SAS - Makros zur Analyse und Darstellung mehrdimensionaler Punktekonfigurationen

Matthias Huerkamp

Bericht Nr. 72

April 1994

Arbeiten aus dem Sonderforschungsbereich 245
"Sprache und Situation"
Heidelberg/Mannheim

Kontaktadresse: M. Huerkamp, Dipl. Psych.
Psychologisches Institut der Universität Heidelberg
Hauptstraße 47-51, 69117 Heidelberg

Dieser Bericht bezieht sich auf Ergebnisse des Projektes B2 "Sprachliche Diskriminierung" im Rahmen des SFB 245 "Sprache und Situation". Wir danken der Deutschen Forschungsgemeinschaft für die Förderung unserer Arbeiten.

ISSN 0937-6224

Inhaltsverzeichnis

Zusammenfassung

Es werden hier SAS-Makros vorgestellt, deren gemeinsamer Zweck in der Analyse und graphischen Repräsentation mehrdimensionaler Punktekonfigurationen besteht.
Diese Konfigurationen sind z.B. das typische Ergebnis multidimensionaler Skalierungsprozeduren, wie etwa ALSCAL oder das neuere SAS-MDS.
Die Makros geben die Möglichkeit, die Konfigurationen graphisch auszugeben (G3DID), die numerische Information einer Konfiguration zu extrahieren (KOORDIST), eine Konfiguration an eine andere Konfiguration so nah wie möglich anzunähern, ohne die Skalierungsinformation zu verändern (PROKRUST), und schließlich ermöglichen sie den Vergleich zweier Konfigurationen (SIMIL). Der Programmcode der Makros ist beigefügt.

Abstract

In this report, four SAS-macros are going to be presented, whose common purpose is the analysis and graphical representation of configurations of points in a multidimensional space.
These configurations are for example the typical output of multidimensional scaling procedures like the results of ALSCAL or the newer SAS-MDS.
The macros provide the possibility to produce graphical output of the configurations (G3DID), to extract the numerical information of a configuration (KOORDIST), to transform one configuration to approximate another configuration as closely as possible without changing the scaling information (PROKRUST), and finally to compare two configurations (SIMIL). The source-code of the macros is included.

1. Einleitung

Die Aufgabe des vorliegenden "gelben Heftes"[1] besteht darin, die bei der Arbeit mit Punktekonfigurationen, wie sie das typische Ergebnis von multidimensionalen Skalierungsversuchen sind, entstandenen und entwickelten SAS-Makros einerseits im Sinne eines technischen Berichts zu dokumentieren, andererseits aber auch einer größeren Öffentlichkeit bekannt zu machen, deren Feedback der Autor hiermit ausdrücklich wünscht.
Trotz der wachsenden Erfordernis, etwa im Rahmen der Facettentheorie, mehrdimensionale Punktekonfigurationen darzustellen und zu interpretieren, sind die Mittel dies zu tun, zur Zeit nicht sehr zugänglich. Sie erfordern entweder besondere mathematische Kenntnisse, langwierige Berechnungen oder sehr spezialisierte, wenig verbreitete Software.

Der Verwendungszweck der vier Makros mag durch einen Gang durch die Historie ihrer Entwicklung besser verständlich werden.
Nachdem 1989 feststand, daß die methodische Entwicklung des Teilprojektes B2 "Sprachliche Diskriminierung" im Rahmen der Facettentheorie mit den Mitteln der Multidimensionalen Skalierung (MDS) geleistet werden sollte, wurden zunächst einmal Programme für die Berechnung der Skalierungslösungen gesucht.
In Heidelberg standen für die MDS zwei Programme zur Verfügung: MINISSA (Roskam & Lingoes 1970) und ALSCAL in seinen verschiedenen Formen, als Einzelprogramm (Young & Lewyckyj 1979) mit einer fast ebenso archaischen Bedienbarkeit wie MINISSA, und als Teile der Programmpakete Statistical Package for the Social Sciences (SPSS) (SPSS Inc. 1985) sowie des Statistical Analysis System (SAS) (Young, Lewyckyj & Takane, 1986). Die Entscheidung für den ALSCAL-Algorithmus als Teil des SAS fiel aufgrund des Optionsreichtums der ALSCAL-Prozedur, der guten Programmierbarkeit der SAS-Sprache und des flexiblen Dateisystems nicht schwer.

Zunächst war geplant gewesen, die graphische Darstellung der errechneten Punktekoordinaten mit einem PC-Programm zu bewerkstelligen; es stellte sich jedoch heraus, daß die wenigsten Programme dreidimensionale Punktekonfigurationen angemessen darstellen konnten. Auch im SAS war diese Aufgabe zunächst nicht ganz einfach, bis ein Makro gefunden wurde (Bundschuh 1989), das für unsere Zwecke adaptiert werden konnte. Das adaptierte Makro G3DID wird in Kap. 6 vorgestellt.

In der neuen Version 6.08 des SAS wurde einerseits ein Ersatz für die Prozedur ALSCAL in der SAS - eigenen Prozedur MDS verfügbar, die mit höherer numerischer Präzision arbeitet und noch mehr Optionen hat. Andererseits wurde mit SAS/Insight erstmals für die PC-Version eine Möglichkeit gegeben, dreidimensionale Punktekonfigurationen darzustellen und sogar interaktiv mit Maus-Bedienung im Raum zu rotieren. Trotzdem behält das Makro G3DID

---

[1] Für das ausdauernde Testen der Makros gilt mein Dank Kurt Imminger.

seinen Wert, weil dort numerisch definierte, nicht-interaktive Drehungen und Kippbewegungen zu realisieren sind.

Das wichtigste Mittel der Auswertung der facettentheoretisch aufbereiteten Fragestellung basiert auf der graphischen Darstellung der Punktekonfigurationen. Eine Eigenschaft der in diesem Zusammenhang errechneten dreidimensionalen MDS-Lösungen besteht darin, daß sie nur bis auf eine Anzahl zulässiger Transformationen festgelegt sind, oder anders ausgedrückt: die gesamte Information ist im **Verhältnis der Abstände der Punkte zueinander** enthalten. Um diese Information numerisch sichtbar zu machen, wurde das Makro KOORDIST entwickelt. Dieses Makro wird im Kap. 5 erläutert.

In der Untersuchung U1 des Teilprojektes B2 (Wagner, Huerkamp, Jockisch & Graumann 1990) entstand die Notwendigkeit, zwei mit unterschiedlichen Methoden gewonnene, aber äquivalente - sich punktweise entsprechende - MDS-Lösungen zu vergleichen. Dieser Vergleich ist aber erst dann möglich, wenn alle zulässigen - die Information nicht ändernden - Transformationen im Sinne der Angleichung ausgeschöpft werden. Dieses wird durch prokrustische Ähnlichkeitstransformationen mit dem Makro PROKRUST ermöglicht, welches im Kap. 3 vorgestellt wird.

Nachdem die zwei Lösungen aneinander angenähert sind und durch die Betrachtung der graphischen Darstellung in einem gemeinsamen Raum die (Un-)Ähnlichkeiten sichtbar werden, will man jedoch wissen, **wie** ähnlich die beiden Konfigurationen sind - möglichst in einem numerischen Wert zwischen 0 und 1 (bzw. -1 und +1). Wenn dieser Wert hohe Ähnlichkeit anzeigt, dann ist von Interesse, ob die nach Prokrustes-Transformation verbleibenden Unterschiede zufallsbedingt sind oder interpretierbare Unterschiede anzeigen. In Kap. 4 wird mit dem Makro SIMIL versucht, Lösungsansätze zu bieten.

Der Autor will keinesfalls den Eindruck erwecken, hier ein komplettes Set von Prozeduren für die Bearbeitung mehrdimensionaler Punktekonfigurationen zur Verfügung zu stellen; es bleibt noch vieles zu wünschen und zu tun übrig. Wie (fast) immer ist auch hier weitere Forschung vonnöten.

2. Allgemeine Überlegungen

Makros sind keine Sache von Statistik-Programmen allein. So verschiedene Software - Gattungen wie Textverarbeitungen (z.B. Word oder Word für Windows), Editoren (z.B. KEDIT), Programmiersprachen (wie etwa Makro-Assembler oder C) oder eben auch Statistik-Programme (wie SAS) bieten die Möglichkeit, Makros zu verwenden.

Was muß man sich unter einem Makro vorstellen?

Der Duden Informatik definiert zunächst weniger den Begriff 'Makro' als dessen Funktion:

> "Wenn in einem Programm mehrere Anweisungen, Befehle oder Deklarationen in einer bestimmten Reihenfolge häufig vorkommen, so ist es sinnvoll, diese abkürzend zu einer Einheit, dem Makrobefehl zusammenzufassen." (1988, S.347)

Kaltenbach, Reetz & Woerrlein (1988) definieren sehr allgemein:

> "Ein Makro oder eine Makroinstruktion ist eine Anweisung, die durch eine Folge anderer Anweisungen definiert ist." (S. 133)

Wie aber paßt folgende Definition zum vorher Gesagten?

> "Ein Macro ist ein (gespeicherter) Text, der durch einen Namen identifiziert ist. Wenn man später den Text benötigt, braucht man dann nicht mehr den Text einzugeben, sondern kann den Text durch seinen Namen aufrufen. Macros erfüllen also eine ähnliche Funktion wie Textbausteine in einem Textverarbeitungsprogramm." (Gogolok, Schuemer & Ströhlein, 1992, S.364)

Diese Definition ist mit den vorhergehenden dann in Einklang zu bringen, wenn man voraussetzt, daß der Text eben nicht einfach beliebiger Text ist, sondern Anweisungen an das umgebende Programm enthält. Das Programm würde also durch Textersetzung anstelle des Kürzels, welches das Makro identifiziert, einen längeren Text bzw. eine Anweisungsfolge setzen; dies wird dann als 'Makroexpansion' bezeichnet.

Wenn aber ein Makro wie ein Textbaustein in der Textverarbeitung funktioniert, was sind dann im Unterschied dazu die Makros in der Textverarbeitung?

Der springende Punkt ist hier, daß Anweisungen an das umgebende Programm nicht nur in Form von Texteingaben erfolgen können, sondern auch als Ein-Tasten-Eingaben oder Tastenkombinationen, oder auch durch Betätigung anderer Eingabegeräte, wie z.B. der Maus. Makros sind dann Zusammenfassungen mehrerer solcher Eingaben und können entweder über einen vereinbarten Namen aufgerufen werden, oder ihrerseits durch einen bestimmten Tastendruck aktiviert werden.

Bei den oben aufgeführten Programmiersprachen gibt es neben der Definition von Makros teilweise auch die Möglichkeit Unterprogramme zu verwenden. Makros unterscheiden sich von Unterprogrammen dadurch, daß in einem Programm ein Unterprogramm nur *einmal* existiert und bei jedem Aufruf erst an die Stelle gesprungen werden muß, an der das Unterprogramm lokalisiert ist; ein Makro hingegen ersetzt bei *jedem* Vorkommen den vereinbarten Namen durch den expandierten Text bzw. die im Text enthaltenen Anweisungen. Makros vergrößern

also das entstehende Resultat, vermeiden aber aufwendige Sprünge im Programmcode; der Programmierer muß abwägen, welches im Einzelfall die bessere Lösung ist.

Makroeinrichtung im SAS

Um das Folgende besser zu verstehen, ist es sinnvoll sich den allgemeinen Aufbau eines SAS-Programms zu verdeutlichen.

Ein SAS-Programm besteht aus einer Abfolge von Daten-Schritten (data-steps), in denen Daten definiert oder verändert werden, und - davon getrennt und unabhängig - den auf diese Daten angewandten Prozeduren (proc-steps). Programmierbar ist das SAS zunächst nur **innerhalb** eines data-steps. Eine mehrere data- und/oder proc-steps übergreifende Programmierung ist nur mithilfe einer der eigentlichen SAS-Programmausführung vorgeschalteten und übergeordneten Makroeinrichtung möglich (Graf, Bundschuh & Kruse, 1993, S. 42).

> "Die Macroeinrichtung ('macro facility') besteht aus einem Macroprozessor und einer Macrosprache ('macro processor' bzw. 'macro language');..." (Gogolok et al., 1992, S.364)

Der Makroprozessor arbeitet als Präprozessor: SAS-Programme werden zunächst auf das Vorhandensein von Elementen der Makrosprache durchsucht; erst nachdem diese verarbeitet worden sind, reicht der Präprozessor das Programm zur eigentlichen Ausführung weiter.

Ein grundlegendes Element der Makrosprache sind symbolische Variablen, sogenannte Makrovariablen. Diese sind im Namen dadurch kenntlich, daß sie als ersten Buchstaben ein '&' haben (z.B. &makvar). Solchen Makrovariablen können durch die Anweisung 'call symput' oder durch die Makroanweisung '%let' Werte zugewiesen werden. Wie der Leser vielleicht schon mutmaßt, sind Anweisungen, die an den Makroprozessor gerichtet sind, im Namen am führenden '%'-Zeichen erkennbar (%do, %end, %if, %then, %else etc.). So wird nach folgender Zuweisung:

```
%let makvar = Dies soll eine Titelzeile werden;
```

jedes Vorkommen von &makvar durch den zugewiesenen Text ersetzt. Also würde die folgende Anweisung:

```
title "&makvar";
```

nach der Expansion durch den Makroprozessor als:

```
title "Dies soll eine Titelzeile werden";
```

zur Ausführung an den SAS-Interpreter weitergegeben. Zu beachten ist, daß es sich wirklich um Textersetzung handelt; mit Makrovariablen können, selbst wenn der Text nur aus Zahlen besteht, keine Berechnungen durchgeführt werden (außer innerhalb eines Makros unter Verwendung spezieller Funktionen).

Prinzipien für die Entwicklung von SAS-Makros

Die Qualitäten eines Programmes liegen in der Korrektheit und der Effizienz; die folgenden Tips beziehen sich hauptsächlich auf die Programmierer- und Resourcen-Effizienz und auf die Fehlervermeidung.

