Vorwort

Das *Statistical Analysis System*, kurz SAS genannt und vor rund 25 Jahren als statistisches Auswertungssystem entwickelt, ist heute weitaus mehr als ein reines Programm zur Durchführung von statistischen Auswertungen. Im Rahmen der zunehmenden Bedeutung von elektronisch gesteuerten Rechenanlagen in den Bereichen der Forschung und der Wirtschaft wird es immer wichtiger, Daten so zu halten und zu verwalten, daß sie leicht zugängig sind, um bei Bedarf Informationen, die an verschiedenen Stellen abgelegt sind, zusammenzuführen und gemeinsam auszuwerten und schließlich die Ergebnisse der Auswertung in ansprechender Form zu präsentieren. Die genannten Vorgänge werden heute gerne unter dem Begriff *Data Warehouse* zusammengefaßt: ein Warenhaus für die Daten mitsamt passender Infrastruktur, um jederzeit auf die notwendigen Informationen zugreifen zu können.

Das SAS-System, wie es sich heute präsentiert, möchte diese Aufgaben erfüllen. Für den Austausch und die Zusammenführung von unterschiedlichen Datenquellen stellt es Schnittstellen zu Datenbanksystemen wie dBASE, Oracle, DB2, Ingres oder MS Access zur Verfügung und gestattet den Zugriff auf Daten, die auf anderen Rechnern mit eventuell auch anderen Betriebssystemen abgelegt sind. Eine eigene Programmiersprache und die Möglichkeit, die *Structured Query Language* (SQL) zu verwenden, erlauben nahezu jede beliebige Operation auf vorhandenen Daten sowie den Zugriff auf Daten, die noch nicht in einem anderen System integriert vorliegen. Fertige Routinen ermöglichen die Berechnung von einfachen statistischen Kennwerten bis hin zu hochkomplexen Datenmodellen, wie verallgemeinerten linearen Modellen, Zeitreihenverfahren oder Neuronalen Netzwerken. Die Ergebnisse der Analysen können in Form von Grafiken und Tabellen aufbereitet werden, die anschließend an diverse Text- und Präsentationsprogramme zur Einbindung in Berichte weitergeleitet werden können. Während in den Anfangsjahren Universitäten und Forschungseinrichtungen die Hauptnutzer der Software waren, kommt das SAS-System heutzutage auch in so unterschiedlichen Bereichen zum Einsatz wie im Bank- und Versicherungswesen, in der Automobil- und Werkzeugfertigung und der pharmazeutischen Industrie.

Bei der Fülle der Funktionalitäten ist es unmöglich, den gesamten Umfang der SAS-Software zu beschreiben. Dieses Buch ist daher eher als Einführung gedacht. Es möchte den Leser an die Software heranführen, ihn mit ihr vertraut machen und dem Anwender das Gefühl der Hilflosigkeit nehmen, das ihn in Anbetracht der Komplexität des Systems am Anfang leicht überkommen kann. Ein erfahrener SAS-Anwender unterscheidet sich von dem Anfänger nicht etwa dadurch, daß er fehlerfrei programmiert, sondern dadurch, daß er auf die vielfältigen System- und Fehlermeldungen richtig reagieren kann. Der Schwerpunkt dieser Einführung liegt daher auch auf der Datenhaltung und -verarbeitung und auf dem Umgang mit dem SAS-System an sich. Anhand einiger grundlegender Statistik- und Grafikprozeduren werden dem Anwender Wege aufgezeigt, wie Analysen durchgeführt und Ergebnisse interpretiert und präsentiert werden können.

Das Buch wendet sich an Einsteiger, die das SAS-System im Rahmen ihres Studiums oder ihrer Berufstätigkeit einsetzen (müssen). Kleinere Aufgaben am Ende der einzelnen Kapitel unterstützen den tutoriellen Charakter des Buchs. Da der Kurs auf der im Moment aktuellen Version SAS 6.12 basiert und einige der darin verfügbaren interaktiven Möglichkeiten des Systems aufzeigt, kann dieses Buch auch für SAS-Anwender nützlich sein, die sich autodidaktisch in das Programm eingearbeitet haben oder die mit älteren Programmversionen „groß geworden“ sind und auf die neue Version umsteigen möchten.

Als Vorlage zu diesem Buch dienten Unterlagen zu Einführungskursen in das SAS-System, die am Universitätsrechenzentrum Heidelberg in den vergangenen vier Jahren gehalten wurden. Dank gebührt in diesem Zusammenhang Frank Elßner von der Universität Osnabrück, dessen Skript „Einführung in das Statistical Analysis System“ mich motivierte, meinen eigenen Kurs gründlich zu überarbeiten. Danken möchte ich auch Sabine Haag, Thomas Bruckner und Heike Pich für die sorgfältige Durchsicht des Manuskripts, Luzia Dietsche für die Umsetzung meiner formalen Wünsche in LaTeX-Befehle, Christa Preisendanz von ITP für ihren festen Glauben an die Fertigstellung dieses Buchs, dem Universitätsrechenzentrum Heidelberg für die großzügige Unterstützung und schließlich meinem Mann für seine Geduld und Toleranz.

Dieser Kurs wurde mit großer Sorgfalt erstellt, die Programme wurden getestet, die Screenshots eigenhändig erzeugt. Dennoch kann nicht ausgeschlossen werden, daß die Programme Fehler enthalten oder daß sie in einzelnen Umgebungen nicht fehlerfrei laufen. Für Hinweise auf Fehler und Inkonsistenzen wäre ich Ihnen daher sehr dankbar (e-mail: Carina.Ortseifen@urz.uni-heidelberg.de).

Heidelberg, im August 1997 Carina Ortseifen

Inhaltsverzeichnis

1

Zum Gebrauch des Buchs

Mit dem SAS-System ist es wie mit anderen Anwendungsprogrammen auch: Man lernt erst dann richtig mit ihnen umzugehen, wenn man sie selbst benutzt. Allein die Lektüre von Hand- und Lehrbüchern reicht nur wenigen Menschen aus, um ein Programm bedienen zu können. Dem Anfänger sei daher empfohlen, das Buch direkt neben einem Computer zu lesen, auf dem die SAS-Software installiert ist, damit die behandelten Verfahren und Aktionen selbst ausprobiert werden können. Die Aufgaben am Ende der jeweiligen Kapitel sollen weitere Anregungen zum praktischen Arbeiten geben.

Die SAS-Software ist eine Software, die auf nahezu allen verfügbaren Plattformen eingesetzt werden kann, auf Großrechnern unter MVS oder VM, auf Workstations unter AIX, HP oder Solaris und auf PCs unter Windows, OS/2 und Macintosh. Die Entwickler des Systems bemühen sich, sowohl die internen Abläufe als auch die Oberflächen einheitlich zu gestalten. Für den Anwender spielt es dabei so gut wie keine Rolle, ob er an einer Workstation oder an einem PC mit der SAS-Software arbeitet. Kleinere Unterschiede zwischen den Versionen gibt es natürlich auch, denn sobald eine Funktionalität vom Betriebssystem abhängt, kann sie nicht auf die anderen Betriebssysteme übertragen werden. Die Screenshots dieses Kurses zeigen die Situation, die sich dem Anwender mit SAS 6.12 unter Windows zeigt. Die Programmbeispiele wurden ebenfalls mit dieser Version ausgeführt und getestet. An einigen Stellen wird darauf hingewiesen, wie die entsprechenden Varianten unter einem anderen Betriebssystem aussehen könnten. Dennoch kann keine Garantie dafür übernommen werden, daß alle Programme unter jeder SAS-Version fehlerfrei laufen.

Als Einführung in das SAS-System beschäftigt sich das Buch im 2. Kapitel mit dem generellen Aufbau des Systems und den verschiedenen Arbeitsmodi. Die Benutzeroberfläche im interaktiven Betrieb wird vorgestellt und der Aufbau der Online-Hilfe ausführlich erläutert, da diese immer größere Bedeutung erlangt.

Im 3. Kapitel werden grundlegende Begriffe erläutert, die für die Arbeit mit dem SAS-System notwendig sind, und an einem einfachen Beispiel dessen Arbeitsweise vorgestellt. Der Aufbau der interaktiven Umgebung wird im 4. Kapitel behandelt. Der Leser lernt Kommandos und wichtige Funktionen des SAS-eigenen Editors kennen.

Im 5. Kapitel werden die verschiedenen Aspekte der Erstellung von SAS-Dateien behandelt. Der Leser erfährt, wie er diese Dateien dauerhaft anlegt und sich zu einem späteren Zeitpunkt einen Überblick über bereits erstellte Dateien verschaffen kann. Die zuvor angelegten Dateien können mit den in Kapitel 6 behandelten Prozeduren in Tabellenform angezeigt werden.

Das 7. Kapitel kehrt zum Datenmanagement zurück. Neben den Funktionalitäten zur Erzeugung neuer Variablen, zur Bildung von Teilmengen und zum Verknüpfen mehrerer Dateien, werden Möglichkeiten zum Umbenennen von Merkmalen und zur Vergabe aussagekräftiger Bezeichnungen vorgestellt.

In Kapitel 8 werden anhand der statistischen Verfahren zum Vergleich von Häufigkeiten und Mittelwerten und der Berechnung von Konfidenzintervallen und Zusammenhangsmaßen gezeigt, wie die Statistikprozeduren eingesetzt und die Ergebnisse interpretiert werden können.

In Kapitel 9 werden die Möglichkeiten der deskriptiven Statistik aufgezeigt und die notwendigen Schritte erläutert, die zur Einbindung der Grafiken in andere Anwendungsprogramme erforderlich sind.

Das 10. Kapitel stellt globale Anweisungen vor und zeigt Möglichkeiten auf, das Aussehen und die Funktionsweise des Systems an eigene Bedürfnisse anzupassen.

Im letzten Kapitel werden, nicht mehr so ausführlich wie in den übrigen Passagen, weitere Themenbereiche angesprochen, die in der praktischen Anwendung dieser Software von Bedeutung sind: Import von Fremddateien in das SAS-System, Dateneingabe über Masken, menügeführte SAS-Benutzung und Programmierung innerhalb des SAS-Systems.

Am Ende der Kapitel 3-10 befinden sich jeweils mehrere Aufgaben, die den Leser anregen sollen, praktisch mit dem SAS-System zu arbeiten, um eigene Erfahrungen zu sammeln. Betrachten Sie die Aufgaben als Anregungen. Variieren Sie die Beispiele und versuchen Sie, eigene Problemstellungen umzusetzen.

Im Anhang finden Sie Lösungsvorschläge zu den einzelnen Aufgaben. Außerdem werden dort Hintergrundinformationen zur Installation gegeben und die in den Beispielen verwendeten Dateien erläutert. Alle vorgestellten Kommandos, Prozedur- und Datenschritte sowie häufig auftretende Fehlermeldungen wurden zusammengestellt und können auch später hier nachgeschlagen werden.

Dem Anfänger wird empfohlen, die ersten sieben Kapitel des Buchs der Reihe nach durchzuarbeiten und sich ausreichend Zeit für die Bearbeitung der Aufgaben und eigener Fragestellungen zu nehmen. Die Kapitel zur Statistik, zur Grafik und zur Änderung globaler Einstellungen sowie das letzte „Schnupper-Kapitel“ können anschließend in beliebiger Reihenfolge bearbeitet werden.

Zur übersichtlichen Gestaltung und zur Erleichterung des Verständnisses wurden verschiedene Vereinbarungen im Schriftbild getroffen. Die Darstellung der Sprachelemente des SAS-Systems entspricht im wesentlichen den üblichen Konventionen der Handbücher von SAS Institute und erleichtert dem Leser das Zurechtfinden in der Originalliteratur.

- Wichtige Begriffe oder Übersetzungen werden durch Kursivschrift hervorgehoben.

  Beispiel: Das Lernprogramm (engl. *Computer Based Tutorial*) ...

- Die Namen der SAS-Module, Prozeduren, Anweisungen und Optionen sowie feste Datei-Endungen werden in Kapitälchen ausgegeben.

  Beispiel: Die Prozedur PRINT kennt u. a. die beiden Anweisungen VAR und SUM.

- Menüpunkte werden kursiv gesetzt und mit einem Pfeil verbunden.

  Beispiel: Wählen Sie dazu *Help* → *Online Training*.

- Schaltflächen mit Text innerhalb von Fenstern werden ebenfalls kursiv gesetzt.

  Beispiel: Nachdem Sie *OK* anklicken, werden die Eintragungen übernommen.

- Kommandos, die direkt vom Anwender über die Tastatur eingegeben werden, werden in einer nicht-proportionalen Schreibmaschinenschrift (Typewriter) angezeigt. Lange Kommandos können zumeist abgekürzt werden. Die Großbuchstaben weisen den erforderlichen Teil auf, die Kleinbuchstaben vervollständigen das Kommando zu einem verständlichen Wort.

  Für das Kommando `INClude` sind beispielsweise nur die ersten drei Buchstaben notwendig. Der Rest kann, muß aber nicht eingegeben werden.

- Eingaben vom Anwender, die in direkter Beziehung zu einem Programm stehen, werden ebenfalls in der Schreibmaschinenschrift ausgegeben.

  Beispiel: Der Dateiname `adressen` kann durch `firma` ersetzt werden.

- Die Funktions- und die Steuerungstasten werden umrandet dargestellt. Innerhalb des Rahmens erscheint die Bezeichnung, die auf einer deutschen

Tastatur vorgefunden wird. Werden die Tasten durch ein „+“ verbunden, muß die erste Taste festgehalten werden, wenn die zweite gedrückt wird.

Beispiel: Mit F1 rufen Sie die Online-Hilfe des Systems auf, mit ALT+F4 beenden Sie die Sitzung.

➤ Die Schaltflächen (das sind die kleinen Symbole, die jedes Programm mit grafischer Oberfläche kennt) werden als Symbol wiedergegeben.

Beispiel: Um die Online-Hilfe aufzurufen, können Sie auch  anklicken.

Die Aufgaben am Ende der Kapitel 3-10 werden durch den stilisierten Bleistift am Rand symbolisiert, der den Wunsch an den Leser ausdrücken soll, zunächst diese Aufgabenstellungen zu bearbeiten, bevor mit dem nächsten Kapitel fortgefahren wird.

Wichtige Aspekte innerhalb des Textes, Erklärungen, die besonders herausgestellt werden sollen, Tips, die dem Anfänger helfen können und Antworten auf zuvor aufgeworfene Fragen werden am Rand durch das Smiley-Zeichen (das lachende Gesicht) gekennzeichnet.

In den Beispielprogrammen, angezeigt durch die laufende Person, werden die festen SAS-Sprachelemente groß geschrieben, die vom Anwender frei wählbaren Namen klein. Die großgeschriebenen Schlüsselworte, Optionen und Prozedurnamen müssen ebenso wie die Satzzeichen in genau der gleichen Art und Weise übernommen werden. Die kleingeschriebenen Namen für Dateien und Variablen können beliebig ausgetauscht werden.

```
PROC CONTENTS DATA=biblio.adressen POSITION;
RUN;
```

In diesem Beispiel wird die Prozedur CONTENTS aufgerufen. Die Worte `PROC CONTENTS` leiten den Prozedurschritt ein und sind unbedingt erforderlich; ebenso die beiden Optionen `DATA=` und `POSITION`, wenn, wie z. B. im Abschnitt 5.5.2, die Variablen in alphabetischer Reihenfolge aufgeführt werden sollen. Die Datei `biblio.adressen` kann durch jede andere vorhandene SAS-Datei ersetzt werden.

Das geschwungene Pfeil-Symbol kennzeichnet die allgemeinen Syntaxbeschreibungen einzelner Anweisungen oder ganzer Prozeduren. Wie in den Beispielprogrammen werden die festen SAS-Sprachelemente großgeschrieben, die vom Anwender frei wählbaren klein. Im Unterschied zu konkreten Variablen- und Dateinamen werden allgemeine Bezeichnungen verwendet, wie beispielsweise `datei`. Optionen und Anweisungen, die nicht unbedingt für den jeweiligen Programmschritt erforderlich sind, werden in spitze Klammern (<>) eingeschlossen. Können mehrere Optionen alternativ verwendet werden, werden diese durch einen senkrechten Strich (|) getrennt.

```
PROC FREQ <DATA=datei> <ORDER=DATA|FORMATTED|FREQ>;
     TABLES variablen-liste </ <NOCOL> <NOROW> <NOPERCENT>
                      <EXPECTED> <DEVIATION> <CELLCHI2>
                      <CHISQ> <EXACT>>;
```

Die Anweisungen `PROC FREQ;` und `TABLES ... ;` sind erforderlich. Die Optionen `DATA=`, `ORDER=`, `NOCOL` etc. sind optional, weshalb sie in spitze Klammern eingeschlossen wurden. Fehlen diese Angaben, führt das SAS-System die Prozedur mit festgelegten Voreinstellungen aus. Der Option `DATA=` kann ein beliebiger Dateinamen folgen. Die Option `ORDER=` kann dagegen nur einen der durch | getrennten Werte `DATA`, `FORMATTED` oder `FREQ` annehmen. Die Namen der Variablen (`variablen-liste`) wird vom Anwender festgelegt. Sie enthält einen oder mehrere Variablennamen. Der Schrägstrich nach der `variablen-liste` ist ebenfalls optional: Er ist nur erforderlich, wenn eine der nachfolgenden Optionen `NOCOL`, `NOROW` etc. genannt wird.

Alle Beispielprogramme, die mit einer Diskette am Rand gekennzeichnet werden, zeigen, in Kommentarzeichen eingeschlossen, in der ersten Zeile den Namen an, unter dem sie auf der Begleitdiskette zu diesem Buch im Unterverzeichnis PROGRAMS abgelegt sind. `/*--- KA04-01.SAS ---*/` beispielsweise ist das 1. Beispielprogramm des 4. Kapitels.

Die konkreten Beispielprogramme und die Syntaxbeschreibungen werden, wie oben gezeigt, ebenso wie Wiedergaben von Protokoll- und Ausgabefenstern umrandet und in nicht-proportionaler Schreibmaschinenschrift dargestellt.

Die Ausgabefenster, in denen die Ergebnisse von SAS-Programmschritten angezeigt werden, Tabellen, Listen und einfache Grafiken, werden zusätzlich noch durch nebenstehendes Randsymbol gekennzeichnet.

Fehler im Programmaufruf und Rückmeldungen zu ausgeführten Daten- und Prozedurschritten und der benötigten Zeit erscheinen im Protokollfenster. Die Symbole, mit denen die drei Basisfenster (Programm-, Ausgabe- und Protokollfenster) gekennzeichnet sind, wurden entsprechend einer Konvention des SAS-Systems gewählt: Sie erscheinen in der Dateiauswahlsicht, wenn man mit SAS 6.12 unter Windows 95 arbeitet.

Zu dem Buch gehört eine Begleitdiskette, die alle mit dem Diskettensymbol gekennzeichneten Beispielprogramme enthält. Auf ihr finden Sie auch die verwendeten Beispieldaten sowie weitere Dateien, die zur Bearbeitung der Aufgaben notwendig sind. Eine ausführliche Beschreibung des Inhalts der Diskette befindet sich im Anhang B.

Die verwendeten SAS-Programme wurden alle mit SAS 6.12 unter Windows 3.1 auf einem Compaq Contura 410C mit 8 MB Hauptspeicher getestet. Die SAS-Software war nicht lokal installiert, sondern wurde von einem externen CD-ROM Laufwerk gestartet. Die präsentierten Meldungen und Resultate beruhen auf den originalen Protokoll- und Ausgabefensterinhalten. An einigen Stellen wurden lediglich für die Darstellung in diesem Buch Leerzeilen oder -spalten entfernt.

2

Spielen mit Legos – Der modulare Aufbau des SAS-Systems

Das SAS-System, so wie es heute vorliegt, entstand im Laufe von 25 Jahren. Zu Anfang konnten Daten organisiert und analysiert werden. Nach und nach kamen neue Aufgabenbereiche hinzu. Heute setzt es sich aus mehr als 20 einzelnen Komponenten zusammen, den SAS-*Modulen*. Jedes Modul erfüllt eigene, zu einem Themenkomplex oder Anwendungsgebiet gehörende, von den anderen Komponenten unabhängige Aufgaben. Die Module setzen sich selbst wiederum aus einer oder mehreren *Prozeduren* zusammen. In Abbildung 2.1 wird der Aufbau des Systems anhand einiger ausgewählter Module und Prozeduren veranschaulicht. Aus dem Namen geht meistens die Hauptaufgabe des Moduls oder der Prozedur hervor. So berechnet die Prozedur TTEST einen t-Test zum Vergleich zweier unabhängiger Gruppen und die Prozedur SORT sortiert eine Datei. Welche Gruppen verglichen oder welche Datei nach welchem Kriterium sortiert werden soll, muß der Anwender dem System noch über *Optionen* mitteilen.

Eines der „ältesten“ Module ist SAS/BASE, das Basismodul, das die wichtigsten Funktionalitäten zum Anlegen und Verwalten von Dateien beinhaltet. Es stellt die Basis dar und ist Voraussetzung für alle anderen Module. In ihm findet man etwa die Prozeduren PRINT und SORT zum Ausgeben und Sortieren von Daten sowie MEANS für die Berechnung von statistischen Maßzahlen, die im Zusammenhang mit dem Mittelwert von Bedeutung sind. SAS/STAT, ebenfalls ein Modul der ersten Stunde, bündelt die statistischen Prozeduren. Diese gestatten die Berechnung des t-Tests und des χ^2-Tests und reichen bis hin zu komplexen multivariaten Verfahren wie multipler Regression, allgemeine lineare Modelle und Diskriminanz- und Faktoranalysen. Mit dem neueren

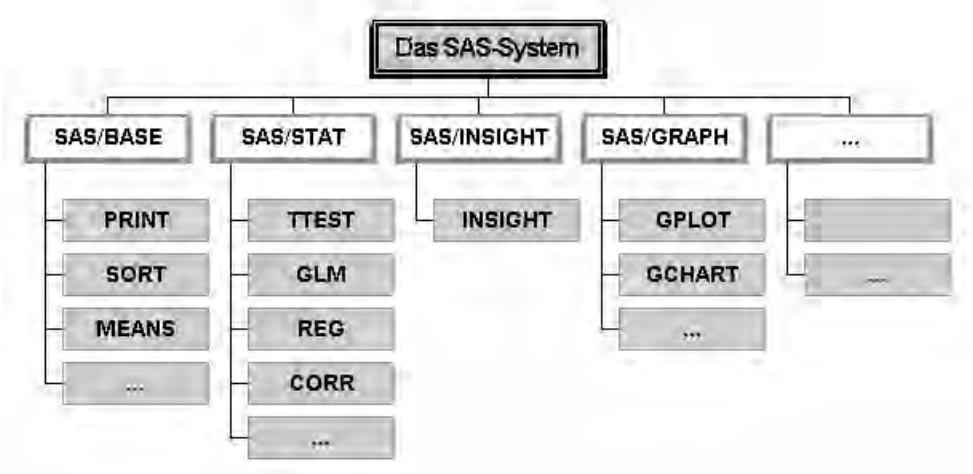


Abb. 2.1: Aufbau des SAS-*Systems mit Modulen und Prozeduren (Ausschnitt)*

Modul SAS/INSIGHT und der einzigen darin enthaltenen Prozedur INSIGHT wird die explorative Datenanalyse nach Hoaglin et.al. [4] unterstützt, indem dieselben Daten auf unterschiedliche Arten gleichzeitig dargestellt und Markierungen oder Ein- und Ausschlüsse interaktiv von einer zur anderen Darstellung übertragen werden können. Dieses Modul ist relativ neu, da es nur auf Rechenanlagen eingesetzt werden kann, bei denen der Anwender interaktiv arbeiten kann. SAS/GRAPH schließt die Grafikprozeduren zur Erzeugung von Streu- und Balkendiagrammen (GPLOT, GCHART) ein und erlaubt die Organisation und Verwaltung der in einem eigenen Format abgelegten SAS-Grafiken (GREPLAY). In diesem Modul sind zahlreiche Landkartenkoordinatendateien integriert, die die Darstellung regional-verteilter Informationen ermöglichen. Mit dem Modul SAS/FSP wird dem Anwender eine bequeme Möglichkeit zur menügesteuerten Dateneingabe über die beiden Prozeduren FSEDIT und FSVIEW zur Verfügung gestellt. Die Prozedur FSLETTER unterstützt darüber hinaus die Erstellung von Formbriefen und Formularen. An den Anfänger wendet sich der Assistent (Modul SAS/ASSIST), der eine menügesteuerte Bedienung des SAS-Systems gestattet. Die dabei durchgeführten Schritte werden protokolliert und können zum späteren Gebrauch in Form von Programmen abgespeichert werden. Ein versierter SAS-Anwender kann aus dem Assistenten ebenfalls Nutzen ziehen, indem er ihm unbekannte Prozeduren des SAS-Systems zunächst über das Menü kennenlernt und anschließend die protokollierten Programmschritte genauer untersucht. Die Analyse von zeitabhängigen Variablen wird von dem Modul SAS/ETS unterstützt. Hier finden sich u. a. Prozeduren zur Modellierung von Zeitreihen wie ARIMA und X11. Mit den beiden Modulen SAS/AF und

SAS/EIS schließlich kann ein SAS-Programmierer eigene bildschirmorientierte Masken und Programme erstellen, die es dem Anwender gestatten, komplexe Analyse- und Datenverarbeitungsschritte ohne eigene SAS-Kenntnisse durchzuführen.

Der Anwender stellt sich nach seinem Anforderungsprofil ein individuelles SAS-System zusammen, d. h., er bestimmt die Module, die er erwerben (korrekt: lizensieren) möchte. Die Lizenzgebühren sind jährlich fällig und richten sich nach der Anzahl der Module, der Anzahl der Anwender und der Maschine, auf der die Software eingesetzt wird. Während an Universitäten aufgrund günstigerer Konditionen das SAS-System meist mit vielen Modulen genutzt werden kann, werden in der freien Wirtschaft nur jene Module erworben, die sich wirtschaftlich rechnen. Als Anwender finden Sie daher vor Ort nicht immer alle hier beschriebenen Komponenten des SAS-Systems vor. Wie Sie in Erfahrung bringen, welche Module bei Ihnen lizensiert sind, erfahren Sie im Anhang A.

2.1 Die Benutzeroberfläche

Das SAS-System kann sich dem Anwender auf zwei recht unterschiedliche Arten präsentieren: zum einen mit einer grafischen Oberfläche, dem *Display Manager System*, bei dem der Anwender direkt über Fenster und Menüleisten mit dem System in Kontakt treten kann; zum anderen als Software, die im Hintergrund eines Rechners verborgen bleibt, und an die Programme übergeben werden und die Protokoll- und Ausgabedateien erzeugt. *Batchmodus* heißt die zweite Variante in der Fachsprache. Beide Arbeitsweisen haben ihre Vor- und Nachteile. Da sich dieses Buch in erster Linie an den Anfänger wendet, der den Umgang mit SAS lernen und an kleinen Beispielen üben will, wird im folgenden nur die erste Variante behandelt. Verfügt der Anwender über ausreichende Erfahrungen in der Erstellung von SAS-Programmen, kann er jederzeit ohne weitere Hilfestellung auf die Verarbeitung im Batchmodus umstellen.

Um SAS unter Windows zu starten, klickt man nebenstehende Schaltfläche im Windows Programm-Manager an oder ruft die im SAS-Hauptverzeichnis stehende, ausführbare Datei SAS.EXE auf.

Das System wird gestartet und präsentiert das Display Manager System (kurz DMS), die typische SAS-Umgebung (Abbildung 2.2).

❶ Das Display Manager System besteht zunächst aus einem großen Fenster, das mit *SAS* überschrieben ist und alle anderen Elemente umfaßt.

❷ Unter der Titelzeile erscheint eine *Menüleiste* mit den unter Windows üblichen Menüs *File, Edit, Help* ...

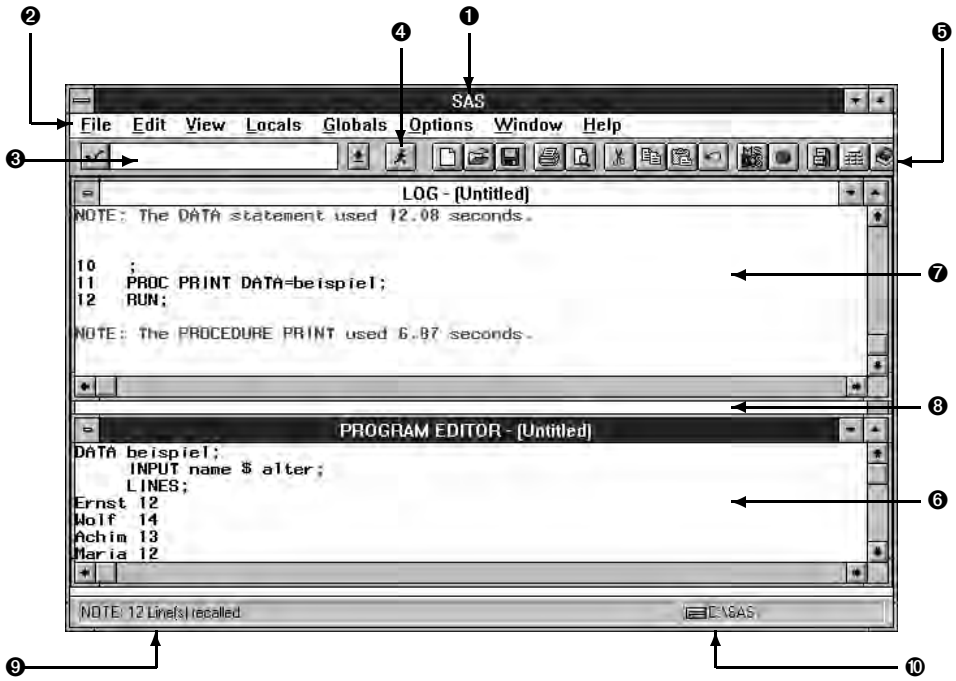


Abb. 2.2: Das Display Manager System

❸ Direkt unterhalb der Menüleiste findet man am linken Rand das *Kommandofenster*, in das Befehle (oder Kommandos) zur Ausführung von Aktionen eingetragen werden können. Wenn Sie den kleinen Pfeil rechts neben der Kommandozeile anklicken, werden die zuletzt ausgeführten Kommandos aufgelistet.

