$n];
43     for i from 1 to n do:
44         b[a[i]] := i;
45     od;
46     return b;
47 end proc;
48
49 # Funktion zur Berechnung des Cayley-Abstandes zweier Permutationen. Es wird
    verwendet das Cayley(a,b)= Cayley(a*b^(-1),id) gilt.
50 cayley := proc(a,b)
51     local c,n,i,j,k;
52     c := pp(a,inverse(b));
53     n := nops(c);
54     k := 0;
55     for i from 1 to n do:
56         if(c[i] <> i) then:
57             k := k+1;
58             # search for i in c and transpose
59             for j from i+1 to n do: # indexes < i are sorted
60                 if(c[j] = i) then:
61                     c[j]:=c[i];
62                     c[i]:=i;
63                 fi;
64             od;

```

```

65     fi;
66   od;
67   return k;
68 end proc;
69
70 # Funktion zur Berechnung von Kendalls tau-Abstand zweier Permutationen. Es
    wid die Formel von Critchlow, Flinger und Verducci verwendet (siehe
    Kapitel II, Abschnitt 3.3)
71 kendall := proc(a,b)
72   local n,i,j,k,a2,b2;
73   n := nops(a);
74   k := 0;
75   a2 := inverse(a);
76   b2 := inverse(b);
77   for i from 1 to n do:
78     for j from i+1 to n do:
79       if((a2[i]-a2[j])*(b2[i]-b2[j]) < 0) then:
80         k := k+1;
81         fi;
82       od;
83     od;
84   return k;
85 end proc;

```

Die Datei `algstat` enthält nur die Funktion `get_solutions`. Eine ausführliche Beschreibung ihrer Funktionsweise kann gefunden werden in Abschnitt 6. Da die 173 Zeilen lange Funktion etwas unübersichtlich ist, wird hier noch ein grober Überblick gegeben: Nach einführenden Kommentaren und Definitionen der lokalen und globalen Variablen (Zeilen 1–16), wird die zu lösende Variable festgelegt und das erste Nicht-0-Polynom in der Liste `Id` bestimmt (Zeilen 21–30). Dann teilt sich die Funktion in zwei große Abschnitte. Der erste Teil (Zeilen 31–77) widmet sich dem Spezialfall, dass nur noch eine zu lösende Variable übrig ist. Hier wird erst der größte gemeinsame Teiler der verbleibenden Polynome bestimmt bevor nachfolgend die Nullstellen bestimmt werden. Im zweiten Teil (Zeilen 78–171) wird der allgemeine Fall behandelt in dem mehr als eine Variable auftauchen können. Hier werden die Fallunterscheidungen gemacht wie sie in Abschnitt 6.2 beschrieben werden, wenn möglich werden Nullstellen bestimmt und rekursiv die nächsten Teillösungen ausgerechnet (Zeile 161), welche mit der bisherigen Teillösung verknüpft und zurück gegeben werden.

Paket `algstat`

```

1 # Funktion zur Berechnung von Loesungen eines polynomialen Gleichungssystems
2 # Input:
3 # Id - Liste von Polynomen die Groebnerbasis mit lex-Ordnung sind,
    faktorisiert und aufteigend nach Leitmonom sortiert (siehe Abschnitt 6)
4 # varnr - Index der Variable die in diesem Schritt geloest werden soll
5 # vars - Liste von Variablen die ueberhaupt geloest werden sollen
6 # sup_sols - Liste der bisher eingesetzten Teilloesungen

```

```

7 # make_coefs_rational - TRUE oder FALSE je nachdem ob numerisch bestimmte
  Werte rationalisiert werden sollen. Achtung: Hierbei koennen Fehler
  entstehen, siehe Abschnitt 6.3
8 # v - Substituierte Variable des Inversionsmodells (siehe Abschnitt 7.3)
9 # Output:
10 # Liste aller Teilloesungen die berechnet werden konnten
11 # Bemerkungen:
12 # - Falls Loesungen nicht bestimmt werden koennen weil die von zu hohem Grad
  sind und mehrere Variablen enthalten wird ein Fehler ausgegeben und die
  Aktuelle Teilloesung sowie der nicht loesbare Faktor in der Datei "lsg<
  filecounter>" abgespeichert. filecounter ist eine globale Variable die
  initial 1 gesetzt wird und dann hochzaehlt.
13 # - Falls Approximationen verwendet werden wird eine Warnung ausgegeben.
14 get_solutions := proc(Id,varnr,vars,sup_sols:=[],make_coefs_rational:=false,
  v:=1)
15   local var,p,k:=1,i,fac,facs,fac_deg,nst,nstwert,nsten:=[],alle_nsten,Id2,
     sub_sol,sub_sols,sols:=[],sId,svars,svarnr,ssup_sols,v2,warnung:=false;
16   global filecounter;
17   # wenn filecounter noch nicht gesetzt ist (also im initialen Aufruf), dann
     1 setzen.
18   if filecounter = 'filecounter' then
19     filecounter := 1;
20   fi;
21   # Zu loesende Variable bestimmen
22   var := vars[varnr];
23   # Erstes Nicht-0 Polynom suchen
24   p := Id[k];
25   while p=0 do
26     k := k+1;
27     if(k<= nops(Id)) then
28       p := Id[k];
29     fi;
30   od;
31   # Wenn die letzte Variable des Systems geloest wird, muss anders gerechnet
     werden, da jetzt alle verbleibenden Polynome einbezogen werden muessen
32   if varnr = 1 then
33     # Dazu: Finde Menge der gemeinsamen Faktoren, d.h. den groessten
     gemeinsamen Teiler
34     while (k<nops(Id)) do
35       k := k+1;
36       p := gcd(p,Id[k]);
37     od;
38     # jetzt die Nst. bestimmen.
39     # falls ein Produkt, dann Faktoren einzeln betrachten
40     if type(p,'*') then
41       facs := op(p);
42     else
43       facs := [p];
44     fi;

```