Wenn Makros von vielen benutzt werden sollen, die mit dem Programmcode nicht vertraut sind, müssen besondere Vorkehrungen zur Benutzungssicherheit und Benutzerfreundlichkeit getroffen werden. Makros zum allgemeinen Gebrauch sollten nach Empfehlung von SAS Institute Inc. (1990) einen eng begrenzten und klar definierten Zweck haben.
Da sie in Programmierumgebungen eingesetzt werden, die beim Programmieren der Makros unbekannt sind, müssen 'Nebenwirkungen' möglichst vermieden werden. Dazu gehört es

- zu vermeiden, Datensätze zu erzeugen, es sei denn, das wäre der definierte Zweck des Makros. Für die Benennung von Datensätzen gilt dasselbe wie für die Makrovariablen (s.u.),
- temporäre Datensätze nachher zu löschen (mit proc datasets),
- lokale Makrovariablen, die außerhalb des Makros nicht bekannt sind, zu verwenden, wo es möglich ist,
- globale Makrovariablen sehr sorgfältig zu benennen, da auf der einen Seite die Funktion klar gemacht werden muß, andererseits eine Variable dieses Namens aber nicht schon existieren darf (Dokumentationspflicht),
- 'Harmlose' Fehlermeldungen und Warnungen, die die Korrektheit des Programmlaufs und der Ergebnisse nicht betreffen, zu vermeiden, da sie zu Irritationen führen können.

Die Dokumentation muß Informationen enthalten über

- den Zweck des Makros,
- die Voraussetzungen für ein korrektes Funktionieren (Input-Menge)
- eine Beschreibung der Ausgabe einschl. der erzeugten globalen Makrovariablen (Output-Menge)
- wer es geschrieben hat, und eine
- Geschichte der Veränderungen (= Versionen).

Zur Resourcen-Effizienz läßt sich generell sagen, daß die Effizienz des erzeugten SAS-Codes Vorrang vor der Effizienz des Makrocodes hat.
Allgemein bekannte Programmiertechniken haben auch hier Geltung. Ausdrücke, die komplizierte Berechnungen erfordern, sollen möglichst außerhalb von Schleifen aufgelöst und in einer Variablen gespeichert werden. Vergleiche oder Berechnungen können dann die Variable benutzen anstatt immer wieder neu zu berechnen. Dies gilt für SAS-Programme sowie für Makros.
Die Anzahl der Variablen soll aus Speicherplatzgründen auf ein Minimum begrenzt werden.
Um Informationen zwischen data-steps und/oder proc-steps zu transferieren, sollten Makrovariablen verwendet werden, ohne den Umweg über eine Datei zu nehmen.

Zur Programmierer-Effizienz tragen alle Mittel der Strukturierung bei (modularer Aufbau).
Die programm-interne Dokumentation sollte durch großzügige Kommentierungen die Struktur des Programms verdeutlichen. Makrokommentar-Anweisungen sollten gegenüber SAS-Kommentar-Anweisungen bevorzugt werden. Die Klarheit sollte gegenüber der Kürze des Codes bevorzugt werden.
Die Makroeinrichtung ist ein Textprozessor: arithmetische Berechnungen sollten vermieden werden, um implizite Umwandlungen von 'Text' (der natürlich aus Ziffern bestehen muß, damit es überhaupt funktioniert) in 'Zahlen' zu vermeiden, welche die Effizienz eines Makros stark herabsetzen können.

Allgemeines über die vorgestellten SAS-Makros

Neben der Tatsache, daß die oben erwähnten Tips zur Strukturierung und Effizienz von SAS-Makros berücksichtigt wurden, lassen sich an den drei Makros PROKRUST, SIMIL und KOORDIST noch mehr Gemeinsamkeiten feststellen: Eingebettet von Makrocode, werden die eigentlichen Berechnungen mit der 'interactive matrix language' (IML) durchgeführt.
Die Verwendung der IML hat den Vorteil, daß komplizierte Matrizenrechnungen, wie z.B. Singulärwertzerlegungen, relativ einfach aus der Literatur übernommen und eingesetzt werden können. Die IML erlaubt außerdem strukturierte Programmierung durch die Verwendung von Unterprogrammen, den 'modules'.
Ein kleiner Nachteil besteht darin, daß die Daten aus der SAS-Datei erst in eine Matrix überführt werden müssen, und entsprechend nach Beenden der Berechnungen mit IML wieder in eine Datei zurückgeschrieben werden müssen. Ein weiterer Nachteil liegt logischerweise darin, daß SAS/IML auf dem Rechner installiert sein muß.

3. PROKRUST - Prokrustische Ähnlichkeitstransformation

Das SAS-Macro PROKRUST führt für zwei mehrdimensionale Punktekonfigurationen X und Y eine prokrustische Ähnlichkeitstransformation durch. Dabei werden eine orthogonale Transformationsmatrix T, die die Rotation und Reflexion beinhaltet, und - je nach gewähltem Modell - ein Dilatationsfaktor K und eine Translationsmatrix TLAT so bestimmt, daß

```
X = Y * T  oder
X = K * Y * T + TLAT
```

im Sinne der least-square-Abweichung von X und Y optimal ist (vgl. Borg 1981, S.471; Borg & Lingoes 1987, S.320).

3.1 Problemstellung

Zunächst entwickelte sich die Prokrustes-Transformation im Rahmen der Rotationsproblematik in der Faktorenanalyse (vgl. Levine 1977). Der Name geht auf Hurley & Cattell (1962) zurück. Erste allgemeine Lösungen des Problem wurden fast zeitgleich von Kristof (1964), Fischer & Roppert (1965) und Schönemann (1966) erarbeitet.
Schönemann & Carroll (1970) stellten einen erweiterten Ansatz bereit, der auch Dilatationen und Translationen zuließ.
Die simultane Transformation mehrerer Konfigurationen wurde schließlich möglich durch die Erweiterungen der Generalized Procrustes Analysis (GPA) (vgl. Gower 1975, Lingoes & Borg 1978).
Neuere Ansätze versuchen auch fehlende Werte zu berücksichtigen. Fehlende Spalten (= ungleiche Dimensionalität der anzunähernden Konfigurationen) werden diskutiert bei Browne (1972), Gower (1975), Peay (1988) und Ten Berge & Knol (1984), fehlende Zeilen (= ungleiche Anzahl beurteilter Objekte) werden in Betracht gezogen von Borg (1977, 1978), und Commandeur (1991). Für beliebige fehlende Werte stellen in einer neueren Arbeit Ten Berge, Kiers & Commandeur (1993) eine Lösung vor.

3.2 Syntax und Parameter

%PROKRUST (INFIX, ID, INROT <, DIMS= > <, MODEL= ORTHO | ALL>
<, VLIST1= > <, VLIST2= > <, DOUT=YES | NO>
<, POUT= NO | YES> <, OUTDSN= >);

3.2.1 Positionsparameter

Positionsparameter müssen angegeben werden, und sie müssen in der genauen Reihenfolge angegeben werden. In Klammern wird zu jedem Parameter angegeben, ob er den Input und/oder Output betrifft.

INFIX (in)

Die Datei, die mit INFIX referenziert wird, enthält die Punktekonfiguration, die das Ziel der Transformation darstellt. Diese Punktekonfiguration muß in derselben Anordnung wie in INROT vorliegen. Diese Datei muß zusätzlich eine ID-Variable beinhalten.

ID (in)

Die ID-Variable wird verwendet, um die Werte des Outputs identifizierbar zu machen. Die ID-Variable kann vom Typ CHAR oder NUMERIC sein. Die gleichnamige und mit den gleichen Werten besetzte Variable im OUTDSN-Datensatz ist unabhängig davon stets vom Typ CHAR. Falls eine solche Variable im Datensatz nicht zur Verfügung steht, kann sie in einem zusätzlichen data-step folgendermaßen sehr einfach erzeugt werden:

```
ID = trim(left(_N_));
```

INROT (in)

Die Datei, die mit INROT referenziert wird, enthält die Punktekonfiguration, zu der eine Transformationsgleichung so bestimmt wird, daß sie im Sinne der least-square-Abweichung von X und Y optimal wird.

3.2.2 Schlüsselwortparameter

Schlüsselwortparameter müssen nicht angegeben werden, und die Reihenfolge, in der sie angegeben werden, ist beliebig.

DIMS= (in)

Dieser Parameter bestimmt die Dimensionalität der Punktekonfiguration.

Voreinstellung: 3 Dimensionen: DIMS = 3

MODEL= (in)

Dieser Parameter bestimmt die zur Berechnung der Ergebnis-Konfiguration zulässigen Transformationen.

ORTHO : zulässig sind nur orthogonale Rotation und Reflexion.

ALL : zusätzlich zu ORTHO sind zentrale Dilatationen und Translationen zulässig.

Voreinstellung: ALL.

VLIST1= (in)

Falls die Koordinaten der Zielkonfiguration nicht unter den Variablennamen DIM1...DIMs vorliegen, müssen sie im Parameter VLIST1 spezifiziert werden. Es ist darauf zu achten, daß die Anzahl der angegebenen Variablennamen der Angabe im DIMS-Parameter entspricht.

VLIST2= (in)

Falls die Koordinaten der Rotationskonfiguration nicht unter den Variablennamen DIM1...DIMs vorliegen, müssen sie im Parameter VLIST2 spezifiziert werden. Es ist darauf zu achten, daß die Anzahl der angegebenen Variablennamen der Angabe im DIMS-Parameter entspricht.

OUTDSN= (out)

OUTDSN enthält den Namen der Ausgabedatei. Diese enthält die Koordinaten der transformierten Rotationskonfiguration unter den Variablennamen DIM1...DIMs und die mit ID spezifizierte Variable. Falls die ID-Variable in INFIX numerisch ist, wird sie in eine CHAR-Variable umgewandelt. Dies ist notwendig, da zur Beschriftung der Ausgabe in PROC IML nur CHAR-Vektoren akzeptiert werden.

Voreinstellung: OUTDSN=PROK_OUT.

DOUT= (out)

Der Parameter DOUT spezifiziert, ob die Distanzen der einzelnen Items im Output ausgegeben werden sollen. Jede andere Angabe als DOUT=YES unterdrückt die Print-Ausgabe.

Voreinstellung: DOUT=NO.

POUT= (out)

Der Parameter POUT spezifiziert, ob die Transformationsmatrizen im Output ausgegeben werden sollen. Jede andere Angabe als POUT=YES unterdrückt die Print-Ausgabe, die Datei-Ausgabe ist davon nicht betroffen.

Voreinstellung: POUT=YES.

3.3 Aufruf

Falls zwei verschiedene Eingabedateien spezifiziert werden, müssen sich die beiden Koordinatensätze sowohl bezüglich der Beobachtungen (=Punkte) als auch der Variablen (=Dimensionen) entsprechen, d.h. gegebenenfalls müssen die Dateien zuvor sortiert werden, bzw. die Reihenfolge der Variablen in den Parametern VLIST1 oder VLIST2 angegeben werden. Wenn nur eine Datei angegeben wird, müssen die sich entsprechenden Koordinaten in **einer** Beobachtung unter **verschiedenen** Namen enthalten sein. Auch hier muß der Benutzer dafür sorgen, daß sich die Variablenreihenfolgen entsprechen.

Wenn die Parameter VLIST1 oder VLIST2 angegeben werden, muß die Anzahl der dort angegebenen Variablennamen mit der Zahl der Dimensionen im DIMS-Parameter übereinstimmen. Fehlende Werte führen zu Fehlern in den Berechnungen, da deren Behandlung nicht implementiert ist.

Einige Beispiele illustrieren den Aufruf:

```
%PROKRUST (koord1, id, koord2);
```

Für die in den Dateien *koord1* und *koord2* jeweils unter den Variablennamen DIM1...DIM3 vorliegenden Punktekonfigurationen werden Transformationsmatrizen bestimmt und im Output ausgegeben. Eine transformierte Konfiguration, die aus den Koordinaten in *koord2* errechnet wird, wird samt ID-Variablen *id* sowohl im Output als auch in die Datei PROK_OUT ausgegeben.

```
%PROKRUST (koordat, bez, koordat, OUTDSN=test
           VLIST2="FAC1" "FAC2" "FAC3");
```

Die Datei *koordat* enthält sowohl die ID-Variable *bez* als auch die dreidimensionale Zielkonfiguration unter DIM1...DIM3 und die Rotationskonfiguration unter FAC1...FAC3. Die Transformationsmatrizen werden im Output ausgegeben. Eine transformierte Konfiguration, die aus den Koordinaten in FAC1...FAC3 errechnet wird, wird unter den Variablennamen DIM1...DIM3 samt ID-Variablen *bez* sowohl im Output als auch in die Datei *test* ausgegeben. Ein weiteres ausführliches Beispiel mit Programmlauf und Ausgabe ist im Anhang F zu finden.

Beispielausgabe

```
559  * Aufruf Macro PROKRUST                                              *;
561  %PROKRUST (BG81X471, nr, BG81Y471, dims=2, outdsn=BORG1);

* ------------------------------------------------- *
*            PROKRUSTES-ROTATION                    *
*                                                   *
*             AUTOR: M. Huerkamp                    *
*           VERSION: 0.45 MAIN                      *
*  LETZTE AENDERUNG: 20.12.93                       *
*                                                   *
* ------------------------------------------------- *

 PROKRUST - Parameter - Ausgabe:
 ------------------------------------------------------
 INFIX       : Zielkonfiguration aus          : BG81X471
 INROT       : Rotationskonfiguration aus     : BG81Y471
 ID          : ID - Variable                  : nr
 DIMS        : Anzahl der Dimensionen         : 2
 MODEL       : gewaehltes Modell              : ALL
 DOUT        : Ausgabe der Einzeldistanzen    : NO
 POUT        : Druckausgabe                   : YES
 OUTDSN      : Ergebniskonfiguration in       : BORG1
 VLIST1      : Liste der Var.namen in INFIX :
 VLIST2      : Liste der Var.namen in INROT :
 ------------------------------------------------------
IML Ready
NOTE: Module IDIST defined.
NOTE: Module CONFCORR defined.
NOTE: Module KONGRU defined.
NOTE: The data set WORK.BORG1 has 4 observations and 2 variables.
Exiting IML.
NOTE: The PROCEDURE IML used 5.12 seconds.
```

Abb.1: Ausgabe des Makros PROKRUST im SAS-LOG

```
                    Transformationsmatrix T

                      -0.86652 -0.49915
                      -0.49915  0.86652

                     Streckungsfaktor K

                           1.99655

                   Translationsvektor TL

                           3.72319
                          -2.46353

                mittl. euclidische Distanzen:

                        vorher   nachher

                       2.67580   0.00893
```

Abb.2: Ausgabe der Transformationsmatrizen im SAS-OUTPUT

Ergebnismatrix

	DIM1	DIM2
1	0.99107	1.99944
2	-0.99506	2.00741
3	-0.99107	-1.99944
4	0.99506	-2.00741

Abb.3: Ausgabe der Ergebnismatrix im SAS-OUTPUT

Kongruenz	Alienation	Korrelation	R-Quadrat
0.999996	0.002822	0.999992	0.999984

Abb.4: Ausgabe der Ähnlichkeitsmaße im SAS-OUTPUT

3.4 Programmstruktur

Die wesentlichen Berechnungen und Ausgaben des Makros werden durch die IML-Module geleistet. Die Macro-Sprache dient zur Erhöhung des Bedienungskomforts beim Aufruf von PROC IML.
Das IML-Modul CONFCORR ist aus einer Korrelationsroutine (SAS Institute Inc., 1985, S. 113) adaptiert worden.
Das Modul KONGRU ist vom Autor nach der Formel in Borg & Leutner 1985, bzw. Borg & Lingoes 1987 programmiert worden.
Das Modul IDIST ist vollständig vom Autor entwickelt.