❹ Rechts neben dem Kommandofenster befinden sich mehrere grafische Symbole, Schaltflächen genannt. Das erste von links ist das *Submit*-Symbol . Klickt man es an, wird das Programm, das sich im Programmfenster (siehe ❻) befindet, ausgeführt, d. h., das Programm wird an das SAS-System übergeben und verarbeitet. Die gleiche Aktion läuft ab, wenn man stattdessen das Kommando `SUBmit` in das Kommandofenster einträgt und die Enter- (oder Return-) Taste drückt.

❺ Nach dem Pfeil- und dem Submit-Symbol folgt eine Reihe weiterer Schaltflächen: die *Tools-Leiste*. Hinter jedem Symbol verbirgt sich eine mehr oder

weniger häufig benötigte Aktion, beispielsweise Öffnen und Speichern, Drucken, Ausschneiden, Kopieren und Einfügen (engl. *Cut and Paste*), die letzte Aktion rückgängig machen (*Undo*) und der Aufruf der Online-Hilfe. Das SAS-System gibt eine Standardanordnung der Schaltflächen auf der Tools-Leiste vor. Der Anwender kann sie jedoch nach eigenen Wünschen und Bedürfnissen gestalten (siehe Abschnitt 10.4).

Bewegt man den Cursor über die einzelnen Symbole, öffnet sich ein kleines *Tool-Tips Fenster*, das angibt, welche Aktion bzw. welches Kommando sich hinter dem Symbol verbirgt. Unter der Submit-Schaltfläche erscheint etwa *Submit*.

❻ Innerhalb des großen „*SAS*“-Fensters gibt es drei weitere Fenster, das Programm-, das Protokoll- und das Ausgabefenster. Im unteren Bildschirmbereich sehen Sie den *PROGRAM EDITOR* oder das *Programmfenster*. Hierin schreiben Sie Ihre Programme, die mit (oder dem Kommando `SUBmit`) ausgeführt werden.

Über die Menüfolge *File* → *Open* (oder) können Programme in das Programmfenster eingelesen werden. Wie diese SAS-Programme im einzelnen aussehen, wird später in Abschnitt 3.1 ausführlich beschrieben.

Im Unterschied zu manch anderen Windows-Programmen werden über die Menüfolge *File* → *Open* keine Datendateien geöffnet und in Tabellenform aufgelistet, sondern Programme im Programmfenster geöffnet, die mit `SUBmit` ausgeführt werden können.

Analog dazu wird mit *File* → *Save* der Inhalt des Programmfensters abgespeichert (oder der Inhalt des gerade aktiven Fensters), nicht die erzeugte SAS-Datei.

Das Programmfenster kann am rechten Rand zusätzlich noch Zeilennummern aufweisen, die dem Anwender die Verwendung der *Zeilenkommandos* (vgl. dazu Abschnitt 4.2) gestatten.

❼ Oberhalb des Programmfensters befindet sich das *Protokollfenster* (*LOG*), in dem, wie der Name andeutet, Systemmeldungen angezeigt und Protokolle über ausgeführte Programme eingetragen werden.

Spezielle Dialogfenster zur Anzeige von Programmfehlern, wie sie bei vielen Windows-Programmen üblich sind, werden vom SAS-System nicht geöffnet. Fehler in Programmen werden im Protokollfenster angezeigt, fehlerhafte Kommandos in der Statuszeile (vgl. ❾).

❽ Im Hintergrund, nur durch einen schmalen Streifen zwischen Programm- und Protokollfenster auszumachen, befindet sich das *Ausgabefenster* (*OUTPUT*). Hier werden Tabellen, Analyseresultate und einfache Grafiken ausgegeben. Das Ausgabefenster rückt in den Vordergrund, wenn etwas Neues

darin ausgegeben wurde. Erzeugt ein Programm keine Ausgabe, verbleibt es im Hintergrund.

❾ Am unteren Ende des *sas*-Fensters erscheint links die *Statuszeile* (*Message line*), in der Systemmeldungen zu ausgeführten Kommandos angezeigt werden.

Bevor Abbildung 2.2 erzeugt werden konnte, wurde zunächst ein aus 12 Zeilen bestehendes Programm ausgeführt. Die entsprechenden Meldungen stehen im Protokollfenster. Anschließend wurde dieses Programm wieder mit dem Kommando `RECall` ins Programmfenster zurückgeholt. In der Statuszeile erscheint die entsprechende Meldung:
`NOTE: 12 Line(s) recalled.`

❿ Rechts unten wird das aktuelle *Verzeichnis* angezeigt. Dieses Verzeichnis ist das Standardverzeichnis für die Aktionen Öffnen und Speichern von Programmen. Im vorliegenden Beispiel ist dies das Unterverzeichnis SAS auf Laufwerk C:. Soll ein Programm aus einem anderen Verzeichnis im Programmfenster geöffnet werden, muß dieses mit komplettem Pfad angegeben werden. Durch einen Doppelklick auf dieses Feld kann ein anderes Verzeichnis zum aktuellen erklärt werden.

Das Display Manager System sieht bis auf individuelle Abweichungen unabhängig von Betriebssystem und installierten Modulen im Prinzip gleich aus. Weder an der Oberfläche noch an den Menüs ist etwas vom modularen Aufbau des SAS-Systems erkennbar. Dem Anwender wird eine einheitliche Benutzeroberfläche präsentiert.

Dadurch kann es passieren, daß man versehentlich einen Menüpunkt auswählt, der ein nicht lizenziertes Modul anspricht. Diese Aktion führt nicht zu einer Katastrophe, sondern zu einer entsprechenden Fehlermeldung im Protokollfenster.

Wurde etwa die Menüfolge *Globals* → *Present* → *SAS/GIS* ausgewählt, das Geografische Informationssystem innerhalb des SAS-Systems, erscheint folgende Fehlermeldung im Protokollfenster:

```
ERROR: The SAS/GIS product with which GIS is associated is
       either not licensed for your system or the product
       license has expired. Please contact your SAS instal-
       lation representative.
```

Bevor in den nächsten Kapiteln der Aufbau von SAS-Programmen und die Aktionen, die neben `SUBmit` und `RECall` noch über das Display Manager System ausgeführt werden können, vorgestellt werden, wird zunächst die Online-Hilfe erläutert, die über den Menüpunkt *Help* aktiviert wird.

2.2 Die Online-Hilfe

„Ein eigener Abschnitt zu diesem Thema?“ werden Sie jetzt vielleicht denken. Aber gerade der Anfänger, der noch nicht sehr viele Erfahrungen mit dem SAS-System hat, sieht zunächst den Wald vor lauter Bäumen nicht. Er irrt von einer zur anderen Seite der Online-Hilfe und findet letztendlich doch nicht, wonach er gesucht hat. Um gezielte Informationen zu einzelnen Prozeduren oder Kommandos aufzufinden, muß man den prinzipiellen Aufbau dieser Online-Hilfe verstanden haben. Wenn Sie sich den Aufbau verinnerlicht haben, können Sie während Ihrer Arbeit mit dem SAS-System die Online-Hilfe als Ersatz für die Handbücher verwenden. Sie finden die komplette Beschreibung einzelner Prozeduren, wenn Sie etwa unsicher sind, wie die exakte Formulierung (Syntax) eines Programms auszusehen hat, aber auch eine Liste aller Formate, die Ihnen z. B. für die Darstellung von Datumsangaben zur Verfügung stehen. Daneben gibt es allgemeinere Beschreibungen, die teilweise tutoriellen Charakter haben und einen Überblick über das System geben, z. B. was beim Drucken zu beachten ist und vieles mehr. Die Online-Hilfe enthält eine Fülle von Informationen, die für das Arbeiten mit dem SAS-System notwendig und hilfreich sind. Aber man muß wissen, wie man zu den Informationen kommt.

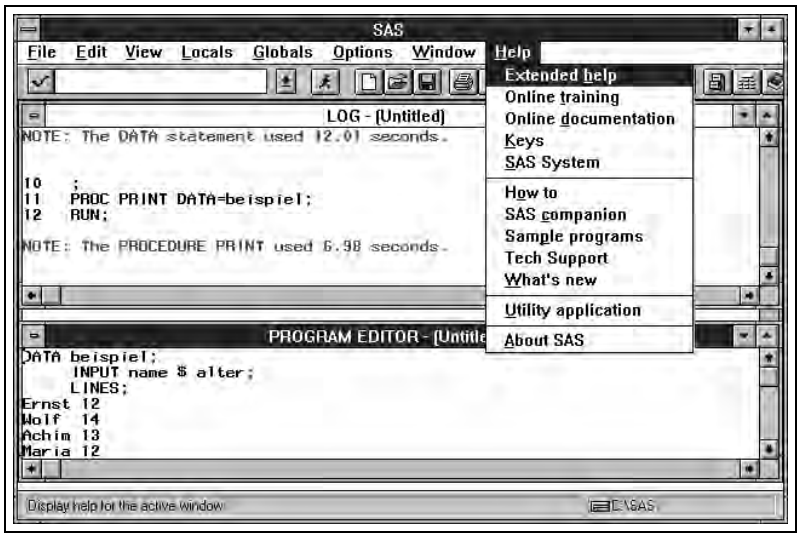


Abb. 2.3: Das Help-Menü

Die Online-Hilfe wird aktiviert, indem man den Menüpunkt *Help* auswählt. Es öffnet sich ein Pop-up Menü mit mehreren Unterpunkten (vgl. Abbildung 2.3), die in vier Gruppen unterteilt sind:

- Erläuterungen zum gesamten SAS-System, seinen Modulen und Prozeduren
- Neuerungen und Betriebssystem-spezifische Hinweise zur aktuellen Version
- Hilfsmittel
- Technische Informationen zur aktuellen Version

Abb. 2.4: Extended Help

Wählt man einen der beiden Menüpunkte *Help* → *Extended Help* oder *Help* → *SAS System*, öffnet sich die Online-Hilfe in Form einer Buchsammlung (Abbildung 2.4), wie es für die Windows-Hilfen allgemein üblich ist. (Sie können diesen Menüunterpunkt auch direkt aufrufen, indem Sie die Schaltfläche benutzen.) Einzelne Bücher können durch Anklicken oder *Open* geöffnet werden, Kapitel in Büchern mittels Anklicken bzw. *Display*. Klickt man ein Buch oder Kapitel erneut an oder klickt man auf *Close*, wird das Buch oder Kapitel wieder geschlossen. Mit *Print* druckt man den Inhalt aus, mit *Cancel* verläßt man die Online-Hilfe. Mittels *Index* kann man nach festvorgegebenen Stichworten suchen, was

praktisch ist, wenn man nicht weiß, in welchem Buch das gewünschte Thema behandelt wird. Mit *Find* dagegen wird zunächst eine Stichwortliste aus allen vorhandenen Texten aufgestellt (was recht zeitaufwendig sein kann), bevor man aktiv nach einem Wort suchen kann.

Neben dem Buch *SAS System Help: Main System* gibt es Informationen zu den Neuerungen der Version 6.12 sowie Hinweise zu den einzelnen Modulen (SAS/AF, SAS/ACCESS usw.) und mit *How to ...* und *SAS Companion ...* zwei Bücher, die bereits im Help-Menü angezeigt (vgl. Abb. 2.3) und weiter unten beschrieben werden.

Das Buch *SAS System Help: Main System* ist für den Anfänger zweifelsfrei das wichtigste. In Abbildung 2.4 sehen Sie das aufgeschlagene Buch und Überschriften wie *Data Management* und *Report Writing*. Unter *Modelling and Analysis Tools* finden Sie alles, was im entferntesten mit Auswertung und Modellformulierung zu tun hat, also alle Module und Prozeduren, die zur Statistik gehören. Unter *Graphics* sind die Prozeduren des Grafikmoduls SAS/GRAPH, aber auch andere Grafik-produzierende Module und Prozeduren aufgeführt. In den Texten wird die Bedeutung und das Einsatzgebiet der jeweiligen Prozeduren erläutert sowie deren Gebrauch (Syntax) aufgeführt. Hier kann der Anwender nachschlagen, wozu und wie bestimmte Anweisungen und Optionen eingesetzt werden. Unter dem Stichwort *Limited Index* finden Sie nicht das, was man normalerweise unter einem Index versteht (ein in alphabetischer Reihenfolge angeordnetes Stichwortverzeichnis), sondern eine nach Modulen angeordnete Zusammenstellung der Prozeduren, aus welcher deren Gebrauch abgelesen werden kann. Wenn Sie also genau wissen, zu welchem Modul die Prozedur gehört, deren Syntax Sie verwenden müssen, können Sie über diesen Menüpunkt direkt zum Ziel gelangen. Unter *Host Information* sind das Betriebssystem betreffende spezifische Hinweise abgelegt, die u. U. nicht für SAS-Versionen auf anderen Plattformen gelten. Mit *Help for SAS Products*, dem letzten Punkt des Hauptmenüs, der hier vorgestellt werden soll, informieren Sie sich über die Einsatzmöglichkeiten der einzelnen Module.

Wenn die Module SAS/CBT1 - SAS/CBT6 lizensiert sind, kann über *Help* → *Online Training* das Lernprogramm (engl. *Computer Based Tutorial*) aufgerufen werden, das sich aus folgenden sechs (CBT-) Teilen zusammensetzt:

- Fundamentals of the SAS System (101)
- Reading Raw Data and Formatting Values with the DATA Step (102)
- Creating, Modifying, and Processing Variables with the DATA Step (103)
- Developing Custom Data Entry Applications (104)
- Creating and Enhancing SAS/GRAPH Output (105)
- Creating Tables with PROC TABULATE (106)

Mit diesem in englischer Sprache abgefaßten Lernprogramm lernt man auf anschauliche Weise das SAS-System kennen, Daten zu erfassen und zu verändern und sie in Form von Grafiken und Tabellen zu präsentieren. Eingestreute praktische Übungen erleichtern das Durcharbeiten der in kurze Lektionen untergliederten Kurse und bieten jederzeit die Möglichkeit, das Erlernte sofort in die Praxis umzusetzen. Ein kurzer Test am Ende jeder Lektion dient der Überprüfung des eigenen Wissensstands. Die Statistikprozeduren werden im Lernprogramm leider nicht behandelt.

Unter *Help* → *Online Documentation* werden Beschreibungen aufgeführt, die dem Anwender während der SAS-Sitzung online zur Verfügung stehen. Die Liste ist, bis auf die Neuerungen der aktuellen Version, nach Modulen sortiert, so daß man sich leicht zurechtfindet, sobald man einen gewissen Überblick über das System hat.

Help → *Keys* zeigt die aktuelle Belegung der Funktions- und anderer Tastenkombinationen an. Diese Tasten können mit Kommandos belegt werden, die zur Arbeit mit dem Display Manager System benötigt werden. Weiter oben wurden die beiden Kommandos `SUBmit` und `RECall` eingeführt, die gewöhnlich auf den Tasten F3 (bzw. F8) und F4 liegen. In Kapitel 4 werden weitere Kommandos, die auf den übrigen Funktionstasten liegen, vorgestellt und ihre Bedeutung erklärt sowie gezeigt, wie sich die Voreinstellung der Belegung ändern läßt (vgl. auch Abschnitt 10.3).

Unter *Help* → *How to ...* finden Sie Antworten auf häufige Fragen zu den Stichpunkten Arbeiten mit dem SAS-System (Aufruf, Ausführen von Programmen, Drucken), Verbindung zu anderen Anwendungen (Austausch von Daten, Aufruf dieser Programme innerhalb der SAS-Umgebung), Zugriff auf SAS- und Fremddateien, Einstellung der Umgebung und spezielle Möglichkeiten für Programmierer (SAS/AF und SAS/EIS). Diese Seiten sind anwendungsbezogen und orientieren sich an den in der Praxis relevanten Problemen.

Unter *Help* → *SAS Companion* werden Betriebssystem-spezifische Aspekte behandelt. Für die diesem Kurs zugrundeliegende Version SAS 6.12 unter Windows sind dies Informationen über die Anzeige von World Wide Web-Seiten, das Verschicken elektronischer Post innerhalb der SAS-Sitzung sowie die Beschreibung von Modulen, Prozeduren und Funktionen, die nur unter Windows verfügbar sind. Diese Seiten kann man zu Rate ziehen, wenn man bislang unter einem anderem Betriebssystem mit der SAS-Software gearbeitet hat oder wenn Probleme auftreten, die mit der Arbeitsumgebung in engem Zusammenhang stehen.

Mit jedem Modul werden in der Regel auch verschiedene Beispielprogramme installiert, auf die über die Online-Hilfe und *Help* → *Sample Programs* zugegriffen werden kann. Zum Teil sind sie auch in einem eigenen Unterverzeichnis (vgl. Anhang A) innerhalb des SAS-Systems abgelegt. Neben einigen der in den

Handbüchern abgedruckten Programme findet man hier zusätzliche Beispiele für den Einsatz der Prozeduren. Eine nach Modulen sortierte Liste erleichtert den Zugriff auf die Beispielprogrammme, und eine Suchroutine (*Search*) erlaubt die Suche nach bestimmten Stichworten. Die Ausführung dieser Beispielprogramme wird zu Beginn des Hilfetextes detailliert beschrieben.

Unter dem Menüpunkt *Help* → *Tech Support* findet man neben den für die aktuelle Version notwendigen Systemanforderungen und den Installationsunterlagen die http-Adresse des SAS Institute WWW-Servers in North Carolina, auf dessen Seiten der Anwender bei auftretenden Problemen selbst nach Lösungsmöglichkeiten suchen kann, bevor er sich telefonisch an den Technical Support von SAS Institute wendet.

Mit *Help* → *What's new* erhalten Sie einen Überblick über die Neuerungen der vorliegenden gegenüber der Vorgängerversion. In SAS 6.12 sind dies u. a. der Import/Export Wizard, der SAS Desktop, das Viewtable Window und die Web Tools. Es wird jeweils eine kurze Beschreibung, um was es sich bei dem Stichwort handelt, und ein Verweis zu ausführlicheren Informationen gegeben.

Die beiden letzten, aus jeweils einem Menüpunkt bestehenden Gruppen sind für den Anwender, der gleichzeitig auch Lizenznehmer und für die Installation zuständig ist, von Bedeutung. *Help* → *About SAS* liefert Detailinformationen zur installierten SAS-Version, wie Release-Nummer und Stand des Technical Support-Levels (TS), Name des Lizenznehmers und Informationen zum Betriebssystem. (z. B. die DOS-, die Windows- und die Win32S-Version). Diese Angaben werden benötigt, wenn wegen Problemen mit der Technical Support-Abteilung von SAS Institute Kontakt aufgenommen werden muß.

Mit der *Help* → *Utility application* können Schriftarten für SAS/GRAPH eingerichtet und Landkartendateien komprimiert bzw. dekomprimiert werden, was manchmal aus Platzgründen notwendig werden kann. Als Anwender ohne Schreibrechte in dem SAS-Systemverzeichnis können Sie diese Aktionen jedoch nicht ausführen.

Nähern Sie sich als Anfänger der Online-Hilfe zunächst über den Menüpunkt *Help* → *Extended Help* → *SAS System Help: Main Menu*. Hier ist die komplette Syntax unterteilt nach Sachgebieten (Datenschritt, Basisprozeduren, Grafik, Modellierung und Auswertung) abgelegt. Später wird *Help* → *How to* .. für eine übergreifende Darstellung grundlegender Aspekte an Bedeutung gewinnen. Den erfahreneren Anwender interessieren dagegen in erster Linie die Neuerungen der aktuellen Version (*Help* → *What's New in Release 6.12*) und den Umsteiger von einer anderen SAS-Version die Betriebssystem-spezifischen Hinweise (*Help* → *SAS Companion*).

Neben der oben erwähnten Online-Dokumentation gibt es natürlich auch „richtige“ Handbücher auf Papier: einführende, die einen Überblick geben, und nach

Modulen ausgerichtete. Bei den Modul-orientierten Handbüchern unterscheidet man zwischen dem *Usage Guide*, der aus Anwendersicht die Möglichkeiten der Prozeduren beschreibt und dem *Reference Guide*, in dem die Prozeduren alphabetisch angeordnet beschrieben werden: Neben einer kurzen Einleitung wird das Einsatzgebiet der Prozedur erläutert, die Syntax aufgeführt, Hintergrundinformationen und weiterführende Literaturhinweise gegeben sowie anhand von Beispielen der Gebrauch und die Deutung der Ergebnisse erläutert.

Zum Basismodul SAS/BASE gehört der Language Reference Guide [9], in dem das Display Manager System, die Kommandos sowie die Funktionen und Formate beschrieben werden, und der Procedures Guide [11], der die Basisprozeduren enthält. Die anderen Modul-orientierten Handbüchern werden entsprechend dem Modul bezeichnet, etwa SAS/GRAPH Software Usage Guide [13].

Die Änderungen der neuen Releases werden in Technical Reports dokumentiert, erst mit der Version 7 sind komplett neue Handbücher angekündigt. Bei SAS Institute erhält man ein Publikationsverzeichnis aller verfügbaren aktuellen Handbücher und anderer Literatur zur SAS-Software.

3

Der große Unterschied – SAS-Programme und SAS-Dateien

Im vorherigen Kapitel wurde im Zusammenhang mit der Benutzeroberfläche die Tatsache betont, daß über die Menüfolge *File* → *Open* keine Datendatei, sondern ein Programm im Programmfenster geöffnet wird. Dieses Programm wird an das SAS-System übergeben und von diesem ausgeführt. Seine einzelnen Bestandteile werden im Protokollfenster wiedergegeben und sein Ergebnis, sofern es sichtbar ist, im Ausgabefenster präsentiert.

Die für das Datenmanagement erforderlichen Datensammlungen werden in Form von SAS-Dateien gespeichert[1], die mit Hilfe von SAS-Programme erzeugt werden können und auf Betriebssystemebene durch die Endung *.SD2 auf dem PC bzw. *.SSD01 unter UNIX erkennbar sind. Bevor die zur Erzeugung von SAS-Dateien notwendigen Prozedurschritte im Detail in Kapitel 5 erläutert werden, sollen die beiden wichtigen Begriffe SAS-Programm und SAS-Datei ausführlicher beschrieben und an einem ersten einfachen Beispiel die Arbeitsweise des SAS-Systems demonstriert werden.

[1] Die Vorsilbe SAS wird weggelassen, wenn keine Verwechslungen zwischen SAS-Programmen, SAS-Dateien und externen Dateien auftreten können.

3.1 Programme

Die SAS-Programme, häufig erkennbar an der Endung *.SAS, werden direkt in das Programmfenster geschrieben oder, wenn sie bereits existieren, über die Menüfolge *File* → *Open* im Programmfenster geöffnet. Während ihre Länge nahezu unbegrenzt ist, dürfen die einzelnen Zeilen höchstens 132 Zeichen breit sein, was jedoch keine Einschränkung sein sollte. Denn auf dem Bildschirm sehen Sie sowieso höchstens ungefähr 70 Zeichen, und ständig nach rechts blättern zu müssen, ist recht umständlich.

Selbst wenn ein SAS-Programm aus mehreren hundert oder gar tausend Programmzeilen besteht, können Sie einen strikten Aufbau aus drei Grundelementen erkennen: *Globale Anweisungen*, *Daten-* und *Prozedurschritte.*

Mit Hilfe der globalen Anweisungen legt der Anwender fest, auf welchem Ausgabegerät die Grafik gezeigt und welcher Titel verwendet werden soll sowie welche Farben zur Verfügung stehen, um nur ein paar Beispiele aufzuzählen.

In den Datenschritten (*DATA step*, vgl. Abbildung 3.1) werden SAS-Dateien neu angelegt, bestehende Dateien durch neue Merkmale ergänzt, Merkmale modifiziert, Untermengen gebildet oder mehrere Dateien zusammengefügt. Die Datenwerte, aus denen die Dateien aufgebaut werden, können entweder neu eingegeben werden oder man greift auf Werte zu, die bereits an anderer Stelle im Rechner abgelegt sind.

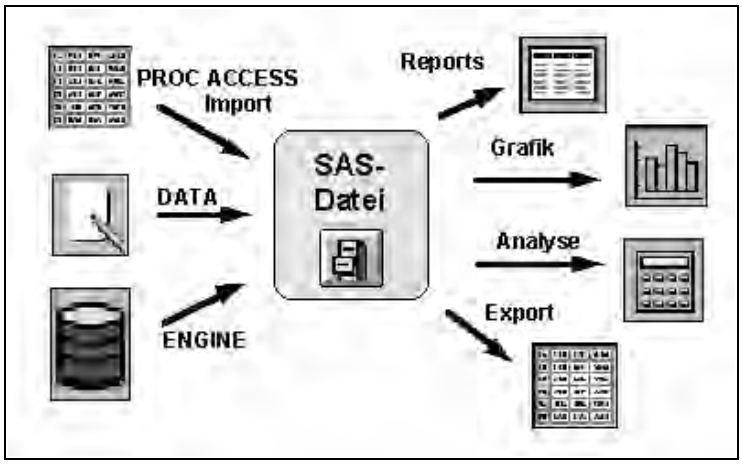


Abb. 3.1: Das zentrale Element des Systems: Die SAS-*Datei*

In den Prozedurschritten (*PROC step*) werden SAS-Dateien verarbeitet, Analysen durchgeführt und Tabellen oder Grafiken erzeugt. Manche Prozedurschritte

erlauben auch, daß die Analyseresultate in neue SAS-Dateien eingetragen sowie bestehende Dateien verändert werden.

In einem SAS-Programm erkennen Sie die Datenschritte an dem Wort DATA, das meist zu Beginn einer neuen Zeile steht. Der Prozedurschritt beginnt mit dem Wort PROC. Die RUN-Anweisung signalisiert das Ende dieser Programmschritte. Alle Anweisungen außerhalb von Daten- und Prozedurschritten[2] sind globale Anweisungen, von denen hier beispielhaft nur OPTIONS zur Festlegung von Systemeinstellungen sowie TITLE zur Definition einer Titelzeile erwähnt werden. Die Programme und Programmschritte können mit (/* ... */) kommentiert werden.

Folgende Struktur sollten Sie somit in einen SAS-Programm erkennen können:

```
OPTIONS ...; /* Globale Einstellungen          */
TITLE ...  ; /* Titelzeile                     */
...          /* Weitere globale Anweisungen    */

DATA ... ;   /* Datenschritt                   */
     ...
RUN;

PROC ... ;   /* Prozedurschritt                */
     ...
RUN;

...  ...;    /* Weitere Daten- und/oder        */
     ...     /* Prozedurschritte               */
RUN;
```

Ein SAS-Programm kann aus mehreren Daten- und Prozedurschritten bestehen, deren Reihenfolge insoweit eine Rolle spielt, daß das SAS-System diese linear von oben nach unten abarbeitet. Wenn Sie für einen Prozedurschritt eine Datei benötigen, die im Programm erst weiter unten erzeugt wird, kann der Prozedurschritt nicht fehlerfrei ausgeführt werden. Die Datei muß angelegt werden, bevor sie analysiert werden kann. Außerdem werden Daten- und Prozedurschritte zunächst beendet, bevor mit dem nächsten Programmschritt begonnen wird, denn Parallelverarbeitung ist nicht möglich. Sie können beispielsweise innerhalb eines Datenschritts keine Prozedur aufrufen.

Über die globalen Anweisungen werden Systemeinstellungen festgelegt, die nachfolgende Daten- und Prozedurschritte beeinflussen. Sie müssen daher vor

[2] Globale Anweisungen können jedoch auch innerhalb von Daten- und Prozedurschritten gesetzt werden (vgl. Kapitel 10).

dem Ende des jeweiligen Daten- oder Prozedurschritts erscheinen, für den sie erstmals gelten sollen.

Die Daten- und Prozedurschritte eines SAS-Programms bestehen gewöhnlich aus mehreren *Anweisungen* (engl. *statement*). Jede Anweisung wird durch ein Schlüsselwort eingeleitet und endet mit einem Semikolon (;). Die Anweisungen sind nicht auf eine Programmzeile beschränkt, sondern können sich über mehrere Zeilen erstrecken. Beispiele für diese Schlüsselworte sind etwa DATA, PROC PRINT, OPTIONS und TITLE.

Den Schlüsselworten folgen Datei-, Prozedur- oder Variablennamen sowie *Optionen*, die bestimmte Aktionen veranlassen oder Voreinstellungen ändern. Das folgende einfache Beispiel veranschaulicht diese Begriffe.