```
45 # Fuer jeden Faktor..
46 for fac in facs do
47   # von mehrfachen Faktoren Exponent entfernen
48   if type(fac,'**') then
49     fac := op(1,fac);
50   fi;
51   # den Grad bestimmen
52   fac_deg := degree(fac);
53   if fac_deg = FAIL then fac_deg:=5 fi;
54   # Bei kleinem Grad exakt rechnen
55   if (fac_deg<4) then
56     alle_nsten := [solve(fac,var)];
57   else # sonst numerisch
58     alle_nsten := [fsolve(fac,var)];
59   fi;
60   # Alle Nullstellen auf Gueltigkeit ueberpruefen und Liste der gueltigen
    Nst. zurueck geben
61   for nst in alle_nsten do
62     # dies war letzte Variable, hier kann Gueltigkeit immer geprueft
        werden
63     if evalf(nst) > 0 then
64       # Spezialfall Inversionsmodell: Substituiere bisherige Loesung in v
        , dies darf nicht 0 werden
65       v2 := subs(var=nst,v);
66       for i from 1 to nops(sup_sols) do
67         v2 := subs(vars[-i]=sup_sols[-i],v2); # sup_sols enhaelt die
            letzten paar Variablen, daher negative Indizes verwenden
68       od;
69       if(simplify(v2) <> 0) then:
70         # Nullstelle an Liste an fuegen
71         nsten := [op(nsten),[nst]];
72       fi;
73     fi;
74   od;
75 od;
76 # Fuer jeden Faktor sind alle gueltigen Loesungen bestimmt worden, diese
    Liste zurueck geben
77 return nsten;
78 # Wenn varnr nicht 1 ist, koennen mehrere Variablen auftreten
79 else
80   # Falls ein Produkt, dann Faktoren einzeln betrachten
81   if type(p,'*') then
82     facs := op(p);
83   else
84     facs := [p];
85   fi;
86   # Fuer jeden Faktor
87   for fac in facs do
88     # nur wenn gueltige Loesungen ueberhaupt moeglich sind
```

```

89   if (StringTools[Search]("-",convert(fac,string))<>0) then
90     # falls mehrfacher Faktor, Exponent entfernen
91     if type(fac,'**') then
92       fac := op(1,fac);
93     fi;
94     # wenn der Grad klein ist, exakt loesen, sonst numerisch
95     fac_deg := degree(fac,var);
96     # Gradbestimmung kann fehlschlagen, z.b. wenn Wurzel enthalten, dann
97     # kann aber auch nicht exakt geloest werden
98     if fac_deg = FAIL then fac_deg:=5; fi;
99     if fac_deg < 4 then
100      alle_nsten := [solve(fac,var)];
101    else
102    # numerisch loesen geht nur wenn keine anderen variablen drin sind
103    if nops(indets(fac,atomic) intersect convert(vars[1..varnr],set))=1
104      then
105      alle_nsten := [fsolve(fac,var)];
106      # wenn noch keine Warnung ausgegeben wurde, muss das jetzt
107      # gemacht werden
108      if not warnung then
109        print("Warnung: Numerische Loesungen verwendet. Moeglicherweise
110              sind Loesungen verloren gegangen.");
111        warnung := true; # die Warnung muss nur ein mal ausgegeben
112        # werden
113      fi;
114      # grad > 4 und mehrere variablen kann nicht geloest werden
115    else
116    # also die halbe Loesung abspeichern und Fehler ausgeben
117    # Funktionsparameter koennen nicht gespeichert werden, also erst
118    # Kopien anfertigen, dann speichern
119    sId := Id; svars := vars; svarnr:=varnr; ssup_sols:=sup_sols;
120    save(sId,svars,svarnr,ssup_sols,cat("lsg",filecounter));
121    print(cat("Faktor nicht loesbar: ",convert(fac,string)));
122    print(cat("Teilloesung abgespeichert in: lsg",filecounter));
123    filecounter := filecounter + 1;
124    alle_nsten := [];
125    fi;
126    # Wenn Maple fsolve(ausdruck) nicht loesen kann, dann kommt 'fsolve
127    # (ausdruck)' zurueck
128    # Dieser fall muss noch einzeln abgefangen werden:
129    if StringTools[Search]("fsolve(",convert(alle_nsten,string)) <> 0
130      then
131      sId := Id; svars := vars; svarnr:=varnr; ssup_sols:=sup_sols;
132      save(sId,svars,svarnr,ssup_sols,cat("lsg",filecounter));
133      print(cat("Numerische Loesung ist fehlgeschlagen fuer Faktor: ",
134              convert(fac,string)));
135      print(cat("Teilloesung abgespeichert in: lsg",filecounter));
136      filecounter := filecounter + 1;
137      alle_nsten := [];

```