3.5 Fehlerbehandlung

Im Makro werden nur die Parameterfehler abgefangen. Wenn zu wenige obligatorische Positionsparameter, also weniger als drei, angegeben werden, oder der DIMS-Parameter auf kleiner als 2 gesetzt wird, wird die im Anhang A dokumentierte Fehlermeldung ausgegeben.

Innerhalb von PROC IML wird überprüft, ob die Konfigurationen gleiche Anzahl von Punkten und Dimensionen haben. Falls dies nicht der Fall ist, bricht PROC IML mit der im Anhang A berichteten Fehlermeldung ab.

3.6 Entwicklung und Planung

Version 0.45

- weiter verbesserte Seiten-Ausgabe
- bei POUT = NO überhaupt keine Druckausgabe mehr (außer der Kontrollausgabe), vorher wurde immer die Ergebnis-Matrix ausgedruckt.
- Berechnung von Kongruenz- & Alienationskoeffizienten.
- Modularisierung
- Kontrollausgabe der Parameter
- Zusätzliche Optionen MODEL= und DOUT=
- Dokumentation

Version 0.41

- verbesserte Ausgabe: reset noname
- Korrelation, etc.

Version 0.3

- Fehlerprüfung
- Defaultbelegung

Version 0.2

- Macro ruft PROC IML auf

Version 0.1

- "pures" IML-Module

Planung:

- Notwendigkeit für die Angabe einer ID-Variablen beseitigen durch Generierung einer Default-Belegung.
- Vermeiden der Umwandlung der ID-Variablen im Output-Datensatz in den Typ CHAR.
- Prüfung auf konsistente Belegung der VLIST1/2 und DIMS-Parameter und entsprechende Fehlermeldung.
- Erlauben von fehlenden Werten unter Berücksichtigung der oben angeführten Arbeiten.

4. SIMIL - Ähnlichkeit von Konfigurationen

Das Macro SIMIL stellt für die Beurteilung der Ähnlichkeit zweier mehrdimensionaler Punktekonfigurationen vier Koeffizienten zur Verfügung:

a) Kongruenzkoeffizient C,
b) Alienationskoeffizient K,
c) Korrelationskoeffizient R,
d) Determinationskoeffizient RQUADRAT

Die Maße c) und d) sind nur interpretierbar, wenn die beiden Punktekonfigurationen unter Ausnutzung aller zulässigen Transformationen aneinander angenähert wurden (etwa mit einer Prokrustes-Rotation).

4.1 Problemstellung

Verschiedene Gründe können dazu führen, daß man sich für den Vergleich zweier Punktekonfigurationen interessiert. Dies kann z.B. ein Methodenvergleich in dem Sinne sein, daß mit verschiedenen Erhebungs-Verfahren die Distanzen oder (Un-)Ähnlichkeiten derselben Objekte ermittelt wurden (z.B. Wagner et al. 1990). Man könnte sich auch dafür interessieren, ob und in wieweit verschiedene Skalierungsalgorithmen - etwa ALSCAL und SAS-MDS - dieselben oder unterschiedliche Ergebnisse zeitigen.
In der Literatur werden zwei Typen von Ähnlichkeitsmaßen diskutiert: Koordinaten-basierte Maße und distanzen-basierte Maße (vgl. Borg & Leutner 1985). Koordinaten-basierte Maße sind hier die Korrelation R, bzw. der durch Quadrieren gewonnene "Determinationskoeffizient" RQUADRAT. Diese Maße sind als Ähnlichkeitsmaße nur nach vorhergehender Prokrustes-Transformation interpretierbar. Für diesen Fall allerdings liefert Langeheine (1980, 1982) approximative statistische Normen.
Borg & Leutner (1985) schlagen die distanzen-basierten Maße des Kongruenzkoeffizienten C und den daraus umgeformten Alienationskoeffizienten K als Alternative vor. Sie bieten gleichfalls approximative statistische Normen für den Alienationskoeffizienten K an. Die Beziehungen zwischen C und K auf der einen, und R und RQUADRAT auf der anderen Seite scheinen komplexer Natur zu sein, so daß es geraten scheint, beide Werte zu konsultieren.

4.2 Syntax und Parameter

%SIMIL (INDSN1, ID, INDSN2 <, DIMS = > <, VLIST1 = > <, VLIST2 = >
<, OUTDSN = SIMILOUT >);

4.2.1 Positionsparameter

Positionsparameter müssen angegeben werden, und sie müssen in der genauen Reihenfolge angegeben werden.

INDSN1 (in)

Name der Datei, die einen der beiden Sätze von Punktekoordinaten enthält. Falls die Koordinaten unter anderen Variablennamen als DIM1...DIMs gespeichert sind, müssen diese in VLIST1 angegeben werden. INDSN1 kann auch mit INDSN2 identisch sein, dann muß VLIST1 und/oder VLIST2 angegeben sein. Die Reihenfolge der Punkte in der Datei muß identisch sein mit der von INDSN2. Zusätzlich muß eine ID-Variable enthalten sein.

ID (in)

ID-Variable, die in INDSN1 enthalten sein muß.

INDSN2 (in)

Name der Datei, die den zweiten der beiden Sätze von Punktekoordinaten enthält. Falls die Koordinaten unter anderen Variablennamen als DIM1...DIMs gespeichert sind, müssen diese in VLIST2 angegeben werden. INDSN2 kann auch mit INDSN1 identisch sein, dann muß VLIST2 und/oder VLIST1 angegeben sein. Die Reihenfolge der Punkte in der Datei muß identisch sein mit der von INDSN1.

4.2.2 Schlüsselwortparameter

Schlüsselwortparameter müssen nicht angegeben werden, und die Reihenfolge, in der sie angegeben werden, ist beliebig.

DIMS= (in)

Anzahl der Dimensionen. Die Anzahl der in DIMS spezifizierten Dimensionen muß mit der Anzahl der gegebenfalls in VLIST1 oder VLIST2 angegebenen entsprechen. Die Anzahl der Dimensionen muß größer oder gleich 2 sein.

Voreinstellung: 3 Dimensionen.

VLIST1= (in)

Falls die Koordinaten der Zielkonfiguration nicht unter den Variablennamen DIM1...DIMs vorliegen, müssen sie im Parameter VLIST1 spezifiziert werden. Es ist darauf zu achten, daß die Anzahl der angegebenen Variablennamen der Angabe im DIMS-Parameter entspricht.

VLIST2= (in)

Falls die Koordinaten der Rotationskonfiguration nicht unter den Variablennamen DIM1...DIMs vorliegen, müssen sie im Parameter VLIST2 spezifiziert werden. Es ist darauf zu achten, daß die Anzahl der angegebenen Variablennamen der Angabe im DIMS-Parameter entspricht.

OUTDSN= (out)

Der Name der Datei, die die vier Ähnlichkeits-Koeffizienten unter den Variablennamen C, K, R, RQUADRAT enthält.

Voreinstellung: SIMILOUT.

4.3 Aufruf

SAS/IML muß zum korrekten Funktionieren des Makros installiert sein. Einige Beispiele zeigen die Anwendung des Makros:

```
%SIMIL (KOORD1, ID, KOORD2, OUTDSN=OUTDAT);
```

Die 3-dimensionalen Koordinaten stehen unter den Variablennamen DIM1...DIM3 in den Dateien *KOORD1* und *KOORD2*, die ID-Variable in der Datei *KOORD1* trägt den Namen *ID*, und die Ergebnisse werden in der Datei *OUTDAT* gespeichert.

```
%SIMIL (KOORDAT, BEZ, KOORDAT, DIMS=2, VLIST2="FAC1"
        "FAC2" );
```

Die Datei *KOORDAT* enthält sowohl die ID-Variable *BEZ*, als auch die beiden 2-dimensionalen Punktekonfigurationen unter den Variablennamen DIM1...DIM2 und *FAC1-FAC2*. Die Ergebnisse werden in der Datei *SIMILOUT* gespeichert. Ein weiteres Beispiel ist mit Programmlauf und Ausgabe in Anhang F zu finden.

Beispielausgabe

```
567  %SIMIL (BG81X471, nr, BG81Y471, dims=2);
* ------------------------------------------------- *
*  Kongruenz-, Alienations- & Korrelationskoeffizienten *
*                                                   *
*             AUTOR: M. Huerkamp                    *
*           VERSION: 0.3 OS/2                       *
*  LETZTE AENDERUNG: 19.03.1994                     *
*                                                   *
* ------------------------------------------------- *

 SIMIL - Parameter - Ausgabe:
 ---------------------------------------------------
 Name der 1. Eingabe-Datei       : BG81X471
 Name der 2. Eingabe-Datei       : BG81Y471
 ID - Variable                   : nr
 Anzahl der Dimensionen          : 2
 Name der Ausgabedatei           : SIMILOUT
 Liste der Var.namen in INDSN1   :
 Liste der Var.namen in INDSN2   :
 ---------------------------------------------------
IML Ready
NOTE: Module CORR defined.
NOTE: Module KONGRU defined.
Exiting IML.
NOTE: The PROCEDURE IML used 1.46 seconds.
```

Abb.5: Beispiels-Ausgabe des Makros SIMIL im SAS-LOG

```
                                    CONGRU
                  Kongruenz :     0.999996

                                     ALIEN
                  Alienation:    0.0028217

                                      KORR
                  R         :    0.4423177

                                       RSQ
                  R-Quadrat :     0.195645
```

Abb.6: Beispiels-Ausgabe des Makros SIMIL im SAS-OUTPUT

4.4 Programmstruktur

Das umgebende Makro bildet die aufrufende Umgebung für die SAS-Prozedur IML. In dieser werden die Berechnungen durchgeführt. Das IML-Modul KONGRU bestimmt nur die untere Dreiecksmatrix der Distanzen (identisch mit dem gleichnamigen Modul im Makro Prokrust).

4.5 Fehlerbehandlung

Fehlerhafte Eingaben sollten möglichst früh zurückgewiesen werden.
Leider werden zur Zeit mehrere Fehler erst nach dem Aufruf der Prozedur IML abgefangen:

- Inkonsistenzen des DIMS-Parameters mit VLIST1 und/oder VLIST2.
- Die Dateien haben nicht die gleiche Anzahl von Beobachtungen, bzw. die Matrizen haben nicht die gleiche Zeilenzahl.
- Fehlende Werte werden zur Zeit nicht behandelt.

4.6 Entwicklung und Planung

Version 0.3

- Umgebendes Makro

Version 0.1

- 'pures' IML-Programm

Planung

- verbesserte Fehlerbehandlung im Makro
- Signifikanz - Prüfung der Koeffizienten

5. KOORDIST - Distanzen der Punkte einer Konfiguration

Das Makro KOORDIST erstellt aus einem Satz von mehrdimensionalen Punktekoordinaten eine Matrix der wechselseitigen euklidischen Distanzen (vergleichbar einer Tabelle von Städtedistanzen).

5.1 Problemstellung

Die eigentliche Information, die in einer MDS-Lösung enthalten ist, liegt wie schon eingangs erwähnt, in dem Verhältnis der Distanzen der Punkte zueinander. Insofern könnte man fast sagen, daß dieses Makro ein Manko der SAS-MDS-Prozedur wett macht, denn dieses gibt in der OUTPUT-Datei nur die Koordinaten-Matrix der Skalierungslösung an. Für denjenigen, der mit der Interpretation graphischer Repräsentationen im dreidimensionalen Raum Schwierigkeiten hat, bietet dieses Makro 'handfeste' numerische Information, etwa in der folgenden Art: Punkt 1 ist doppelt so weit von Punkt 2 entfernt wie von Punkt 3 etc... .

5.2 Syntax und Parameter

%KOORDIST (INDSN, ID <, DIMS=3 > <, VLIST= >);

5.2.1 Positionsparameter

Positionsparameter müssen angegeben werden, und sie müssen in der genauen Reihenfolge angegeben werden.

INDSN (in)

Name der Datei, die die Punktekoordinaten enthält. Falls die Koordinaten unter anderen Variablennamen als DIM1...DIMs enthalten sind, müssen diese in VLIST angegeben werden. Zusätzlich muß eine ID-Variable enthalten sein.

ID (in)

ID-Variable, die in INDSN enthalten sein muß.

5.2.2 Schlüsselwortparameter

Schlüsselwortparameter müssen nicht angegeben werden, und die Reihenfolge, in der sie angegeben werden, ist beliebig.

DIMS= (in)

Anzahl der Dimensionen. Die Anzahl der in DIMS spezifizierten Dimensionen muß mit der Anzahl der gegebenenfalls in VLIST angegebenen entsprechen. Die Anzahl der Dimensionen muß größer oder gleich 2 sein.

Voreinstellung: 3 Dimensionen.

VLIST= (in)

Falls die Koordinaten nicht unter den Variablennamen DIM1...DIMs vorliegen, müssen sie im Parameter VLIST spezifiziert werden. Es ist darauf zu achten, daß die Anzahl der angegebenen Variablennamen der Angabe im DIMS-Parameter entspricht.

5.3 Aufruf

Ein Beispiel illustriert den Aufruf:

```
%KOORDIST(loes3d, nam, VLIST="koord1" koord2" koord3");
```

Die euklidischen Distanzen der Punkte einer Koordinaten-Matrix, die in den Variablen *koord1...koord3* der Datei *loes3d* enthalten ist, wird berechnet und ausgegeben. Ein ausführliches Beispiel mit Programmlauf und Ausgabe ist in Kap. 9 zu finden.