Die bestehende SAS-Datei `adressen` wird mit der Prozedur SORT absteigend nach dem Merkmal `name` sortiert. Folgendes SAS-Programm besteht aus einem Prozedurschritt und erledigt diese Aufgabe:

```
PROC SORT DATA=adressen;
     BY DESCENDING name;
RUN;
```

Der Prozedurschritt, gekennzeichnet durch das Schlüsselwort `PROC`, besteht aus drei Anweisungen, die mit den Schlüsselworten PROC SORT, BY und RUN eingeleitet werden.

`PROC` kennzeichnet den Beginn des Verarbeitungschritts, hier ein Prozedurschritt, und gleichzeitig den Beginn der Anweisung mit dem Schlüsselwort PROC.

`SORT` kennzeichnet die zu verwendende Prozedur, hier SORT für das Sortieren von Dateien.

Nach dem Schlüsselwort PROC muß stets ein Prozedurname genannt werden, weshalb man auch von der PROC SORT-Anweisung sprechen kann.

`DATA=` Die DATA=-Option bestimmt die zu verarbeitende Datei.

`adressen` ist der Name der zu sortierenden Datei.

`;` ist das Ende der PROC SORT-Anweisung.

`BY` Das Schlüsselwort BY leitet die Anweisung zur Bestimmung des Sortiervorgangs ein.

`DESCENDING` Diese Option legt die Sortierreihenfolge von Z nach A fest.

Ohne diese Option würde aufsteigend von A nach Z sortiert werden.

`name` ist das Merkmal (die Variable) in der Datei `adressen`, nach dem die Datei sortiert wird.

`;` bildet das Ende der BY-Anweisung.

`RUN` Das Schlüsselwort RUN leitet den Abschluß des Prozedurschritts ein. Diese Anweisung benötigt keine weiteren Optionen.

`;` beendet die Anweisung RUN und den Prozedurschritt.

Die globalen Anweisungen und Datenschritte lassen sich auf die gleiche Art in Schlüsselworte, Optionen und Namen von Dateien und Variablen zerlegen (siehe Aufgabe ❶). Wird im weiteren Verlauf dieses Kurses von der XYZ-Anweisung oder der Anweisung XYZ die Rede ist, ist damit stets die Anweisung gemeint, die mit dem Schlüsselwort XYZ beginnt und mit einem Semikolon (;) endet, auch wenn dies nicht ausdrücklich erwähnt wird.

Schreiben Sie Ihre eigenen Programme übersichtlich: Beginnen Sie mit den globalen Einstellungen, die sich nicht im Laufe des Programms ändern. Heben Sie Daten- und Prozedurschritte deutlich voneinander ab. Rücken Sie die zu einem Daten- oder Prozedurschritt gehörenden Anweisungen etwas nach rechts ein. Bevor Sie Programme anderer Anwender einsetzen, verschaffen Sie sich einen Überblick über deren Struktur.

Der Aufbau eines Prozedurschritts ist mit der Auswahl einer bestimmten Prozedur so gut wie festgelegt. Die *Prozedursyntax*, die man kennt bzw. in der Online-Hilfe oder dem Handbuch nachschlagen kann (vgl. Abschnitt 2.2), gibt Auskunft darüber, welche Anweisungen und Optionen in welcher Kombination erlaubt sind. Für den Datenschritt gilt im wesentlichen das gleiche, nur ist er weitaus vielfältiger hinsichtlich der in Frage kommenden Anweisungen und Optionen. Der Datenschritt stellt fast schon eine eigene Programmierumgebung dar. Er erlaubt die Verwendung von Programmschleifen und das bedingte Ausführen von Anweisungen. Man kann mit ihm Auswertungen realisieren, die es an anderer Stelle im SAS-System nicht gibt und von den SAS-Prozeduren unabhängige Tabellen und Listen erzeugen.

Der fortgeschrittene SAS-Anwender, der über ausreichende Programmierkenntnisse von Daten- und Prozedurschritten verfügt, kann mit Hilfe der Makrosprache globale Anweisungen, Daten- und Prozedurschritte zu einem Aufruf zusammenfassen und durch Definition geeigneter Aufrufparameter den Ablauf „seines“ Programms (=Makros) steuern. In Kapitel 11 wird dies an einem Beispiel vorgestellt.

3.2 Dateien

„SAS-Datei“ ist neben „SAS-Programm“ der zweite zentrale Begriff innerhalb des SAS-Systems. Damit die SAS-Datei `adressen` mit obigem Prozedurschritt fehlerfrei sortiert werden konnte, mußte sie zuvor in einem Datenschritt angelegt werden. In diesem Datenschritt organisiert das SAS-System die Datensammlung in der dafür erforderlichen Form. Erst wenn die SAS-Datei angelegt ist, kann mit Prozeduren auf diese zugegriffen werden.

Eine solche SAS-Datei kann man sich als tabellarische Darstellung der Eintragungen einer Kartei vorstellen. Zur Illustration wird eine kleine (fiktive) Firma im Rhein-Neckar Kreis ausgewählt. Für jeden der fünf beschäftigten Mitarbeiter werden in der Adreßkartei vier Eigenschaften notiert. Diese Datensammlung umfaßt somit 20 Werte, die sich wie folgt in einer Tabelle anordnen lassen:

Name	Vorname	Wohnort	Alter
Fritz	Petra	Heidelberg	31
Fischer	Ulrich	Heidelberg	46
Meier	Hans	Walldorf	25
Rost	Werner	Mannheim	34
Schulz	Karin	Mannheim	27

Die Informationen der einzelnen Mitarbeiter wurden in die Zeilen der Tabelle eingetragen. In den Spalten werden deren Eigenschaften notiert: `Name`, `Vorname`, `Wohnort` und `Alter`. Diese Form läßt sich nahezu identisch auf eine SAS-Datei übertragen, welche die Daten ebenfalls in Form einer rechteckigen Tabelle abspeichert.

Die Tabelle mit allen Zeilen und Spalten entspricht einer SAS-Datei (engl. *data set*). In vorliegendem Beispiel ist dies die komplette Adreßkartei der Mitarbeiter, weshalb die gesamte Tabelle in der folgenden Abbildung grau unterlegt wurde.

Name	Vorname	Wohnort	Alter
Fritz	Petra	Heidelberg	31
Fischer	Ulrich	Heidelberg	46
Meier	Hans	Walldorf	25
Rost	Werner	Mannheim	34
Schulz	Karin	Mannheim	27

Die oberste, grau-unterlegte Zeile in der Tabelle enthält die Spaltenüberschriften. In der SAS-Datei entsprechen diese den Namen der Merkmale (oder Variablen). Hier lauten diese `Name`, `Vorname` etc.

Name	Vorname	Wohnort	Alter
Fritz	Petra	Heidelberg	31
Fischer	Ulrich	Heidelberg	46
Meier	Hans	Walldorf	25
Rost	Werner	Mannheim	34
Schulz	Karin	Mannheim	27

Außer den Variablennamen ist für das SAS-System noch wichtig, von welchem Typ die Variablen sind. In der *Variablendeklarationen* wird dazu unterschieden, ob es sich um numerische, also um reine Zahlenwerte, oder um Textvariablen (auch alphanumerische oder String-Variablen genannt) handelt, die auch aus Zeichenketten mit Buchstaben bestehen können. Zusätzlich wird bestimmt, wie die Werte eingelesen bzw. ausgegeben werden sollen (Informat, Format, vgl. Abschnitt 7.5.2).

Die Variablen der vorliegenden Adreßkartei sind mit einer Ausnahme Textvariablen. Nur bei der Altersangabe handelt es sich um eine numerische Variable. Spezielle Informate und Formate werden nicht verwendet. Das SAS-System legt in diesem Fall automatisch `$8.` für die Text- und `8.` für die numerischen Variablen fest. Die Variablendeklaration kann über das Variablenfenster des SAS-Systems (vgl. Kapitel 4) abgefragt werden.

Die Zeilen der Tabelle (mit Ausnahme der obersten) enthalten die *Beobachtungen* (engl. *observation*). Sie umfassen die Ausprägungen aller zu einer Person erhobenen Merkmale oder *Variablen* (engl. *variable*). Jede Zeile steht für eine eigene Beobachtung. In vorliegendem Beispiel bilden alle Informationen, die zu Herrn Fischer in der Adreßkartei vorliegen, eine Beobachtung.

Name	Vorname	Wohnort	Alter
Fritz	Petra	Heidelberg	31
Fischer	Ulrich	Heidelberg	46
Meier	Hans	Walldorf	25
Rost	Werner	Mannheim	34
Schulz	Karin	Mannheim	27

Die Spalten der Tabelle enthalten die Merkmalsausprägungen oder Werte der Variablen. In einer Spalte stehen somit alle Werte, die zu einer bestimmten Variablen gehören. Von den Firmenmitarbeiter wohnen zwei in Heidelberg, zwei in Mannheim und einer in Walldorf. Die Zahlenwerte 25, 27, 31, 34 und 46 spiegeln die Altersangaben (in alphabetischer Reihenfolge) aller fünf Mitarbeiter wider. Soll der Altersdurchschnitt berechnet werden, genügt es, sich auf diese Spalte zu beschränken.

Name	Vorname	Wohnort	Alter
Fritz	Petra	Heidelberg	31
Fischer	Ulrich	Heidelberg	46
Meier	Hans	Walldorf	25
Rost	Werner	Mannheim	34
Schulz	Karin	Mannheim	27

Die einzelnen Zellen schließlich repräsentieren die Ausprägungen oder Datenwerte (engl. *data value*) der entsprechenden Variable (=Spalte) bei der Beobachtung (=Zeile). `Walldorf` beispielsweise ist der `Wohnort` von Herrn `Meier`.

Name	Vorname	Wohnort	Alter
Fritz	Petra	Heidelberg	31
Fischer	Ulrich	Heidelberg	46
Meier	Hans	Walldorf	25
Rost	Werner	Mannheim	34
Schulz	Karin	Mannheim	27

Eine SAS-Datei besteht aus Variablen und Beobachtungen. Für jede Beobachtung liegen Datenwerte zu allen Variablen vor. Bei fehlenden Werten zu einer Beobachtung wird die Merkmalsausprägung durch das SAS-System besonders gekennzeichnet.

Spätestens wenn Sie die Beispielprogramme des nächsten Kapitels betrachten, wird Ihnen auffallen, daß es nicht ausreicht, nur eine Datei angelegt zu haben. Unterschiedliche Typen von Informationen sollen in verschiedenen Dateien abgelegt werden. Dazu vergibt man unterschiedliche Dateinamen, beispielsweise `adressen` und `klasse`. Alle SAS-Dateien werden innerhalb eines physikalischen Bereichs auf einem Datenspeichermedium abgelegt. Der Zugriff durch das SAS-System erfolgt über eine SAS-Bibliothek (engl. *library*). Im PC- oder UNIX-Bereich entspricht dieser physikalische Bereich einem Unterverzeichnis, unter MVS einer TSO-Datei und unter VM einer Minidisk. Alle Dateien innerhalb dieses Verzeichnisses gehören zur gleichen Bibliothek, die Dateien in einem anderen Verzeichnis zu einer anderen Bibliothek. Mit der globalen LIBNAME-Anweisung werden den physikalischen Bereichen Namen für die SAS-Bibliotheken zugeordnet. Ohne speziellen Bibliotheksnamen werden die Dateien nur temporär gespeichert. Beim Verlassen der SAS-Sitzung werden diese Dateien automatisch gelöscht.

In einer Bibliothek können auch SAS-*Kataloge* abgelegt werden, in denen Grafiken, Funktionstastenbelegungen, Formate oder Bildschirmmasken gespeichert wurden. In den Abschnitten 5.4 und 5.5.1 werden Sie erfahren, wie Sie Verzeichnisse gezielt als Bibliotheken definieren bzw. wie Sie einsehen können, welche Dateien und Kataloge bereits vorhanden sind.

Außer in Datenschritten können SAS-Dateien auch mit speziellen Prozeduren (PROC DBF, PROC BMDP, PROC ACCESS), mittels ODBC (*Open Database Connectivity*) und dem Import/Export Wizard aus Dateien erzeugt werden, die außerhalb des SAS-Systems nicht in Textformat sondern als Systemdateien anderer Anwendungsprogramme vorliegen (siehe Abbildung 3.1, S. 20). In Kapitel 11 wird für einige ausgewählte Anwendungen gezeigt, wie die Übertragung der Daten in eine SAS-Datei im konkreten Fall aussehen könnte.

3.3 Ein einführendes Beispiel

Nachdem die Begriffe Datei und Programm in den beiden vorausgegangenen Abschnitten eher theoretisch erläutert wurden, wird es nun praktischer. Aus obiger Adreßkartei wird nun in einem Datenschritt eine SAS-Datei erzeugt. Anschließend werden die Datenwerte der fünf Beobachtungen mit einem Prozedurschritt in Tabellenform dargestellt. Die Beschriftung der Tabelle wird über eine globale Anweisung gesteuert.

In das Programmfenster werden die erforderlichen Programmzeilen formatfrei eingetragen, d. h. ohne Berücksichtigung einer festen Zahl von Leerzeichen innerhalb der Anweisungen. Einzelne Anweisungen können sich somit über mehrere Zeilen erstrecken, ohne daß dadurch Fehler entstehen würden.

```
00001 /*--- KA03-01.SAS ---*/
00002 OPTIONS NODATE;           *-- Globale Anweisungen;
00003 TITLE 'Adressen';
00004 DATA adressen;            *-- Datenschritt;
00005      INPUT name $ vorname $ wohnort $ alter;
00006      LINES;               *-- Datensammlung;
00007 Fritz   Petra   Heidelberg 31
00008 Fischer Ulrich  Heidelberg 46
00009 Meier   Hans    Walldorf   25
00010 Rost    Werner  Mannheim   34
00011 Schulz  Karin   Mannheim   27
00012 RUN;
00013 PROC PRINT DATA=adressen; *-- Prozedurschritt;
00014 RUN;
```

Um das Beispiel selbst durchzuspielen, können Sie entweder die 14 Zeilen ohne die Zeilennummern ins Programmfenster eintippen oder Sie nehmen die Begleitdiskette zur Hand und öffnen das Programm `KA03-01.SAS` aus dem Unterverzeichnis PROGRAMS über das Menü *File* - *Open* im Programmfenster.

Bei der ersten Programmzeile handelt es sich um einen Kommentar, der vom SAS-System nicht interpretiert wird. Er zeigt Ihnen an, unter welchem Namen dieses Beispielprogramm abgelegt ist.

Kommentierende Bemerkungen können, wie in der ersten Programmzeile von obigem Beispiel geschehen, in die Zeichen `/*` und `*/` eingeschlossen werden. In der zweiten Programmzeile sehen Sie die zweite Kommentierungsmöglichkeit: Anstelle eines Schlüsselworts beginnt die Anweisung mit einem `*`. Bis zum nächsten Semikolon ignoriert das SAS-System diese Anweisung. Mit der ersten Variante können Sie auf einfache Weise komplette, aus mehreren Anweisungen bestehende Programmschritte aus einem Programm ausschließen, während die Kommentar-Anweisung eher geeignet ist, kurze Beschriftungen aufzunehmen oder einzelne Anweisungen auszublenden.

3.3.1 Ausführen des Programms

Solange das Programm im Programmfenster steht, kann es noch verändert oder gelöscht werden, ohne daß das SAS-System davon Kenntnis nimmt. Die im Programm enthaltenen Anweisungen werden erst dann verarbeitet, wenn sie dem System zur Ausführung übergeben werden. Die Übergabe kann mit dem Kommando `SUBmit` vollzogen werden. Um das Kommando auszuführen, gibt es vier Möglichkeiten:

1. Man klickt auf die Schaltfläche  rechts neben der Kommandozeile (vgl. Abbildung 2.2).

2. Man drückt die Funktionstaste F3, die das Kommando `SUBmit` ausführt (vgl. Abschnitt 10.3).

3. Man schreibt das Kommando `SUBmit` (oder als Abkürzung `SUB`) direkt in das Kommandofenster.

4. Man aktiviert die Menüfolge *Locals* → *Submit*.

Probieren Sie alle Varianten zur Programm-Ausführung aus und wählen Sie für die Zukunft diejenige, die Ihnen am bequemsten erscheint. Während das Programm vom System ausgeführt wird, erscheinen über dem Programmfenster, auf manchen Rechnern nur für Sekundenbruchteile sichtbar, nacheinander die folgenden Meldungen:

```
Processing submitted statements
DATA STEP running
Processing submitted statements
PROC PRINT running
```

Anschließend wird das Ausgabefenster, in dem die Daten der fünf Mitarbeiter in Tabellenform angezeigt werden, automatisch aktiviert und in den Vordergrund geschoben.

Adressen

OBS	NAME	VORNAME	WOHNORT	ALTER
1	Fritz	Petra	Heidelbe	31
2	Fischer	Ulrich	Heidelbe	46
3	Meier	Hans	Walldorf	25
4	Rost	Werner	Mannheim	34
5	Schulz	Karin	Mannheim	27

Die Tabelle erscheint auf den ersten Blick ganz ordentlich, nur beim Wohnort `Heidelberg` fehlen die letzten beiden Buchstaben. Hier hat das SAS-System ein Standardformat gewählt, das mit dem 10 Zeichen langen Ortsnamen Heidelberg nicht zurechtkommt. Wie Sie dies ändern können, wird in Kapitel 5 beschrieben. Hier geht es zunächst um die grundlegenden Aktionen.

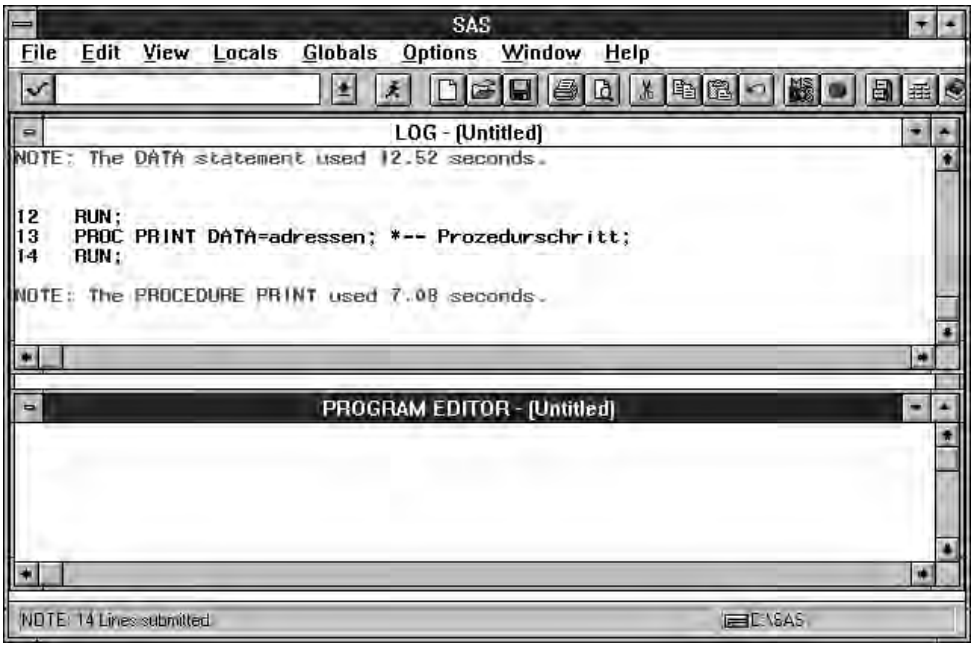


Abb. 3.2: Die SAS*-Umgebung nach Verlassen des Ausgabefensters*

Nachdem Sie das Ergebnis der Prozedur gründlich studiert haben, verlassen Sie das Ausgabefenster, indem Sie die Taste F3 drücken, die mit dem Kommando END belegt ist, oder die Menüfolge *File* → *End* wählen. Begehen Sie nicht den Fehler, das Fenster zu verlassen, indem Sie es schließen. Denn in diesem Falle würde es *iconisiert*, d. h. zu einem Symbol verkleinert, und wenn das nächste Programm wieder eine Ausgabe erzeugt, kann das Ausgabefenster nicht geöffnet werden, um die Ausgabe anzuzeigen. Erst wenn Sie es von Hand öffnen (*Windows* → *Output*), sehen Sie die Tabelle.

Vermeiden Sie, das Ausgabefenster zu schließen, um zum Programmfenster zurückzukehren. Verwenden Sie F3 oder das Kommendo END.

Wenn Sie das Ausgabefenster verlassen, können Sie im Protokollfenster die letzten Systemmeldungen erkennen (Abbildung 3.2). Das Programmfenster ist leer. Wechseln Sie in das Protokollfenster, indem Sie es einfach anklicken (oder das Kommando LOG in die Kommandozeile eintragen oder die Menüfolge *Window* → *Log* aufrufen) und gehen Sie zum Anfang der Meldungen.

```
NOTE: Copyright (c) 1989-1996 by SAS Institute Inc.,
      Cary, NC, USA.
NOTE: SAS (r) Proprietary Software Release 6.12  TS020

1    /*--- KA03-01.SAS ---*/
2    OPTIONS NODATE;           *-- Globale Anweisungen;
3    TITLE 'Adressen';
4    DATA adressen;            *-- Datenschritt;
5         INPUT name $ vorname $ wohnort $ alter;
6         LINES;

6                              *-- Datensammlung;
NOTE: The data set WORK.ADRESSEN has 5 observations and 4
      variables.
NOTE: The DATA statement used 12.52 seconds.

12   RUN;
13   PROC PRINT DATA=adressen; *-- Prozedurschritt;
14   RUN;

NOTE: The PROCEDURE PRINT used 7.08 seconds.
```

In den ersten Zeilen des Protokollfensters erscheinen Copyright-Hinweise und Informationen zur installierten SAS-Version (hier: Release 6.12 und Technical Support Level TS020). Im Anschluß werden die ausgeführten Programmzeilen durchnumeriert wiedergegeben. In der 2. und 3. Programmzeile erscheinen die

beiden globalen Anweisungen OPTIONS und TITLE. Vom anschließenden Datenschritt werden nur die Programmzeilen mit Anweisungen ausgegeben. Die fünf Zeilen mit den Datenwerten, die nach der LINES-Anweisung folgen, erscheinen nicht, da sie ausschließlich Daten, aber keine Anweisungen enthalten. Die letzten beiden Zeilen 13 und 14 gehören zum Prozedurschritt.

Würden Sie nun ein weiteres Programm schreiben und ausführen, oder das gleiche Programm erneut ausführen, würde die Zeilennumerierung mit 15, 16 usw. fortgesetzt und nicht, wie Sie vielleicht erwartet hätten, wieder mit 1 beginnen.

Außer der Protokollierung der Programmzeilen gibt das SAS-System aber auch Rückmeldung über Aktionen, die es ausgeführt hat. Zuerst wurde der Datenschritt `DATA adressen; ...` in 12.52 Sekunden bearbeitet. In ihm wurde die Datei `WORK.ADRESSEN` mit fünf Beobachtungen und vier Variablen angelegt. Nach dem Datenschritt wird der Prozedurschritt `PROC PRINT` bearbeitet, in 7.08 Sekunden. Sein Resultat ist im Ausgabefenster sichtbar und wurde bereits diskutiert. Die Ausführung der globalen Anweisungen wird nicht mit Zeitangaben kommentiert. Und das war es schon: Sie haben Ihr erstes SAS-Programm ausgeführt!

Bemerkungen

- Der obige Datenschritt hat keine eigene Ausgabe im Ausgabefenster produziert. Er hat die Datei `adressen` aus den nach der Anweisung LINES folgenden Datenwerten im SAS-System angelegt.
- Die Anweisung OPTIONS mit der Option NODATE unterdrückt die Ausgabe des aktuellen Datums in der Ausgabe.
- Die Anweisung TITLE definiert eine Überschrift, die bei der Ausgabe der Firmenadressen im nachfolgenden Prozedurschritt eingesetzt wird.
- Mit der Prozedur PRINT wird die Datei `adressen` im Ausgabefenster angezeigt. Die linke, mit `OBS` überschriebene Spalte wird vom System automatisch an die Tabelle angefügt. Sie enthält die interne Variable OBS zur Beschreibung der Beobachtungsnummer.

Meldungen im Protokollfenster

Die Systemmeldungen über die Dauer der Daten- und Prozedurschritte werden im Protokollfenster in der gleichen Farbe wie die ausgeführten und numerierten Programmzeilen ausgegeben. Neben diesen (standardmäßig) blauen Hinweisen (NOTE) können im Protokollfenster aber auch grüne Warnungen und rote Fehlermeldungen auftauchen. Die NOTE-Hinweise sind meist harmlos. Sie zeigen die Aktionen an, die ausgeführt wurden und wieviel Zeit dafür benötigt wurde. Während für das System alles fehlerfrei und ohne Komplikationen verlief, sollten Sie als Anwender dennoch ein Auge auf diese Hinweise haben.

Das Wort WARNING dagegen weist den Anwender darauf hin, daß das System bei der Ausführung eines Programmschritts auf einen Fehler gestoßen ist, der behoben werden konnte. Der Programmschritt wurde komplett ausgeführt; ob korrekt, kann das SAS-System nicht entscheiden. Da ist der Anwender gefragt.

Bei einem ERROR handelt es sich um einen Fehler, der nicht behoben werden konnte. Die Störung ist so schwerwiegend, daß der Programmschritt abgebrochen wurde. Schließen sich an einen abgebrochenen Programmschritt weitere Daten- oder Prozedurschritte an, werden diese vom System ausgeführt, allerdings mit wenig Erfolg, wenn auf den vorausgegangenen Schritten aufgebaut wird. Diese Folgefehler können Sie zunächst ignorieren. Korrigieren Sie den zuerst aufgetretenen Fehler und führen Sie das Programm erneut aus.

Auch wenn Ihnen nach der Ausführung eines Programms zuerst das Ausgabefenster angezeigt wird, sollten Sie das Protokollfenster auf eventuelle Hinweise, Warnungen und Fehlermeldungen absuchen. Denn was nützt die schönste Analyse, wenn Sie nur auf einem Bruchteil der Daten oder falschen Werten beruht?

3.3.2 Abspeichern des Programms

Zeigt das Protokollfenster nach der Ausführung des Beispielprogramms keinerlei Fehlermeldungen an, können Sie das Programm, bevor Sie die Sitzung verlassen, abspeichern, damit Sie in der nächsten Sitzung wieder darauf zugreifen können. Doch das Programmfenster ist leer und im Protokollfenster stehen zwar die Programmzeilen, dazwischen befinden sich jedoch auch die Hinweise und Zeilennummern. Außerdem fehlen die Datenwerte. Zum Glück hat das SAS-System ein gutes Gedächtnis, den Programmspeicher (engl. *program buffer*, vgl. Abbildung 4.3). Alle Programme, die ausgeführt wurden, werden dort der Reihe nach abgelegt und können in umgekehrter Reihenfolge auch wieder ausgelesen werden. Umgekehrt, weil das zuletzt ausgeführte Programm zuerst kommt und ganz am Ende das zuerst ausgeführte. Jedes Programm kann in diesem Moment nur einmal zurückgeholt werden. Erst nachdem ein weiteres Programm ausgeführt wurde, kann wieder erneut auf alle ausgeführten Programme zugegriffen werden.

Um nun das Einführungsprogramm zurückzuholen, wechselt man in das Programmfenster, entweder durch Anklicken des Fensters, über das Menü *Window* oder mit der Funktionstaste F3, die mit dem Kommando `END` belegt ist. Das Programm wird jetzt mit dem Kommando `RECall` zurückgeholt (vgl. Abbildung 2.2, ❾), indem auf die mit dem Kommando belegte Funktionstaste F4 gedrückt oder die Menüfolge *Locals* → *Recall* ausgewählt wird. Ein erneuter Rückholversuch würde scheitern, da bisher nur ein Programm ausgeführt wurde und daher auch kein weiteres zum Zurückholen vorliegt. Jetzt, wo das Programm im Programmfenster erscheint, kann es über *File* → *Save as* (oder die Schaltfläche

der Toolsleiste) abgespeichert werden. Das SAS-System schlägt als Endung des Programms *.SAS vor, als Verzeichnis das aktuelle Verzeichnis (vgl. Abbildung 2.2, ⓾). Die Vorschläge können überschrieben bzw. geändert werden. Die Endung *.SAS sollten Sie jedoch belassen. Dies hat den Vorteil, daß das SAS-System beim Öffnen von Programmen auch genau diese *.SAS-Programme zur Auswahl anzeigt, so daß Sie es sich ersparen, den Filter ständig verändern zu müssen, wenn Sie andere Erweiterungen verwenden.

Natürlich können auch die Inhalte des Protokoll- und des Ausgabefensters abgespeichert werden. Dazu aktiviert man das Fenster und geht genauso wie beim Programmfenster vor. Um das Programm auch später noch von dem Protokoll und dem Resultat unterscheiden zu können, bieten sich die in Tabelle 3.1 aufgezeigten Endungen an.

Endung	Dateityp
*.SAS	Programme
*.LOG	Protokollfenster
*.OUT	Ausgabefenster
*.DAT	Datenwerte (nicht als SAS-Datei abgelegt)

Tab. 3.1: Empfohlene Endungen (Dateierweiterungen)

Als Alternative zum Abspeichern über das Menü oder die Schaltflächen gibt es auch ein entsprechendes Kommando `FILE`. Für das Beispiel könnte es wie folgt aussehen: `FILE 'C:\Kurs\beispiel.sas'`.

3.3.3 Beenden der SAS-Sitzung

Nachdem das Programm abgespeichert wurde, kann die SAS-Sitzung beendet werden. Dazu drückt man, wie bei Windows-Anwendungen allgemein gebräuchlich, gleichzeitig die beiden Tasten ALT+F4 oder wählt die Menüfolge *File* → *Exit*. Oder, für Anhänger der Kommandosprache: `BYE`.