```
129     fi;
130 fi;
131 # Nullstellen wurden (falls moeglich) bestimmt, also nun die
      gueltigen (= postiven) Nullstellen herausfiltern
132 nsten := [];
133 for nst in alle_nsten do
134     # In der Loesung koennen auch andere variablen enthalten sein aus
      der exakten Loesung. Nur wenn keine andere Variable enthalten
      ist, kann Gueltigkeit geprueft werden.
135     if nops(indets(nst,atomic) intersect convert(vars[1..varnr-1],set))
      =0 then
136         if evalf(nst) > 0 then
137             nsten := [op(nsten),nst];
138         fi;
139     # Ansonsten einfach hinzunehmen, da Gueltigkeitspruefung nicht
      moeglich
140     else
141         nsten := [op(nsten),nst];
142     fi;
143 od;
144 # Fuer jede gueltige Nullstelle...
145 for nst in nsten do
146     # Im Spezialfall des Inversionsmodells muss v ueberprueft werden, d
      .h. es muss kontrolliert werden ob v bei einsetzen der
      aktuellen Teilloesung nicht schon 0 wird.
147     v2 := subs(var=nst,v);
148     for i from 1 to nops(sup_sols) do
149         v2 := subs(vars[-i]=sup_sols[-i],v2); # sup_sols enthaelt die
      letzten paar Variablen, darum negative Indizes verwenden
150     od;
151     # Nur wenn nicht 0 herauskommt ist eine gueltige Loesung moeglich.
152     if simplify(v2) <> 0 then
153         # Die Nullstelle in die anderen Polynome einsetzen
154         # Gegebenenfalls Koeffizienten kuenstlich 'exakt' machen
155         if make_coefs_rational then
156             Id2 := factor(convert(simplify(map(numer,subs(var=nst,subs(fac
      =0,Id)))),rational));
157         else
158             Id2 := factor(simplify(map(numer,subs(var=nst,subs(fac=0,Id)))
      ));
159         fi;
160         # Nach dem Einsetzen rekursiven Aufruf starten
161         sub_sols := get_solutions(Id2,varnr-1,vars,[nst,op(sup_sols)],
      make_coefs_rational,v);
162         # Jede moegliche Erweiterung mit bisheriger Loesung verknuepfen
163         for sub_sol in sub_sols do
164             sols := [op(sols),[op(sub_sol),nst]];
165         od;
166     fi;
```

```

167     od;
168     fi;
169     od;
170     # Fuer alle gueltigen Faktoren sind alle gueltigen Loesungen bestimmt,
        also zurueckgeben
171     return sols;
172     fi;
173 end proc;

```

B Plackett-Luce

Für das Plackett-Luce-Modell existieren 5 Prozeduren: `testrun_pl` (Zeile 8) arbeitet alle Fälle von statistischen Daten ab, in denen genau zwei Permutationen genau einmal auftauchen und versucht die Lösungen zu bestimmen; `test_first_polynomial_pl` (Zeile 37) betrachtet die selben Fälle auf vereinfachte Weise indem nur das erste Polynom ausgewertet wird und berechnet keine Lösungen; `MLfct_pl` (Zeile 63) erstellt zu gegebenen statistischen Datensatz die Likelihood-Funktion; `prob_pl` (Zeile 82) berechnet die Wahrscheinlichkeit einer konkreten Permutation; `var_params_pl` (Zeile 101) erstellt einen allgemeinen Parametervektor für das Modell.

Paket plackett

```

1 # Funktion zur automatischen Abarbeitung aller Faelle von statistischen Daten
        in denen genau zwei Permutationen genau ein mal auftreten fuer die
        Problemgroesse n
2 # Input:
3 # n - Problemgroesse
4 # Output:
5 # - Liste der n!-1 Loesungen
6 # Bemerkungen
7 # - Wie zur Funktion get_solutions beschrieben koennen nicht immer alle
        Faelle geloest werden. In so einem Fall werden Fehlermeldungen ausgegeben
        und Teilloesungen abgespeichert
8 testrun_pl := proc(n)
9     local i,j,a,L,Id,Sn,allsols:=[],sols;
10    Sn := S(n);
11    for i from 2 to n! do
12        # Den aktuellen Fall ausgeben, damit man waehrend der Rechnung sehen kann
            , wo es gerade moeglicherweise lange dauert
13        print(Sn[i]);
14        # Stat. Datensatz erstellen
15        a := [1,0$i-2,1,0$(n!-i)];
16        # Likelihoodfunktion berechnen
17        L := MLfct_pl(n,a);
18        # setze vorab x1=1 damit die Rechnung moeglichst einfach ist
19        Id := factor(Basis(map(numer,convert(grad(subs(x1=1,L),[x|(2..n)]),list)
            ),plex(x|(2..n))));
20        # Loesungen bestimmen

```