Beispielausgabe

```
570  %KOORDIST (BG81X471, nr, dims=2);

* ------------------------------------------------ *
*           DISTANZEN DER PUNKTE EINER             *
*           KOORDINATEN-MATRIX KOORDIST            *
*                                                  *
*                AUTOR: M. HUERKAMP                *
*              VERSION: 0.11 - OS/2                *
*     LETZTE AENDERUNG: 28.3.1994                  *
*                                                  *
* ------------------------------------------------ *

 KOORDIST-Parameter-Ausgabe:
 ---------------------------------------------
 Eingangsmatrix            : BG81X471
 ID-Variable               : nr
 Anzahl der Dimensionen    : 2
 Liste der Variablennamen :
 ---------------------------------------------
IML Ready
NOTE: Module KOORD defined.
Exiting IML.
```

Abb.7: Ausgabe des Makros KOORDIST im SAS_LOG

DISTANZMATRIX:

DIST			
0.000	2.000	4.472	4.000
2.000	0.000	4.000	4.472
4.472	4.000	0.000	2.000
4.000	4.472	2.000	0.000

Abb.8: Ausgabe der Distanzmatrix im SAS-OUTPUT

5.4 Programmstruktur

Das umgebende Makro ruft, nach der obligatorischen Parameterüberprüfung, die Prozedur IML auf, die ihrerseits die Berechnungen durchführt und ausgibt.

5.5 Fehlerbehandlung

Fehlende Werte führen zu Fehlern in der Berechnung der Distanzen.

5.6 Entwicklung und Planung

Version 0.11

Aufruf als Makro

Version 0.1

'pures' IML- Module

6. G3DID - dreidimensionale graphische Darstellung

Das Makro G3DID unterscheidet sich von den bis jetzt vorgestellten Makros in Zweck, Entstehungsart und Programmstruktur. Es dient der Darstellung dreidimensionaler Punktekonfigurationen und wurde ausgehend von einem Makro von Bundschuh (1989) angepaßt und weiterentwickelt.

6.1 Problemstellung

Da die graphische Darstellung Grundlage der Interpretation der Ergebnisse ist, mußte eine Darstellungsart gefunden werden, die es erleichtert, sich in der Graphik zu orientieren. Dazu gehört die eindeutige Kennzeichnung der dargestellten Punkte, ein räumliches Koordinatensystem mit Gitterlinien und Fallinien, welche die Position des Punktes auf der Grundfläche markieren.

6.2 Syntax und Parameter

%G3DID (TIEFE, BREITE, HOEHE, ID <, INDSN=_LAST_ > <, SIZE=1 >
<, AXMAX=2 > <, POS=3 > <, SSIZE=0.001 > <, TICKS=5 > <, ROTATE=70 >
<, TILT=70 > <, POUT=NO > <, TEMP=T_E_M_P > <, OUTDSN=G3DIDOUT>);

6.2.1 Positionsparameter

Positionsparameter müssen angegeben werden, und sie müssen in der genauen Reihenfolge angegeben werden.

TIEFE (in)

Gibt an, welche der drei Koordinaten als Tiefendimension darzustellen ist.

BREITE (in)

Gibt an, welche der drei Koordinaten als Breitendimension darzustellen ist.

HOEHE (in)

Gibt an, welche der drei Koordinaten als Höhendimension darzustellen ist.

ID (in)

ID-Variable, die als Bezeichner der Punkte eingesetzt wird.

6.2.2 Schlüsselwortparameter

Schlüsselwortparameter müssen nicht angegeben werden, und die Reihenfolge, in der sie angegeben werden, ist beliebig.

INDSN= (in)

Name der Eingabedatei, die die Koordinaten und die Bezeichnervariable ID enthält.

Voreinstellung: _LAST_, die letzte referenzierte Datei

OUTDSN= (out)

Name der Ausgabedatei

Voreinstellung: G3DIDOUT

POUT= (in)

Mit diesem Parameter kann spezifiziert werden, ob die Ausgabe der Annotate-Datei gewünscht wird oder nicht.

Voreinstellung: NO, keine Ausgabe der Annotate-Datei.

ROTATE= (in)

Der Rotationswinkel gibt an, wie sehr die Darstellung um die Höhenachse gedreht wird.

Voreinstellung: 70, dies entspricht der Voreinstellung des SAS.

TILT= (in)

Der Kipp-Winkel gibt an, um wieviel die Grundfläche dem Betrachter entgegengekippt wird.

Voreinstellung: 70, dies entspricht der Voreinstellung des SAS.

SIZE= (in)

Die Größe des ID-Zeichens wird hier angegeben.

SSIZE= (in)

Dieser Parameter betrifft die Größe des Plotsymbols.

TICKS= (in)

Gibt die Anzahl der Achsenmarkierungen auf allen drei Achsen an.

Voreinstellung: 5

AXMAX= (in)

Gibt die dargestellte Achsenlänge auf allen drei Koordinaten an.

Voreinstellung: 2

TEMP= (in)

Der Name der temporären Datei

Voreinstellung: T_E_M_P

6.3 Aufruf

SAS/GRAPH muß natürlich auf dem Rechnersystem installiert sein.
Beispiel:

```
%G3DID (dim2, dim1, dim3, id, INDSN=indat,
        TICKS=7 AXMAX=5 );
```

Die Punkte, deren Koordinaten in der Datei *indat* enthalten sind, werden mit *dim2* als Tiefenkoordinate, *dim1* als horizontale Achse und *dim3* als vertikale Achse dreidimensional dargestellt. Als Bezeichner werden die in *id* enthaltenen Werte neben den Punkten ausgegeben.
Die Achsen reichen von -5 bis +5 und sind in 6 gleiche Teilstücke unterteilt.
Ein weiteres Beispiel ist mit Programmlauf und Plot-Ausgabe in Anhang F zu finden.

6.4 Programmstruktur

Die hauptsächliche Funktion des Makros besteht in einem Aufruf der SAS-Prozedur G3D mit einer Annotate-Datei, die die Bezeichner für die Punkte enthält. Diese Annotate-Datei wird vom Makro generiert.

6.5 Fehlerbehandlung

Bei Aufruf-Fehlern wird die Fehlermeldung im Anhang A ausgegeben.

6.6 Entwicklung und Planung

Version 0.2

Neue Parameter: ROTATE und TILT

Version 0.1

Anpassung des ursprünglichen Makros

7. Desiderata

Die vorgestellten Makros haben sich im harten Projektalltag bewährt. Es bleibt trotzdem noch einiges zu wünschen übrig.

In bezug auf die Programmierung könnte die Fehlerbehandlung in den Makros noch verbessert werden. Einige Fehleingaben könnten früher abgefangen werden. Sie könnten schon im umgebenden Makro behandelt werden statt später im IML-Teil des Makros.

Einige Makros könnten auch mit noch mehr Optionen ausgestattet werden (vgl. die jeweiligen Kapitel 'Entwicklung und Planung').

Im Hinblick auf die Aufgabenstellung dieser Makros etwa aus der Sicht der facettentheoretischen Interpretation der Ergebnisse vor allem Wünsche offen bezüglich der Prokrustes-Transformationen.

Es wäre sehr wünschenswert, Verfahren an die Hand zu bekommen, die nicht eine Annäherung einer Konfiguration an eine zweite bewirken, sondern maximale Distanz der Punkte bezüglich einer vordefinierten Trennungslinie oder Ebene herstellt. Dies könnte das Testen regionaler Hypothesen sehr vereinfachen.

Wünschenswert wären auch numerische oder graphentheoretische Verfahren, die die Einhaltung (oder Verletzung) regionaler Hypothesen direkt ermitteln. Hier stellt sich natürlich zum Schluß die Frage, ob dann nicht ein Neu-Ansatz, der auf die Darstellung im euklidischen Raum verzichtet und strukturelle Modelle direkter testet, sinnvoller wäre.

Literatur

Borg, I. (1977). SFIT: matrix fitting when points correspond in some substantive sense only. **Journal of Marketing Research, 14**, 556-558.

Borg, I. (1978). Procrustean analysis of matrices with different row order. **Psychometrika, 43**, 277-278

Borg, I. (1981). **Anwendungsorientierte multidimensionale Skalierung** (= Lehr- und Forschungstexte Psychologie, Bd. 1). Berlin: Springer.

Borg, I., & Leutner, T. (1985). Measuring the similarity of MDS configurations. **Multivariate Behavioral Research, 20**, 325-334.

Borg, I., & Lingoes, J. (1987). **Multidimensional similarity structure analysis**. New York: Springer.

Browne, M.W. (1972). Orthogonal rotation to a partially specified target. **British Journal of Mathematical and Statistical Psychology, 25**, 207-212.

Bundschuh, W. (1989). **Das SAS-Macro IDG3D**. In W. Bundschuh, J. Hefner, & H. Hellwig, Lokale SAS-Prozeduren, Lokale SAS-Funktionen, Lokale SAS-Macros (30-31). Heidelberg: Universitätsrechenzentrum, 3. Aufl.

Commandeur, J.J.F. (1991). **Matching configurations**. Leiden: DSWO.

Engesser, H. (Hrsg.). (1988). **Duden Informatik**. Mannheim: Dudenverlag

Fischer, G., & Roppert, J. (1965). Ein Verfahren der Transformationsanalyse faktorenanalytischer Ergebnisse. In J. Roppert & G. Fischer (Hrsg.), **Lineare Strukturen in Mathematik und Statistik**, Wien: Physika.

Gogolok, J., Schuemer, R., & Ströhlein, G. (1992). **Einführung in das Programmsystem, Datenmanagement und Auswertung** (=Datenverarbeitung und statistische Auswertung mit SAS, Bd. 1). Stuttgart: Gustav Fischer.

Gower, J.C. (1975). Generalized Procrustes analysis. **Psychometrika, 40**, 33-51.

Graf, A., Bundschuh, W., & Kruse, H.-G. (1993). **Effektives Arbeiten mit SAS**: Grundlagen und Programmierung. Mannheim: B.I..

Hurley, J.R., & Cattell, R.B. (1962). The Procrustes program: Producing direct rotation to test a hypothesized factor structure. **Behavioral Science**, 7, 258-262.

Kaltenbach, T., Reetz, U., & Woerrlein, H. (1988). **Taschenlexikon der Computerwelt.** Haar: Markt & Technik.

Kristof, W. (1964). Die beste orthogonale Transformation zur gegenseitigen Überführung zweier Faktormatrizen. **Diagnostica, 10,** 87-90.

Langeheine, R. (1980). Erwartete Fit-Werte für Zufallskonfigurationen in PINDIS. **Zeitschrift für Sozialpsychologie, 11,** 38-49.

Langeheine, R. (1982). Statistical evaluation of measures of fit in the Lingoes-Borg procrustean individual differences scaling. **Psychometrika, 47,** 427-442.

Levine, M.S. (1977). **Canonical analysis and factor comparison.** Beverly Hills: Sage.

Lingoes, J.C., & Borg, I. (1978). Procrustean individual differences scaling. **Journal of Marketing Research, 13,** 406-407.

Peay, E.R. (1988). Multidimensional rotation and scaling of configurations to optimal agreement. **Psychometrika, 53,** 199-208.

Roskam, E.E.C.I., & Lingoes, J.C. (1970). MINISSA-I: A FORTRAN IV (G) Program for the smallest space analysis of square symmetric matrices. **Behavioral Science, 15,** 204.

SAS Institute Inc. (1985). **SAS/IML guide for personal computers**. Version 6 edition. Cary, NC: SAS Institute Inc.

SAS Institute Inc. (1990). **SAS guide to macro processing**. Version 6 second edition. Cary, NC: SAS Institute Inc.

SPSS Inc. (1985). **SPSS-X Release 2.1: New procedures and facilities**. SPSS Inc.

Schönemann, P.H. (1966). A generalized solution of the orthogonal Procrustes problem. **Psychometrika, 31,** 1-10.

Schönemann, P.H. & Carroll, R.M. (1970). Fitting one matrix to another under choice of a central dilation and a rigid motion. **Psychometrika, 35,** 245-256.

Ten Berge, J.M.F., & Knol, D.L. (1984). Orthogonal rotations to maximal agreement for two or more matrices of different column orders. **Psychometrika, 49,** 49-55.

Ten Berge, J.M.F., Kiers, H.A.L., & Commandeur, J.J.F. (1993). Orthogonal Procrustes rotation for matrices with missing values. **British Journal of Mathematical and Statistical Psychology, 46,** 119-134.

Wagner, F., Huerkamp, M., Jockisch, H., & Graumann, C.F. (1990). **Sprachlich realisierte soziale Diskriminierungen: empirische Überprüfung eines Modells expliziter Diskriminierung**. (Arbeiten aus dem SFB 245 "Sprache und Situation" Nr. 23). Heidelberg/Mannheim.

Young, F.W., & Lewyckyj, R. (1979). **ALSCAL - 4 user's guide**. Chapel Hill, NC: Data Analysis and Theory Associates, 2. Aufl..

Young, F.W., Lewyckyj, R., & Takane, Y. (1986). **The ALSCAL Procedure**. In SAS Institute Inc., SUGI Supplemental Library user's guide. Version 5 edition. Cary, NC: SAS Institute Inc., 1-16

Anhang A: Fehlermeldungen

Fehlermeldungen des Makros PROKRUST

Fehlermeldung falls eine zu geringe Anzahl (<3) obligatorischer Parameter angegeben wurde, oder der DIMS-Parameter kleiner als 2 gewählt wurde:

```
 ERROR: Fehlerhafter Aufruf Macro PROKRUST
 ERROR: Zu wenige obligatorische Parameter
 ERROR: oder DIMS kleiner als 2 gesetzt.
 ------------------------------------------------------
Syntax:
PROKRUST (positionsparms, schluesselwortparms)
   positionsparms (obligatorisch):
     INFIX    Name der Ziel-Eingabedatei
     ID       ID-Variable, (in INFIX)
     INROT    Name der zu rotierenden Datei
   schluesselwortparms (optional):
     DIMS=    Anzahl der Dimensionen, >= 2
              Voreinstellung: 3 );
     VLIST1= Liste der Variablennamen in INFIX
              Voreinstellung: DIM1..DIMs
     VLIST2= Liste der Variablennamen in INROT;
              Voreinstellung: DIM1..DIMs
     MODEL=   gewaehltes Modell: ORTHO,
              oder ALL
              Voreinstellung: ALL
     DOUT=    Ausgabe der Einzeldistanzen
              Voreinstellung: NO
     POUT=    Druckausgabe der Ergebnisse
              Voreinstellung: YES
     OUTDSN= Name der Ausgabedatei, enthaelt die
              Koordinaten DIM1..DIMs und ID
              Voreinstellung: PROK_OUT
Beispiel:
  PROKRUST %(ziel, id, rotier, DIMS=3, OUTDSN=outdat,
            VLIST2="FAC1" "FAC2" "FAC3" %)
 ------------------------------------------------------
```

Abb.9: Macro-Fehlermeldung des Makros PROKRUST

Fehlermeldung falls die Konfigurationen nicht die gleiche Anzahl von Punkten oder Dimensionen aufweisen:

```
ERROR(PROKRUST): Eingabefehler bei Prokrustes Rotation
      Matrizen X und Y haben unvertraegliches Format
      oder sind nicht definiert
```