Gewöhnlich verlangt das SAS-System eine Bestätigung, daß die Sitzung tatsächlich verlassen werden soll, da damit alle nichtgespeicherten Programme und Daten verloren gehen. Gibt man diese, befindet man sich wieder auf der Ebene des Betriebssystems, z. B. im Windows Programm-Manager.

3.4 Aufgaben

Starten Sie eine SAS-Sitzung.

❶ Zerlegen Sie das Beispielprogramm `KA03-01.SAS` analog zu dem Prozedur SORT-Beispiel auf Seite 22 in dessen einzelne Programmschritte, Anweisungen, Optionen und Datei- und Variablennamen. Informieren Sie sich mit der Online-Hilfe oder den Handbüchern über die Bedeutung der einzelnen Programmelemente.

❷ Öffnen Sie das Beispielprogramm `KA03-01.SAS` von der Begleitdiskette (Unterverzeichnis PROGRAMS) im Programmfenster. Fügen Sie zwischen dem Daten- und dem Prozedurschritt einen weiteren Prozedurschritt ein, mit dem die Datei `adressen` absteigend nach dem Nachnamen sortiert wird. (Sie finden ein Beispiel für den Sortiervorgang auf den ersten Seiten dieses Kapitels und weitere Hinweise in der Online-Hilfe.)

Führen Sie das Programm aus und speichern Sie es mitsamt Ausgabe- und Protokollfenster auf Ihrem Rechner in einem Übungsverzeichnis auf der Festplatte oder auf einer Diskette ab.

❸ Tragen Sie eine andere Überschrift in der globalen Anweisung TITLE ein und führen Sie das veränderte Programm aus. Speichern Sie das fehlerfreie Programm und dessen Resultat im Ausgabefenster ebenfalls, aber unter anderem Namen, ab.

4

Oberflächlich – Der Display Manager

Die grafische Oberfläche des SAS-Systems, das Display Manager System (kurz DMS), das beim interaktiven Arbeiten mit dem System in Erscheinung tritt, wurde bereits im zweiten Kapitel mit seinen Grundelementen beschrieben. Neben dem großen SAS-Fenster sind dies die Menü- und die Tools-Leiste, die Kommandozeile sowie die drei sich beim Programmstart öffnenden Basisfenster: das *Programmfenster* zum Eingeben, Speichern und Ausführen von Programmen; das *Protokollfenster*, in das Systemmeldungen beim Ausführen der Programme eingetragen werden und das *Ausgabefenster*, in dem Tabellen und Analyseresultate präsentiert werden. Sofern Sie die Aufgaben des letzten Kapitels bearbeitet haben, konnten Sie erste eigene Erfahrungen mit dem DMS sammeln und einige seiner Funktionalitäten einsetzen. In diesem Kapitel lernen Sie diese Oberfläche besser kennen: weitere Fenster, neue Kommandos und die Eigenschaften des SAS-eigenen Editors.

Neben den genannten Basisfenstern gibt es weitere Fenster, die erst auf ausdrücklichen Wunsch hin geöffnet werden.

➤ Die Bedeutung der Funktionstasten wurde bei der Bearbeitung des einführenden Beispiels bereits deutlich. Sie sind mit Kommandos belegt, die ein schnelles Arbeiten ermöglichen. Das *Keys-Fenster* zeigt die aktuelle Belegung der Funktionstasten an und erlaubt deren Veränderung. Der Aufruf erfolgt über die Menüfolge *Help* → *Keys* oder F2. In Abbildung 4.1 können Sie die Standardbelegung ablesen. In Kapitel 10 (Abschnitt 10.3) wird ausführlich beschrieben, wie sich die Standardbelegung ändern und dauerhaft speichern läßt.

Abb. 4.1: Die Belegung der Funktionstasten

- Das *Options-Fenster* wird mit *Globals* → *Options* → *Global options* aufgerufen und zeigt allgemeine Systemeinstellungen an. Einige dieser Optionen können vom Anwender während der laufenden SAS-Sitzung verändert werden (etwa mit der Anweisung OPTIONS, wie im einführenden Beispiel), andere dagegen müssen bereits vor dem Systemstart gesetzt werden. Nähere Hinweise dazu finden Sie in Abschnitt 10.2.

- Alle aktuell in Zugriff befindlichen SAS-Bibliotheken und die darin enthaltenen Dateien werden vom *Libraries-Fenster* und vom *Libname-Fenster* angezeigt. Der Informationsgehalt in beiden Fenstern unterscheidet sich etwas. Das Libraries-Fenster, als das historisch jüngere Fenster, zeigt sofort die zu einer Bibliothek gehörenden Dateien und Kataloge an, während man vom Libname-Fenster aus erst das *Directory*-Fenster öffnen muß. Das *Var*-Fenster schließlich zeigt die Deklaration der Variablen einer Datei an. Genauere Ausführungen dazu folgen im nächsten Kapitel.

- Im *Grafikfenster* werden die hochauflösenden Grafiken der SAS/GRAPH-Prozeduren ausgegeben. („Normale “ Tabellen und die Grafiken der Basis-Prozeduren werden im Unterschied dazu im Ausgabefenster angezeigt.) Wenn eine Grafik-Prozedur ausgeführt wird, öffnet sich das Grafikfenster zum Anzeigen der Grafik. Mit F3 wird es wieder geschlossen. Ausführliche Erläuterungen und Beispiele zum Grafikfenster finden Sie in Kapitel 9.

Die Fenster haben in der Regel keine eigene Menüleiste. Stattdessen paßt sich die Menüleiste des großen, mit SAS überschriebenen Fensters (vgl. Abbildung 2.2) dem jeweils aktiven Fenster an: Sie ist kontext-sensitiv. Es können weitere Menüpunkte dazukommen, vorhandene Punkte inaktiviert werden (wie beispielsweise *File → Open*) oder das Menü komplett durch ein anderes ersetzt werden. Damit die Maus aber nicht immer an den oberen Bildschirmrand verschoben werden muß, um einen Menüpunkt aufzurufen, reicht ein Klick auf die rechte Maustaste. An der Stelle des Mauszeigers öffnet sich ein Popup-Menü, wie in Abbildung 4.2 dargestellt.

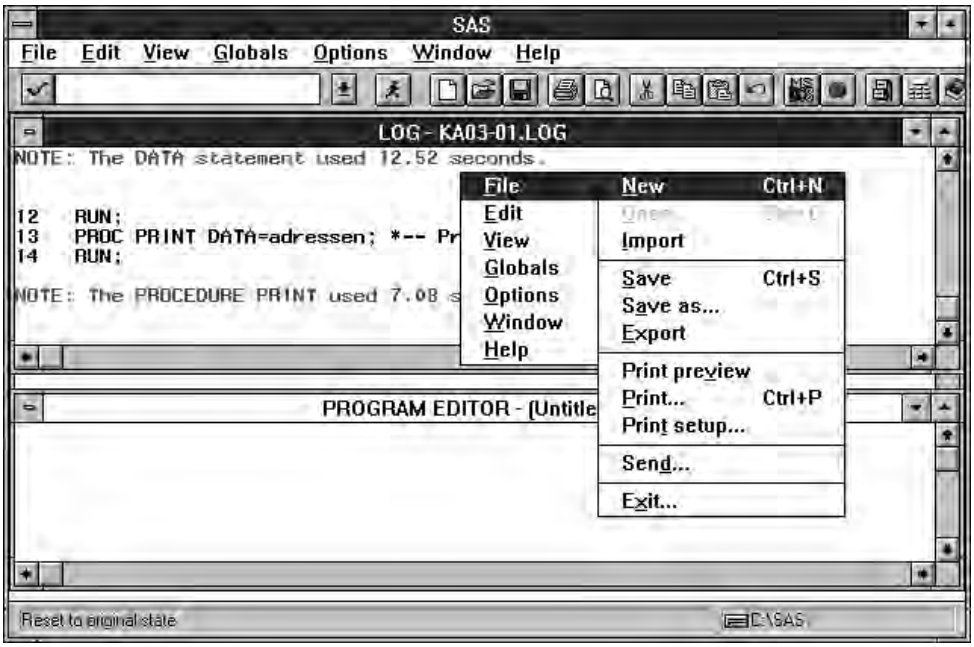


Abb. 4.2: Popup-Menü mit rechter Maustaste

Über die Menüpunkte werden Kommandos an das SAS-System übergeben, ähnlich wie dies über Schaltflächen oder die Funktionstasten (wie im einführenden Beispiel gezeigt wurde) möglich ist, sofern diese mit den entsprechenden Befehlen belegt sind.

Aber nicht jedes Kommando ist in jedem Fenster gültig. So kann man etwa nur in dem Programmfenster Programme öffnen. Im Protokoll- oder Ausgabefenster ist dies nicht erlaubt. Dagegen können die Inhalte aller drei Basisfenster mit dem Kommando FILE in externen Dateien abgespeichert werden. Menüpunkte, die für das gerade aktive Fenster verboten sind, werden vom System

inaktiviert, was Ihnen als Anwender mit einer grauen Schrift sichtbar gemacht wird. Im Protokollfenster beispielsweise hat ausschließlich das System Schreibrechte. Anwender dürfen dessen Inhalte nur lesen und abspeichern. Der Menüpunkt *Open* erscheint daher in Abbildung 4.2 in grauer Farbe, die anderen, aktiven Menüpunkte dagegen schwarz.

4.1 Kommandos

Bereits im einführenden Beispiel konnten Sie erleben, daß sich dem Anwender mehrere Möglichkeiten bieten, mit dem SAS-System zu kommunizieren: über die Kommandozeile, das Menü, die Funktionstasten oder über die Schaltflächen der Tools-Leiste. Alle vier Varianten haben eines gemeinsam: Es wird das jeweilige Kommando ausgeführt, das sich hinter der Funktionstaste, der Schaltfläche oder dem Menüpunkt verbirgt. Die Kommandos sind somit die Basis des Arbeitens mit und in den Fenstern des Display Manager Systems. Das Kommando `SUBmit` veranlaßt etwa, daß die im Programmfenster enthaltenen Anweisungen ausgeführt werden. Das gleiche passiert aber auch, wenn Sie auf die entsprechende Schaltfläche klicken, F3 drücken oder die Menüfolge *Locals* → *Submit* auswählen. Einige wichtige Kommandos werden im folgenden mit Ausführungsvarianten und Beispielaufrufen genauer erläutert.

4.1.1 Öffnen, Speichern und Ausführen

Die wichtigsten Kommandos betreffen das Öffnen, Ausführen und Abpeichern von Programmen und Ergebnissen. Sie haben sie bereits in Zusammenhang mit dem Einführungsbeispiel kennengelernt: `INClude`, `FILE`, `SUBmit`, `RECall` und `CLEAR`.

Mit dem Kommando `INClude` werden Programme im Programmfenster geöffnet, die außerhalb des SAS-Systems abgelegt sind. Dem Kommando muß als Argument der Name des Programms in Anführungszeichen mitgegeben werden:

```
INClude 'Programm'
```

Damit das einführende Beispielprogramm geöffnet wird, genügt das Kommando `INClude 'A:\programs\ka03-01.sas'`.

Wesentlich einfacher, weil man nicht den gesamten Pfad des Programms selbst eintragen muß, ist das zu öffnende Programm über die Schaltfläche  oder die Menüfolge *File* → *Open* auszuwählen. In einem entsprechenden Fenster werden alle vorhandenen Laufwerke, die Verzeichnisse und Programme angezeigt. Per Mausklick wird das gewünschte Programm ausgewählt.

Ob einfache oder doppelte Anführungszeichen verwendet werden, spielt bei den Kommandos und den Anweisungen keine Rolle. Wichtig ist nur, daß die ver-

schiedenen Anführungszeichen nicht gemischt werden: Beginnt man mit einem einfachen Anführungszeichen, endet man auch mit dem einfachen. Beginnt man mit dem doppelten, endet man mit dem doppelten. Und wenn man sich doch mal vertan hat, passiert in der Regel zunächst nichts, auch nicht das, was man eigentlich beabsichtigt hatte. Das SAS-System wartet, bis die Zeichenkette geschlossen wird. Sie müssen daher das oder die fehlenden Anführungszeichen eintragen. Weitere Hinweise zur Fehlerbeseitigung finden Sie im Anhang E.

Mit dem Kommando `FILE` werden die Programme im Programmfenster und die Inhalte des Protokoll- und Ausgabefensters abgespeichert. Ruft man *File* → *Save* bzw. *File* → *Save as* oder die Schaltfläche auf, wird wieder, analog zum Öffnungsvorgang, ein Fenster geöffnet, in dem man das Laufwerk, das Verzeichnis und den Namen, unter dem der Fensterinhalt abgelegt werden soll, auswählen kann. Speichert man direkt mit dem Kommando, muß der Name explizit in Anführungszeichen angegeben werden:

```
FILE 'name'
```

Vorsicht ist geboten beim Speichern über *File* → *Save* bzw. . Wurde bereits ein Programm abgespeichert, besteht die Gefahr, daß beim erneuten Abspeichern dieses Programm überschrieben wird. *File* → *Save as* ist in Zweifelsfällen die sicherere Variante.

Mit dem `File`-Kommando wird der komplette Inhalt des aktiven Fensters abgespeichert, nicht nur der für den Anwender gerade sichtbare Teil. Um etwa das komplette Protokollfenster nach Ausführung des Beispielprogramms abzuspeichern, muß das Protokollfenster aktiviert und folgende Zeile in die Kommandozeile geschrieben werden: `file 'C:\kurs\ka03-01.log'`.

Die Ergebnisse im Ausgabefenster können mit dem *Output Manager* einzeln gespeichert werden. Im Programmfenster kann man sich der Hilfe des Editors bedienen, um überflüssige Zeilen vor dem Speichern zu entfernen.

Das Kommando `SUBmit` wurde wiederholt vorgestellt. Mit ihm wird das aktuell im Programmfenster eingetragene Programm ausgeführt. Alternativ zum Kommando `SUBmit` kann die Schaltfläche verwendet werden, F3 oder die Menüfolge *Locals* → *Submit*.

Besteht ein Programm aus mehreren Programmschritten, können die einzelnen Schritte getrennt ausgeführt werden. Man markiert dazu den jeweiligen Programmschritt mit der Maus, so daß er invers dargestellt wird (weißer Text auf schwarzem Grund statt wie gewohnt schwarz auf weiß), und führt das Kommando `SUBmit` aus. Es werden nur die markierten Zeilen ausgeführt, die nichtmarkierten bleiben unbeachtet. Im Unterschied zum „normalen“ `SUBmit` verbleibt in diesem Fall das komplette Programm im Programmfenster. Lief der markierte Programmschritt fehlerfrei durch, kann der nächste Programmschritt markiert und ausgeführt werden. Traten jedoch Fehler auf, werden diese korrigiert

und der gleiche Schritt nochmals ausgeführt. Auf diesem Weg können große Programme einfach und schnell getestet werden.

Wird ein komplettes Programm ohne Markierung mit der Maus ausgeführt, verschwindet es aus dem Programmfenster und wird im Programmspeicher aufbewahrt. Mit dem Kommando `RECall` können diese intern gespeicherten Programme zurückgeholt werden. Das `RECall`-Kommando kann auch mit F4 oder der Menüfolge *Locals - Recall* ausgeführt werden.

Der Programmspeicher arbeitet nach dem Prinzip: Last In - First Out (LIFO). Das Programm kommt als erstes zurück (= aus dem Speicher heraus), das als letztes ausgeführt (= in den Programmspeicher eingetragen) wurde. Bildlich stellt man sich diesen Speicher am besten als einen Stapel von Programmen vor (Abbildung 4.3). Wenn ein Programm ausgeführt wird, wird es auf dem Stapel als oberstes abgelegt. Ganz unten, am Fuß des Stapels, liegt das Programm, das als erstes ausgeführt wurde. Soll nun eines der ausgeführten Programme zurückgerufen werden, kann man zunächst nur auf das oberste zugreifen. Sobald dieses im Programmfenster erscheint, wird das nächste zugänglich. An das erste, unterste Programm kommt man also erst heran, wenn der Stapel ganz abgeräumt wurde, d. h., wenn alle bisher in dieser Sitzung ausgeführten Programme zurückgeholt wurden.

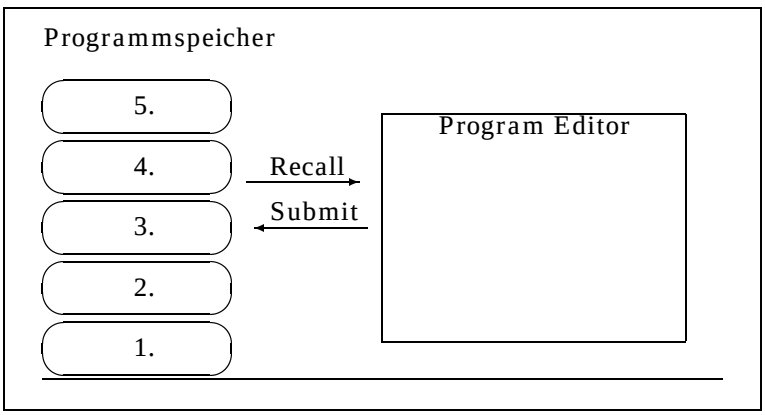


Abb. 4.3: Funktionsweise des Programmspeichers (LIFO-Algorithmus)

Wenn auch das unterste Programm zurückgeholt wurde, ist der Stapel und damit der Programmspeicher leer. Ein weiteres `RECall` ist in diesem Moment nicht möglich. Erst wenn erneut ein Programm ausgeführt wird, wird der Stapel neu zusammengesetzt und die bislang ausgeführten Programme der Reihe nach angeordnet. Das neue Programm, das zuletzt ausgeführt wurde, steht ganz oben.

Nun kann man auch wieder auf die älteren Programme zugreifen, indem man entsprechend oft RECall ausführt.

Werden im Laufe einer Sitzung zahlreiche oder längere Programme ausgeführt, umfassen das Protokoll- und u. U. auch das Ausgabefenster viele Seiten, so daß es sich empfiehlt, von Zeit zu Zeit diese Inhalte mit dem Kommando CLEAR zu löschen. Teile des Ausgabefensters können auch mit Hilfe des Output Managers gelöscht werden. Für das Protokollfenster gibt es keine selektive Möglichkeit. Hier gilt nur: Alles oder nichts. Das Kommando CLEAR, das auch über die Tastenkombination STRG+C bzw. die Menüfolge *Edit* → *Clear text* ausgeführt werden kann, kann selbstverständlich auch im Programmfenster verwendet werden.

4.1.2 Bewegen innerhalb der Fenster

In jedem Windows-Fenster gibt es Randlaufleisten, mit denen man sich innerhalb der Fenster nach oben, unten, rechts und links bewegen kann. Daneben gibt es innerhalb des SAS-Systems auch Kommandos, die die gleichen Funktionen erfüllen: BACKward, FORward, LEft, RIght, TOP und BOTTOM.

- Mit BACKward (oder BILD ↑) blättert man im aktiven Fenster eine Seite zurück, mit FORward (bzw. BILD ↓) eine Seite vor.
- Die beiden Kommandos RIght und LEft bewegen das Fenster in horizontaler Richtung nach rechts und links.
- Um ganz an den Anfang des Fensters zu gelangen, blättert man entweder solange, bis man oben ist, oder man gibt das Kommando TOP an. Mit dem Kommando BOTtom springt man zur letzten Seite. Hierzu gibt es leider keine Schaltflächen bzw. vorbelegten Funktionstasten.

4.1.3 Öffnen bzw. Aktivieren von Fenstern

Die wichtigsten Fenster können über Kommandos oder vorbelegte Funktionstasten aktiviert bzw. geöffnet werden. Nach dem Systemstart ist gewöhnlich das Programmfenster aktiviert. Um ein anderes, bereits offenes Fenster zu aktivieren, klickt man es mit dem Mauszeiger an. Um ein Fenster zu öffnen, führt man eines der folgenden Kommandos aus: PGM, LOG, OUTput, MANAGER, HELP, KEYS, DLGLIB, LIBname, DIRectory, VAR.

- Die Kommandos PGM, LOG und OUTput aktivieren (oder öffnen) das jeweilige Basisfenster. Die Kommandos sind auf die drei Funktionstasten F5, F6 und F7 gelegt. Sind die Fenster geöffnet, können sie auch über das Menü *Windows* aktiviert werden.

- Das Kommando MANAGER ruft den Output Manager auf, mit dem die Ausgaben bearbeitet und prozedurweise gedruckt und abgespeichert werden können.
- Die Online-Hilfe kann über das Kommando HELP aufgerufen werden oder auch über das Menü *Help*, die Funktionstaste F1 und die Schaltfläche  .
- Mit dem Kommando KEYS (F9 oder *Help* → *Keys*) wird das Keys-Fenster geöffnet zur Ansicht und Änderung der Funktionstastenbelegung.
- Das Libraries-Fenster wird über die Schaltfläche  geöffnet, mit der Menüfolge *Globals* → *Access* → *Display Libraries* oder mit dem Kommando DLGLIB, während das Kommando LIBname (STRG+B) das Libname-Fenster öffnet. DIRectory (STRG+D) öffnet das Directory-Fenster und VAR das Variablenfenster.

4.1.4 Kontakt zur Betriebssystemebene

Wie viele andere Windowsprogramme auch, kann die SAS-Sitzung mit ALT+F4 oder mit der Menüfolge *File* → *Exit* beendet werden. Aber Sie können auch das Kommando Bye oder die Anweisung ENDSAS (ohne weitere Optionen) verwenden, um zum Betriebssystem zurückzukehren.

Noch bevor es interaktive Dialog-Betriebssysteme wie OS/2 oder Windows gab, konnte ein Anwender aus einer laufenden SAS-Sitzung heraus Befehle an das Betriebssystem abgeben. Mit dem Kommando X (oder ) können Sie auch in der aktuellen Version zur Betriebssystemebene wechseln, dort andere Programme aufrufen und Befehle ausführen und mit EXIT wieder zur SAS-Ebene zurückwechseln. Solange Sie sich auf der Ebene des Betriebssystems bewegen, haben Sie allerdings keine Möglichkeit, SAS-Kommandos auszuführen.

Folgt dem Kommando X direkt der Befehl oder der Aufruf des externen Programms, genügt es, wenn Sie nach Ausführung des Befehls oder des Programms eine beliebige Taste drücken, um zur SAS-Ebene zurückzukehren. Mit dem Kommando X dir beispielsweise wechselt das System zur DOS-Ebene, führt das DIR-Kommando aus und kehrt anschließend wieder zum SAS-System zurück.

Unter Windows, OS/2 und UNIX kann man auch einfach ein weiteres Fenster öffnen, um Befehle auf Betriebssystemebene auszuführen, mit dem Vorteil, daß jederzeit auch SAS-Kommandos ausgeführt werden können.

Die meisten Kommandos können mit den ersten drei Buchstaben abgekürzt werden: LIB anstelle von LIBname öffnet ebenfalls das Libname-Fenster, SUB statt SUBmit führt das Programm aus. Nur wenn mehrere Kommandos existieren, die mit der gleichen Buchstabenfolge beginnen, wird vom System ein eindeutiges Kommando mit mehr als drei Buchstaben verlangt.

Kommandos können auch kombiniert werden. Dazu schreibt man sie durch Semikolon (;) getrennt hintereinander. Um beispielsweise das Protokollfenster zu aktivieren, um dessen Inhalt anschließend zu löschen, muß man die beiden Kommandos `LOG` und `CLEAR` ausführen. Als kombiniertes Kommando sieht das folgendermaßen aus: `LOG;CLEAR`.

Um alle drei Standardfenster zu löschen und anschließend das zuletzt ausgeführte Programm zurückzuholen, führt man folgende Kommandokombination aus:

```
OUT;CLEAR;LOG;CLEAR;PGM;CLEAR;RECall
```

Diesen langen Bandwurm will natürlich kein Anwender häufiger tippen. Hier bietet sich eine erste Gelegenheit, eine Funktionstaste nach eigenen Wünschen zu belegen oder eine eigene Schaltfläche zu definieren. Näheres dazu finden Sie in Kapitel 10.

4.1.5 Unterscheidung von Kommandos und Anweisungen

Für den Anfänger sind die zahlreichen Begriffe, mit denen er bei der Arbeit mit dem SAS-System konfrontiert wird, verwirrend, und es kommt häufig zu Verwechslungen zwischen Kommandos und Anweisungen. Aber wenn Sie im Auge behalten, wo Kommandos und Anweisungen ihren Platz haben und was sie bewirken, sollte nichts schiefgehen.

- *Position*: Anweisungen sind Teile der Programme und haben ihren Platz innerhalb des Programmfensters; Kommandos werden dagegen über Funktionstasten, Schaltflächen, Menüs oder die Kommandozeile ausgeführt.
- *Wirkung*: Anweisungen sind Bestandteile der Programme. Mit ihnen präzisieren Sie die Aufgabenstellung, die das SAS-System ausführen soll. Die Kommandos dagegen sorgen dafür, daß Ihr Programm an das System übertragen wird, Fenster gelöscht werden, das Ergebnis ausgedruckt wird etc.

Leider gibt es Bezeichnungen, die sowohl zu einem Kommando als auch zu einer Anweisung gehören. Oder Kommandos haben den gleichen Namen wie eine Prozedur. Beispiele sind etwa LIBNAME, FILENAME und INSIGHT. Das macht es dem Anfänger natürlich nicht leicht, die Übersicht zu bewahren. Um Verwechslungen auszuschließen, sollten Sie sich angewöhnen, das Wort *Prozedur*, *Kommando* oder *Anweisung* hinzuzufügen. Sprechen Sie etwa von der LIBNAME-Anweisung, dem Kommando `FILE`, der INSIGHT-Prozedur, wenn Sie mit anderen Anwendern über Ihre Programme und/oder Probleme diskutieren. Merken Sie sich folgende Regel:

Anweisungen sind Teile der Programme und haben ihren Platz innerhalb des Programmfensters. Kommandos lösen Aktionen aus und werden über das Menü, die Funktionstasten, Schaltflächen oder die Kommandozeile ausgeführt.

Mit den bisher vorgestellten Kommandos sind Sie in der Lage, die wichtigsten Fenster des SAS-Systems zu öffnen und zu schließen, Inhalte abzuspeichern und zu löschen sowie Programme auszuführen und zurückzuholen. Im Programmfenster haben Sie darüber hinaus die Möglichkeit, neue Programme einzutragen bzw. bestehende Programme zu öffnen und zu verändern (Editorfunktion). Zugegeben, der SAS-Editor ist nicht so komfortabel wie manch anderer Editor, aber zahlreiche Aktionen können mit ihm zufriedenstellend ausgeführt werden.

4.2 Der Editor

Nicht alle Wünsche eines SAS-Anwenders werden durch die Menüs erfüllt. Man kommt selten darum herum, eigene Programme zu schreiben oder Programme anderer Anwender abzuändern. Dazu kann man entweder einen externen Editor oder den Editor, den das SAS-System im Programmfenster anbietet, verwenden.

Ein Editor ist ein Hilfsprogramm, das es ermöglicht, Dateien für Programme, Daten oder Texte zu erstellen oder vorhandene Dateien aufzubereiten bzw. nach Bedarf abzuändern. Ein Editor sollte das Kopieren von Zeilen oder einzelnen Begriffen erlauben, das Verschieben von Zeilen und Worten, Löschen, Ersetzen und Suchen und einiges mehr. Zeichen- und Absatzformatierungen wie fett, kursiv, zentriert oder Blocksatz, wie man sie von Textverarbeitungsprogrammen kennt, werden dagegen von den wenigsten Editoren unterstützt, da diese Funktionen für die Erstellung von Programmen unwichtig sind. Zum Umfang von fast jedem Betriebssystem gehört ein eigener Editor. EDIT und VI sind etwa die Standardeditoren im PC-DOS- bzw. UNIX-Bereich. Daneben gibt es Editoren von Softwareherstellern, die leistungsfähiger und bedienungsfreundlicher sind als diese Standardeditoren. Allerdings müssen sie extra beschafft werden. Das SAS-System stellt einen eigenen Editor bereit, der den Vorteil bietet, daß bei der Arbeit mit dem System kein Zusatzprogramm aufgerufen werden muß.

Als Anwender können Sie nun entscheiden, ob Sie Ihre Programme außerhalb des SAS-Systems mit einem externen Editor schreiben und anschließend im Batchmodus verarbeiten (vgl. Kapitel 2.1) oder die Programme im Programmeditor öffnen bzw. ob Sie den SAS-eigenen Editor verwenden, um die Programme direkt in das Programmfenster einzugeben.

Ausschneiden, in die Zwischenablage kopieren und aus der Zwischenablage einfügen kann man entweder über die Schaltflächen  ,  und  realisieren,

nachdem man die zu löschende bzw. zu kopierende Textstelle mit der Maus oder über die entsprechenden Unterpunkte im *Edit* markiert hat. Das *Edit*-Menü unterstützt zusätzlich das Suchen und Ersetzen von Begriffen (*Find*, *Repeat find*, *Replace*) und eine Rechtschreibprüfung (*Check spelling*).

Weitere Aktionen kann man über sogenannte *Zeilenkommandos* ausführen. Das sind Kommandos, die nicht in die Kommandozeile eingetragen werden und auch nicht über das Menü gefunden werden können. Zeilenkommandos werden in die Zeilennummern des Programmsfensters geschrieben. In Abbildung 2.2 wurde das Display Manager System ohne Zeilennummern dargestellt. Mit dem Kommando[1] `Nums on` (oder über die Menüfolge *Edit* → *Options* → *Numbers*) werden im Programmfenster am linken Rand fünfstellige Zeilennummern angezeigt, wie in Abbildung 4.4 dargestellt.