```

21   sols := get_solutions(Id,n-1,[x|(2..n)]);
22   # anschliessend x1=1 an die Ergebnisse anfüegen
23   for j from 1 to nops(sols) do
24     sols[j] := [1,op(sols[j])];
25   od;
26   # Loesungen zur Liste aller Loesungen hinzufuegen
27   allsols := [op(allsols),sols];
28   od;
29   return allsols;
30 end proc;
31
32 # Funktion zur oberflaechlichen Abarbeitung aller Faelle von statistischen
    Daten in denen genau zwei Permutationen genau ein mal auftreten fuer die
    Problemgrosse n. Hier wird nur die Groebnerbasis berechnet und
    nachgeguckt ob das erste Polynom in der Groebnerbasis mit gueltigen
    Parametern loesbar ist oder nicht. Es werden keine Loesungen berechnet
33 # Input:
34 # n - Problemgrosse
35 # Output:
36 # - Liste der Laenge n!-1, welche die Werte 'keine' und 'moeglich' enthaelt.
    'keine' bedeutet, das sicher keine gueltigen Loesungen existieren, '
    moeglich' bedeutet, dass die Existenz von Loesungen nicht am ersten
    Polynom der Groebnerbasis abgelesen werden kann.
37 test_first_polynomial_pl := proc(n)
38   local Sn,i,L,Id,p,sols:=[];
39   Sn := S(n);
40   for i from 2 to n! do
41     print(Sn[i]);
42     L := MLfct_pl(n,[1,0$i-2,1,0$(n!-i)]);
43     # setze vorab x1=1 damit die Rechnung moeglichst einfach ist
44     Id := Basis(map(numer,convert(grad(subs(x1=1,L),[x|(2..n)]),list)),plex(
        x|(2..n)));
45     # Faktorisiere nur erstes Polynom und schaue nach ob Loesung ueberhaupt
        moeglich
46     p := factor(Id[1]);
47     if StringTools[Search]("-",convert(p,string)) <> 0 then
48       sols := [op(sols),'moeglich'];
49     else
50       sols := [op(sols),'keine'];
51     fi;
52   od;
53   sols;
54 end proc;
55
56 # Funktion zur Erstellung der Likelihood-Funktion bei gegebenem Vektor
    statistischer Daten
57 # Input:
58 # n - Problemgrosse

```

```

59 # a - Statistischer Datenvektor der Laenge n!. a[i]=k, bedeutet, dass die i-
    te Permutation in S_n (Wortschreibweise, lexikographisch sortiert) genau
    k mal aufgetreten ist.
60 # params - (optional) Vektor der Parameter. Default ist [x1,x2...,xn].
61 # Output:
62 # L - Likelihoodfunktion. Dies ist eine rationale Funktion in uebergebenen
    Parametern params
63 MLfct_pl := proc(n,a,params := var_params_pl(n))
64   local Sn,L,k,pi,fac,i,j;
65   if(nops(a) = n!) then
66     Sn := S(n);
67     # Likelihoodfunktion ist Produkt der Wahrscheinlichkeiten der
        auftretenden Permutationen jeweils mit Vielfachheit (Exponent) ihres
        Vorkommens
68     L := mul(mul(params[i]/add(params[Sn[k][j]],j=i..n),i=1..n)**a[k],k=1..n
        !);
69     return L;
70   else
71     print(cat("Fehler: Problemgroesse ist ",n,",aber Datenvektor der Laenge
        ",nops(a)," erhalten."));
72     return;
73   fi;
74 end proc;
75
76 # Funktion zur berechnung der Wahrscheinlichkeit einer bestimmten Permutation
77 # Input:
78 # pi - Permutation, dessen Wahrscheinlichkeit berechnet werden soll.
79 # params - (optional) Liste von Parametern des Modells. Default ist [x1,x2
    ,...,xn].
80 # Output:
81 # L - rationaler Ausdruck in den Parametern, welche die Wahrscheinlichkeit
    der gegebenen Permutation bezueglich der gegebenen Paramter ausdrueckt.
82 prob_pl := proc(pi,params:=var_params_pl(nops(pi)))
83   local n,j,L,substitutions;
84   if(nops(pi) <> nops(params)) then:
85     print(cat("Fehler: Problemgroesse ",nops(pi)," und Paramterzahl ",nops(
        params)," stimmen nicht ueberein."));
86     return;
87   fi;
88   n := nops(pi);
89   # Formel fuer die Wahrscheinlichkeit der Permutation pi im Plackett-Luce-
        Modell
90   L := mul(cat(x,i)/add(cat(x,pi[k]),k=i..n),i=1..n);
91   # Alle noetigen Substitutionen erst zu Menge zusammenfassen, damit sie
        gleichzeitig ausgefuehrt werden
92   substitutions := {};
93   for j from 1 to n do
94     substitutions := substitutions union { cat(x,j)=params[j] };
95   od;

```

```

96 L := subs(substitutions,L);
97 L;
98 end proc;
99
100 # Hilfsfunktion die die Standardliste fuer allgemeine Parameter zurueck gibt.
101 var_params_pl := proc(n::nonnegint)
102   return [x|(1..n)];
103 end proc;

```

C Bradley-Terry-Mallows

Für das Bradley-Terry-Mallows-Modell existieren 7 Prozeduren: `testrun_btm` (Zeile 9) arbeitet alle Fälle von statistischen Daten ab, in denen genau zwei Permutationen genau einmal auftauchen und versucht die Lösungen zu bestimmen; `test_first_polynomial_btm` (Zeile 34) betrachtet die selben Fälle auf vereinfachte Weise und berechnet keine Lösungen; `MLfct_btm` (Zeile 60) erstellt zu gegebenen statistischen Datensatz die Likelihood-Funktion; `prob_btm` (Zeile 75) berechnet die Wahrscheinlichkeit einer konkreten Permutation; `f_btm` und `c_btm` (Zeilen 80 und 85) sind Hilfsfunktionen für die beiden Vorgenannten Funktionen und berechnen die Werte der Funktionen f und c^* aus Definition 4.7; `var_params_btm` (Zeile 90) erstellt einen allgemeinen Parametervektor für das Modell.

Paket btm