Abb. 10: IML-Fehlermeldung des Makros PROKRUST

Fehlermeldungen des Makros SIMIL

```
* ------------------------------------------------- *
*  Kongruenz-, Alienations- & Korrelationskoeffizienten *
*                                                   *
*              AUTOR: M. Huerkamp                   *
*            VERSION: 0.2                           *
* ------------------------------------------------- *

 ERROR: Fehlerhafter Aufruf Macro SIMIL);
 ---------------------------------------------------
Syntax:
SIMIL (positionsparms, schluesselwortparms)
   positionsparms (obligatorisch):
     INDSN1  Name der 1. Eingabedatei
     ID      ID-Variable, (in INDSN1, char )
     INDSN2  Name der 2. Eingabedatei
   schluesselwortparms (optional):
     DIMS=   Anzahl der Dimensionen      Default=3
     OUTDSN= Name der Ausgabedatei       Default=SIMILOUT
             enthält die Koeffizienten
     VLIST1= Liste der Variablennamen in INDSN1
             ohne Angabe: Präfix DIM, Suffix 1..DIMS
     VLIST2= Liste der Variablennamen in INDSN2);
             ohne Angabe: Präfix DIM, Suffix 1..DIMS
Beispiel:
  SIMIL (KOORD1, ID, KOORD2, DIMS=3, OUTDSN=OUTDAT,
           VLIST2="FAC1" "FAC2" "FAC3" )
 ---------------------------------------------------
```

Abb. 11 Fehlermeldung des Makros SIMIL

Anhang B: Source-Code des Makros PROKRUST

```
%MACRO PROKRUST (INFIX,ID,INROT,
                 DIMS=3,
                 VLIST1=, VLIST2=,
                 MODEL=ALL,
                 DOUT=NO,
                 POUT=YES,
                 OUTDSN=PROK_OUT);
%* ------------------------------------------------------ *;
%* ------------------------------------------------------ *;
%put %str( );
%put %str(* ------------------------------------------------------ *);
%put %str(*             PROKRUSTES-ROTATION                        *);
%put %str(*                                                        *);
%put %str(*              AUTOR: M. Huerkamp                        *);
%put %str(*           VERSION: 0.45 MAIN                           *);
%put %str(*  LETZTE AENDERUNG: 20.12.93                            *);
%put %str(*                                                        *);
%put %str(* ------------------------------------------------------ *);
%put %str( );
%if &INROT= or &DIMS < 2 %then %do;
  %put %str( ERROR: Fehlerhafter Aufruf Macro PROKRUST              );
  %put %str( ERROR: Zu wenige obligatorische Parameter              );
  %put %str( ERROR: oder DIMS kleiner als 2 gesetzt.                );
  %put %str( ------------------------------------------------------ );
  %put %str(Syntax:                                                 );
  %put %str(PROKRUST (positionsparms, schluesselwortparms)          );
  %put %str(   positionsparms (obligatorisch):                      );
  %put %str(     INFIX   Name der Ziel-Eingabedatei                 );
  %put %str(     ID      ID-Variable, (in INFIX)                   );
  %put %str(     INROT   Name der zu rotierenden Datei              );
  %put %str(   schluesselwortparms (optional):                      );
  %put %str(     DIMS=   Anzahl der Dimensionen, >= 2               );
  %put %str(             Voreinstellung: 3                          );
  %put %str(     VLIST1= Liste der Variablennamen in INFIX          );
  %put %str(             Voreinstellung: DIM1..DIMs                 );
  %put %str(     VLIST2= Liste der Variablennamen in INROT          );
  %put %str(             Voreinstellung: DIM1..DIMs                 );
  %put %str(     MODEL=  gewaehltes Modell: ORTHO,                  );
  %put %str(             oder ALL                                   );
  %put %str(             Voreinstellung: ALL                        );
  %put %str(     DOUT=   Ausgabe der Einzeldistanzen                );
  %put %str(             Voreinstellung: NO                         );
  %put %str(     POUT=   Druckausgabe der Ergebnisse                );
  %put %str(             Voreinstellung: YES                        );
  %put %str(     OUTDSN= Name der Ausgabedatei, enthaelt die        );
  %put %str(             Koordinaten DIM1..DIMs und ID              );
  %put %str(             Voreinstellung: PROK_OUT                   );
  %put %str(Beispiel:                                               );
  %put %str(  PROKRUST %(ziel, id, rotier, DIMS=3, OUTDSN=outdat,   );
  %put %str(            VLIST2="FAC1" "FAC2" "FAC3" %)              );
  %put %str( ------------------------------------------------------ );
  %put %str( );
  %goto ENDE;
%end;
%else %do;
  %let POUT = %upcase(&POUT);
  %let DOUT = %upcase(&DOUT);
  %let MODEL = %upcase(&MODEL);
  %put %str( PROKRUST - Parameter - Ausgabe: );
  %put %str( ------------------------------------------------------ );
  %put %str( INFIX      : Zielkonfiguration aus       : &INFIX      );
  %put %str( INROT      : Rotationskonfiguration aus  : &INROT      );
```

```
  %put %str( ID          : ID - Variable                    : &ID        );
  %put %str( DIMS        : Anzahl der Dimensionen           : &DIMS      );
  %put %str( MODEL       : gewaehltes Modell                : &MODEL     );
  %put %str( DOUT        : Ausgabe der Einzeldistanzen      : &DOUT      );
  %put %str( POUT        : Druckausgabe                     : &POUT      );
  %put %str( OUTDSN      : Ergebniskonfiguration in         : &OUTDSN    );
  %put %str( VLIST1      : Liste der Var.namen in INFIX     : &VLIST1    );
  %put %str( VLIST2      : Liste der Var.namen in INROT     : &VLIST2    );
  %put %str( ------------------------------------------------------------ );
%end;

PROC IML;

%* - Generieren der Module der Druckausgabe und Aufruf   *;
%if &POUT=YES %then %do;
 start IDIST;
   DIF = X - Y;
   DIST = DIF##2;
   DIST = DIST(|,+|);
   VORHER = sqrt(DIST);
   MDIS1 = sum(VORHER)/ENN;
   DIF = X - ITEM;
   DIST = DIF##2;
   free DIF;
   DIST = DIST(|,+|);
   DISTANZ = VORHER || (sqrt(DIST));
   MITTEL = MDIS1 || (sum(sqrt(DIST))/ENN);
   free DIST;
   dtitel = {"vorher" "nachher"};
   %if DOUT=YES %then %do;
    print 'euclidische Distanzen:',,
                              DISTANZ(|format=8.5 r=&ID c=DTITEL|);
   %end;
   print 'mittl. euclidische Distanzen:',,
                              MITTEL(|format=8.5 c=DTITEL|) ;
 finish;

 start CONFCORR(A, B);
   ENN = nrow(A);
   EMM = ncol(A);
   N   = ENN * EMM;
   CMAT = shape(A, N) || shape(B, N);
   SUMME = CMAT(|+,|);
   XPX = CMAT` * CMAT - SUMME` * SUMME / N;
   S = diag(1/sqrt(vecdiag(XPX)));
   CORR = S * XPX * S;
   return (CORR(|1,2|));
 finish;


 start KONGRU (X, Y);
   ENN = nrow(X);
   EMM = ncol(X);
   DISTX = j(ENN,ENN,0);
   DISTY = j(ENN,ENN,0);

   do J = 1 to EMM;
     DIFX = j(ENN,ENN,0);
     DIFY = j(ENN,ENN,0);
     do I = 2 to ENN;
       do K = 1 to (I-1);
         DIFX(|I,K|) = X(|I,J|) - X(|K,J|);
         DIFY(|I,K|) = Y(|I,J|) - Y(|K,J|);
       end;
     end;
     DIFX = DIFX##2;
```

```
      DIFY = DIFY##2;
      DISTX = DISTX + DIFX;
      DISTY = DISTY + DIFY;
    end;
    DISTX = sqrt(DISTX);
    DISTY = sqrt(DISTY);
    return (sum(DISTX#DISTY)/sqrt(sum(DISTX##2)*sum(DISTY##2)));

 finish;

%* - Generieren der Default-Belegungen   --------------- *;
 PRAE="DIM";
  %if &VLIST1= %then %do;
   VLIST1 = concat (cshape (PRAE,ncol(1:&DIMS),1,length(PRAE)),
                    char ((1:&DIMS)`,1));
  %end;
  %else %do;
   VLIST1 = {&VLIST1};
  %end;

  %if &VLIST2= %then %do;
   VLIST2 = concat (cshape (PRAE,ncol(1:&DIMS),1,length(PRAE)),
                    char ((1:&DIMS)`,1));
  %end;
  %else %do;
   VLIST2 = {&VLIST2};
  %end;

 VARS    = concat (cshape (PRAE,ncol(1:&DIMS),1,length(PRAE)),
                    char ((1:&DIMS)`,1));

%* - Einlesen aus der/den Datei(en)   ------------------ *;
 use &INFIX;
 read all var {&ID} into &ID;
 if type(&ID) = "N" then &ID = char(&ID);
 read all var VLIST1 into X;
 close &INFIX;

 use &INROT;
 read all var VLIST2 into Y;
 close &INROT;


%* - Fehlerpruefung    --------------------------------- *;
 ENNY= nrow(Y);
 ENN = nrow(X);
 EMMY= ncol(Y);
 EMM = ncol(X);
 if (ENNY ^= ENN) | (EMMY ^= EMM) then do;
   reset log;
   print 'ERROR(PROKRUST): Eingabefehler bei Prokrustes Rotation:   ',
         '       die Matrizen X und Y haben unvertraegliches Format.';
   stop;
 end;
 else if ((EMMY = 0)|(EMM = 0))|((EMMY ^= &DIMS)|(EMM ^= &DIMS)) then do;
   reset log;
   print 'ERROR(PROKRUST): Eingabefehler bei Prokrustes Rotation:',
         '       Mindestens eine der Matrizen X oder Y ist nicht ',
         '       definiert oder die Variablenlisten sind nicht   ',
         '       kompatibel mit den Dateien.                     ';
 end;
 else do;

  %* - Generieren der Module   -------------------------- *;

  ZET = i(ENN)-(j(ENN,ENN,1) / ENN);
```

```
 CEH = Y`*ZET*X;
 call SVD (PEH, DIAG, KUH, CEH);
 T = PEH * KUH`;
 free PEH KUH DIAG;
 K   = trace(T`*CEH) / trace(Y`*ZET*Y);
 free CEH ZET;
 TL  = ((X - (K*Y*T))`*j(ENN,1,1))/ENN;
  %if &POUT=YES %then %do;
   reset noname;
   print ' Transformationsmatrix T',, T (|format=8.5|);
    %if &MODEL=ALL %then %do;
     print ' Streckungsfaktor K',, K (|format=8.5|);
     print ' Translationsvektor TL',, TL (|format=8.5|);
    %end;
  %end;
 ITEM = (Y*T);
 %if &MODEL=ORTHODIL %then %do;
  ITEM = (K*ITEM);
 %end;
 %if &MODEL=ALL %then %do;
  TLAT = shape (TL, ENN, EMM);
  ITEM = (K*ITEM) + TLAT;
 %end;


 C = KONGRU (X,Y);
 K = sqrt(1-C##2);
 run IDIST;
 R = CONFCORR (X,ITEM);
 RQUADRAT = R##2;
 print 'Ergebnismatrix ',, ITEM (|format=8.5 c=VARS r=&ID|);
 KTITEL = {"Kongruenz" "Alienation" "Korrelation" "R-Quadrat"};
 reset fw=14;
 print (C || K || R || RQUADRAT) (|c=KTITEL f=10.6|);
 reset name;
%end;



%* ------------------------------------------------------ *;
%* - Hauptprogramm  ------------------------------------- *;
%* ------------------------------------------------------ *;

create &OUTDSN from ITEM (|c=VARS r=&ID|);
append from ITEM (|r=&ID|);
close &OUTDSN;

end;

IMLENDE:
QUIT;

%ENDE:
%MEND PROKRUST;
```