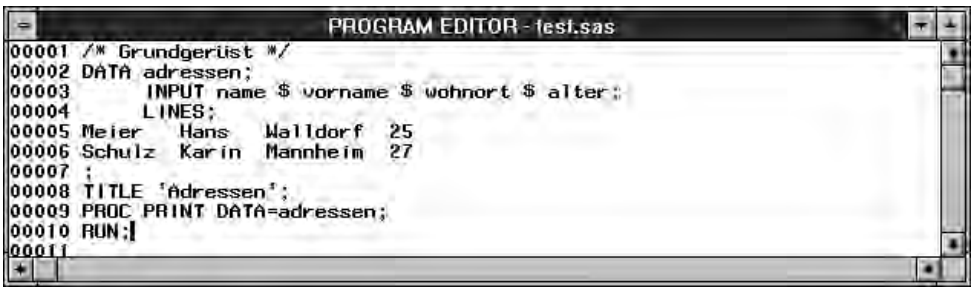


Abb. 4.4: Das Programmfenster mit Zeilennummern

Die Zeilennummern haben weder Einfluß auf das Programm noch auf die Breite der Zeilen. Mit dem Kommando `Nums off` werden sie wieder entfernt. Die Zeilenkommandos sind höchstens fünf Zeichen lang und werden direkt über die jeweiligen Zeilennummern geschrieben. Mit ihnen kann man nun ebenfalls neue Zeilen einfügen (`I` wie *Insert*), Zeilen kopieren (`C`, *Copy*), Zeilen verschieben (`M`, *Move*) und Zeilen löschen (`D`, *Delete*). Der Gebrauch und die Wirkung der Zeilenkommandos wird im folgenden an einem Beispielprogramm demonstriert. Die Zeilen, an denen Zeilenkommandos eingesetzt werden, sind mit einem `-->` gekennzeichnet.

Das in Kapitel 3 eingeführte Beispiel der Adreßkartei wird hier fortgesetzt: Sie werden beauftragt, die Adressen der Mitarbeiter zu erfassen. Ein Kollege hat Ihnen folgendes Programmgerüst erstellt.

```
00001 /* Grundgerüst */
00002 DATA adressen;
00003      INPUT name $ vorname $ wohnort $ alter;
```

[1] Kein Zeilenkommando, sondern ein Kommando über die Kommandozeile

```
   00004      LINES;
   00005 Meier   Hans   Walldorf  25
   00006 Schulz  Karin  Mannheim  27
   00007 ;
   00008 TITLE 'Adressen';
   00009 PROC PRINT DATA=adressen;
   00010 RUN;
```

Dieses Programm muß nun erweitert werden, da in der Firma weitere Mitarbeiter beschäftigt werden. Um Leerzeilen für diese neuen Werte einzufügen, tragen Sie in die 6. Zeile des Progammfensters das Zeilenkommando `I4` (für `Insert`) ein, das das System veranlaßt, vier Leerzeilen einzufügen.

```
   00001 /* Grundgerüst, ergänzt um vier Leerzeilen */
   00002 DATA adressen;
   00003      INPUT name $ vorname $ wohnort $ alter;
   00004      LINES;
   00005 Meier   Hans   Walldorf   25
-->I4006 Schulz  Karin  Mannheim   27
   00007 ;
   00008 TITLE 'Adressen';
   00009 PROC PRINT DATA=adressen;
   00010 RUN;
```

In welche Spalte das Kommando eingetragen wird, spielt keine Rolle. Es macht keinen Unterschied, ob es ganz links oder ganz rechts steht. Drückt man nun die Enter-Taste, verschieben sich die nachfolgenden Zeilen nach unten.

```
   00001 /* Grundgerüst, ergänzt um vier Leerzeilen */
   00002 DATA adressen;
   00003      INPUT name $ vorname $ wohnort $ alter;
   00004      LINES;
   00005 Meier   Hans   Walldorf   25
   00006 Schulz  Karin  Mannheim   27
   00007
   00008
   00009
   00010
   00011;
   00012 TITLE 'Adressen';
   00013 PROC PRINT DATA=adressen;
   00014 RUN;
```

In die vier neuen Leerzeilen können nun weitere Beobachtungen aufgenommen werden.

```
   00001 /* Aufnahme weiterer Mitarbeiter */
   00002 DATA adressen;
   00003      INPUT name $ vorname $ wohnort $ alter;
   00004      LINES;
   00005 Meier   Hans   Walldorf   25
   00006 Schulz  Karin  Mannheim   27
   00007 Fischer Ulrich Heidelberg 46
   00008 Fritz   Petra  Heidelberg 31
   00009
   00010
   00011;
   00012 TITLE 'Adressen';
   00013 PROC PRINT DATA=adressen;
   00014 RUN;
```

Nachdem zwei neue Mitarbeiter erfaßt wurden, stellen Sie fest, daß zu viele Leerzeilen eingefügt wurden. Um diese wieder zu löschen, tragen Sie das Zeilenkommando `Delete` ein.

```
   00001 /* Löschen überflüssiger Zeilen */
   00002 DATA adressen;
   00003      INPUT name $ vorname $ wohnort $ alter;
   00004      LINES;
   00005 Meier   Hans   Walldorf   25
   00006 Schulz  Karin  Mannheim   27
   00007 Fischer Ulrich Heidelberg 46
   00008 Fritz   Petra  Heidelberg 31
-->D0009
-->0D010
   00011;
   00012 TITLE 'Adressen';
   00013 PROC PRINT DATA=adressen;
   00014 RUN;
```

Nachdem die ENTER-Taste gedrückt wurde, sind die Leerzeilen verschwunden.

Zum Schluß sollen die Beobachtungen noch von Hand nach Namen sortiert werden[2]. Dazu wird das Zeilenkommando `Move` verwendet. Um die 7. Zeile `Fischer Ulrich Heidelberg 46` direkt im Anschluß an die 4. Zeile (nach der LINES-Anweisung) zu plazieren, muß in die 7. Zeile ein `M` eingetragen wer-

[2] Das Sortieren von SAS-Dateien mit dem SAS-Editor ist an dieser Stelle nur als Übung gedacht, um die Editorkommandos praktisch zu erklären. Normalerweise wird für das Sortieren von Dateien die SORT-Prozedur eingesetzt.

den und in die 4. Zeile ein A für *after*. Mit B für *before* würde die Zeile vor der Anweisung LINES eingefügt werden. Zu dem Zeilenkommando M gehört also stets noch die Positionierung, also die Zeile, vor oder nach der die andere Zeile eingefügt werden soll.

```
   00001 /* Verschieben von Zeilen */
   00002 DATA adressen;
   00003      INPUT name $ vorname $ wohnort $ alter;
-->A0004      LINES;
   00005 Meier   Hans   Walldorf   25
   00006 Schulz  Karin  Mannheim   27
-->M0007 Fischer Ulrich Heidelberg 46
   00008 Fritz   Petra  Heidelberg 31
   00009;
   00010 TITLE 'Adressen';
   00011 PROC PRINT DATA=adressen;
   00012 RUN;
```

Schließlich müßte noch `Fritz ...` nach `Fischer ...` eingereiht werden, damit die alphabetische Reihenfolge perfekt ist. Dazu schreibt man in die 8. Zeile M und in die 5. Zeile A.

Beim Kopieren muß ebenfalls die Positionierung der zu kopierenden Zeile angegeben werden. Um einen fünften Mitarbeiter einzutragen, der ebenso wie Frau Schulz in Mannheim wohnt, wird deren 6. Zeile ans Ende der Datenwerte kopiert:

```
   00001 /* Zeilen kopieren */
   00002 DATA adressen;
   00003      INPUT name $ vorname $ wohnort $ alter;
   00004      LINES;
   00005 Fischer Ulrich Heidelberg 46
   00006 Meier   Hans   Walldorf   25
-->C0007 Schulz  Karin  Mannheim   27
   00008 Fritz   Petra  Heidelberg 31
-->0B009;
   00010 TITLE 'Adressen';
   00011 PROC PRINT DATA=adressen;
   00012 RUN;
```

Die Zeile 7 wird kopiert (Copy) und vor der Zeile 9 eingefügt (B). Die Datenwerte der kopierten Zeile werden abschließend noch angepaßt.

```
   00001 /* Firmenadressen */
   00002 DATA adressen;
   00003      INPUT name $ vorname $ wohnort $ alter;
   00004      LINES;
   00005 Fischer Ulrich Heidelberg 46
   00006 Meier   Hans   Walldorf   25
   00007 Schulz  Karin  Mannheim   27
   00008 Fritz   Petra  Heidelberg 31
-->00009 Rost    Werner Mannheim   34
   00010;
   00011 TITLE 'Adressen';
   00012 PROC PRINT DATA=adressen;
   00013 RUN;
```

Die Adreßkartei und damit das SAS-Programm sind nun komplett. Das Programm kann ausgeführt und abgespeichert werden.

Soll das Grundgerüst ohne die persönlichen Daten abgespeichert werden, muß man zuvor die Zeilen 5-9 aus dem Programm löschen. Bevor man dazu aber fünfmal das Zeilenkommando `Delete` einträgt, löscht man eleganter die Zeilen als Block. Man verdoppelt den Buchstaben des Zeilenkommandos und markiert damit Anfang und Ende des Blocks. Alle Zeilen innerhalb des Blocks werden so auf einmal gelöscht.

```
   00001 /* Grundgerüst */
   00002 DATA adressen;
   00003      INPUT name $ vorname $ wohnort $ alter;
   00004      LINES;
-->DD005 Fischer Ulrich Heidelberg 46
   00006 Meier   Hans   Walldorf   25
   00007 Schulz  Karin  Mannheim   27
   00008 Fritz   Petra  Heidelberg 31
-->00DD9 Rost    Werner Mannheim   34
   00010;
   00011 TITLE 'Adressen';
   00012 PROC PRINT DATA=adressen;
   00013 RUN;
```

Vergißt man aus Versehen das Ende eines Blocks zu definieren, wird folgende Meldung in der Statuszeile ausgegeben: *Pending line command on line* ... und der Anfang des Blocks rot markiert. Alle bislang behandelten Zeilenkommandos können für Blöcke eingesetzt werden: `CC` und `MM`. Die Angabe zur Positionierung muß dagegen nicht verdoppelt werden: `A` (after) und `B` (before).

Neue Zeilen kann man natürlich auch im Einfügemodus erhalten, indem man die Enter-Taste drückt. Um aber mehrere Zeilen einzufügen, müßte man ent-

sprechend oft Enter drücken. Hier ist das Zeilenkommando `In` geeigneter, wobei `n` die Anzahl der einzufügenden Zeilen angibt. Mit `Ib` bzw. `Ibn` werden die Leerzeilen vor der aktuellen Zeile eingefügt. Mehrere Zeilen können auf einmal auch gelöscht (`Dn`), verschoben (`Mn`) und kopiert (`Cn`) werden.

Mit dem Zeilenkommando `O` für `Overlay` wird die mit `O` gekennzeichnete Zeile von der mit zu bewegenden (`M`) bzw. kopierenden (`C`) überlagert, d. h., leere Stellen werden durch an gleicher Position in der 2. Zeile stehende ersetzt. Zwei Zeilen können mit `TC` (`Connect`) verbunden werden, bzw. eine Zeile mit `TS` (`Split`) an der aktuellen Cursorposition in zwei Zeilen aufgeteilt werden.

Die Zeilenkommandos `JC`, `JL` und `JR` (`Justify Center`, `Left`, `Right`) zentrieren die aktuelle Zeile, bzw. richten sie links- oder rechtsbündig aus. Mit > verschiebt man die aktuelle Zeile nach rechts, mit < nach links. Gibt man zusätzlich eine Zahl `n` an, wird die Zeile um `n` Positionen verschoben.

Das Zeilenkommando `COLS` (*COLumnS*) legt ein Spaltenlineal an, das es erlaubt, im Programmfenster Spalten leichter abzulesen.

```
   00001 /* Firmenadressen */
   00002 DATA adressen;
   00003      INPUT name $ vorname $ wohnort $ alter;
   00004      LINES;
-->*COLS ----|----10---|----20---|----30---|----40---|----
   00005 Fischer Ulrich Heidelberg 46
   00006 Meier   Hans   Walldorf   25
   00007 Schulz  Karin  Mannheim   27
   00008 Fritz   Petra  Heidelberg 31
   00009 Rost    Werner Mannheim   34
   00010;
   00011 TITLE 'Adressen';
   00012 PROC PRINT DATA=adressen;
   00013 RUN;
```

In obigem Beispiel kann man etwa ablesen, daß der Nachname in den Spalten 1 bis 7 steht, der Vorname in den Spalten 9-14, der Ort in 16-25 und das Alter in den Spalten 27 und 28. Damit das Spaltenlineal wieder verschwindet, schreibt man einfach `D` (für `Delete`) in die Linealzeile.

Wenn Zeilenkommandos fehlerhaft eingegeben wurden und mit Betätigen der Enter-Taste nicht verschwinden, müssen Sie den Cursor auf die Zeilennummer setzen und das Kommando, oder die verbliebenen Reste davon, mit der Taste ENTF löschen.

Wenn Sie viel mit dem Programmeditor arbeiten und die Zeilenkommandos nutzen, können Sie häufig benötigte Zeilenkommandos wie die anderen Kommandos auch auf Funktionstasten ablegen bzw. Zeilenkommandos über die Kommandozeile eingeben. Zur Unterscheidung von den „normalen“ Kommandos müssen Sie vor das Zeilenkommando einen Doppelpunkt setzen. `:D` in der Kommandozeile löscht die aktuelle Zeile im Programmfenster. `:Cols` trägt das Spaltenlineal ein, auch wenn Sie ohne Zeilennummern arbeiten.

4.3 Aufgaben

Starten Sie eine SAS-Sitzung.

❶ Öffnen Sie das Programm `KA03-01.SAS` von der Begleitdiskette (Unterverzeichnis PROGRAMS) im Programmfenster (oder schreiben Sie das Grundgerüst ab), löschen Sie die Datenwerte der fünf Mitarbeiter und tragen Sie stattdessen Namen von Kollegen oder Familienangehörigen ein. Verschieben Sie einzelne Zeilen, üben Sie die Zeilenkommandos.

Ändern Sie die Überschrift in der Anweisung TITLE und führen Sie das Programm aus.

❷ Rufen Sie in der Online-Hilfe den Menüpunkt *Sample Programs* auf und öffnen Sie aus SAS/BASE das Beispielprogramm *SAS Language and Procedures: Usage Chapter 3*. Markieren Sie mit Hilfe der linken Maustaste den ersten Datenschritt und den darauf folgenden Prozedurschritt. Kopieren Sie danach diese beiden Programmschritte und fügen Sie sie in Ihr Programmfenster ein.

Führen Sie das Programm aus, betrachten Sie die Meldungen im Protokoll- und im Ausgabefenster und holen Sie das Programm aus dem Programmspeicher wieder ins Programmfenster zurück.

Speichern Sie das Programm nun in Ihrem Übungsverzeichnis ab.

Löschen Sie einzelne Clubmitglieder, fügen Sie neue hinzu und sortieren Sie die Mitglieder nach der Identifikationsnummer `idno`.

5

Am Anfang war die Schöpfung – Dateien anlegen

Daten können mit der SAS-Software erst analysiert und präsentiert werden, nachdem sie in eine SAS-Datei übertragen wurden. Diese Übertragung wird in den meisten Fällen in einem Datenschritt, gekennzeichnet durch die Anweisung DATA, vollzogen. (In Kapitel 11 werden zwei weitere Methoden zum Anlegen von SAS-Dateien vorgestellt: die Prozeduren FSEDIT und INSIGHT.) In dem Datenschritt werden dem System die notwendigen Informationen über die zukünftige SAS-Datei mitgeteilt: die Anzahl der Variablen, deren Namen und Anordnung in den Rohwerten, sowie der physikalische Bereich, in dem die Rohwerte zu finden sind.

Man unterscheidet, ob die Datenwerte direkt über das Programmfenster eingetragen werden oder ob sie bereits in maschinenlesbarer Form, in einer sogenannten Text- oder ASCII-Datei[1], außerhalb des SAS-Systems vorliegen. Weiterhin wird unterschieden, ob die einzelnen Werte in Spalten angeordnet oder durch ein besonderes Zeichen, etwa ein Leerzeichen, ein Stern oder ein Semikolon, getrennt werden, ob nur die Werte einer Beobachtung oder die Werte mehrerer Beobachtungen in den einzelnen Zeilen bzw. die Werte einer Beobachtung gar auf mehrere Zeilen verteilt werden.

Daneben gibt es noch weitere Möglichkeiten (die Werte werden etwa in Abhängigkeit von der ersten Spalte den Variablen zugeordnet oder in verschiedenen SAS-Dateien abgelegt), die z. B. im Lernprogramm (*Help → Online trai-*

[1] Unter einer Textdatei wird im weiteren Verlauf eine außerhalb des SAS-Systems mit jedem beliebigen Editor lesbare Datei verstanden.

ning) ausführlich besprochen und dem fortgeschrittenen Anwender überlassen werden.

Neben der Anweisung DATA, die den Datenschritt einleitet, werden im Datenschritt somit Anweisungen benötigt, die dem System den Weg zu den Daten weisen (INFILE, LINES/DATALINES und CARDS) und die Anordnung der Daten beschreiben (INPUT).

5.1 Der Beginn jeden Datenschritts – DATA

Jeder Datenschritt beginnt mit der DATA-Anweisung und hat in der Regel die Erzeugung einer SAS-Datei zum Ziel. Nach dem Schlüsselwort DATA folgt der logische Name, unter dem die Datei vom SAS-System abgelegt wird und über den auf die Datei in späteren Prozedurschritten zugegriffen werden kann. Auf dem Datenträger, z. B. der Festplatte, kann die Datei einen anderen Namen tragen als in der DATA-Anweisung festgelegt (vgl. Abbildung 5.2, S. 77).

Die Namen von SAS-Dateien dürfen wie alle SAS-Bezeichner höchstens 8 Zeichen lang sein und müssen mit einem Buchstaben oder einem Unterstrich (`_`) beginnen. Sie dürfen keine Umlaute und andere Sonderzeichen wie „`*`“ oder „`.`“ enthalten. Groß- und Kleinschreibung wird nicht unterschieden. Die Namen `test` und `klinik1` sind zulässig, während `männer` wegen des Umlauts und `krankenhaus` wegen der Länge unzulässig sind.

Um, wie im einführenden Beispiel in Abschnitt 3.3, die Firmenadressen in die SAS-Datei `adressen` zu überführen, muß der Datenschritt mit der Anweisung `DATA adressen;` beginnen.

Wird der Dateiname vergessen oder absichtlich nicht angegeben, wird der Datenschritt dennoch ohne Fehlermeldung ausgeführt. In diesem Fall legt das System eine Datei namens `DATA1` an. Falls diese Datei bereits existiert, wird `DATA2`, danach `DATA3` usw. angelegt. Zum Testen eines einzelnen Datenschritts ist dies recht hilfreich. Sobald man aber mit mehreren Dateien arbeitet und in nachfolgenden Prozedurschritten auf bestimmte Dateien zugreifen will, benötigt man deren Dateinamen, um Verwechslungen und Fehlentwicklungen auszuschließen.

Jede Regel hat ihre Ausnahme. Auch diese, daß mit jedem Datenschritt eine SAS-Datei angelegt wird. Wird als Dateiname `_NULL_` gewählt, werden alle Anweisungen innerhalb des Datenschritts ausgeführt, aber am Ende keine Datei angelegt. Die dahinter stehende Logik ist nicht auf den ersten Blick erkennbar. In Kapitel 7 werden jedoch weitere Möglichkeiten aufgezeigt, wie man im Datenschritt Berechnungen durchführen und die Ergebnisse in ansprechender Form präsentieren kann.

Werden statt einem gleich zwei oder mehrere Dateinamen nach dem Schlüsselwort DATA angegeben, werden entsprechend viele Dateien erzeugt. Beginnt

der Datenschritt beispielsweise mit der Anweisung `DATA ad1 ad2 ad2;`, werden die drei Dateien `ad1`, `ad2` und `ad3` angelegt. Ob es sich bei diesen drei Dateien um identische Kopien handelt oder um Dateien, die unterschiedliche Beobachtungen oder Variablen enthalten, hängt von den weiteren Anweisungen des Datenschritts ab. Die Syntax, also die korrekte Schreibweise der DATA-Anweisung, hat die Form:

```
DATA <datei|_NULL_|datei1 datei2 ...>;
```

5.2 Der Weg zu den Daten – LINES oder INFILE

In vielen Lehrbüchern zur SAS-Software werden die Datenwerte direkt in das Programmfenster eingetragen, damit der Anfänger sofort den Bezug zwischen dem Datenschritt und den Daten herstellen kann. In der Praxis kommt es dagegen weitaus häufiger vor, daß die Daten in einer Textdatei abgelegt wurden und erst für die Auswertung in das SAS-System übertragen werden müssen.

Die für die Realisierung der ersten Variante des Überführens von Rohwerten in SAS-Dateien notwendige Anweisung erinnert an die „Steinzeit“ im Computerwesen, in der Informationen nur über Lochkarten an den Rechner übergeben werden konnten: CARDS. Diese Anweisung muß verwendet werden, wenn die Daten direkt im Programmfenster am Ende des restlichen Programms (genauer am Ende des Datenschritts) eingegeben werden. Etwas zeitgemäßer sind die synonymen Bezeichnungen LINES und DATALINES, die in allen folgenden Beispielen anstelle von CARDS verwendet werden. Die Anweisung LINES weist an, daß bis zum nächsten Semikolon nur noch Datenwerte folgen. Erinnern Sie sich an den Datenschritt aus Kapitel 3?

```
DATA adressen;
     INPUT name $ vorname $ wohnort $ alter;
     LINES;
Fritz   Petra   Heidelberg 31
Fischer Ulrich  Heidelberg 46
Meier   Hans    Walldorf   25
Rost    Werner  Mannheim   34
Schulz  Karin   Mannheim   27
RUN;
```

Die Anweisung LINES in der dritten Zeile des Datenschritts teilt dem System mit, daß bis zum nächsten Semikolon in der 9. Zeile Datenwerte kommen. Danach ist der gesamte Datenschritt beendet. Definitiv! Es kann anschließend nur

ein neuer Datenschritt, ein Prozedurschritt oder eine globale Anweisung folgen. Wichtig ist, daß das abschließende Semikolon nach der letzten Datenzeile steht.

Die RUN-Anweisung zeigt an, daß der Datenschritt an dieser Stelle zu Ende ist. Sie könnten das Schlüsselwort RUN genauso gut weglassen und einfach ein Semikolon eintragen oder direkt den nächsten Programmschritt anschließen, da die Datenzeilen keine Anweisung im üblichen SAS-Verständnis sind und daher nicht mit einem Semikolon abgeschlossen werden müssen.

Der Übersichtlichkeit und besseren Lesbarkeit wegen sollten Sie die RUN-Anweisung allerdings eintragen. Sie beendet jeden Prozedurschritt[2] und Datenschritte, die mit der INFILE-Anweisung arbeiten. Hier, in Verbindung mit LINES, ruft sie keinen Fehler hervor, hilft aber Ihnen als Anfänger, das Ende des Datenschritts zu erkennen.

Wurden die Datenwerte bereits in einer Textdatei `adressen.dat` im Unterverzeichnis `kurs` auf Laufwerk A: abgelegt, kommt die Anweisung INFILE zum Einsatz[3]:

```
/*--- KA05-01.SAS ---*/
DATA adressen;
     INFILE 'A:\kurs\adressen.dat';
     INPUT name $ vorname $ wohnort $ alter;
RUN;
```

Die Textdatei `adressen.dat` kann außerhalb des SAS-Systems betrachtet werden, z. B. mit dem DOS-Kommando TYPE: `type A:\kurs\adressen.dat`.

Der Weg zu dieser Textdatei wird dem System über die Anweisung INFILE mitgeteilt, die direkt nach der Anweisung DATA erscheinen muß. Der Pfad der Textdatei steht nach dem Schlüsselwort in (einfachen oder doppelten) Anführungszeichen. Befindet sich die Datei im aktuellen Arbeitsverzeichnis, das rechts unten am Bildschirm angezeigt wird (⓾ in Abbildung 2.2), genügt es, innerhalb der Anführungszeichen den Namen der Textdatei einzutragen: `INFILE 'adressen.dat';`. Aber mit Angabe des kompletten Pfads liegt man immer richtig, gleichgültig von welchem Verzeichnis das SAS-System gestartet wird.

Wird eine Textdatei innerhalb eines Programms häufiger benötigt, ist es praktisch, wenn man nicht mehrfach den Pfad- und Textdatei-Namen eingeben muß. Man kann in diesem Fall eine Abkürzung, einen *Referenznamen*, für die Textdatei vereinbaren, und in der Anweisung INFILE anstelle des Pfadnamens diesen Referenznamen einsetzen.

[2] Ausnahme sind die interaktiven Prozeduren, die mit QUIT; beendet werden.

[3] Zum Ausführen des Programms müssen Sie die Begleitdiskette einlegen oder den Pfadnamen entsprechend ändern.

Der Referenzname wird mit der globalen FILENAME-Anweisung vereinbart. Nach dem Schlüsselwort erscheint zuerst der Referenzname und anschließend in Anführungszeichen der Name der Textdatei samt ihrem kompletten Pfad.

```
/*--- KA05-02.SAS ---*/
FILENAME in 'A:\kurs\adressen.dat';
DATA adressen;
     INFILE in;
     INPUT name $ vorname $ wohnort $ alter;
RUN;
```

Der vereinbarte Referenzname `in` ersetzt nun bei der Anweisung INFILE den Namen der Textdatei. Wenn während der gleichen SAS-Sitzung diese Textdatei nochmals benötigt wird, genügt die Nennung des Referenznamens `in`.

```
DATA adres-neu;
     INFILE in;
     INPUT name $ vorname $ wohnort $ alter;
RUN;
```

Die RUN-Anweisung schließt den Datenschritt ab. Würde man sie weglassen, würde das SAS-System trotz des Kommandos `Submit` auf weitere Anweisungen für diesen Datenschritt warten. Erst nachdem die Anweisung RUN oder ein weiterer Daten- oder Prozedurschritt ausgeführt wird, kann der erste Datenschritt abgeschlossen und bearbeitet werden.

```
/* Rohwerte im Programmfenster */
LINES;
rohwerte
;

/* Rohwerte in einer externen Textdatei */
INFILE 'Pfad+Dateiname'|referenzname;
```

Damit ist die erste Aufgabe erfüllt: Dem System kann die Lokalisation der Rohwerte mitgeteilt werden. Es bleibt noch die Festlegung der Variablennamen und die Deklaration ihrer Anordnung.

5.3 Jede Variable braucht einen Namen – INPUT

Die Rohwerte werden mit dem Datenschritt in eine SAS-Datei übertragen. Um in weiteren Programmschritten bequem auf die Werte der einzelnen Variablen zugreifen zu können und nicht von der m-ten Spalte oder n-ten Variablen sprechen zu müssen, vereinbart man mit der INPUT-Anweisung Variablennamen. Diese *Variablennamen* müssen der Namenskonvention genügen, die bereits in Abschnitt 5.1 für Dateinamen vorgestellt wurde: Variablennamen dürfen höchstens acht Zeichen lang sein, müssen mit einem Buchstaben oder einem Unterstrich beginnen und dürfen keine Sonderzeichen enthalten. Gültige Variablennamen sind `Name`, `Vorname` und `Alter`, aber nicht `Geschlecht` (mehr als acht Zeichen) oder `Ängste` (Sonderzeichen).

Neben den Variablennamen müssen Sie dem System auch den Variablentyp mitteilen, d. h., ob die zu einer Variablen gehörenden Datenwerte rein numerischer Natur und die Dezimalstellen durch Punkt abgetrennt sind oder ob Textzeichen auftreten. Die Textvariablen müssen in der Anweisung INPUT nach dem Variablennamen mit einem Dollarzeichen ($) gekennzeichnet werden (z. B. `nachname $`). Aus diesem Grund dürfen Variablennamen auch nicht mit dem $-Zeichen beginnen, denn in der Anweisung `INPUT name $;` wäre nicht eindeutig zu klären, ob es sich bei dem `$` um eine zweite Variable oder um den Typ der Variablen `name` handelt.

Die numerischen Variablen werden nicht gekennzeichnet. Erscheint nur ein Variablenname, wird der *Variablentyp* als numerisch angenommen. Neben Text- und numerischen Variablen unterscheidet das SAS-System keine weiteren Variablentypen. Insbesondere kann das System nicht zwischen numerisch kodierten nominalen, ordinalen und intervallskalierten Merkmalen unterscheiden.

Numerische Datenwerte werden rechtsbündig in der SAS-Datei abgelegt bzw. im Ausgabefenster angezeigt, Datenwerte von Textvariablen linksbündig. Anhand zweier Variablen, einer numerischen und einer Textvariablen, wird dies verdeutlicht. In der linken Spalte sehen Sie die Eingabe, in der rechten Spalte die zu den beiden Variablen gehörige Ausgabe, rechtsbündig die Zahlen, linksbündig die Texte.