```

1 # Funktion zur automatischen Abarbeitung aller Faelle von statistischen Daten
   in denen genau zwei Permutationen genau ein mal auftreten fuer die
   Problemgroesse n
2 # Input:
3 # n - Problemgroesse
4 # make_coeffs_rational - TRUE oder FALSE je nachdem ob numerisch bestimmte
   Werte rationalisiert werden sollen. Achtung: Hierbei koennen Fehler
   entstehen, siehe Abschnitt 6.3. Default ist FALSE
5 # Output:
6 # - Liste der n!-1 Loesungen
7 # Bemerkungen
8 # - Wie zur Funktion get_solutions beschrieben koennen nicht immer alle
   Faelle geloest werden. In so einem Fall werden Fehlermeldungen ausgegeben
   und Teilloesungen abgespeichert
9 testrun_btm := proc(n,make_coeffs_rational:=false)
10   local i,j,a,L,Id,Sn,allsols:=[],sols;
11   Sn := S(n);
12   for i from 2 to n! do
13     print(Sn[i]);
14     a := [1,0$(i-2),1,0$(n!-i)];
15     L := MLfct_btm(n,a);
16     # Setze vorab v1=1 damit die Rechnung moeglichst einfach ist
17     Id := factor(Basis(map(numer,convert(grad(subs(v1=1,L),[v|(2..n)]),list)
   )),plex(v|(2..n)));
18     sols := get_solutions(Id,n-1,[v|(2..n)],[],make_coeffs_rational);

```

```

19 # Anschliessend noch v1=1 an die Ergebnisse anfüegen
20 for j from 1 to nops(sols) do
21     sols[j] := [1,op(sols[j])];
22 od;
23 # Loesungen zur Liste der Loesungen hinzufuegen
24 allsols := [op(allsols),sols];
25 od;
26 allsols;
27 end proc;
28
29 # Funktion zur oberflaechlichen Abarbeitung aller Faelle von statistischen
    Daten in denen genau zwei Permutationen genau ein mal auftreten fuer die
    Problemgroesse n. Hier wird nur die Groebnerbasis berechnet und
    nachgeguckt ob das erste Polynom in der Groebnerbasis mit gueltigen
    Parametern loesbar ist oder nicht. Es werden keine Loesungen berechnet
30 # Input:
31 # n - Problemgroesse
32 # Output:
33 # - Liste der Laenge n!-1, welche die Werte 'keine' und 'moeglich' enthaelt.
    'keine' bedeutet, das sicher keine gueltigen Loesungen existieren, '
    moeglich' bedeutet, dass die Existenz von Loesungen nicht am ersten
    Polynom der Groebnerbasis abgelesen werden kann.
34 test_first_polynomial_btm := proc(n)
35     local Sn,i,L,Id,p,sols:=[];
36     Sn := S(n);
37     for i from 2 to n! do
38         print(Sn[i]);
39         L := MLfct_btm(n,[1,0$i-2,1,0$(n!-i)]);
40         # setze vorab v1=1 damit die rechnung moeglichst einfach ist
41         Id := Basis(map(numer,convert(grad(subs(v1=1,L),[v||(2..n)]),list)),plex(
            v||(2..n)));
42         # Faktorisiere nur erstes Polynom und schau nach ob Loesung ueberhaupt
            moeglich
43         p := factor(Id[1]);
44         if StringTools[Search]("-",convert(p,string)) <> 0 then
45             sols := [op(sols),'moeglich'];
46         else
47             sols := [op(sols),'keine'];
48         fi;
49     od;
50     sols;
51 end proc;
52
53 # Funktion zur Erstellung der Likelihood-Funktion bei gegebenem Vektor
    statistischer Daten
54 # Input:
55 # n - Problemgroesse
56 # a - Statistischer Datenvektor der Laenge n!. a[i]=k, bedeutet, dass die i-
    te Permutation in S_n (Wortschreibweise, lexikographisch sortiert) genau

```

```

    k mal aufgetreten ist.
57 # params - (optional) Vektor der Parameter. Default ist [v1,v2...,vn].
58 # Output:
59 # L - Likelihoodfunktion. Dies ist eine rationale Funktion in uebergebenen
    Parametern params
60 MLfct_btm := proc(n,a,params:=var_params_btm(n))
61   local c,L,Sn;
62   c := c_btm(params);
63   Sn := S(n);
64   L := mul(prob_btm(Sn[i],params,c)**a[i],i=1..n!);
65   return L;
66 end proc;
67
68 # Funktion zur berechnung der Wahrscheinlichkeit einer bestimmten Permutation
69 # Input:
70 # pi - Permutation, dessen Wahrscheinlichkeit berechnet werden soll.
71 # params - (optional) Liste von Parametern des Modells. Default ist [v1,v2
    ,...,vn].
72 # c - (optional) Vorab berechneter Normierungswert c. Dieser kann mit Hilfe
    der Funktion c_btm vorab berechnet werden, falls diese Funktion mehrfach
    aufgerufen werden soll. Default ist der Wert der Funktion c_btm(params)
73 # Output:
74 # - rationaler Ausdruck in den Parametern, welche die Wahrscheinlichkeit der
    gegebenen Permutation bezueglich der gegebenen Paramter ausdrueckt.
75 prob_btm := proc(pi,params,c:=c_btm(params))
76   return 1/c*f_btm(pi,params);
77 end proc;
78
79 # Implementierung der Funktion f aus Definition 4.7
80 f_btm := proc(pi,params)
81   return mul(params[pi[i]]**(nops(params)-i),i = 1..nops(params));
82 end proc;
83
84 # Implementierung der Funktion c aus Definition 4.7
85 c_btm := proc(params)
86   return add(f_btm(pi,params),pi in S(nops(params)));
87 end proc;
88
89 # Hilfsfunktion die die Standardliste fuer allgemeine Parameter zurueck gibt.
90 var_params_btm := proc(n::nonnegint)
91   return [v||(1..n)];
92 end proc;

```

D Inversions-Modell

Für das Inversions-Modell existieren 10 Prozeduren: `testrun_inv` (Zeile 9) arbeitet alle Fälle von statistischen Daten ab, in denen genau drei Permutationen genau einmal auftauchen und versucht die Lösungen zu bestimmen; `solve_case_inv` (Zeile 33) versucht

einen einzelnen Fall zu gegebener Problemgröße und gegebenem Parametervektor zu lösen; `test_criterion_inv` (Zeile 68) testet bei den selben Parametern die notwendige Bedingung aus Satz 7.3; `gls_inv` (Zeile 95) ist bei ebenfalls selbigen Parametern eine Hilfsfunktion zur Erstellung des zu lösenden Gleichungssystems und übernimmt dabei die Auflösung der Nebenbedingung und Substitution in die Likelihood-Funktion; `MLfct_inv` (Zeile 126) erstellt zu gegebenen statistischen Datensatz die Likelihood-Funktion; `N_inv` (Zeile 138) erstellt die Nebenbedingung zu gegeben Modellparametern; `prob_inv` (Zeile 149) berechnet die Wahrscheinlichkeit einer konkreten Permutation; `rand_params_inv` (Zeile 180) erstellt zu gegebene Problemgröße zufällige Modellparameter in Matrixform für heuristische Versuche, die aber nicht notwendigerweise die Nebenbedingung erfüllen; `var_params_inv` (Zeile 194) erstellt eine allgemeine Parametermatrix für das Modell; `var_list_inv` (Zeile 207) gibt selbige Parameter in Listenform zurück.

Paket inversions