Anhang C: Source-Code des Makros SIMIL

```
%MACRO SIMIL (INDSN1, ID, INDSN2, DIMS=3,
              VLIST1=, VLIST2=,
              OUTDSN=SIMILOUT);

%put %str(* ------------------------------------------------- *);
%put %str(*  Kongruenz-, Alienations- & Korrelationskoeffizienten *);
%put %str(*                                                   *);
%put %str(*              AUTOR: M. Huerkamp                   *);
%put %str(*            VERSION: 0.3 OS/2                      *);
%put %str(*  LETZTE AENDERUNG: 19.03.1994                     *);
%put %str(*                                                   *);
%put %str(* ------------------------------------------------- *);
%put %str( );

%if &INDSN2= or &dims < 2 %then %do;
  %put %str(                                                  );
  %put %str( ERROR: Fehlerhafter Aufruf Macro SIMIL           );
  %put %str( ERROR: zu wenige obligatorische Parameter        );
  %put %str( ERROR: oder DIMS kleiner als 2 gesetzt.          );
  %put %str( ------------------------------------------------ );
  %put %str(Syntax:                                           );
  %put %str(SIMIL (positionsparms, schluesselwortparms)       );
  %put %str(   positionsparms (obligatorisch):                );
  %put %str(     INDSN1  Name der 1. Eingabedatei             );
  %put %str(     ID      ID-Variable, (in INDSN1, char )      );
  %put %str(     INDSN2  Name der 2. Eingabedatei             );
  %put %str(   schluesselwortparms (optional):                );
  %put %str(     DIMS=   Anzahl der Dimensionen     Default=3 );
  %put %str(     OUTDSN= Name der Ausgabedatei      Default=SIMILOUT);
  %put %str(             enthaelt die Koeffizienten           );
  %put %str(     VLIST1= Liste der Variablennamen in INDSN1   );
  %put %str(             ohne Angabe: Praefix DIM, Suffix 1..DIMS );
  %put %str(     VLIST2= Liste der Variablennamen in INDSN2   );
  %put %str(             ohne Angabe: Praefix DIM, Suffix 1..DIMS );
  %put %str(Beispiel:                                         );
  %put %str(  SIMIL %(KOORD1, ID, KOORD2, DIMS=3, OUTDSN=OUTDAT, );
  %put %str(            VLIST2="FAC1" "FAC2" "FAC3" %)        );
  %put %str( ------------------------------------------------ );
  %put %str( );
  %goto ENDE;
%end;

%else %do;
  %put %str( SIMIL - Parameter - Ausgabe: );
  %put %str( ------------------------------------------------ );
  %put %str( Name der 1. Eingabe-Datei       : &INDSN1        );
  %put %str( Name der 2. Eingabe-Datei       : &INDSN2        );
  %put %str( ID - Variable                   : &ID            );
  %put %str( Anzahl der Dimensionen          : &DIMS          );
  %put %str( Name der Ausgabedatei           : &OUTDSN        );
  %put %str( Liste der Var.namen in INDSN1   : &VLIST1        );
  %put %str( Liste der Var.namen in INDSN2   : &VLIST2        );
  %put %str( ------------------------------------------------ );
%end;


PROC IML;

%* -  Generieren der Module  ------------------------------- *;

start CORR (X);
  N = nrow(X);
```

```
  SUMME = X[+,];
  XPX = X` * X - SUMME` * SUMME / N;
  S = diag(1/sqrt(vecdiag(XPX)));
  return (S*XPX*S);
finish;


start KONGRU (X, Y);
  ENN = nrow(X);
  EMM = ncol(X);
  DISTX = shape(0,ENN,ENN);
  DISTY = shape(0,ENN,ENN);

  do J = 1 to EMM;
    DIFX = shape(0,ENN,ENN);
    DIFY = shape(0,ENN,ENN);
    do I = 2 to ENN;
      do K = 1 to (I-1);
        DIFX[I,K] = X[I,J] - X[K,J];
        DIFY[I,K] = Y[I,J] - Y[K,J];
      end;
    end;
    DIFX = DIFX##2;
    DIFY = DIFY##2;
    DISTX = DISTX + DIFX;
    DISTY = DISTY + DIFY;
  end;
DISTX = sqrt(DISTX);
DISTY = sqrt(DISTY);
return (sum(DISTX#DISTY)/sqrt(sum(DISTX##2)*sum(DISTY##2)));

finish;

%* -  Generieren der Default-Belegungen  ---------------------------- *;

   PRAE="DIM";
   %if &VLIST1= %then %do;
    VLIST1 = concat (cshape (PRAE,ncol(1:&DIMS),1,length(PRAE)),
                     char ((1:&DIMS)`,1));
   %end;
   %else %do;
    VLIST1 = {&VLIST1};
   %end;

   %if &VLIST2= %then %do;
    VLIST2 = concat (cshape (PRAE,ncol(1:&DIMS),1,length(PRAE)),
                     char ((1:&DIMS)`,1));
   %end;
   %else %do;
    VLIST2 = {&VLIST2};
   %end;

   VARS = concat (cshape (PRAE,ncol(1:&DIMS),1,length(PRAE)),
                  char ((1:&DIMS)`,1));

%* -  Einlesen aus der/den Datei(en)  ------------------------------ *;

  use &INDSN1;
  read all var {&ID} into &ID;
  read all var VLIST1 into X;
  close &INDSN1;

  use &INDSN2;
  read all var VLIST2 into Y;
  close &INDSN2;
```

```
%* -  Fehlerpruefung  ------------------------------------------------ *;
  ENNY= nrow(Y);
  ENN = nrow(X);
  EMMY= ncol(Y);
  EMM = ncol(X);
  if (nrow(Y) ^= ENN) | (ncol(Y) ^= EMM) | (ENNY = 0 | ENN = 0) then do;
    reset log;
    print 'ERROR(SIMIL): Eingabefehler bei Koeffizienten-Makro',
          '       Matrizen X und Y haben unvertraegliches Format';
    stop;
  end;
  else if ((ENNY = 0)|(ENN = 0))|((EMM ^= &DIMS)|(EMMY ^= &DIMS)) then do;
    reset log;
    print 'ERROR(SIMIL): Mindestens eine der Matrizen X und Y',
          '       ist nicht definiert oder die Variablenliste',
          '       ist nicht kompatibel mit den Dateien.';
    stop;
  end;
  else do;


  %* ------------------------------------------------------------------ *;
  %* -  Hauptprogramm  ------------------------------------------------ *;
  %* ------------------------------------------------------------------ *;


  CONGRU=KONGRU(X, Y);
  PRINT "Kongruenz : "CONGRU,;
  ALIEN= sqrt(1-CONGRU##2);
  PRINT "Alienation: "ALIEN,;
    XS = shape(X, EMM*ENN);
    IS = shape(Y, EMM*ENN);
    temp=CORR(XS||IS);
    korr=temp[1,2];
    RSQ= (korr##2);

  PRINT "R         : " Korr,;
  PRINT "R-Quadrat : "RSQ,;

  end;

%IMLENDE:
QUIT;

%ENDE:
%MEND SIMIL;
```

Anhang D: Source-Code des Makros KOORDIST

```
%MACRO KOORDIST (INDSN,ID,
                 DIMS=3,
                 VLIST=);

%* ------------------------------------------------------------ *;
%* ------------------------------------------------------------ *;
%put %str( );
%put %str(* ------------------------------------------------- *);
%put %str(*          DISTANZEN DER PUNKTE EINER               *);
%put %str(*          KOORDINATEN-MATRIX KOORDIST              *);
%put %str(*                                                   *);
%put %str(*              AUTOR: M. HUERKAMP                   *);
%put %str(*            VERSION: 0.11 - OS/2                   *);
%put %str(*  LETZTE AENDERUNG: 28.3.1994                      *);
%put %str(*                                                   *);
%put %str(* ------------------------------------------------- *);
%put %str( );

%if &ID= or &DIMS < 2 %then %do;
  %put %str( ERROR: Fehlerhafter Aufruf Macro KOORDIST          );
  %put %str( ERROR: Zu wenige obligatorische Parameter          );
  %put %str( ERROR: oder DIMS kleiner als 2 gesetzt.            );
  %put %str( ---------------------------------------------- );
  %put %str( Syntax:                                            );
  %put %str( KOORDIST (positionsparms, schluesselwortparms)     );
  %put %str(   positionsparms (obligatorisch):                  );
  %put %str(     INDSN  Name der Eingabematrix                  );
  %put %str(     ID       ID-Variable, in INDSN                 );
  %put %str(   schluesselwortparms (optional):                  );
  %put %str(     DIMS=    Anzahl der Dimensionen, >=2           );
  %put %str(              Voreinstellung: 3                     );
  %put %str(     VLIST=   Liste der Variablennamen in INDSN     );
  %put %str(              Voreinstellung: DIM1..DIMs            );
  %put %str(                                                    );
  %put %str( Beispiel:                                          );
  %put %str(   KOORDIST (U2HYP,nr,DIMS=2,VLIST="FAC1" "FAC2")  );
  %put %str( ---------------------------------------------- );
  %put %str( );
  %goto ENDE;
%end;
%else %do;
  %put %str( KOORDIST-Parameter-Ausgabe:                        );
  %put %str( ---------------------------------------------- );
  %put %str( Eingangsmatrix           : &INDSN                  );
  %put %str( ID-Variable              : &ID                     );
  %put %str( Anzahl der Dimensionen   : &DIMS                   );
  %put %str( Liste der Variablennamen : &VLIST                  );
  %put %str( ---------------------------------------------- );
%end;

PROC IML;

%* - Generieren des Moduls   *;

  START KOORD(X);

    ENN = NROW(X);
    EMM = NCOL(X);
    DIST = SHAPE(0,ENN,ENN);
    DO J = 1 TO EMM;
      DIF = SHAPE(0,ENN,ENN);
      DO I = 2 TO ENN;
```

```
        DO K = 1 TO (I-1);
          DIF[I,K] = X[I,J] - X[K,J];
        END;
      END;
      DIF = DIF##2;
      DIST = DIST + DIF;
    END;
    DIST = DIST##0.5;
    DIST = DIST + DIST`;
    PRINT 'DISTANZMATRIX: ',,DIST[format=5.3];
  FINISH;


%* -  Generieren der Default-Belegungen  ------------------------- *;

   PRAE="DIM";
   %if &VLIST= %then %do;
    VLIST = concat (cshape (PRAE,ncol(1:&DIMS),1,length(PRAE)),
                    char ((1:&DIMS)`,1));
   %end;
   %else %do;
    VLIST = {&VLIST};
   %end;

   VARS = concat (cshape (PRAE,ncol(1:&DIMS),1,length(PRAE)),
                  char ((1:&DIMS)`,1));

%* -  Einlesen aus der/den Datei(en)  ---------------------------- *;

  use &INDSN;
  read all var {&ID} into &ID;
  IF type(&ID) = "N" then &ID = char(&ID);
  read all var VLIST into X;
  close &INDSN;


%* ------------------------------------------------------------------ *;
%* -  Hauptprogramm  ------------------------------------------------ *;
%* ------------------------------------------------------------------ *;

run KOORD(X);

QUIT;

%ENDE:
%MEND KOORDIST;
```

Anhang E: Source-Code des Makros G3DID

```
%MACRO G3DID (TIEFE, BREITE, HOEHE, ID,
              INDSN=_LAST_,
              SIZE=1, AXMAX=2, POS=3,
              SSIZE=.001, TICKS=5, ROTATE=70, TILT=70,
              POUT=NO, TEMP=T_E_M_P,
              OUTDSN=G3DIDOUT);
%PUT %str( );
%PUT %str(* ------------------------------------------------ *);
%PUT %str(*   G3DID: dreidimensionaler Plot mit Bezeichner   *);
%PUT %str(*                                                  *);
%PUT %str(*              AUTOR: M. Huerkamp                  *);
%PUT %str(*            VERSION: 0.2  OS/2                    *);
%PUT %str(*  LETZTE AENDERUNG: 19.03.1994                    *);
%PUT %str(* ------------------------------------------------ *);
%PUT %str( );
%IF &ID= %THEN %DO;
 %PUT %str(ERROR: Fehlerhafter Aufruf Macro G3DID                 );
 %PUT %str(ERROR: zu wenige obligatorische Parameter              );
 %PUT %str( ------------------------------------------------     );
 %PUT %str(Syntax:                                               );
 %PUT %str(G3DID (positionsparms, schluesselwortparms)           );
 %PUT %str(   positionsparms (obligatorisch):                    );
 %PUT %str(     TIEFE   Tiefendimension                          );
 %PUT %str(     BREITE  Breitendimension                         );
 %PUT %str(     HOEHE   Hoehendimension                          );
 %PUT %str(     ID      ID-Variable                              );
 %PUT %str(   schluesselwortparms (optional):                    );
 %PUT %str(     INDSN=  Name der Eingabedatei          Default=_LAST_   );
 %PUT %str(     OUTDSN= Name der Ausgabedatei          Default=G3DIDOUT);
 %PUT %str(     POUT=   Ausgabe der Annotate-Datei     Default=NO       );
 %put %str(     ROTATE= Rotationswinkel                          );
 %put %str(     TILT=   Kipp-Winkel                              );
 %put %str(     SIZE=   Groesse des ID-Zeichens                  );
 %put %str(     SSIZE=  Groesse des Plotsymbols                  );
 %PUT %str(     TICKS=  Anzahl der Achsenmarkierungen Default=5  );
 %PUT %str(     AXMAX=  Maximale Koordinate           Default=2  );
 %PUT %str(     TEMP=   Name der Temporaeren Datei    Default=T_E_M_P);
 %PUT %str(Beispiel:                                             );
 %PUT %str(  G3DID %(dim2, dim1, dim3, id, INDSN=indat, OUTDSN=outdat,);
 %PUT %str(         TICKS=7 AXMAX=5 %)                           );
 %PUT %str( ------------------------------------------------     );
 %PUT %str( );
 %GOTO ENDE;
%END;
%ELSE %DO;
 %LET POUT = %upcase(&POUT);
 %PUT %STR( G3DID - Parameter - Ausgabe:                         );
 %PUT %str( ------------------------------------------------     );
 %PUT %str( Tiefendimension              : &TIEFE                );
 %PUT %str( Breitendimension             : &BREITE               );
 %PUT %str( Hoehendimension              : &HOEHE                );
 %PUT %str( ID-Variable                  : &ID                   );
 %PUT %str( Name der Eingabedatei        : &INDSN                );
 %PUT %str( Name der Ausgabedatei        : &OUTDSN               );
 %PUT %str( Name der temporaeren Datei   : &TEMP                 );
 %PUT %str( Ausgabe der Annotatedatei    : &POUT                 );
 %PUT %str( Rotations-Winkel             : &ROTATE               );
 %PUT %str( Kipp-Winkel                  : &TILT                 );
 %PUT %str( Groesse des ID-Zeichens      : &SIZE                 );
 %PUT %str( Groesse des Plot-Symbols     : &SSIZE                );
 %PUT %str( Anzahl der Achsenmarkierungen : &TICKS               );
 %PUT %str( Maximale Koordinate          : &AXMAX                );
```

```
 %PUT %str( ------------------------------------------------- );
 %PUT %str( );
%END;

%* -  DATA-Steps  ---------------------------------------- *;

DATA &TEMP;
  retain &ID " ";
  &TIEFE  = -&AXMAX;
  &BREITE =  &AXMAX;
  &HOEHE  = -&AXMAX; OUTPUT;
  &TIEFE  =  &AXMAX;
  &BREITE = -&AXMAX;
  &HOEHE  =  &AXMAX; OUTPUT;

DATA &TEMP;
  set &INDSN &TEMP;

DATA &OUTDSN;
  SET &INDSN;
  LENGTH TEXT FUNCTION COLOR STYLE $ 8;
  LENGTH XSYS YSYS HSYS WHEN POSITION $ 1;
  RETAIN XSYS YSYS "2" HSYS "4" POSITION "&POS" WHEN "A";
  RETAIN ANGLE ROTATE 0;
  RETAIN STYLE "SIMPLEX" COLOR "BLACK" FUNCTION "LABEL   ";
  TEXT=TRIM(LEFT(&ID));
  X = &TIEFE;
  Y = &BREITE;
  Z = &HOEHE;
  SIZE = &SIZE;
  OUTPUT;

%IF &POUT=NO %THEN ;
%ELSE %DO;
  proc print data=&OUTDSN;
%END;

%* ------------------------------------------------------- *;
%* -  Haupt-Programm  ----------------------------------- *;
%* ------------------------------------------------------- *;

PROC G3D DATA=&TEMP ANNO=&OUTDSN;
  SCATTER &BREITE*&TIEFE=&HOEHE
          /SIZE=&SSIZE SHAPE='BALLOON' COLOR='black' grid
          rotate=&ROTATE tilt=&TILT
          xticknum=&TICKS yticknum=&TICKS zticknum=&TICKS;
  TITLE F=SIMPLEX "G3D-PLOT Datei &INDSN am &SYSDATE &SYSTIME ";
%ENDE:
%MEND G3DID;
```