Eingabe		*Ausgabe*	
1	Eva	1	Eva
10	Ulrich	10	Ulrich
100	Joachim	100	Joachim
1000	Matthias	1000	Matthias

5.3.1 Listengesteuertes Einlesen

Neben den Variablennamen und den Variablentypen muß auch die Reihenfolge festgelegt werden, in der die Rohwerte angeordnet sind. In obigem Datenschritt (`DATA adressen;...`) beginnen die Rohwertzeilen mit dem Nachnamen des Mitarbeiters, gefolgt von dessen Vorname, dann kommt der Wohnort und als letzte Eintragung das Alter. Entsprechend wird die INPUT-Anweisung im Beispiel formuliert: Die Variablennamen werden in der Reihenfolge aufgelistet, in der die zu den Variablen gehörenden Datenwerte in den Rohwertzeilen erscheinen. Name, Vorname und Wohnort sind Textvariablen und werden daher mit einem $-Zeichen gekennzeichnet. Die komplette INPUT-Anweisung, mit der alle vier Variablen vollständig vereinbart werden und deren Reihenfolge und Typ festgelegt wird, hat die Gestalt

```
INPUT name $ vorname $ wohnort $ alter;
```

Diese Form des Dateneinlesens ist die einfachste, die man sich vorstellen kann: Man nennt die Variablennamen, die Typen und die Reihenfolge, führt das Programm aus und fertig ist die SAS-Datei. Diese Variante wird als *Freies Einlesen* oder auch *listengesteuertes Einlesen* bezeichnet, da die Rohwerte „frei“ eingelesen werden. Sie stehen in einer Liste, nicht notgedrungen spaltengebunden wie in obigem Beipiel, und die Werte einer Beobachtung sind durch ein Leer- oder anderes Trennzeichen separiert.

Je zwei Datenwerte müssen beim listengesteuerten Einlesen durch mindestens ein Leerzeichen getrennt werden: „`Fritz Petra Heidelberg 31`“ und nicht „`FritzPetra...`“.

Datenwerte von Textvariablen dürfen höchstens acht Zeichen lang sein und kein Leerzeichen enthalten. Rufen Sie sich dazu das im Ausgabefenster gezeigte Ergebnis des einführenden Beispielprogramms ins Gedächtnis:

Adressen

OBS	NAME	VORNAME	WOHNORT	ALTER
1	Fritz	Petra	Heidelbe	31
2	Fischer	Ulrich	Heidelbe	46
3	Meier	Hans	Walldorf	25
4	Rost	Werner	Mannheim	34
5	Schulz	Karin	Mannheim	27

Von den angegebenen Wohnorten (Heidelberg, Walldorf und Mannheim) werden nur die ersten acht Buchstaben in die Variable übernommen, der Rest wird igno-

riert. Zusammengesetzte Städtenamen, wie z. B. New York oder St. Ilgen, rufen beim Ausführen des Programms eine Reihe von Hinweisen und Fehlermeldungen hervor. Lautet die Datenzeile

```
Fritz Petra St. Ilgen 31,
```

erscheinen im Protokollfenster folgende Meldungen:

```
NOTE: Invalid data for ALTER in line 5 21-25.
RULE:----+----1----+----2----+----3----+----4----+----5----+
5    Fritz   Petra   St. Ilgen  31
NAME=Fritz VORNAME=Petra WOHNORT=St. ALTER=. _ERROR_=1 _N_=1
NOTE: The PROCEDURE PRINT used 6.37 seconds.
```

Der Name und der Vorname werden korrekt übernommen: `NAME=Fritz`, `VORNAME=Petra`. Der dritte Bestandteil der Rohwerte `St.` wird für den Ort eingelesen: `WOHNORT=St.`. Das nächste Wort `Ilgen` wird als Datenwert der vierten Variable interpretiert. Da es sich bei `alter` allerdings um eine numerische Variable handelt, der ein Text zugewiesen wird, erscheint sofort eine entsprechende Meldung im Protokollfenster: `Invalid data for ALTER`. Die Zeilennummer 5 bezieht sich auf die Programmzeile 5, die nach der Meldung ausgegeben wird. Die Zahlen `21-25` bezeichnen die Spaltenpositionen, an denen der Fehler auftritt. Mit Hilfe des eingetragenen Spaltenlineals kann der Anwender erkennen, daß es sich um den Begriff `Ilgen` handelt.

Fehlende Datenwerte müssen durch einen Punkt gekennzeichnet werden. Liegt von einem Mitarbeiter, der neu eingestellt wird, die genaue Adresse noch nicht vor, können die übrigen Daten bereits in die Kartei aufgenommen werden. An der Stelle des Wohnorts wird ein Punkt zur Kennzeichnung des fehlenden Werts eingetragen: „`Karl Helmut . 51`“.

Ohne den Punkt würde die Zahl `51` anstelle des Wohnortes eingetragen und das Alter würde aus der nächsten Zeile eingelesen. Fehlen zwei hintereinander stehende Angaben, müssen zwei Punkte, getrennt durch ein Leerzeichen, eingegeben werden: „`. . Walldorf 51`“.

Die Variablen können nur in der Reihenfolge eingelesen werden, in der die Rohwerte eingegeben wurden. In obigem Beispiel ist es daher nicht möglich, zuerst den Vornamen und danach den Nachnamen zu erfassen.

Andere Trennzeichen – Zurück zu INFILE

In den bisherigen Beispielen wurden die Datenwerte nur durch Leerzeichen getrennt. Die Rohwerte können beim listengesteuerten Einlesen aber auch durch andere Zeichen, etwa * oder + separiert werden. Damit diese Trennzeichen (engl. *delimiter*) nicht mit normalen Werten von Textvariablen verwechselt werden,

müssen diese Trennzeichen mittels der DLM-Option (als Abkürzung für *Delimiter*) der INFILE-Anweisung vereinbart werden.

Stehen die Rohwerte in einer externen Datei und wird im Datenschritt die INFILE-Anweisung verwendet, um dem System den Weg zu den Rohwerten zu zeigen, muß diese Anweisung um die Option DLM= erweitert werden. Werden die Rohwerte wie im folgenden Beispiel dagegen direkt ins Programmfenster eingetragen, gibt es keine Anweisung INFILE, die erweitert werden könnte. Sie muß daher zusätzlich eingefügt werden.

```
/*--- KA05-03.SAS ---*/
DATA adressen;
     INFILE CARDS DLM='*';
     INPUT name $ vorname $ wohnort $ alter;
     LINES;
Fritz*Petra*Heidelberg*31
Fischer*Ulrich*Heidelberg*46
Meier*Hans*Walldorf*25
Rost*Werner*Mannheim*34
Schulz*Karin*Mannheim*27
RUN;
```

Der Referenzname CARDS ist ein festeingetragener Name im System und zeigt an, daß die Rohwerte, wie im vorliegenden Beispiel, direkt im Anschluß an die LINES- bzw. CARDS-Anweisung folgen und nicht in einer externen Datei abgelegt sind. Mit der Option DLM= wird der „*“ als Trennzeichen vereinbart.

Das Semikolon hat innerhalb des SAS-Systems eine besondere Rolle, da es die Anweisungen abschließt. Dient das Semikolon jedoch als Trennzeichen für die Rohwerte, muß man neben der Option DLM= in der INFILE-Anweisung zwei weitere Veränderungen im Programm vornehmen:

❶ Anstelle der Anweisungen LINES oder CARDS muß nun entweder LINES4 oder CARDS4 verwendet werden.

❷ Nach den Rohwertzeilen genügt nun nicht mehr ein Semikolon, sondern es müssen vier (!) Semikolons angefügt werden, entsprechend LINES4. Denn ein Semikolon könnte ja ein Trennzeichen für weitere Werte sein.

Die Zahl von vier Semikolons ist dabei ein festgelegter Standard und hat nichts mit den vier Variablen der Adreßdatei zu tun.

Das Einleseprogramm für die durch Semikolon getrennten Rohwerte sieht folgendermaßen aus:

```
/*--- KA05-04.SAS ---*/
DATA adressen;
     INFILE CARDS DLM=';';
     INPUT name $ vorname $ wohnort $ alter;
     LINES4;
Fritz;Petra;Heidelberg;31
Fischer;Ulrich;Heidelberg;46
Meier;Hans;Walldorf;25
Rost;Werner;Mannheim;34
Schulz;Karin;Mannheim;27
RUN;;;;
```

5.3.2 Spaltengesteuertes Einlesen

Wenn die Rohwerte von vornherein streng in Spalten angeordnet vorliegen, treten keine der beim listengesteuerten Einlesen genannten Nachteile auf. Denn die Rohwerte werden allein durch Angabe der entsprechenden Spaltenbereiche eingelesen. In der INPUT-Anweisung wird dafür nach dem Variablennamen und dem eventuell notwendigen $-Zeichen für Textvariablen der Spaltenbereich der zu der Variablen gehörenden Rohwerte angegeben. Übertragen auf obiges Beispiel nimmt das Programm folgende Form an, wobei zur Verdeutlichung das Zeilenlineal (Kommando COLS) eingeblendet wurde:

```
/*--- KA05-05.SAS ---*/
DATA adressen;
     INPUT name $ 1-7 vorname $ 9-15 wohnort $ 17-26
           alter 28-29;
     LINES;
----|----10---|----20---|----30---|----40---|----
Fritz   Petra   Heidelberg 31
Fritz   Petra   Heidelberg 31
Fischer Ulrich  Heidelberg 46
Meier   Hans    Walldorf   25
Rost    Werner  Mannheim   34
Schulz  Karin   Mannheim   27
Neuer   Anne    St. Ilgen  53
RUN;
```

Für die Variable name werden die Spalten 1-7 eingelesen, für vorname 9-15 usw. Der Leerraum zwischen zwei Werten braucht bei dieser Art des Einlesens

nicht berücksichtigt zu werden. Man könnte für den Nachnamen auch die Spalten 1-8 und für den Vornamen 9-16 einlesen.

Die Spaltenpositionen lassen sich in dem Fall, daß die Daten direkt in das Programmfenster eingetragen werden oder bei kleineren externen Dateien, die sich im Programmfenster öffnen lassen, recht einfach mit Hilfe des Spaltenlineals (Zeilenkommando `Cols`, vgl. Abschnitt 4.2) ablesen. Bei größeren Dateien, die breiter als 132 Spalten sind, ist man dagegen auf externe Hilfe (Dateibeschreibung auf Papier oder einen leistungsfähigen Editor) angewiesen.

Einzige Voraussetzung für das spaltengesteuerte Einlesen ist, daß die Datenwerte exakt in den Spalten positioniert sind und die Spalten so breit angelegt werden, daß der längste Wert darin Platz findet. Im Unterschied zum listengesteuerten Einlesen können nun auch Mitarbeiter korrekt erfaßt werden, die in Heidelberg und St. Ilgen wohnen.

```
OBS    NAME       VORNAME     WOHNORT      ALTER
 1     Fritz      Petra       Heidelberg     31
 2     Fischer    Ulrich      Heidelberg     46
 3     Meier      Hans        Walldorf       25
 4     Rost       Werner      Mannheim       34
 5     Schulz     Karin                      27
 6     Neuer      Anne        St. Ilgen      53
```

Bei fehlenden Datenwerten werden die entsprechenden Positionen frei gelassen, die Eintragung eines Punktes zur Kennzeichnung wie bei listengesteuertem Einlesen ist überflüssig.

Die in festen Spalten angeordneten Rohwerte können in einer veränderten Reihenfolge eingelesen werden. Hiermit läßt sich beispielsweise der Vorname vor dem Nachnamen (`INPUT vorname $ 9-15 name $ ...;`) einlesen, wodurch die Variablen in der Datei in vertauschter Reihenfolge abgelegt werden.

```
DATA adressen;
     INPUT vorname $ 9-15 name $ 1-7 wohnort $ 17-26
           alter 28-29;
     LINES;
Fischer Ulrich  Heidelberg 46
Meier   Hans    Walldorf   25
Rost    Werner  Mannheim   34
Schulz  Karin              27
Neuer   Anne    St. Ilgen  53
RUN;
```

5.3.3 Formatgesteuertes Einlesen und zusätzliche Bemerkungen

Als dritte Einlesevariante können Rohwerte auch formatgebunden eingelesen werden. Bei dieser Form wird in der Anweisung INPUT zuerst die Spaltenposition angegeben, an der die zu einer Variablen gehörenden Rohwerte beginnen, danach der Variablenname, eventuell das $ für Textvariablen und anschließend das für den Einlesevorgang notwendige *Einlese-* oder *Informat*.

```
/*--- KA05-06.SAS ---*/
DATA adressen;
     INPUT @1 name $7. @9 vorname $7. @17 wohnort $10.
           @28 alter 2.0;
     LINES;
Fritz   Petra   Heidelberg 31
Fischer Ulrich  Heidelberg 46
Meier   Hans    Walldorf   25
Rost    Werner  Mannheim   34
Schulz  Karin   Mannheim   27
RUN;
```

Der absolute Spaltenzeiger @N beschreibt die Spaltenposition, an der der Einlesevorgang beginnt. Bei `@1` wird die Variable Name eingelesen, bei `@9` der Vorname usw. Die Einleseformate für die drei Textvariablen `name, vorname` und `wohnort` haben die allgemeine Form $W., wobei W die Anzahl der Spalten bezeichnet (Breite, engl. *width*): Name und Vorname sind sieben Zeichen breit, Wohnort 10. Das Einleseformat für das Alter hat die Form W.D, wobei W die Breite angibt, die insgesamt (inklusive Trennpunkt) eingelesen wird, und D die Anzahl der Dezimalstellen. `2.0` bedeutet konkret, daß zwei Stellen ohne Nachkommastelle eingelesen werden.

Die Einleseformate enden stets mit einem Punkt oder enthalten zumindest einen Punkt. Die Einleseformate für Textvariablen beginnen mit dem $-Zeichen.

```
DATA adressen;
     INPUT @1 name $7. +1 vorname $7. +1 wohnort $10.
           +1 alter 2.0;
     LINES;
Fritz   Petra   Heidelberg 31
Fischer Ulrich  Heidelberg 46
Meier   Hans    Walldorf   25
Rost    Werner  Mannheim   34
Schulz  Karin   Mannheim   27
;
```

Mit dem relativen Spaltenzeiger, +N, wird die Spaltenposition um die angegebene Zahl nach rechts versetzt. Die Rohwerte werden im Datenschritt zeilenweise verarbeitet. Ein (interner) Spaltenzeiger vermerkt die Position, an der gerade gelesen wird. Beim formatgesteuerten Einlesen wird an der ersten Position damit begonnen, die Variable name einzulesen. Der Spaltenzeiger rückt sieben Positionen nach rechts: Er steht, nachdem der Name eingelesen wurde, auf der Position 8. Von dort muß er um eine weitere Position nach rechts rücken, um den Vornamen einlesen zu können: Mit +1 wird er dazu bewegt, diesen einen Schritt zu machen, während er mit @9 „ohne Nachzudenken“ direkt zur 9. Position springen würde. Während man mit dem absoluten Spaltenzeiger die Reihenfolge der Variablen beim Einlesen verändern kann, kann sich der relative Spaltenzeiger nur nach rechts bewegen.

Nichtstandard-Daten, wie etwa Datumsangaben, und numerische Werte, bei denen die Nachkommastellen mit einem Komma abgetrennt sind, werden am einfachsten formatgesteuert eingelesen. Welche Einleseformate es für Datumsangaben gibt, wird ausführlicher im Abschnitt 7.6 beschrieben.

Die verschiedenen Arten der Einlesesteuerung können innerhalb einer INPUT-Anweisung gemeinsam verwendet werden. Man kann beispielsweise spaltengesteuert beginnen, formatgesteuert fortfahren und schließlich listengesteuert enden.

Beim listengesteuerten Einlesen können durch Angabe eines Doppelpunkts (:) und zusätzlichen Formatangaben auch Zeichenketten eingelesen werden, die länger als 8 Zeichen sind. Durch Angabe eines &-Zeichens werden Datenwerte mit Leerzeichen korrekt verarbeitet. Damit kann man zwei der oben genannten Nachteile ausgleichen.

Um die Rohwerte des Beispiels korrekt listengesteuert einzulesen, müssen die längeren Wohnorte berücksichtigt werden. Nach dem Variablennamen und dem $ wird ein Doppelpunkt und die Anzahl der Stellen, die ein Wohnort maximal haben kann, hier 20, eingetragen.

```
DATA adressen;
     INPUT  name $ vorname $ wohnort $ :20. alter;
     LINES;
Fritz Petra Heidelberg 31
Fischer Ulrich Heidelberg 46
Meier Hans Walldorf 25
Rost Werner Mannheim 34
Schulz Karin Mannheim 27
;
```

Mit dem Zusatz `:20.` wird die Standardlänge der Variablen von acht auf 20 Zeichen erweitert. Alternativ dazu könnte man die LENGTH-Anweisung verwenden.

```
DATA adressen;
     LENGTH wohnort $20.;
     INPUT  name $ vorname $ wohnort alter;
     LINES;
Fritz Petra Heidelberg 31
Fischer Ulrich Heidelberg 46
Meier Hans Walldorf 25
Rost Werner Mannheim 34
Schulz Karin Mannheim 27
;
```

Die Anweisung LENGTH vor der INPUT-Anweisung definiert die Variable `wohnort` als Textvariable der Länge 20. In der INPUT-Anweisung ist dann das $-Zeichen bei `wohnort` überflüssig.

Damit auch Wohnorte eingelesen werden, die ein Leerzeichen enthalten, ergänzt man die Angabe um ein & -Zeichen und rückt die Altersangaben um ein weiteres Leerzeichen, insgesamt also zwei, vom Wohnort ab.

```
DATA adressen;
     INPUT  name $ vorname $ wohnort $ &:20. alter;
     LINES;
Fritz Petra Heidelberg  31
Fischer Ulrich Heidelberg  46
Meier Hans Walldorf  25
Rost Werner Mannheim  34
Schulz Karin Mannheim  27
Neuer Anne St. Ilgen  53
;
```

In den bisherigen Beispielen paßten alle zu einer Beobachtung gehörenden Datenwerte in eine einzige Zeile der Rohwertedatei. Die nächsten beiden Abschnitte beschreiben die Fälle, daß mehrere Zeilen pro Beobachtung benötigt werden bzw. mehrere Beobachtungen in einer Zeile stehen.

5.3.4 Mehrere Eingabezeilen pro Beobachtung

Bei größeren Erhebungen mit mehreren hundert Merkmalen kann man leicht den Überblick über die Daten verlieren, wenn die Datenwerte einer Beobach-

tung in einer einzigen Zeile stehen und man ständig gezwungen ist, nach rechts oder links zu blättern, um die Werte zu kontrollieren. Komfortabler ist es stattdessen, wenn die Datenwerte auf mehrere Zeilen verteilt werden, damit man höchstens nach oben und unten blättern muß, um alle zu einer Beobachtung gehörigen Werte zu sehen.

Analog zum Spaltenzeiger gibt es einen Zeilenzeiger, mit dem man in der Anweisung INPUT steuert, in welchen Zeilen welche Variablen stehen und in welcher Reihenfolge diese eingelesen werden. Man unterscheidet zwischen dem absoluten Zeilenzeiger # und dem relativen Zeilenzeiger /. Mit dem absoluten Zeilenzeiger kann man beliebig zwischen den Zeilen springen, während der relative Zeilenzeiger nur den Sprung in die nächste Zeile gestattet.

Die Rohwerte der fünf Mitarbeiter werden nun um eine weitere, zweite Zeile ergänzt, die den Beginn der Beschäftigung und die Abteilung enthält, der der Mitarbeiter angehört.

```
/*--- KA05-07.SAS ---*/
DATA adressen;
     INPUT  #1 name $ vorname $ wohnort $ &:20. alter
            #2 jahr 7-8 abt $;
     LINES;
Fritz Petra Heidelberg  31
15/01/91 A
Fischer Ulrich Heidelberg  46
01/06/90 B
Meier Hans Walldorf  25
01/06/90 A
Rost Werner Mannheim  34
01/07/91 A
Schulz Karin Mannheim  27
15/08/94 B
Neuer Anne St. Ilgen  53
01/01/92 A
RUN;
```

Zunächst werden die in der jeweils ersten Zeile (#1) stehenden Werte der Variablen name, vorname, wohnort und alter eingelesen, anschließend die beiden Variablen jahr und abt der zweiten Zeile (#2). Bis auf den Beschäftigungsbeginn werden die Werte listengesteuert eingelesen. Von dem Datum wird über die Spalteneingabe nur das Jahr erfaßt.

Bei der Verwendung von Zeilenzeigern ist es wichtig, daß für jede Beobachtung gleich viele Rohwertzeilen vorliegen, also im vorliegenden Fall zwei Zeichen pro Beobachtung. Ist dies nicht der Fall, kommt der Datenschritt beim Abarbeiten

der Anweisung INPUT durcheinander und produziert neben vielen Fehlermeldungen unvollständige Daten.

5.3.5 Mehrere Beobachtungen pro Eingabezeile

In dem Fall, daß mehrere Rohwertzeilen pro Beobachtung vorliegen, sorgt man durch Angabe der Zeilenzeiger dafür, daß der Einlesezeiger in die folgenden Zeilen springt. Liegen nun im umgekehrten Fall mehrere Beobachtungen pro Datenzeile vor, muß man verhindern, daß der Einlesezeiger nach dem Einlesen einer Beobachtung in die nächste Zeile springt. Dazu setzt man den doppelten Zeilenhalter @@ am Ende der Anweisung INPUT, direkt vor dem Semikolon, ein. Der Zeilenhalter sorgt dafür, daß der Zeilenzeiger nach dem Einlesen der ersten Beobachtung nicht in die nächste Zeile springt, sondern in der gleichen Zeile die nächste Beobachtung einliest. Dieser Vorgang wird solange wiederholt, bis alle Datenwerte, die in der Zeile stehen, in die SAS-Datei übertragen wurden.

Ein Mitarbeiter der Firma lädt zu seinem Geburtstag ein. Für ein gemeinsames Geschenk wird Geld gesammelt und der jeweilige Betrag mit dem Namen des Geldgebers notiert.

```
/*--- KA05-08.SAS ---*/
DATA adressen;
     INPUT  name $ &20. betrag @@;
     LINES;
Fritz P.  5.00 Fischer U.  10.00 Meier H.  5.00
Rost W.  5.00 Schulz K.  15.00
RUN;
```

In der ersten Datenzeile (nach der Anweisung LINES) wurden drei Beobachtungen aufgenommen, in der zweiten Zeile zwei weitere. Die Namen sind wegen des & -Zeichens durch zwei Leerzeichen vom Betrag abgetrennt.

Der Zeilenhalter @@ weist das System während der Ausführung des Datenschritts an (nachdem `Fritz, P.` und `5.00` eingelesen wurden), mit `Fischer` für die zweite Beobachtung in der gleichen Zeile fortzufahren. Es erscheint der folgende Hinweis im Protokollfenster:

```
NOTE: SAS went to a new line when INPUT statement reached
      past the end of a line.
```

Diese Meldung weist den Anwender darauf hin, daß der Zeilenhalter das Ende der Datenzeile erreicht hat und in der nächsten Zeile der Einlesevorgang fortgesetzt wird.

Das Einlesen mehrerer Beobachtungen pro Eingabezeile kann nicht mit dem spaltengesteuerten Einlesen kombiniert werden. Der Zeilenhalter hindert das System daran, in die nächste Zeile zu springen, und die Spaltensteuerung erlaubt nur, die jeweils gleichen Spalten zu lesen. Der Einlesevorgang dreht sich im Kreis: eine Endlosschleife, die erst endet, wenn das System nicht mehr genügend Platz zur Speicherung der Datei finden kann oder abgebrochen wird.

Abschließend werden die verschiedenen Formen, die die INPUT-Anweisung annehmen kann, in ihrer allgemeinen Syntax zusammengestellt. Dabei wird jede Variante getrennt dargestellt, obwohl alle Methoden miteinander kombinierbar sind.

```
/* Listengesteuertes Einlesen */
INPUT variabl1 <$> <&> <:> variabl2 ...;

/* Spaltengesteuertes Einlesen */
INPUT variabl1 <$> n1-n2 variabl2 ...;

/* Formatgesteuertes Einlesen */
INPUT @1 variabl1 <$> einleseformat @i|+j variabl2 ... ;

/* Mehrere Eingabezeilen pro Beobachtung */
INPUT #1 variabl1 ...       #2|/ variabl... ...;

/* Mehrere Beobachtungen pro Eingabezeile */
INPUT variabl1 ... @@;
```

5.3.6 (Fehler-) Meldungen – Nützliches mit INFILE

Da der Datenschritt für die weitere Arbeit wichtig ist, wenn z. B. die erzeugten Dateien in nachfolgenden Prozedurschritten verarbeitet werden, sollte der Anwender jede Meldung, die im Protokollfenster ausgegeben wird, sei es eine Fehlermeldung, eine Warnung oder eine Notiz, sehr sorgfältig betrachten und sich Gedanken über deren Ursache machen. Die Meldung

```
NOTE: SAS went to a new line when INPUT statement reached
      past the end of a line.
```

kann, wie oben gezeigt, auftreten, wenn der Zeilenhalter @@ eingesetzt wird. Das System springt nicht automatisch in die nächste Zeile, nachdem die erste Beobachtung eingelesen wurde, sondern liest weiter Werte ein, solange in der

ersten Zeile Rohwerte vorhanden sind. Erst wenn dort kein weiterer Wert mehr vorhanden ist, wird die Zeile gewechselt und die Meldung ausgegeben. Die Meldung kann den Anwender aber auch auf einen möglicherweise unterlaufenen Fehler beim listengesteuerten Einlesen aufmerksam machen, wie das nächste Beispiel zeigt.

Die externe Textdatei `A:\kurs\zahl.dat` besteht aus 3 Zeilen, die jeweils die Zahlen 1 bis 4 enthalten, wobei allerdings in der 2. Zeile die letzten beiden Zahlen fehlen. (Sie finden diese Datei auf der Begleitdiskette.)

```
1 2 3 4
1 2
1 2 3 4
```

Um diese Werte in eine SAS-Datei mit vier Variablen `a`, `b`, `c` und `d` zu übertragen, führt man einen Datenschritt aus. Mit dem PROC PRINT-Schritt werden die Werte der SAS-Datei im Ausgabefenster angezeigt.

```
/*--- KA05-09.SAS ---*/
DATA zahlen;
     INFILE 'A:\kurs\zahl.dat';
     INPUT a b c d;
RUN;
PROC PRINT DATA=zahlen;
RUN;
```

Nach Ausführung des Programms erscheinen im Ausgabefenster

```
OBS    A    B    C    D

 1     1    2    3    4
 2     1    2    1    2
```

und im Protokollfenster u.a. die Meldungen

```
NOTE: 3 records were read from the infile 'C:\kurs\zahl.dat'
      The minimum record length was 3.
      The maximum record length was 7.
NOTE: SAS went to a new line when INPUT statement reached
      past the end of a line.
NOTE: The data set WORK.ZAHLEN has 2 observations and 4
      variables.
```

Was ist bei der Ausführung des Datenschritts passiert? Wo ist die dritte Beobachtung geblieben? Warum wurden die Werte `1` und `2` der dritten Beobachtung anstelle der fehlenden Werte eingetragen?

Betrachtet man die Werte im Ausgabefenster genauer, fällt zunächst auf, daß die erste Beobachtung korrekt eingelesen wurde. Die beiden vorhandenen Werte der zweiten Beobachtung wurden auch eingetragen. Da die für listengesteuertes Einlesen notwendige Kennzeichnung von fehlenden Werten durch den Punkt (.) fehlt, springt der Einlesezeiger in die nächste Zeile und füllt mit den dort befindlichen Werten `1 2` die Werte der zweiten Beobachtung auf. Zum Einlesen der Werte der dritten Beobachtung springt der Zeiger in die nächste Zeile, die aber schon nicht mehr vorhanden ist. Der dritte und vierte Wert der dritten Rohwertzeile wird nicht erfaßt. Welche Auswege gibt es aus diesem Dilemma?

❶ Man ergänzt die Textdatei `zahl.dat` um die Punkte zur Kennzeichnung der fehlenden Werte.

❷ Man teilt dem SAS-System mit, daß es Zeilen, in denen am Ende der Zeile Beobachtungswerte fehlen, automatisch mit fehlenden Werten auffüllen soll.

Bei kleinen Textdateien ist die Eintragung von Punkten anstelle der fehlenden Werte machbar.

```
1 2 3 4
1 2 . .
1 2 3 4
```

Bei größeren Dateien jedoch würde dies einen erheblichen Aufwand bedeuten und im modernen Computerzeitalter ist ein solches Vorgehen nicht zeitgemäß und auch nicht professionell. Daher ist die zweite Variante die elegantere: Sie nutzen die Option MISSOVER der Anweisung INFILE, um dem System mitzuteilen, daß fehlende Werte am Ende einer Zeile automatisch aufgefüllt werden sollen.

```
DATA zahlen;
     INFILE 'C:\kurs\zahl.dat' MISSOVER;
     INPUT a b c d;
RUN;
PROC PRINT DATA=zahlen;
RUN;
```

Im Protokollfenster bleibt daraufhin der Hinweis aus, daß das System in eine neue Zeile springen mußte, und die Datei enthält am Ende drei Beobachtungen mit den entsprechenden Werten.

Nun gut, werden Sie jetzt vielleicht denken, die Werte könnten doch auch, als dritte Möglichkeit, spaltengebunden eingelesen werden? Auch dabei springt der Einlesezeiger nicht ungewollt in die nächste Zeile, wenn ihm Werte fehlen. Mit dieser Einleseart werden fehlende Werte direkt erkannt und übertragen. In Ordnung, der Versuch soll zeigen, ob mit spaltengebundenem Einlesen das Problem behoben werden kann.