```

1 # Funktion zur automatischen Abarbeitung aller Faelle von statistischen Daten
  in denen genau drei Permutationen genau ein mal auftreten fuer die
  Problemgroesse n
2 # Input:
3 # n - Problemgroesse
4 # make_coeffs_rational - TRUE oder FALSE je nachdem ob numerisch bestimmte
  Werte rationalisiert werden sollen. Achtung: Hierbei koennen Fehler
  entstehen, siehe Abschnitt 6.3. Default ist FALSE.
5 # Output:
6 # - Liste der  $(n!-1)(n!-2)/2$  gefundenen Loesungen
7 # Bemerkungen
8 # - Wie zur Funktion get_solutions beschrieben koennen nicht immer alle
  Faelle geloest werden. In so einem Fall werden Fehlermeldungen ausgegeben
  und Teilloesungen abgespeichert
9 testrun_inv := proc(n,make_coeffs_rational:=false)
10 local Sn,i,j,a,sol,solutions:=[];
11 Sn := S(n);
12 for i from 2 to n! do
13   for j from i+1 to n! do
14     # datenvektor erstellen
15     a := [1,0$(i-2),1,0$(j-i-1),1,0$(n!-j)];
16     print(cat(Sn[1],Sn[i],Sn[j]));
17     sol := solve_case_inv(n,a,make_coeffs_rational);
18 solutions := [op(solutions),sol];
19   od;
20 od;
21 return solutions;
22 end proc;
23
24 # Diese Funktion versucht einen konkreten Fall im Inversionsmodell mit Hilfe
  der Funktionen get_gls und get_solutions zu loesen und gibt die
  gefundenen Loesungen zurueck
25 # Input:
26 # n - Problemgroesse

```

```
27 # a - Vektor statistischer Daten der Laenge n!. a[i]=k, bedeutet, dass die i-
    te Permutation in S_n (Wortschreibweise, lexikographisch sortiert) genau
    k mal aufgetreten ist.
28 # make_coeffs_rational - TRUE oder FALSE je nachdem ob numerisch bestimmte
    Werte rationalisiert werden sollen. Achtung: Hierbei koennen Fehler
    entstehen, siehe Abschnitt 6.3. Default ist FALSE.
29 # Ouput:
30 # - Liste der gefundenden Loesungen
31 # Bemerkungen
32 # - entsprecheden den Beschraenkungen der Funktion get_solutions (siehe dort
    und Abschnitt 6.3) koennen nicht immer alle Loesungen bestimmt werden. In
    dem Fall werden entsprechende Fehlermeldungen ausgegeben und
    Teilloesungen abgespeichert.
33 solve_case_inv := proc(n,a,make_coeffs_rational:=false)
34   local varlist,glS,L,v,Id,sols,i,j;
35   # Erst notwendiges Kriterium pruefen
36   if not test_criterion_inv(n,a) then
37     print("Notwendige Bedingung nicht erfuehlt. Keine Loesungen.");
38     return [];
39   else
40     print("Notwendige Bedingung erfuehlt. Berechne Loesungen...");
41     # Gleichungssystem mit entsprecheden Substitutionen erstellen
42     glS := glS_inv(n,a);
43     L := glS[2];
44     v := glS[3];
45     varlist := glS[4];
46     Id := factor(Basis(PolyId(map(numer,glS[1])),plex(op(varlist))));
47     # Loesungen bestimmen
48     sols := get_solutions(Id,n*(n-1)/2,varlist,[],make_coeffs_rational,v);
49     # in allen Loesungen v12 und die anderen v_ij=1 hinzufuegen
50     for i from 1 to nops(sols) do
51       v := glS[3];
52       for j from 1 to nops(varlist) do
53         v := subs(varlist[j]=sols[i][j],v);
54       od;
55       v := simplify(v);
56     sols[i] := [op(sols[i]),v,1$(nops(sols[i])-1)];
57     od;
58     return sols;
59   fi;
60 end proc;
61
62 # Funktion zum Testen des notwendigen Kriteriums zur Existenz von Maxima im
    Inversions-Modell, vgl. Satz 7.3
63 # Input:
64 # n - Problemgroesse
65 # a - Vektor statistischer Daten der Laenge n!. a[i]=k, bedeutet, dass die i-
    te Permutation in S_n (Wortschreibweise, lexikographisch sortiert) genau
    k mal aufgetreten ist.
```