Anhang F: Beispiels-Programmläufe

Demonstrationsprogramm PROSIMKO für die Makros PROKRUST, SIMIL und KOORDIST:

```
* ------------------------------------------------------- *;
* PROSIMKO.SAS                                            *;
* Test der Macros PROKRUST, SIMIL und KOORDIST            *;
* ------------------------------------------------------- *;
title  'PROSIMKO.SAS';
%include 'd:\progs\kurt\prokrust.mac';
%include 'd:\progs\kurt\simil.mac';
%include 'd:\progs\kurt\koordist.mac';

data BG81X471;
  input dim1 dim2;
  NR = left(trim(_N_));
cards;
 1  2
-1  2
-1 -2
 1 -2
;

data BG81Y471;
  input dim1 dim2;
  NR = left(trim(_N_));
cards;
 .07 2.62
 .93 3.12
1.93 1.38
1.07 0.88
;

* ------------------------------------------------------- *;
* Aufruf Macro PROKRUST                                   *;
* ------------------------------------------------------- *;
%PROKRUST (BG81X471, nr, BG81Y471, dims=2, outdsn=BORG1);
run;

proc print data=BORG1;
run;

%SIMIL (BG81X471, nr, BG81Y471, dims=2);
run;

%KOORDIST (BG81X471, nr, dims=2);
run;
```

Das Log-Protokoll des Programms PROSIMKO:

```
1133  * ------------------------------------------------------ *;
1134  * PROSIMKO.SAS                                           *;
1135  * Test der Macros PROKRUST, SIMIL und KOORDIST           *;
1136  * ------------------------------------------------------ *;
1137  title  'PROSIMKO.SAS';
1138  %include 'd:\progs\kurt\prokrust.mac';
1383  %include 'd:\progs\kurt\simil.mac';
1555  %include 'd:\progs\kurt\koordist.mac';
1664
1665  data BG81X471;
1666    input dim1 dim2;
1667    NR = left(trim(_N_));
1668  cards;
NOTE: Numeric values have been converted to character values
      at the places given by:
      (Line):(Column).
      1667:18
NOTE: The data set WORK.BG81X471 has 4 observations and 3 variables.
NOTE: The DATA statement used 0.9 seconds.
1673  ;
1674
1675  data BG81Y471;
1676    input dim1 dim2;
1677    NR = left(trim(_N_));
1678  cards;

NOTE: Numeric values have been converted to character values
      at the places given by:
      (Line):(Column).
      1677:18
NOTE: The data set WORK.BG81Y471 has 4 observations and 3 variables.
NOTE: The DATA statement used 1.54 seconds.

1683  ;
1684
1685  * ------------------------------------------------------ *;
1686  * Aufruf Macro PROKRUST                                  *;
1687  * ------------------------------------------------------ *;

1688  %PROKRUST (BG81X471, nr, BG81Y471, dims=2, outdsn=BORG1);
* ------------------------------------------------- *
*            PROKRUSTES-ROTATION                    *
*                                                   *
*             AUTOR: M. Huerkamp                    *
*           VERSION: 0.45 MAIN                      *
*  LETZTE AENDERUNG: 20.12.93                       *
*                                                   *
* ------------------------------------------------- *

 PROKRUST - Parameter - Ausgabe:
 -------------------------------------------------------
 INFIX      : Zielkonfiguration aus        : BG81X471
 INROT      : Rotationskonfiguration aus   : BG81Y471
 ID         : ID - Variable                : nr
 DIMS       : Anzahl der Dimensionen       : 2
 MODEL      : gewaehltes Modell            : ALL
 DOUT       : Ausgabe der Einzeldistanzen  : NO
 POUT       : Druckausgabe                 : YES
 OUTDSN     : Ergebniskonfiguration in     : BORG1
 VLIST1     : Liste der Var.namen in INFIX :
 VLIST2     : Liste der Var.namen in INROT :
 -------------------------------------------------------
```

```
IML Ready
NOTE: Module IDIST defined.
NOTE: Module CONFCORR defined.
NOTE: Module KONGRU defined.
NOTE: The data set WORK.BORG1 has 4 observations and 3 variables.
Exiting IML.
NOTE: The PROCEDURE IML used 2.0 seconds.
1689  run;
1690
1691  proc print data=BORG1;
1692  run;
NOTE: The PROCEDURE PRINT used 0.46 seconds.

1693
1694  %SIMIL (BG81X471, nr, BG81Y471, dims=2);
* ------------------------------------------------- *
*  Kongruenz-, Alienations- & Korrelationskoeffizienten *
*                                                   *
*             AUTOR: M. Huerkamp                    *
*           VERSION: 0.3 OS/2                       *
*  LETZTE AENDERUNG: 19.03.1994                     *
*                                                   *
* ------------------------------------------------- *

 SIMIL - Parameter - Ausgabe:
 ---------------------------------------------------
 Name der 1. Eingabe-Datei       : BG81X471
 Name der 2. Eingabe-Datei       : BG81Y471
 ID - Variable                   : nr
 Anzahl der Dimensionen          : 2
 Name der Ausgabedatei           : SIMILOUT
 Liste der Var.namen in INDSN1   :
 Liste der Var.namen in INDSN2   :
 ---------------------------------------------------
IML Ready
NOTE: Module CORR defined.
NOTE: Module KONGRU defined.
Exiting IML.
NOTE: The PROCEDURE IML used 1.28 seconds.
1695  run;
1696
1697  %KOORDIST (BG81X471, nr, dims=2);
* ------------------------------------------------- *
*          DISTANZEN DER PUNKTE EINER               *
*          KOORDINATEN-MATRIX KOORDIST              *
*                                                   *
*              AUTOR: M. HUERKAMP                   *
*            VERSION: 0.11 - OS/2                   *
*    LETZTE AENDERUNG: 28.3.1994                    *
*                                                   *
* ------------------------------------------------- *
 KOORDIST-Parameter-Ausgabe:
 ---------------------------------------------
 Eingangsmatrix           : BG81X471
 ID-Variable              : nr
 Anzahl der Dimensionen   : 2
 Liste der Variablennamen :
 ---------------------------------------------
IML Ready
NOTE: Module KOORD defined.
Exiting IML.
NOTE: The PROCEDURE IML used 0.84 seconds.

1698  run;
```

Der Output des Programms PROSIMKO:

PROSIMKO.SAS 22:46 Tuesday, November 1, 1994 6

Transformationsmatrix T

-0.86652 -0.49915
-0.49915 0.86652

Streckungsfaktor K

1.99655

Translationsvektor TL

3.72319
-2.46353

mittl. euclidische Distanzen:

vorher nachher

2.67580 0.00893

Ergebnismatrix

	DIM1	DIM2
1	0.99107	1.99944
2	-0.99506	2.00741
3	-0.99107	-1.99944
4	0.99506	-2.00741

Kongruenz	Alienation	Korrelation	R-Quadrat
0.999996	0.002822	0.999992	0.999984

OBS	DIM1	DIM2	NR
1	0.99107	1.99944	1
2	-0.99506	2.00741	2
3	-0.99107	-1.99944	3
4	0.99506	-2.00741	4

CONGRU
Kongruenz : 0.999996

ALIEN
Alienation: 0.0028217

KORR
R : 0.4423177

RSQ
R-Quadrat : 0.195645

DISTANZMATRIX:

DIST
0.000 2.000 4.472 4.000
2.000 0.000 4.000 4.472
4.472 4.000 0.000 2.000
4.000 4.472 2.000 0.000

Demonstrationsprogramm G3DID.SAS für das Makro G3DID:

```
* ------------------------------------------------------------ *;
* G3DID.SAS                                                    *;
* Test des Macros G3DID.MAC                                    *;
* ------------------------------------------------------------ *;
title  'G3DID.SAS';
%include 'd:\progs\kurt\g3did.mac';

data testg3d;
  input s $ dim1-dim3;
cards;
pyra 1    2   -1
diam 2   -2    1
cube .5  -.5   2
;

filename graph 'd:\progs\kurt\test.hpg';

goptions device = hp7475 noprompt
         colors = (black)
         gsfmode= replace
         gsfname= graph   ;

%G3DID(dim1, dim2, dim3, s, indsn=testg3d, ssize=3, pos=5, axmax=3);
run;
```

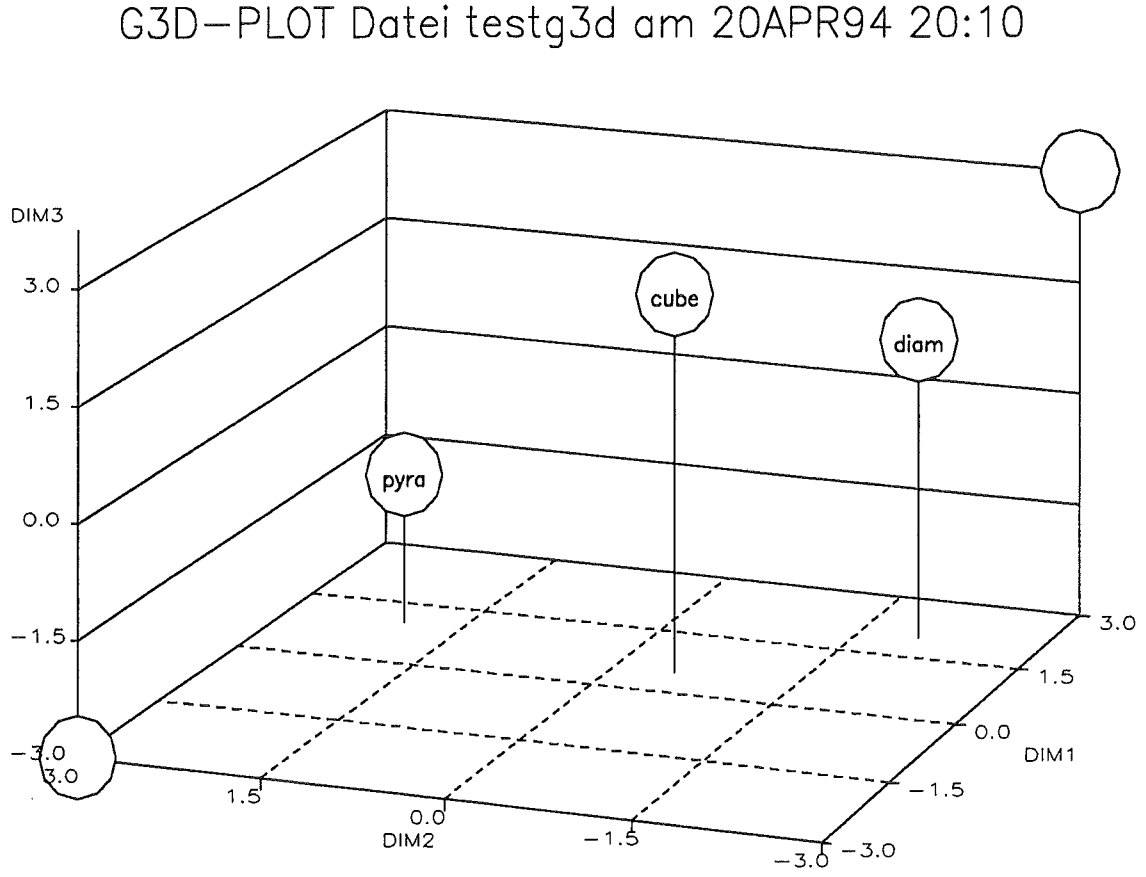

Abb. 12: Beispielsgraphik aus dem vorhergehenden Testprogramm mit dem Makro G3DID

Verzeichnis der Arbeiten aus dem Sonderforschungsbereich 245

Heidelberg/Mannheim

Nr. 1 Schwarz, S., Wagner, F. & Kruse, L.: Soziale Repräsentation und Sprache: Gruppenspezifische Wissensbestände und ihre Wirkung bei der sprachlichen Konstruktion und Rekonstruktion geschlechtstypischer Episoden. Februar 1989.

Nr. 2 Wintermantel, M., Laux, H. & Fehr, U.: Anweisung zum Handeln: Bilder oder Wörter. März 1989.

Nr. 3 Herrmann, Th., Dittrich, S., Hornung-Linkenheil, A., Graf, R. & Egel, H.: Sprecherziele und Lokalisationssequenzen: Über die antizipatorische Aktivierung von Wie-Schemata. April 1989.

Nr. 4 Schwarz, S., Weniger, G. & Kruse, L. (unter Mitarbeit von R. Kohl): Soziale Repräsentation und Sprache: Männertypen: Überindividuelle Wissensbestände und individuelle Kognitionen. Juni 1989.

Nr. 5 Wagner, F., Theobald, H., Heß, K., Schwarz, S. & Kruse, L.: Soziale Repräsentation zum Mann: Gruppenspezifische Salienz und Strukturierung von Männertypen. Juni 1989.

Nr. 6 Schwarz, S. & Kruse, L.: Soziale Repräsentation und Sprache: Gruppenspezifische Unterschiede bei der sprachlichen Realisierung geschlechtstypischer Episoden. Juni 1989.

Nr. 7 Dorn-Mahler, H., Grabowski-Gellert, J., Funk-Müldner, K. & Winterhoff-Spurk, P.: Intonation bei Aufforderungen. Teil I: Theoretische Grundlagen. Juni 1989.

Nr. 8 Dorn-Mahler, H., Grabowski-Gellert, J., Funk-Müldner, K. & Winterhoff-Spurk, P.: Intonation bei Aufforderungen. Teil II: Eine experimentelle Untersuchung. Dezember 1989.

Nr. 9 Sommer, C. M. & Graumann, C. F.: Perspektivität und Sprache: Zur Rolle von habituellen Perspektiven. August 1989.

Nr. 10 Grabowski-Gellert, J. & Winterhoff-Spurk, P.: Schreiben ist Silber, Reden ist Gold. August 1989.

Nr. 11 Graf, R. & Herrmann, Th.: Zur sekundären Raumreferenz: Gegenüberobjekte bei nicht-kanonischer Betrachterposition. Dezember 1989.

Nr. 12 Grosser, Ch. & Mangold-Allwinn, R.: Objektbenennung in Serie: Zur partnerorientierten Ausführlichkeit von Erst- und Folgebenennungen. Dezember 1989.

Nr. 13 Grosser, Ch. & Mangold-Allwinn, R.: Zur Variabilität von Objektbenennungen in Abhängigkeit von Sprecherzielen und kognitiver Kompetenz des Partners. Dezember 1989.

Nr. 14 Gutfleisch-Rieck, I., Klein, W., Speck, A. & Spranz-Fogasy, Th.: Transkriptionsvereinbarungen für den Sonderforschungsbereich 245 „Sprechen und Sprachverstehen im sozialen Kontext". Dezember 1989.

Nr. 15 Herrmann, Th.: Vor, hinter, rechts und links: das 6H-Modell. Psychologische Studien zum sprachlichen Lokalisieren. Dezember 1989.