In der INPUT-Anweisung müssen dazu die Spaltenbereiche, die in diesem Fall aus genau einer Spalte bestehen, für die vier Variablen ergänzt werden:

```
/*--- KA05-10.SAS ---*/
DATA zahlen;
     INFILE 'A:\kurs\zahl.dat';
     INPUT a 1 b 3 c 5 d 7;
RUN;
PROC PRINT DATA=zahlen;
RUN;
```

Das Ausgabefenster zeigt nach der Ausführung wieder zwei Beobachtungen an, aber mit unterschiedlichen Werten in der zweiten Beobachtung.

```
OBS    A    B    C    D

 1     1    2    3    4
 2     1    2    1    4
```

Die erste Beobachtung wurde korrekt übernommen. Die Werte für die Variablen a und b der zweiten Beobachtung wurden eingetragen, anschließend erscheint allerdings eine 1 bei der Variable c und eine 4 bei d. Die dritte Beobachtung fehlt. Und zur allgemeinen Überraschung erscheint im Protokollfenster die gleiche Meldung, wie beim listengesteuerten Einlesen, daß das System in eine neue Zeile wechseln mußte, obwohl spaltengebunden eingelesen wurde.

Die Ursache dieses Phänomens liegt an der internen Speicherung der Textdatei zahl.dat. Um Speicherplatz einzusparen, werden Textdateien in der Regel nicht als Rechteckdateien abgelegt, worin jede Zeile gleich lang wäre, sondern ein internes Zeilenende markiert die Spaltenposition, an der das letzte Zeichen steht. Für die Textdatei zahl.dat wird dieses Zeilenende in der folgenden Darstellung durch einen * für den Leser sichtbar gemacht.

```
1 2 3 4*
1 2*
1 2 3 4*
```

Man erkennt, daß die erste und dritte Zeile gleich lang und jeweils 7 Zeichen breit sind. Die zweite Zeile dagegen ist nur drei Zeichen breit. Soll nun hier, wie im obigen Datenschritt gefordert, die Spalte 5 eingelesen werden, kann das System diese Spalte in der zweiten Zeile nicht finden, sondern behilft sich damit, daß es in die nächste Zeile springt und dort die erst (-beste) Zahl übernimmt. Für die vierte Variable d wird in der dritten Zeile die 7. Spalte abgelesen.

Ein Ausweg aus dieser Situation bietet die Option PAD der INFILE-Anweisung. Sie veranlaßt das System, bevor dieses auf die Textdatei zugreift, alle Zeilen gleich breit zu machen, indem sie mit Leerzeichen aufgefüllt werden. Das interne Zeilenende steht dabei in der jeweils gleichen Spalte. Im Fall der Textdatei zahl.dat könnte man es sich wie folgt vorstellen:

```
1 2 3 4*
1 2   *
1 2 3 4*
```

Der zugehörige Datenschritt zum spaltengesteuerten Einlesen hat die folgende Form:

```
DATA zahlen;
     INFILE 'A:\kurs\zahl.dat' PAD;
     INPUT a 1 b 3 c 5 d 7;
RUN;
PROC PRINT DATA=zahlen;
RUN;
```

Neben diesen mehr technischen Optionen, die zur Fehlerbeseitigung beim Einlesen von Textdateien hilfreich sind, bietet die Anweisung INFILE zahlreiche Optionen, von denen die wichtigsten kurz vorgestellt werden:

- Das Erstellen eines Datenschritts kann sich bei größeren Textdateien als schwierig erweisen. Müssen viele Variablen über die Anweisung INPUT vereinbart werden, sollte man den Datenschritt zunächst an einer kleinen Zahl von Beobachtungen testen. Anstatt nun die Textdatei editieren und bearbeiten zu müssen, bestimmt man über die beiden Optionen FIRSTOBS= und OBS= der Anweisung INFILE, bei welcher Zeile in der Textdatei der Einlesevorgang begonnen und bis zur wievielten Beobachtung dieser fortgesetzt werden soll.

  Mit FIRSTOBS=10 OBS=3 werden beispielsweise die 10., 11. und 12. Beobachtung eingelesen. Wird nur die Option FIRSTOBS= angegeben, werden alle Beobachtungen ab der genannten übertragen, wird dagegen nur OBS= gesetzt, werden nur die ersten n Beobachtungen übertragen.

Die Option FIRSTOBS= ist auch sehr hilfreich, wenn im Kopf der Textdatei ein Begleittext eingetragen wurde, etwa die Dateibeschreibung, ein Kommentar oder Erläuterungen. Man muß diese Textzeilen nicht aus der Rohwertedatei entfernen, sondern überspringt diese durch Angabe der ersten Zeile, in der die eigentlichen Rohwerte beginnen.

➤ Standardmäßig werden vom SAS-System nur solche Textdateien eingelesen, deren Zeilenlänge kürzer als 256 Zeichen ist. Da größere Satzlängen aber gerade bei automatisch erstellten Dateien vorkommen können, kann man über die Option LRECL= (Abkürzung für engl. *Logical record length*) die Zeilenlänge festlegen. Mit `LRECL=1000` können 1000 Zeichen pro Spalte eingelesen werden.

Eine vollständige Liste aller Optionen der Anweisung INFILE findet man in der Online-Hilfe oder im Language Reference Guide [9, S. 377ff.]. Es folgt eine Zusammenstellung der hier besprochenen Optionen und der Syntax der INFILE-Anweisung.

```
INFILE 'Pfad+Dateiname'|referenzname|CARDS
       <MISSOVER> <PAD>
       <FIRSTOBS=n> <OBS=n>
       <LRECL=n>;
```

5.4 Von bleibendem Wert – Permanente Dateien

In den bisherigen Beispielen wurden die Dateien `adressen`, `adres-neu` und `zahlen` erzeugt. Beendet man die SAS-Sitzung und startet später das System neu, sind diese Dateien nicht mehr da. Ein Prozedurschritt, der auf eine der zuvor angelegten Dateien zugreifen will, würde zu einer Fehlermeldung führen. Erst nachdem die Datenschritte erneut ausgeführt werden, kann man auf die Dateien `adressen`, `adres-neu` und `zahlen` wieder mit einem PROC PRINT-Schritt zugreifen. Mit dem Beenden einer SAS-Sitzung wird also „aufgeräumt“ und alles, was nicht ausdrücklich gespeichert wurde, gelöscht.

Studiert man die Meldungen genauer, die nach der Ausführung eines Datenschritts im Protokollfenster erscheinen, erkennt man, daß das SAS-System vor dem Dateinamen einen Zusatz ausgibt.

```
NOTE: The data set WORK.ADRESSEN has 5 observations and 4
      variables.
```

Die Datei `adressen` wird nicht einfach mit `ADRESSEN` bezeichnet, sondern mit `WORK.ADRESSEN`. Der Name `WORK` (= *Arbeit*) deutet daraufhin, daß es sich um eine Arbeitsdatei handelt, die am Ende der SAS-Sitzung gelöscht wird. Die Arbeitsdateien werden auch als *temporäre Dateien* bezeichnet, da sie nur zeitlich begrenzt existieren, im Unterschied zu den beständigen, *permanenten Dateien*, die nach Beendigung einer Sitzung erhalten bleiben.

Der Zusatz `WORK` weist gleichzeitig auf die SAS-Bibliothek (vgl. Abschnitt 3.2) hin, in der die temporären Dateien abgelegt sind: der temporäre Arbeitsbereich. Dieser Arbeitsbereich wird beim Starten der SAS-Sitzung automatisch in Form eines Unterverzeichnisses angelegt. Alle im Laufe der Sitzung erzeugten temporären Dateien werden dort abgelegt, und nach der Beendigung der Sitzung werden die Einträge in diesem Unterverzeichnis gelöscht.

Die temporäre Bibliothek wird über den Namen WORK angesprochen. Anstatt den Datenschritt mit der Anweisung `DATA adressen;` einzuleiten, könnte man auch `DATA work.adressen;` schreiben. Denn genau unter diesem Namen wird die Datei abgelegt. Aber wozu der Mehraufwand? Man muß sich nur merken, daß mit einteiligen Dateinamen diese immer im temporären Arbeitsbereich abgelegt werden.

Wenn man mit dem SAS-System „ernsthaft“ arbeitet, ist es sehr unkomfortabel, wenn die im Laufe einer Sitzung erstellten Dateien am Ende gelöscht werden. Man möchte permanent mit den Dateien arbeiten können, die man einmal angelegt hat. Dazu muß man dem SAS-System aber mitteilen können, daß die Dateien nicht in der Bibliothek `WORK`, sondern in einer permanenten Bibliothek gespeichert werden sollen. Für die Umsetzung werden zwei Schritte benötigt:

❶ Zunächst definiert man den Namen der permanenten Bibliothek, indem eine Verbindung zwischen dem Namen und dem (existierenden) Unterverzeichnis über das Libraries-Fenster oder über die LIBNAME-Anweisung hergestellt wird.

❷ Anschließend vergibt man im Datenschritt in der Anweisung DATA einen zusammengesetzten, zweiteiligen Dateinamen: *Bibliothek.Datei*.

Für das folgende Beispiel wird wieder auf die Adreßkartei der Firma zugegriffen. Um den Datenschritt aus Kapitel 3 nicht ständig wiederholen zu müssen, soll die Datei nun permanent angelegt werden. Als Bibliothek dient das Übungsverzeichnis `C:\kurs`. Dort soll die Datei abgespeichert werden. Für die Festlegung des Bibliotheksnamens wird das Libraries-Fenster durch Auswählen der Schaltfläche  (vgl. Abbildung 5.3, Abschnitt 5.5.1) aufgerufen und über *New Library* das Eingabefenster (vgl. Abbildung 5.1) geöffnet, in das die notwendigen Informationen zum Anlegen einer neuen Bibliothek eingetragen werden können.

Unter *Library* gibt man den Namen der Bibliothek an. Für das Beispiel wurde `biblio` gewählt. Das Unterverzeichnis `C:\kurs` trägt man entweder direkt

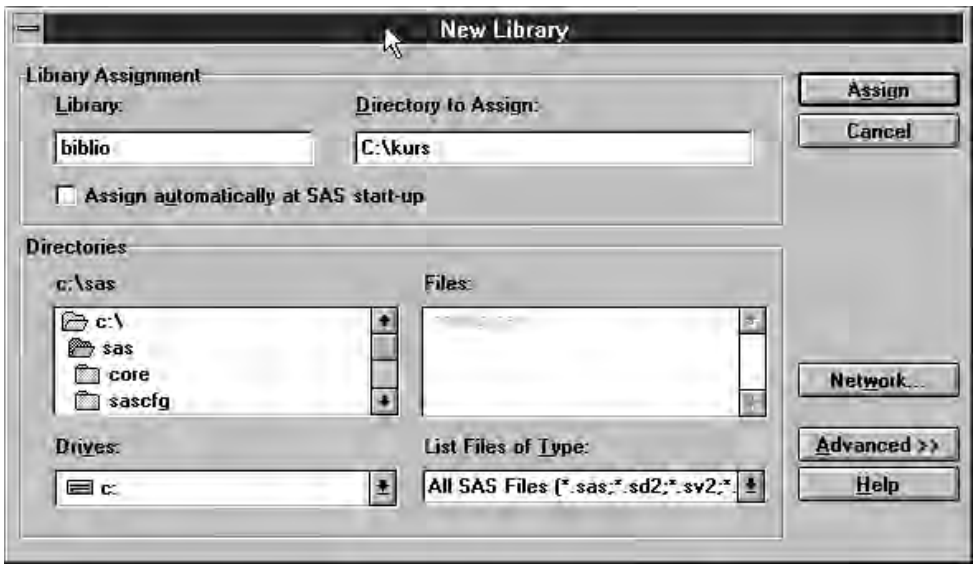


Abb. 5.1: Das New Library-Fenster

unter *Directory to Assign* ein, oder man wählt über *Directories* Laufwerk und Unterverzeichnis per Mausklick aus. Bei *Files* werden alle SAS-typischen Dateien in dem gerade gewählten Verzeichnis bzw. die mit *List Files of Type* eingeschränkten Dateien angezeigt. Wurden alle Eintragungen vorgenommen, beendet man mit *Assign* den Dialog, damit die Verknüpfung zwischen Bibliotheksnamen und Unterverzeichnis eingetragen werden kann.

Wird der Name eines noch nicht existierenden Unterverzeichnisses eingetragen, erfolgt eine Rückmeldung vom System und man muß bestätigen, ob dieses neue Verzeichnis angelegt werden darf. Der Bibliotheksname kann nach den üblichen Namenskonventionen des SAS-Systems frei gewählt werden. Er ist für die aktuelle Sitzung mit dem Unterverzeichnis verbunden, wird aber am Ende der Sitzung gelöscht, so daß die Verbindung in einer späteren Sitzung auch unter einem anderen Bibliotheksnamen eingerichtet werden kann.

Nachdem die Verbindung zur Bibliothek über das New Library-Fenster hergestellt wurde, kann der Datenschritt angepaßt und ausgeführt werden. Der Dateiname `adressen` wird um den Bibliotheksnamen `biblio` erweitert:

```
DATA biblio.adressen;
     INFILE 'A:\kurs\adressen.dat';
     INPUT name $ vorname $ wohnort $:12. alter;
RUN;
```

Wird dieser Datenschritt ausgeführt, erscheint im Protokollfenster die Meldung

```
NOTE: The data set BIBLIO.ADRESSEN has 5 observations and 4
      variables.
```

Damit wird bestätigt, daß nun anstelle der temporären Datei `WORK.ADRESSEN` eine permanente Datei `BIBLIO.ADRESSEN` angelegt wurde. Der Name der Datei ist zweiteilig: Der erste Teil, `BIBLIO`, weist auf die Bibliothek hin, der zweite Teil, `ADRESSEN`, ist der eigentliche Dateiname, unter dem die Datei auf Betriebssystemebene abgespeichert und geführt wird. Man kann sich alle Dateien in einer Bibliothek als große Familie vorstellen, die einen gemeinsamen Namen hat, den Bibliotheksnamen. Will man mit einem bestimmten Familienmitglied sprechen, muß man aber dessen Namen kennen, den Dateinamen. In der Bibliothek `BIBLIO` gibt es bislang nur ein Mitglied: die Datei `ADRESSEN`.

Nachdem die Datei permanent abgelegt wurde, kann die Sitzung beendet werden. Mit dem Windows-Dateimananger, dem Windows 95-Explorer oder dem DOS-Kommando `Dir` kann man sich jetzt davon überzeugen, daß die Datei tatsächlich im Unterverzeichnis `C:\kurs` vorhanden ist. Wie in Abbildung 5.2 zu erkennen ist, wurde die permanente Datei unter dem Namen `adressen.sd2` abgespeichert. Der Bibliotheksname `biblio` ist auf Betriebssystemebene nicht zu erkennen. Er ist nur ein logischer Name innerhalb des SAS-Systems. Die Dateierweiterung `sd2` dürfen Sie nicht verändern, da dies für das System der einzige Anhaltspunkt ist, daß es sich bei der Datei um eine systemeigene Datei handelt. (Unter SAS für UNIX erhalten die SAS-Dateien die Endung *.SSD01. Kataloge haben die Endung *.SC2 bzw. *.SCT01.)

Abb. 5.2: SAS-*Datei, mit dem Windows-Dateimanager angezeigt*

Wenn in der nächsten SAS-Sitzung wieder auf die permanente Firmenkartei zugegriffen werden soll, muß zunächst die Bibliothek über das Libraries-Fenster referenziert werden. Der Datenschritt ist nicht notwendig, da die Datei bereits existiert. Stattdessen wird sofort ein Prozedurschritt zur Darstellung der Werte im Ausgabefenster ausgeführt.

```
/*--- KA05-11.SAS ---*/
PROC PRINT DATA=biblio.adressen;
RUN;
```

Der Bibliotheksname wird außerhalb der SAS-Umgebung nicht mit der Datei gespeichert. In der zweiten Sitzung muß daher nicht notwendigerweise der gleiche Bibliotheksname verwendet werden. Stattdessen kann über das Libraries-Fenster ein anderer Bibliotheksname für das Unterverzeichnis `C:\kurs` vereinbart werden, etwa `firma` statt `biblio`. Der analoge Prozedurschritt zur Anzeige der Datenwerte der permanenten Adreßdatei erhält damit die folgende Form:

```
PROC PRINT DATA=firmen.adressen;
RUN;
```

Da man mit dem SAS-System auch im Batchbetrieb arbeiten kann, gibt es natürlich auch eine Anweisung, über die die Verbindung zwischen dem Bibliotheksnamen und dem Unterverzeichnis hergestellt wird: LIBNAME. Diese globale Anweisung fügt man in das Programm vor den Datenschritt ein, damit zuerst das Unterverzeichnis `C:\kurs` mit dem Bibliotheksnamen referenziert wird und danach der Daten- oder Prozedurschritt erfolgen kann.

```
LIBNAME firmen 'C:\kurs';
PROC PRINT DATA=firmen.adressen;
RUN;
```

Nach dem Schlüsselwort LIBNAME wird zuerst der Bibliotheksname angegeben und danach in Anführungszeichen das Unterverzeichnis. Wird die LIBNAME-Anweisung ausgeführt, erscheint im Protokollfenster eine entsprechende Vollzugsmeldung:

```
1    LIBNAME firmen 'C:\kurs';
NOTE: Libref FIRMEN was successfully assigned as follows:
      Engine:        V612
      Physical Name: C:\kurs
```

Der Anwender erfährt, daß der Bibliotheksname (auch `Libref` genannt) erfolgreich mit dem physikalischen Verzeichnis `C:\kurs` verknüpft werden konnte. Die `Engine` gibt dem Benutzer Auskunft über den Typ der Dateien und Kataloge innerhalb der Bibliothek, hier `V612`. Dies wird wichtig, wenn auch auf Dateien

aus älteren SAS-Versionen oder anderen Anwendungsprogrammen zugegriffen werden soll (vgl. Kapitel 11).

Während man über das Libraries-Fenster auch neue Unterverzeichnisse anlegen kann, geht dies bei der Anweisung LIBNAME nicht. Ist das genannte Unterverzeichnis nicht vorhanden (oder wurde einfach ein Schreibfehler in der Anweisung übersehen), erscheint nach der Ausführung der Anweisung folgender Hinweis im Protokollfenster:

```
2    LIBNAME biblio 'C:\kusr';
NOTE: Library BIBLIO does not exist.
```

Da die Bibliothek, genauer das Unterverzeichnis `C:\kusr`, nicht existiert, kann keine Zuweisung zum Referenznamen erfolgen, was wiederum fatale Folgen haben kann, wenn in den anschließenden Programmschritten auf diese Bibliothek zugegriffen wird: Fehlermeldungen über Fehlermeldungen, alle eine Folge der nicht ausgeführten LIBNAME-Anweisung.

Wenn Ihr Programm LIBNAME-Anweisungen enthält, kontrollieren Sie im Protokollfenster sehr sorgfältig, ob alle Zuweisungen vollzogen wurden, bevor Sie weitere Prozedur- und Datenschritte ausführen.

Das Anlegen permanenter SAS-Dateien schafft die Voraussetzung für eine effektive Arbeit mit dem System. Dateien können auf diesem Wege anderen Benutzern zugänglich gemacht werden, ohne daß diese den Einlese- und Bearbeitungsvorgang selbst durchführen müssen. Komplexe Analysen müssen nicht innerhalb einer Sitzung abgeschlossen werden, sondern können sich über mehrere Tage, Wochen etc. ausdehnen.

Die beiden Voraussetzungen für die Arbeit mit permanenten Dateien sind die Herstellung einer Verbindung zwischen Bibliotheksnamen und dem Unterverzeichnis über das Libraries-Fenster oder die LIBNAME-Anweisung

```
LIBNAME referenzname 'Pfadname';
```

und die Verwendung zweiteiliger Dateinamen. Der erste Teil besteht aus dem Bibliotheksnamen, der zweite Teil aus dem eigentlichen Dateinamen, und die beiden Bestandteile werden durch einen Punkt verbunden.

5.5 Neugier – Informationen über Dateien erfragen

Mit der Prozedur PRINT wurden die in einer Datei enthaltenen Werte im Ausgabefenster angezeigt. Bei größeren Dateien mit Hunderten von Variablen und

tausend oder mehr Beobachtungen kann man mit dieser Art der Darstellung leicht den Überblick verlieren, und eine Kontrolle der Eintragungen ist kaum möglich. Mit dem Libraries-Fenster  können dagegen gezielte Dateiinformationen interaktiv abgefragt werden.

5.5.1 Das Libraries-Fenster

Über das Libraries-Fenster können, wie bereits beschrieben, Bibliotheksnamen mit Unterverzeichnissen verknüpft werden, aber das Fenster zeigt auch die im Laufe der aktuellen Sitzung referenzierten und noch im Zugriff befindlichen Verzeichnisse an.

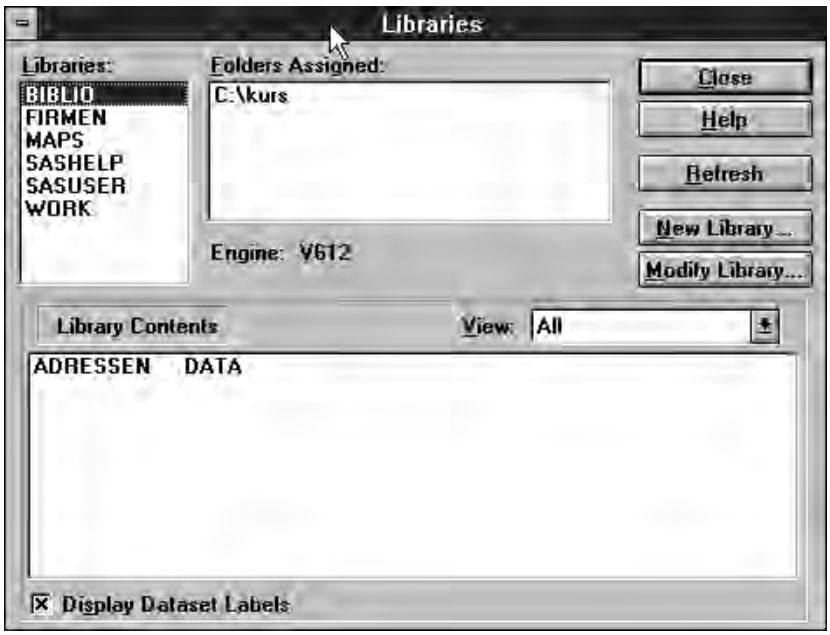


Abb. 5.3: Das Libraries-Fenster

In Abbildung 5.3 sind im linken Fenster die alphabetisch sortierten Bibliotheken zu sehen. Die erste Bibliothek, im vorliegenden Fall `BIBLIO`, wird nach dem Aufruf standardmäßig selektiert (sichtbar durch die farbige bzw. graue Unterlegung). In den anderen Fenstern werden nähere Informationen zu dieser ausgewählten Bibliothek angezeigt. In der Mitte (*Folders Assigned*) erscheint das Unterverzeichnis, das mit dem Bibliotheksnamen verknüpft ist; in der unteren Hälfte (*Library Contents*) die Dateien und Kataloge, die sich in der Bibliothek befinden.

In der Bibliothek BIBLIO befindet sich nur die Datei ADRESSEN. Die Bibliothek FIRMEN verknüpft ebenfalls das Verzeichnis C:\kurs. An letzter Stelle befindet sich die Arbeitsbibliothek WORK, in der die temporären Dateien ADRESSEN, ADRES-NEU und ZAHLEN enthalten sind, sofern alle Übungsaufgaben innerhalb einer Sitzung ausgeführt wurden. Daneben gibt es weitere Bibliotheken, die beim Systemstart automatisch angelegt wurden: Die Bibliothek MAPS[4] enthält Dateien mit Koordinateninformationen für Landkarten, SASHELP beinhaltet Kataloge und Dateien für das SAS-System und SASUSER Kataloge und Dateien mit Windows-Einstellungen und Druckeroptionen.

Die Schaltfläche *Modify Library ...* gestattet die Veränderung bestehender Bibliotheken, etwa die Hinzunahme weiterer Unterverzeichnisse, die Umbenennung von Bibliotheken (nicht Verzeichnissen) oder die Aufhebung der Verbindung zwischen Bibliotheksnamen und Unterverzeichnis.

Um Informationen zu einer speziellen Datei zu erhalten, wählt man zuerst die Bibliothek aus, in der sich die Datei befindet, und klickt die gewünschte Datei einmal mit der linken Maustaste an, so daß die Zeile farbig unterlegt wird. Über die rechte Maustaste zeigt sich nun ein Popup-Menü, über das sich weitere Fenster öffnen lassen:

➤ Mit *Open table view* wird der Dateiinhalt, d. h. alle Variablen und Beobachtungen der Datei, in Form einer Tabelle ausgegeben (Abbildung 5.4).

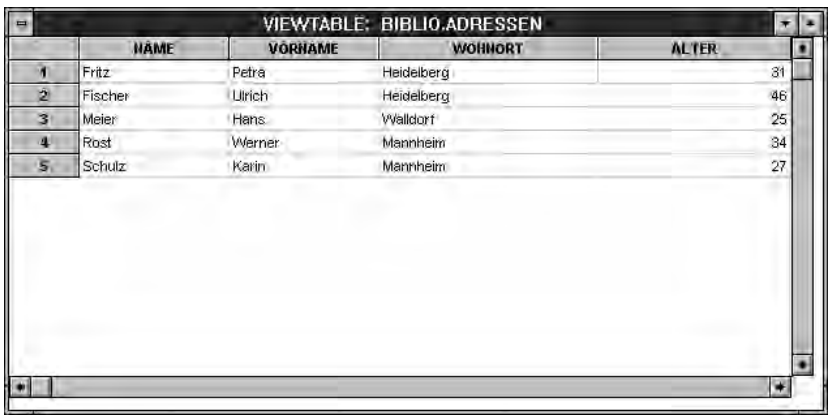


Abb. 5.4: Das Viewtable-Fenster - Tabellenform

[4] MAPS wird bei Ihnen nur angezeigt, wenn das Modul SAS/GRAPH lizensiert und auch installiert ist.

In den Vorgängerversionen 6.08 bis 6.11 wurde an dieser Stelle die Prozedur FSVIEW aufgerufen, in der aktuellen Version 6.12 dagegen erscheint die Tabelle in einem mehr Windows-gemäßen Gewand mit u. a. fett-gedruckten und unterlegten Spaltenbeschriftungen und einem anderen Schrifttyp. Für jede Beobachtung wird eine Zeile in der Tabelle zur Verfügung gestellt, in den Spalten werden die Variablen eingetragen.

➤ Mit *Open form view* wird jede Beobachtung auf einer eigenen Seite dargestellt (Abbildung 5.5). Diese Variante ist bei Dateien mit sehr vielen Variablen geeigneter, da die Darstellung, die man mit der Tabellensicht erhalten würde, sehr breit und dadurch unhandlich werden würde.

In dieser Ansicht werden die Variablen vertikal angeordnet. Über die Richtungstasten kann man sich nach vorne und hinten in der Datei bewegen bzw. an den Beginn und an das Ende der Datei springen.

Abb. 5.5: Das Viewtable-Fenster - Jede Beobachtung für sich

➤ Das Variablenfenster wird im Popup-Menü mit *VAR window* geöffnet. Es zeigt die Variablen mitsamt ihren Eigenschaften auf: Name, Typ, Länge, sowie Einlese- und Ausgabeformat (Abbildung 5.6). Die Spalte links vom Variablennamen ist zur Selektion gedacht. Hier kann man `R` (für *Rename*) eingeben, um die Variable umzubenennen, und `C` (für *Cancel*), um den Umbenennungsvorgang abzubrechen.

➤ *Rename* erlaubt die Umbenennung einer Datei. `adressen` könnte beispielsweise in `firma` umbenannt werden.

➤ *Delete* schließlich gestattet dem Anwender, eine bestehende SAS-Datei, sei sie temporär oder permanent angelegt, zu löschen.

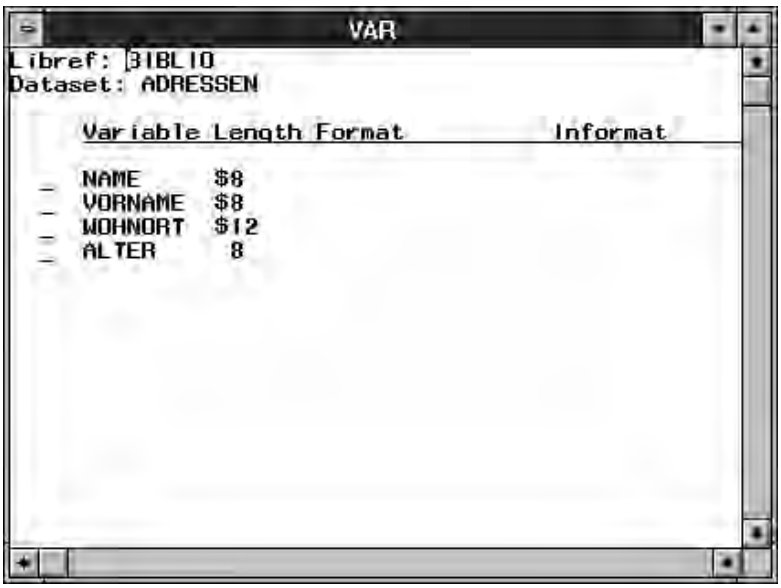


Abb. 5.6: Das Variablenfenster

Für den ausschließlich mit dem Display Manager System arbeitenden Anwender stellt das Libraries-Fenster den idealen Weg dar, um einen schnellen Überblick über Dateistrukturen und -inhalte zu gewinnen. Für den im Batchmodus arbeitenden Anwender und für den Anwender, der die Dateistrukturen für die Dokumentation „schwarz auf weiß“ benötigt, stellt die Prozedur CONTENTS die Alternative dar. Bibliotheken verändern kann man im Batchmodus allerdings nur über die Anweisung LIBNAME, Dateistrukturen und -eigenschaften über die Prozedur DATASETS oder im Datenschritt (vgl. Kapitel 7).