```

66 # Output
67 # - TRUE, falls notwendiges Kriterium erfuehlt, FALSE sonst
68 test_criterion_inv := proc(n,a)
69   local i,Sn:=S(n),invs,invset:={},noninvset:={};
70   # Berechne die Inversionen und Nicht-Inversionen jeder permutation
71   invs := map(get_inversions,Sn);
72   # Vereinige die entsprechenden Mengen fuer alle auftauchenden Permutationen
73   for i from 1 to n! do
74     if a[i] <> 0 then
75       invset := invset union invs[i][1];
76 noninvset := noninvset union invs[i][2];
77     fi;
78   od;
79   # Jedes Tupel muss als Inversion und als Nicht-Inversion gefunden worden
      sein
80   if nops(invset) <> n*(n-1)/2 or nops(noninvset) <> n*(n-1)/2 then
81     return false;
82   else
83     return true;
84   fi;
85 end proc;
86
87 # Hilfsfunktion zur Erstellung des Gleichungssystems bei gegebenem
      statistischem Datenvektor
88 # Input:
89 # n - Problemgroesse
90 # a - Vektor statistischer Daten der Laenge n!. a[i]=k, bedeutet, dass die i-
      te Permutation in S_n (Wortschreibweise, lexikographisch sortiert) genau
      k mal aufgetreten ist.
91 # Output
92 # - Liste mit 4 Werten. Der erste ist die Liste der partiellen Ableitungen
      der Likelihoodfunktion, der zweite ist die Likelihoodfunktion selbst, der
      dritte ist die Substitution die fuer v12 durchgefuehrt wurde und der
      vierte ist die Liste der verbleibenden Parameter
93 # Bemerkungen:
94 # Durch die hohe Redundanz des Inversions-Modells wird v_ij=1 gesetzt fuer {i
      ,j} verschieden von {1,2}, ferner wird die Nebenbedingung des Modells
      nach v_12 aufgeloeset und dies in der Likelihoodfunktion substituiert. Es
      bleibt eine Funktion in der Parametern u_ij uebrig.
95 gls_inv := proc(n,a)
96   local vars,var_list,i,j,N,L,Sn;
97   Sn := S(n);
98   var_list := sort(var_list_inv(n));
99   vars := var_params_inv(n);
100  # alle v[ij] bis auf v[12] koennen 1 gesetzt werden
101  for i from 2 to n do
102    for j from 1 to n-1 do
103      if(i <> 2 or j <> 1) then
104        vars[i][j] := 1;

```

```
105     fi;
106     od;
107 od;
108 # Nebenbedingung nach v12 auflösen
109 N := add(prob_inv(pi,vars),pi in Sn)-1;
110 vars[2,1] := solve(N,vars[2,1]);
111 L := MLfct_inv(n,a,vars);
112 # Ableitungen in die Richtungen u[ij] genuegen, da die anderen Variablen
    schon festgelegt sind.
113 return([convert(grad(L,var_list[1..(n*(n-1)/2])),list),
114         L,
115         vars[2,1],
116         var_list[1..(n*(n-1)/2)]]);
117 end proc;
118
119 # Funktion zur Berechnung der Likelihood-Funktion im Inversions-Modell
120 # Input:
121 # n - Problemgroesse
122 # a - statistische Daten. 1-dim Liste der Laenge n!
123 # params - 2-dim Liste der Dimesion n*n, Parameter des Inversions-Modells in
    Matrixform
124 # Output:
125 # L - Likelihoodfunktion
126 MLfct_inv := proc(n,a,params)
127   local Sn,L;
128   Sn := permute(n);
129   L := mul(prob_inv(Sn[i],params)**a[i],i=1..n!);
130   return L;
131 end proc;
132
133 # Funktion zur Erstellung der Nebenbedingung im Inversionsmodell
134 # Input:
135 # n - Problegroesse
136 # Output:
137 # Nebenbedingung N, vlg. Abschnitt 4.3
138 N_inv := proc(n,params:=var_params_inv(n))
139   local Sn := S(n);
140   return add(prob_inv(pi,params),pi in Sn)-1;
141 end proc;
142
143 # Funktion zur Berechnung der Wahrscheinlichkeit einer Permutation im
    Inversions-Modell
144 # Input:
145 # pi - Permuatation in S_n
146 # params - 2-dim Liste der Dimesion n*n, Parameter des Inversions-Modells in
    Matrixform
147 # Output:
148 # Wahrscheinlichkeit der gegebenen Permutation im Inversions-Modell
149 prob_inv := proc(pi,params)
```