Nr. 16 Dittrich, S. & Herrmann, Th.: „Der Dom steht hinter dem Fahrrad." – Intendiertes Objekt oder Relatum? März 1990.

Nr. 17 Kilian, E., Herrmann, Th., Dittrich, S. & Dreyer, P.: Was- und Wie-Schemata beim Erzählen. Mai 1990.

Nr. 18 Herrmann, Th. & Graf, R.: Ein dualer Rechts-links-Effekt. Kognitiver Aufwand und Rotationswinkel bei intrinsischer Rechts-links-Lokalisation. August 1990.

Nr. 19 Wintermantel, M.: Dialogue between expert and novice: On differences in knowledge and means to reduce them. August 1990.

Nr. 20 Graumann, C. F.: Perspectivity in Language and Language Use. September 1990.

Nr. 21 Graumann, C. F.: Perspectival Structure and Dynamics in Dialogues. September 1990.

Nr. 22 Hofer, M., Pikowsky, B., Spranz-Fogasy, Th. & Fleischmann, Th.: Mannheimer Argumentations-Kategoriensystem (MAKS). Mannheimer Kategoriensystem für die Auswertung von Argumentationen in Gesprächen zwischen Müttern und jugendlichen Töchtern. Oktober 1990.

Nr. 23 Wagner, F., Huerkamp, M., Jockisch, H. & Graumann, C. F.: Sprachlich realisierte soziale Diskriminierungen: empirische Überprüfung eines Modells expliziter Diskriminierung. Oktober 1990.

Nr. 24 Rettig, H., Kiefer, L., Sommer, C. M. & Graumann, C. F.: Perspektivität und soziales Urteil: Wenn Versuchspersonen ihre Bezugsskalen selbst konstruieren. November 1990.

Nr. 25 Kiefer, L., Sommer, C. M. & Graumann, C. F.: Perspektivität und soziales Urteil: Klassische Urteilseffekte bei individueller Skalenkonstruktion. November 1990.

Nr. 26 Hofer, M., Pikowsky, B., Fleischmann, Th. & Spranz-Fogasy, Th.: Argumentationssequenzen in Konfliktgesprächen zwischen Müttern und Töchtern. November 1990.

Nr. 27 Funk-Müldner, K., Dorn-Mahler, H. & Winterhoff-Spurk, P.: Kategoriensystem zur Situationsabhängigkeit von Aufforderungen im betrieblichen Kontext. Dezember 1990.

Nr. 28 Groeben, N., Schreier, M. & Christmann, U.: Argumentationsintegrität (I): Herleitung, Explikation und Binnenstrukturierung des Konstrukts. Dezember 1990.

Nr. 29 Blickle, G. & Groeben, N.: Argumentationsintegrität (II): Zur psychologischen Realität des subjektiven Wertkonzepts – ein experimenteller Überprüfungsansatz am Beispiel ausgewählter Standards. Dezember 1990.

Nr. 30 Schreier, M. & Groeben, N.: Argumentationsintegrität (III): Rhetorische Strategien und Integritätsstandards. Dezember 1990.

Nr. 31 Sachtleber, S. & Schreier, M.: Argumentationsintegrität (IV): Sprachliche Manifestationen argumentativer Unintegrität – ein pragmalinguistisches Beschreibungsmodell und seine Anwendung. Dezember 1990.

Nr. 32 Dietrich, R., Egel, H., Maier-Schicht, B. & Neubauer, M.: ORACLE und die Analyse des Äußerungsaufbaus. Februar 1991.

Nr. 33 Nüse, R., Groeben, N. & Gauler, E.: Argumentationsintegrität (V): Diagnose argumentativer Unintegrität – (Wechsel-)wirkungen von Komponenten subjektiver Werturteile über argumentative Sprechhandlungen. März 1991.

Nr. 34 Christmann, U. & Groeben, N.: Argumentationsintegrität (VI): Subjektive Theorien über Argumentieren und Argumentationsintegrität – Erhebungsverfahren, inhaltsanalytische und heuristische Ergebnisse. März 1991.

Nr. 35 Graf, R., Dittrich, S., Kilian, E. & Herrmann, Th.: Lokalisationssequenzen: Sprecherziele, Partnermerkmale und Objektkonstellationen (Teil II). Drei Erkundungsexperimente. März 1991.

Nr. 36 Hofer, M., Pikowsky, B., &. Fleischmann, Th.: Jugendliche unterschiedlichen Alters im argumentativen Konfliktgespräch mit ihrer Mutter. März 1991.

Nr. 37 Herrmann, Th., Graf, R. & Helmecke, E.: „Rechts" und „Links" unter variablen Betrachtungswinkeln: Nicht-Shepardsche Rotationen. April 1991.

Nr. 38 Herrmann, Th. & Grabowski, J.: Mündlichkeit, Schriftlichkeit und die nicht-terminalen Prozeßstufen der Sprachproduktion. Februar 1992.

Nr. 39 Thimm, C. & Kruse, L.: Dominanz, Macht und Status als Elemente sprachlicher Interaktion. Mai 1991.

Nr. 40 Thimm, C. & Kruse, L.: Sprachliche Effekte von Partnerhypothesen in dyadischen Situationen. September 1993.

Nr. 41 Thimm, C., Maier, S. & Kruse, L.: Statusrelationen in dyadischen Kommunikationssituationen: Zur Rolle von Partnerhypothesen. April 1994.

Nr. 42 Funk-Müldner, K., Dorn-Mahler, H. & Winterhoff-Spurk, P.: Nonverbales Verhalten beim Auffordern – ein Rollenspielexperiment. Dezember 1991.

Nr. 43 Dorn-Mahler, H., Funk-Müldner, K. & Winterhoff-Spurk, P.: AUFF_{KO} – Ein inhaltsanalytisches Kodiersystem zur Analyse von komplexen Aufforderungen. Oktober 1991.

Nr. 44 Herrmann, Th.: Sprachproduktion und erschwerte Wortfindung. Mai 1992.

Nr. 45 Grabowski, J., Herrmann, Th. & Weiß, P.: Wenn „vor" gleich „hinter" ist – zur multiplen Determination des Verstehens von Richtungspräpositionen. Juni 1992.

Nr. 46 Barattelli, St., Koelbing, H.G. & Kohlmann, U.: Ein Klassifikationssystem für komplexe Objektreferenzen. September 1992.

Nr. 47 Haury, Ch., Engelbert, H. M., Graf, R. & Herrmann, Th.: Lokalisationssequenzen auf der Basis von Karten- und Straßenwissen: Erste Erprobung einer Experimentalanordnung. August 1992.

Nr. 48 Schreier, M. & Czemmel, J.: Argumentationsintegrität (VII): Wie stabil sind die Standards der Argumentationsintegrität ? August 1992.

Nr. 49 Engelbert, H. M., Herrmann, Th. & Haury, Ch.: Ankereffekte bei der sprachlichen Linearisierung. Oktober 1992.

Nr. 50 Spranz-Fogasy, Th.: Bezugspunkte der Kontextualisierung sprachlicher Ausdrücke in Interaktionen. Ein Konzept zur analytischen Konstitution von Schlüsselwörtern. November 1992.

Nr. 51 Kiefer, M., Barattelli, St. & Mangold-Allwinn, R.: Kognition und Kommunikation: Ein integrativer Ansatz zur multiplen Determination der lexikalischen Spezifität der Objektklassenbezeichnung. Februar 1993.

Nr. 52 Spranz-Fogasy, Th.: Beteiligungsrollen und interaktive Bedeutungskonstitution. Februar 1993.

Nr. 53 Schreier, M. & Groeben, N.: Argumentationsintegrität (VIII): Zur psychologischen Realität des subjektiven Wertkonzepts. Eine experimentelle Überprüfung für die 11 Standards integeren Argumentierens. Dezember 1992.

Nr. 54 Sommer, C. M., Freitag, B. & Graumann, C. F.: Aggressive Interaction in Perspectival Discourse. März 1993.

Nr. 55 Huerkamp, M., Jockisch, H., Wagner, F. & Graumann, C. F.: Facetten expliziter sprachlicher Diskriminierung: Untersuchungen von Ausländer-Diskriminierungen anhand einer deutschen und einer ausländischen Stichprobe. Februar 1993.

Nr. 56 Rummer, R., Grabowski, J., Hauschildt, A. & Vorwerg, C.: Reden über Ereignisse: Der Einfluß von Sprecherzielen, sozialer Nähe und Institutionalisiertheitsgrad auf Sprachproduktionsprozesse. April 1993.

Nr. 57 Blickle, G.: Argumentationsintegrität (IX): Personale Antezedensbedingungen der Diagnose argumentativer Unintegrität. Juli 1993.

Nr. 58 Herrmann, Th., Buhl, H.M., Schweizer, K. & Janzen, G.: Zur repräsentationalen Basis des Ankereffekts. Kognitionspsychologische Untersuchungen zur sprachlichen Linearisierung. September 1993.

Nr. 59 Carroll, M.: Keeping spatial concepts on track in text production. A comparative analysis of the use of the concept path in descriptions and instructions in German. Oktober 1993.

Nr. 60 Speck, A.: Instruieren im Dialog. Oktober 1993.

Nr. 61 Herrmann, Th. & Grabowski, J.: Das Merkmalsproblem und das Identitätsproblem in der Theorie dualer, multimodaler und flexibler Repräsentationen von Konzepten und Wörtern (DMF-Theorie). November 1993.

Nr. 62 Rummer, R., Grabowski, J. & Vorwerg, C.: Zur situationsspezifischen Flexibilität zentraler Voreinstellungen bei ereignisbezogenen Sprachproduktionsprozessen. November 1993.

Nr. 63 Christmann, U. & Groeben, N.: Argumentationsintegrität (X): Realisierung argumentativer Redlichkeit und Reaktionen auf Unredlichkeit. November 1993.

Nr. 64 Christmann, U. & Groeben, N.: Argumentationsintegrität (XI): Retrognostische Überprüfung der Handlungsleitung subjektiver Theorien über Argumentationsintegrität bei Kommunalpolitikern/innen. November 1993.

Nr. 65 Schreier, M.: Argumentationsintegrität (XII): Sprachliche Manifestationsformen argumentativer Unintegrität in Konfliktgesprächen. Dezember 1993.

Nr. 66 Christmann, U., Groeben, N. & Küppers, A.: Argumentationsintegrität (XIII): Subjektive Theorien über Erkennen und Ansprechen von Unintegritäten im Argumentationsverlauf. Dezember 1993.

Nr. 67 Christmann, U. & Groeben, N.: Argumentationsintegrität (XIV): Der Einfluß von Valenz und Sequenzstruktur argumentativer Unintegrität auf kognitive und emotionale Komponenten von Diagnose- und Bewertungsreaktionen. Dezember 1993.

Nr. 68 Schreier, M., Groeben, N. & Mlynski, G.: Argumentationsintegrität (XV): Der Einfluß von Bewußtheitsindikatoren und (Un-)Höflichkeit auf die Rezeption argumentativer Unintegrität. Februar 1994.

Nr. 69 Thimm, C., Rademacher, U. & Augenstein, S.: "Power-Related Talk (PRT)": Ein Auswertungsmodell. Januar 1994.

Nr. 70 Kiefer, L., Rettig, H., Sommer, C. M. & Graumann, C. F.: Perspektivität und soziales Urteil: Vier Sichtweisen zum Thema "Ausländerstop". Januar 1994.

Nr. 71 Graumann, C. F.: Discriminatory Discourse. Conceptual and methodological problems. 1994.

Nr. 72 Huerkamp, M.: SAS-Makros zur Analyse und Darstellung mehrdimensionaler Punktekonfigurationen. April 1994.

Nr. 73 Galliker, M., Huerkamp, M., Höer, R. & Wagner, F.: Funktionen expliziter sprachlicher Diskriminierung: Validierung der Kernfacetten des Modells sprachlicher Diskriminierung. Juni 1994.

Nr. 74 Buhl, H.M., Schweizer, K. & Herrmann, Th.: Weitere Untersuchungen zum Ankereffekt. April 1994.

Nr. 75 Herrmann, Th.: Psychologie ohne 'Bedeutung'? Zur Wort-Konzept-Relation in der Psychologie. Mai 1994.

Nr. 76 Neubauer, M., Hub, I. & Thimm, C.: Transkribieren mit LaTeX: Transkriptionsregeln, Eingabeverfahren und Auswertungsmöglichkeiten. Mai 1994.

Nr. 77 Thimm, C. & Augenstein, S.: Sprachliche Effekte in hypothesengeleiteter Interaktion: Durchsetzungsstrategien in Aushandlungsgesprächen. Mai 1994.

Nr. 78 Sommer, C. M., Rettig, H., Kiefer, L. & Frankenhauser, D.: "Germany will be one single concrete block ...". Point of View and Reference to Topic Aspects in Adversial Discussions on Immigration. September 1994.

Nr. 79 Maier, S. & Kruse, L.: Ein Design zur Erfassung einer dialogischen Kommunikationssituation: Das Experiment "Terminabsprache". November 1994.

Nr. 80 Grabowski, J.: Schreiben als Systemregulation – Ansätze einer psychologischen Theorie der schriftlichen Sprachproduktion. Oktober 1994.

Nr. 81 Hermanns, F.: Schlüssel-, Schlag- und Fahnenwörter. Zu Begrifflichkeit und Theorie der lexikalischen ‹politischen Semantik›. Dezember 1994.

Nr. 82 Kiefer, L., Rettig, H., Frankenhauser, D., Sommer, C. M. & Graumann, C. F.: Perspektivität und Persuasion: Effektivität perspektivenrelevanter Persuasionsstrategien. Dezember 1994.

Nr. 83 Liebert, W.-A.: Das analytische Konzept "Schlüsselwort" in der linguistischen Tradition. Dezember 1994.

Nr. 84 Buhl, H. M., Schweizer, K. & Herrmann, Th.: Der Einfluß von Räumlichkeit und Reizmodalität auf den Ankereffekt. Dezember 1994.

Nr. 85 Koelbing, H.G., Mangold-Allwinn, R., Barattelli, St., Kohlmann, U. & Stutterheim, C. v.: Welchen Einfluß hat der Ausführende auf den Instruierenden ? Dezember 1994.

Nr. 86 Held, Th. & Maier-Schicht, B.: Benutzerhandbuch und Dokumentation eines Experimentalsystems auf der Basis der Expertensystemschale knoX. Dezember 1994.

Nr. 87 Maier-Schicht, B., Theiss, G. & Held, Th.: Ein Expertensystem als Experimentalsystem. Februar 1995.