```

150 local n,tau,i,j,L,substitutions;
151 n:= nops(pi);
152 tau := inverse(pi);
153 L:=1;
154 for i from 1 to n do:
155     for j from i+1 to n do:
156         if(tau[i]<tau[j]) then:
157             L := L*cat(u,i,j);
158         else
159             L := L*cat(v,i,j);
160         fi;
161     od;
162 od;
163 # W'keit ist nun allgemein berechnet mit u_ij v_ij, nun uebergebene
    Parameter substituieren
164 # Substitutionen zuerst zu Menge zusammenfassen damit alle gleichzeitig
    ausgefuehrt werden
165 substitutions := {};
166 for i from 1 to n do:
167     for j from i+1 to n do:
168         if(tau[i]<tau[j]) then:
169             substitutions := substitutions union { cat(u,i,j)=params[i][j] };
170         else
171             substitutions := substitutions union { cat(v,i,j)=params[j][i] };
172         fi;
173     od;
174 od;
175 L := subs(substitutions,L);
176 L;
177 end proc;
178
179 # Hilfsfunktion fuer heuristische Untersuchungen die einen Satz zufaelligen
    Paramter zurueckgibt. Es wird fuer jeden Paramter (gleichverteilt) ein
    Wert aus der Menge {1/10,2/10,...,20/10} eingesetzt.
180 rand_params_inv := proc(n)
181     local i,j,roll,params;
182     roll := rand(1..20)/10;
183     params := [[0$n]$n];
184     for i from 1 to n do:
185         for j from i+1 to n do:
186             params[i,j] := roll();
187             params[j,i] := roll();
188         od;
189     od;
190     return params;
191 end proc;
192
193 # Hilfsfunktion zur Erstellung einer allgemeinen Paramtermatrix (u_ij,v_ij)
194 var_params_inv := proc(n)

```

```
195 local i,j,params;
196 params := [[0$n]$n];
197 for i from 1 to n do:
198   for j from i+1 to n do:
199     params[i,j] := cat(u,i,j);
200     params[j,i] := cat(v,i,j);
201   od;
202 od;
203 return params;
204 end proc;
205
206 # Hilfsfunktion zur Erstellung einer Liste der allgemeinen Parameter
207 var_list_inv := proc(n)
208   local list,i,j,k;
209   list := [0$n*(n-1)];
210   k := 1;
211   for i from 1 to n do:
212     for j from i+1 to n do:
213       list[k] := cat(u,i,j);
214       list[k+1] := cat(v,i,j);
215       k := k+2;
216     od;
217   od;
218   return list;
219 end proc;
```

Literaturverzeichnis

- [Ada00] William Adams, *An introduction to Gröbner bases*, third ed., American Mathematical Society, 2000.
- [BGK86] W. Boege, R. Gebauer, and H. Kredel, *Some examples for solving systems of algebraic equations by calculating Groebner bases*, *Journal of Symbolic Computation* **2** (1986), 83–98.
- [BT52] R.A. Bradley and M.E. Terry, *Rank analysis of incomplete block designs. I. The method of paired comparisons.*, *Biometrika* **39** (1952), 324–245.
- [CFV91] Douglas E. Critchlow, Michael A. Flinger, and Joseph S. Verducci, *Probability models on rankings*, *Journal of Mathematical Psychology* **35** (1991), 294–318.
- [CLO00] David Cox, John Little, and Donal O’Shea, *Ideals, varieties and algorithms: An introduction to computational algebraic geometry and commutative algebra*, third ed., Springer, 2000.
- [Dan50] H.A. Daniels, *Rank correlation and population models*, *Journal of the Royal Statistical Society, Series B: Statistical Methodology* **12** (1950), 171–181.
- [Dav88] H.A. David, *The method of paired comparisons*, Oxford University Press: New York, 1988.
- [Dia88] P. Diaconis, *Group representations in probability and statistics*, Institute of Mathematical Statistics: Hayward, California, 1988.
- [GKM⁺04] Shuhong Gao, Erich Kaltofen, John May, Zhengfeng Yang, and Lihong Zhi, *Approximate factorization of multivariate polynomials via differential equations*, *Proceedings of the 2004 international symposium on Symbolic and algebraic computation (New York, NY, USA), ISSAC ’04, ACM, 2004*, pp. 167–174.
- [GMN⁺91] A. Gianni, T. Mora, G. Niesi, L. Robbiano, and C. Traverso, *‘One sugar cube please’ or selection strategies in the Buchberger algorithm*, *Proc. International Symposium on Symbolic and Algebraic Computation (1991)*, 49–54.
- [Hec03] André Heck, *Introduction to Maple*, 3rd ed., Springer, 2003.
- [Hul00] Klaus Hulek, *Algebraische Geometrie*, Vieweg, 2000.
- [KBS40] M.G. Kendall and B. Babington Smith, *On the method of paired comparisons*, *Biometrika* **31** (1940), 324–345.

- [Ken70] M.G. Kendall, *Rank correlation methods*, 4th ed., Griffin, 1970.
- [Luc59] R.D. Luce, *Individual choice behavior*, Wiley, 1959.
- [Mal57] C.L. Mallows, *Non-null ranking models. I.*, *Biometrika* **44** (1957), 114–130.
- [Mar95] John I. Marden, *Analyzing and modeling rank data*, 1st ed., Chapman and Hall, 1995.
- [MGH⁺05] Michael B. Monagan, Keith O. Geddes, K. Michael Heal, George Labahn, Stefan M. Vorkoetter, James McCarron, and Paul DeMarco, *Maple 10 programming guide*, Maplesoft, Waterloo ON, Canada, 2005.
- [Pla75] R.L. Plackett, *The analysis of permutations*, *Journal of the Royal Statistical Society, Series C: Applied Statistics* **23** (1975), 193–202.
- [SW11] B. Sturmfels and V. Welker, *Commutative algebra of statistical ranking*, 2011, submitted.
- [Thu27] L.L. Thurstone, *A law of comparative judgement*, *Psychological Reviews* **34** (1927), 273–286.