5.5.2 Die Prozedur CONTENTS

Die Prozedur CONTENTS liefert einen Überblick über die Dateien in einer Bibliothek und gibt neben der Dateibeschreibung zusätzliche Informationen zu einzelnen Dateien aus. Der folgende Aufruf liefert gezielte Informationen zu der Adreßkartei der Beispielfirma, die in einem der letzten Abschnitte als permanente Datei in der Bibliothek `biblio` abgelegt wurde.

```
/*--- KA05-12.SAS ---*/
LIBNAME biblio 'C:\kurs';
PROC CONTENTS DATA=biblio.adressen;
RUN;
```

```
CONTENTS PROCEDURE

Data Set Name: BIBLIO.ADRESSEN      Observations:          5
Member Type:   DATA                 Variables:             4
Engine:        V612                 Indexes:               0
Created:       16:55 Sun,Jun 8,1997 Observation Length:   36
Last Modified: 16:55 Sun,Jun 8,1997 Deleted Observations:  0
Protection:                         Compressed:           NO
Data Set Type:                      Sorted:               NO
Label:

-----Engine/Host Dependent Information-----

Data Set Page Size:       8192
Number of Data Set Pages: 1
File Format:              607
First Data Page:          1
Max Obs per Page:         226
Obs in First Data Page:   5

-----Alphabetic List of Variables and Attributes-----

#    Variable    Type    Len    Pos
-----------------------------------
4    ALTER       Num       8     28
1    NAME        Char      8      0
2    VORNAME     Char      8      8
3    WOHNORT     Char     12     16
```

Im ersten Abschnitt der Darstellung wird neben dem Namen der Datei als wichtige Informationen das Datum, an dem die Datei angelegt wurde (`Created`), und das Datum, an dem sie zuletzt verändert wurde (`Last Modified`), ausgegeben. Rechts daneben kann man die Zahl der Beobachtungen (`Observations`) und Variablen (`Variables`) ablesen sowie Informationen darüber, ob die Datei in komprimiertem Zustand (`Compressed`) oder sortiert vorliegt.

Der zweite Abschnitt enthält Betriebssystem-spezifische Informationen (`Engine/Host Dependent Information`), die für den Anfänger nicht von Bedeutung sind.

Im dritten Abschnitt schließlich werden alle Variablen der Datei in alphabetischer Reihenfolge aufgeführt. Man kann erkennen, an welcher Position innerhalb der Datei die Variable steht (#) und ob es sich bei der Variablen um eine numerische (`Num`) oder um eine Textvariable (`Char`) handelt.

Mit der Option POSITION der PROC CONTENTS-Anweisung werden die Variablen zusätzlich auch in ihrer Originalreihenfolge, also in der Reihenfolge, in der sie in der Datei stehen, ausgegeben.

```
PROC CONTENTS DATA=biblio.adressen POSITION;
RUN;

-----Variables Ordered by Position-----

#    Variable    Type    Len    Pos
-----------------------------------
1    NAME        Char      8      0
2    VORNAME     Char      8      8
3    WOHNORT     Char     12     16
4    ALTER       Num       8     28
```

Befinden sich mehrere Dateien in der Bibliothek, erstellt man mit der Option DIRECTORY eine Liste mit den Namen aller vorhandenen Dateien. Mit dem allgemeinen Dateinamen BIBLIOTHEKSNAME._ALL_ können für alle Dateien einer Bibliothek die Detailinformationen angefordert werden.

```
PROC CONTENTS DATA=biblio._ALL_ DIRECTORY;
RUN;
```

Für die Detailinformationen aller im temporären Arbeitsbereich abgelegten Dateien läßt man den Bibliotheksnamen weg: `DATA=_ALL_`. Die Syntax der Prozedur CONTENTS hat die allgemeine Form:

```
PROC CONTENTS DATA=Bibliotheksname.Datei|_ALL_
              <POSITION>
              <DIRECTORY>;
```

5.6 Aufgaben

❶ Legen Sie eine neue Datei an, in der die Namen und Adressen Ihrer zehn nächsten Angehörigen oder Freunde eingetragen sind. Verwenden Sie die verschiedenen Steuerungsvarianten: listen-, spalten- und formatgesteuert.

Kontrollieren Sie im Libraries-Fenster die korrekte Übernahme Ihrer Angaben. Speichern Sie den oder die Datenschritte in einem Programm ab.

❷ Variieren Sie den Datenschritt so, daß die Datei permanent abgelegt wird. Speichern Sie auch dieses Programm. Verlassen Sie nun die Sitzung und machen Sie sich auf die Suche nach der permanenten Datei. Starten Sie die Sitzung erneut und geben Sie eine Liste Ihrer Angehörigen bzw. Freunde im Ausgabefenster aus, ohne den Datenschritt zu wiederholen.

❸ Übernehmen Sie das Beispielprogramm `KA05-08.SAS`. Ändern Sie es so ab, daß die Namen und Beträge spaltengerecht untereinander stehen und lesen Sie die Werte spaltengesteuert ein. Führen Sie das Programm aus.

Nach einer Weile sollten Sie es mit STRG+UNTBR abbrechen und das Ergebnis betrachten.

Und untersuchen Sie auch, was passieren würde, wenn Sie den Zeilenhalter (@@) vergessen würden, oder der von Herrn Meier gezahlte Betrag in der nächsten Zeile steht.

❹ Auf der Diskette befinden sich im Unterverzeichnis AUFGABEN die Ergebnisse eines Experiments von R. A. Fisher, der je 50 Blüten von drei verschiedenen Iris-Arten untersuchte. Er maß Länge und Breite der Kelch- und Blütenblätter. Die Daten sind unter dem Namen `IRIS.DAT` abgelegt.

Schreiben Sie ein Programm, mit dem Sie die Daten in eine permanente SAS-Datei mit Namen `iris` übertragen können.

❺ Auf der Begleitdiskette (gleiches Verzeichnis) befinden sich die permanenten SAS-Dateien `staedte` und `tour97`. Wieviele Beobachtungen haben diese Dateien und wie lauten deren Variablennamen?

❻ Übertragen Sie die Stammdaten der Firma (`stammdat.dat`) von der Diskette (Unterverzeichnis FIRMA) in eine permanente SAS-Datei `stammdat` in Ihrem Übungsverzeichnis. Eine Beschreibung dieser Datei finden Sie im Anhang B.1. Erfassen Sie analog zum Beispielprogramm `KA05-07.SAS` nur die Jahreszahl der Eintrittsdaten.

6

Erste Früchte ernten – Daten in Tabellenform präsentieren

Der Inhalt der im Datenschritt erzeugten SAS-Dateien kann entweder interaktiv über das Libraries-Fenster angezeigt oder, wie in den bisherigen Beispielen demonstriert, mit der Prozedur PRINT im Ausgabefenster in Form einer Tabelle aufgelistet werden. In den folgenden Abschnitten werden zusätzliche Anweisungen und Optionen der Prozedur PRINT sowie die Prozedur SORT vorgestellt, mit denen Sie als Anwender mehr Einfluß auf die Reihenfolge und Form der Datenwerte im Ausgabefenster nehmen können.

6.1 Einfache Tabellen – Die Prozedur PRINT

Zunächst wird die permanente Datei `punkte` erzeugt, die die Ergebnisse einer vierteiligen Einstellungsprüfung enthält, die alle 31 Firmenmitarbeiter absolvieren mußten. Eine ausführliche Beschreibung sowie die komplette Auflistung dieser Daten können Sie dem Anhang B.2 entnehmen. In einem Datenschritt werden die Rohwerte von der Begleitdiskette in die permanente SAS-Datei `biblio.punkte` übertragen. Zur Kontrolle wird die Datei mit der Prozedur PRINT im Ausgabefenster angezeigt.

Der Name PRINT suggeriert, daß die Datenwerte direkt aus*gedruckt* werden. Ein Trugschluß, denn die Tabelle erscheint im Ausgabefenster. Soll sie tatsächlich auf einem Drucker ausgegeben werden, muß im Ausgabefenster die Schaltfläche  angeklickt oder der Menüpunkt *File* → *Print* aktiviert werden.

```
/*--- KA06-01.SAS ---*/
LIBNAME biblio 'C:\kurs';
DATA biblio.punkte;
     INFILE 'A:\firma\punkte.dat';
     INPUT pnr teil1 teil2a teil2b teil3 teil4;
RUN;
PROC PRINT DATA=biblio.punkte;
RUN;
```

Der Prozedurschritt ist sehr klein, er besteht lediglich aus den beiden Anweisungen PROC PRINT und RUN. Damit werden alle Variablen und alle Beobachtungen in der Reihenfolge aufgeführt, in der sie in der Datei eingetragen sind. (Aus Platzgründen werden im folgenden nur Ausschnitte aus den Listen wiedergegeben.)

OBS	PNR	TEIL1	TEIL2A	TEIL2B	TEIL3	TEIL4
1	1001	9	2	9	8	6
2	1002	6	9	6	5	15
3	1003	6	7	7	5	13
4	1004	7	9	7	7	14
5	1005	8	10	8	8	18
6	1006	2	6	7	7	13
7	1007	8	2	10	7	13
8	1008	8	9	9	10	11
9	1009	6	8	6	6	2
..	....	..	.	.	.	..
22	1022	9	7	7	6	11
23	1023	5	5	8	4	8
24	1024	6	6	9	5	6
25	1025	7	8	4	6	17
26	1026	3	4	6	7	16
27	1027	6	2	7	8	13
28	1028	9	10	2	10	9
29	1029	10	8	1	8	12
30	1030	7	6	8	4	6
31	1031	5	8	10	6	10

Zusätzlich zu den Variablen wird eine Extraspalte eingefügt, OBS, die die Beobachtungsnummer anzeigt.

Soll die Reihenfolge der Variablen verändert oder nur ein Teil der Variablen dargestellt werden, muß zusätzlich die Anweisung VAR angegeben werden.

Nach dem Schlüsselwort werden die Variablen aufgezählt, deren Inhalt gezeigt werden soll. Im folgenden Beispiel werden nur die Ergebnisse der ersten drei Tests, `teil1`, `teil2a`, `teil2b` und als letzte Variable die Personalnummer `pnr` angezeigt.

```
/*--- KA06-02.SAS ---*/
PROC PRINT DATA=biblio.punkte;
     VAR teil1 teil2a teil2b pnr;
RUN;
```

OBS	TEIL1	TEIL2A	TEIL2B	PNR
1	9	2	9	1001
2	6	9	6	1002
3	6	7	7	1003
4	7	9	7	1004
5	8	10	8	1005
..	.	.	.	....
27	6	2	7	1027
28	9	10	2	1028
29	10	8	1	1029
30	7	6	8	1030
31	5	8	10	1031

Besteht die Datei aus sehr vielen Variablen, ist es natürlich unkomfortabel, die Variablen einzeln aufzählen zu müssen, wenn nicht alle ausgedruckt werden sollen. Man kann die Liste abkürzen, indem man nur die jeweils erste und letzte Variable eines Bereichs angibt und die beiden Namen durch einen oder zwei Querstriche verbindet.

- Beginnen die Variablennamen mit der gleichen Vorsilbe (Prefix) und enden sie auf eine Zahl (`item1`, `item2`, `item3` ... `item10`), genügt es, die erste und letzte Variable mit einem Querstrich zu verbinden:

  ```
  VAR item1-item10;
  ```

  Das SAS-System ergänzt automatisch alle dazwischenliegenden Variablen. Die interne Reihenfolge der Variablen spielt keine Rolle.

- Enden die Variablen wie im vorliegenden `punkte`-Beispiel nicht ausschließlich auf Zahlen, sondern kommen auch Buchstaben vor (z. B. `teil2a`) oder haben die Variablennamen keinen gemeinsamen Prefix (`alter`, `sex`, `name`), kennzeichnet man solche Variablenbereiche durch zwei Querstriche.

```
VAR teil2a--teil4
VAR alter--punkte
```

Der Variablenbereich bezieht sich dabei auf die interne Reihenfolge der Variablen in der SAS-Datei (vgl. Variablenfenster und Prozedur CONTENTS in Abschnitt 5.5). Alle Variablen, die in der Datei zwischen der erstgenannten und der letztgenannten Variablen stehen, werden von diesem Bereich erfaßt.

Mit dem folgenden Beispielprogramm werden neben der Personalnummer nur die Ergebnisse der letzten vier Prüfungsteile ausgegeben, die in den Variablen `teil2a`, `teil2b`, `teil3` und `teil4` enthalten sind:

```
/*--- KA06-03.SAS ---*/
PROC PRINT DATA=biblio.punkte NOOBS;
     VAR pnr teil2a--teil4;
RUN;
```

Die Option NOOBS der PROC PRINT-Anweisung unterdrückt die Angabe der internen Zählvariablen `OBS`, da diese im vorliegenden Beispiel, in dem die Beobachtungen bereits durch die Personalnummer eindeutig identifiziert sind, überflüssig ist.

PNR	TEIL2A	TEIL2B	TEIL3	TEIL4
1001	2	9	8	6
1002	9	6	5	15
1003	7	7	5	13
1004	9	7	7	14
1005	10	8	8	18
....	.	.	.	..
1027	2	7	8	13
1028	10	2	10	9
1029	8	1	8	12
1030	6	8	4	6
1031	8	10	6	10

Die Beobachtungen wurden in den bisherigen Beispielprogrammen stets in der Reihenfolge ausgegeben, wie sie in der Datei eingetragen wurden. Soll diese Reihenfolge geändert werden, muß die Datei vor der Ausführung der Prozedur PRINT entsprechend sortiert werden. Das Sortieren übernimmt die Prozedur SORT, die im nächsten Abschnitt ausführlicher behandelt wird. Hier wird sie eingeführt, um die Daten in anderer Sortierreihenfolge mit der Prozedur PRINT

auszugeben. Der Prozedur SORT muß neben dem Dateinamen auch das Sortierkriterium übergeben werden, nach dem die Datei sortiert werden soll. Als Kriterium gibt man eine oder mehrere Variablen an. Das Telefonbuch beispielsweise ist zunächst nach dem Ort sortiert, innerhalb des Ortes nach Nachnamen und noch nach Vornamen.

Die Datei punkte aus dem vorangegangenen Beispiel wird nach den erzielten Punkten im letzten Prüfungsabschnitt sortiert. Standardmäßig sortiert das SAS-System aufsteigend, d. h., zuerst kommen die kleinen Werte, danach die großen. Damit die ursprüngliche Reihenfolge nicht überschrieben wird, wird über die Option OUT= eine neue Datei benannt, die das Sortierergebnis enthält.

```
/*--- KA06-04.SAS ---*/
PROC SORT DATA=biblio.punkte OUT=s_punkte;
     BY teil4;
RUN;
PROC PRINT DATA=s_punkte;
     VAR pnr teil4 teil1;
RUN;
```

Die temporäre Datei s_punkte enthält genauso viele Beobachtungen und Variablen wie die Datei biblio.punkte, doch die Beobachtungen sind anders angeordnet. Ebenso gut hätte man natürlich auch eine permanente Datei der sortierten Daten anlegen können. Mit der Anweisung BY übergibt man die Variablen, nach denen sortiert werden soll (in obigem Beispiel ist dies teil4). Der Programmschritt PROC SORT erzeugt keine Ausgabe im Ausgabefenster. Erst der anschließende PROC PRINT-Schritt zeigt das sortierte Ergebnis an:

```
OBS      PNR    TEIL4    TEIL1

  1     1009        2        6
  2     1020        5        6
  3     1001        6        9
  4     1024        6        6
  5     1030        6        7
 ..     ....       ..        .
 27     1026       16        3
 28     1010       17       10
 29     1025       17        7
 30     1005       18        8
 31     1015       19        6
```

Wiederholt man die BY-Anweisung im PROC PRINT-Schritt, so werden getrennte Tabellen für alle vorkommenden Datenwerte der Variablen `teil4` erzeugt.

```
PROC PRINT DATA=s_punkte;
     VAR pnr teil4 teil1;
     BY teil4;
RUN;
```

Da es 15 verschiedene Werte gibt, werden 15 Tabellen mit den Zwischenüberschriften `TEIL4=2`, `TEIL4=5` ... `TEIL4=19` erzeugt, von denen hier nur ein Ausschnitt wiedergegeben wird:

```
.......

TEIL4=5

OBS     PNR    TEIL4    TEIL1
  2    1020      5        6


TEIL4=6

OBS     PNR    TEIL4    TEIL1
  3    1001      6        9
  4    1024      6        6
  5    1030      6        7

.......
```

Im vorliegenden Beispiel macht der Gebrauch der BY-Anweisung mit der Prozedur PRINT natürlich wenig Sinn. Aber wenn z. B. der Familienstand eine Rolle spielen würde, könnte es interessant sein, die Tabellen in Ledige und Verheiratete aufzuteilen. Im nächsten Abschnitt folgt dazu mehr.

Zunächst soll jedoch die letzte Anweisung der Prozedur PRINT vorgestellt werden, die auch für den Anfänger von Bedeutung sein kann: Die Anweisung SUM. Mit der Prozedur PRINT können nicht nur Tabellen erzeugt werden, sondern es lassen sich mit ihr auch numerische Werte einer Variablen aufsummieren, d. h. alle Werte einer Spalte zusammenzählen.

```
/*--- KA06-05.SAS ---*/
PROC PRINT DATA=biblio.punkte;
     VAR pnr;
     SUM teil1 teil4;
RUN;
```

In diesem Prozedurschritt werden die drei Variablen `pnr`, `teil1` und `teil4` angezeigt, die beiden in der SUM-Anweisung genannten Variablen werden zusätzlich aufsummiert. Theoretisch ließen sich auch die Personalnummern summieren, da es sich dabei ebenfalls um reine Zahlenwerte einer numerischen Variablen handelt, aber was für einen Sinn sollte dies machen?

```
OBS      PNR    TEIL1    TEIL4

  1     1001        9        6
  2     1002        6       15
  3     1003        6       13
  4     1004        7       14
  5     1005        8       18
 ..     ....        .       ..
 27     1027        6       13
 28     1028        9        9
 29     1029       10       12
 30     1030        7        6
 31     1031        5       10
                 =====    =====
                  188      368
```

In einer zusätzlichen Zeile am Ende der Tabelle werden die beiden Summen ausgegeben. Die Summe der Punkte der Abschlußveranstaltung ist mehr als doppelt so groß, wie die Summe der Punkte des ersten Teils (was daran liegt, daß beim Abschluß doppelt so viele Punkte erzielt werden konnten).

Rufen Sie die Prozedur PRINT ohne die Option DATA= auf, wird der Inhalt der in der laufenden Sitzung zuletzt erzeugten SAS-Datei angezeigt. Den Namen dieser Datei können Sie über das Options-Fenster in Erfahrung bringen (vgl. Kapitel 4). Sicherer ist es jedoch, wenn Sie bei jedem Prozeduraufruf explizit angeben, welche Datei bearbeitet werden soll[1].

[1] Diese Eigenschaft gilt für (fast) alle SAS-Prozeduren.

Neben den hier vorgestellten Anweisungen gibt es weitere, die Sie selbst mit Hilfe des Handbuchs (SAS Procedures Guide [11]) oder den Beispielprogrammen der Online-Hilfe studieren können (z. B. ID, SUMBY). Für die genannten gilt die folgende allgemeine Prozedursyntax:

```
PROC PRINT <DATA=Datei> <NOOBS>;
     <VAR Variablen-Liste;>
     <SUM Variablen-Liste;>
     <BY Variablen-Liste;>
RUN;
```

6.2 Dateien sortieren – Die Prozedur SORT

Die Prozedur SORT sortiert eine SAS-Datei nach vorgegebenen Kriterien, die mit Hilfe der Anweisung BY formuliert werden. Sie erzeugt selbst keine Ausgabe im Ausgabefenster, sondern verändert entweder die Datei, die sortiert wurde, oder erzeugt eine neue Datei, in die sie das Sortierergebnis einträgt. Zur Kontrolle des Sortiervorgangs können Sie entweder die Prozedur PRINT einsetzen oder im Libraries-Fenster *Open table view* aufrufen. Mit folgendem Programm wird die permanente Datei `biblio.punkte` nach der Variable `teil4` sortiert:

```
/*--- KA06-06.SAS ---*/
PROC SORT DATA=biblio.punkte;
     BY teil4;
RUN;
```

Die Mitarbeiter mit weniger erzielten Punkten werden vor den Mitarbeitern mit vielen Punkten im 4. Abschnitt des Einstellungstests eingereiht. Aber die Reihenfolge kann unter Verwendung der Option DESCENDING der BY-Anweisung umgekehrt werden:

```
/*--- KA06-07.SAS ---*/
PROC SORT DATA=biblio.punkte;
     BY DESCENDING teil4;
RUN;
PROC PRINT DATA=biblio.punkte;
RUN;
```

Als Ergebnis des zweiten Prozedurschritts, PROC PRINT, erhält man im Ausgabefenster:

OBS	PNR	TEIL1	TEIL2A	TEIL2B	TEIL3	TEIL4
1	1015	6	2	6	6	19
2	1005	8	10	8	8	18
3	1010	10	7	9	8	17
4	1025	7	8	4	6	17
5	1026	3	4	6	7	16
.	....	.	.	.	.	..
27	1001	9	2	9	8	6
28	1024	6	6	9	5	6
29	1030	7	6	8	4	6
30	1020	6	1	8	7	5
31	1009	6	8	6	6	2

Mit der Option DESCENDING wurde die Sortierreihenfolge umgedreht, es wird absteigend sortiert: zuerst die guten Ergebnisse, am Ende die schlechten Ergebnisse. Bei numerischen Werten wird nach der Größe sortiert, bei Textvariablen nach dem Alphabet, also A vor B, D vor E etc.

Da mehrere Mitarbeiter gleich viele Punkte erzielen konnten, ist die Sortierreihenfolge nicht eindeutig. Sowohl die dritte als auch die vierte Beobachtung in der oberen Ausgabe erzielte 17 Punkte. Möchte man innerhalb dieser beiden die Reihenfolge festlegen, vereinbart man eine zweite Variable, nach der bei gleicher Ausprägung in der ersten Variablen sortiert wird.

```
PROC SORT DATA=biblio.punkte;
     BY teil4 teil3;
RUN;
```

Die Datei `biblio.punkte` wird nun zunächst nach den Ausprägungen der Variablen `teil4` sortiert und, wenn gleiche Ausprägungen auftreten, noch nach den Ausprägungen der Variablen `teil3`. Sollen die Ausprägungen in absteigender Reihenfolge sortiert werden, muß die Option DESCENDING vor jeder Variablen, nach der absteigend sortiert werden soll, eingefügt werden.

```
PROC SORT DATA=biblio.punkte;
     BY DESCENDING teil4 DESCENDING teil3;
RUN;
```

In den obigen Beispielen wurde die Datei `biblio.punkte` sortiert und anschließend mit der Prozedur PRINT ausgegeben. Das bedeutet aber, daß die ursprüng-

liche Reihenfolge der Eintragungen verlorengegangen ist, weil sie von dem Sortiervorgang überschrieben wurde. Die Ausgangsdatei wurde verändert. Dies ist aber nicht immer wünschenswert oder zulässig, etwa weil man keine Schreibrechte für die Datei hat. In solch einem Fall würde etwa die folgende Fehlermeldung im Protokollfenster erscheinen beim Versuch, das obige Programm auszuführen:

```
ERROR: User does not have appropriate authorization level
       for file BIBLIO.PUNKTE.DATA.
NOTE: The SAS System stopped processing this step because of
      errors.
```

Der Anwender (`User`) hat nicht genügend Rechte zur Veränderung der Datei `BIBLIO.PUNKTE.DATA`, daher wird der Prozedurschritt abgebrochen.

Damit die Datei dennoch sortiert wird, setzt man die Option OUT= ein. Mit dieser gibt man den Namen einer neuen Datei an, die in dem Prozedurschritt angelegt wird und das Ergebnis des Sortiervorgangs enthält:

```
/*--- KA06-08.SAS ---*/
PROC SORT DATA=biblio.punkte OUT=punkte;
     BY teil1;
RUN;
```

Nach Ausführung dieses Prozedurschritts erscheint im Protokollfenster der Hinweis:

```
NOTE: The data set WORK.PUNKTE has 20 observations and 6
      variables.
```

Es wurde eine neue, temporäre Datei `WORK.PUNKTE` angelegt, die nach den Angaben der Variable `teil1` sortiert ist. Die Ausgangsdatei `biblio.punkte` dagegen liegt in ihrer Originalreihenfolge vor. Seien Sie aber dennoch vorsichtig: Existiert die temporäre Datei `WORK.PUNKTE` vor dem Sortiervorgang, gibt Ihnen das System keinen Hinweis aus, sondern überschreibt, wie im Datenschritt ja auch, die bereits existierende Datei durch die neue.

Die Verwendung der Option OUT= hat noch andere Vorteile: Tritt beim Sortieren der Datei ein Speicherplatzproblem auf (z. B. voller oder defekter Datenträger) bleibt die Ausgangsdatei unberührt. Ohne die Option OUT= kann die Datei jedoch beschädigt oder sogar ganz zerstört werden.

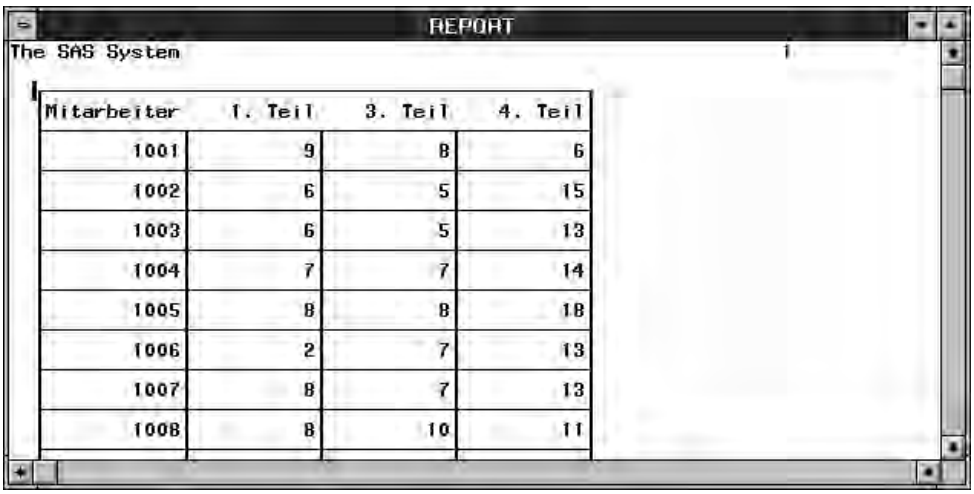
Zum Schluß wird die Syntax der Prozedur SORT zusammenhängend dargestellt:

```
PROC SORT <DATA=Datei> <OUT=Datei>;
     BY <DESCENDING> Variablen-Liste;
RUN;
```

Im weiteren Verlauf werden Sie zusätzliche Gestaltungsmöglichkeiten für diese Tabellen kennenlernen: Titel, Fußzeilen, Bezeichnungen für die Variablen und ihre Werte (Labels und Formate).

Mit der Prozedur REPORT können Dateien ebenfalls in Tabellenform dargestellt werden. Im Unterschied zu den PRINT-Tabellen werden die einzelnen Datenfelder und die gesamte Tabelle umrandet. Der Aufruf erfolgt über das Modul SAS/ASSIST oder mit einem Prozedurschritt REPORT Schritt und der Option PROMPT und verläuft menügesteuert. Das heißt, das SAS-System erfragt vom Anwender die notwendigen Eingaben, ohne daß dieser ein größeres SAS-Programm schreiben muß. Der Prozeduraufruf mit der Option PROMPT sieht wie folgt aus:

```
PROC REPORT DATA=biblio.punkte PROMPT;
RUN;
```



Mitarbeiter	1. Teil	3. Teil	4. Teil
1001	9	8	6
1002	6	5	15
1003	6	5	13
1004	7	7	14
1005	8	8	18
1006	2	7	13
1007	8	7	13
1008	8	10	11

Abb. 6.1: Prozedur REPORT

Nach dem Aufruf der Prozedur wird der Anwender zunächst gefragt, ob die Zahl der Beobachtungen begrenzt werden soll. Für das Beispiel wurde *no limit* gewählt. Anschließend werden im *Data Columns*-Fenster die Variablen bestimmt: `pnr`, `teil1`, `teil3` und `teil4`.

Diesen Variablen müssen nun im *Definition*-Fenster Rollen zugeteilt werden. Im vorliegenden Beispiel wurden alle vier Variablen als *DISPLAY*-Variablen gekennzeichnet, womit ihre Werte einfach nur angezeigt werden. Außerdem legt man hier die Spaltenbreite (*Width*: `11` für die Variable `pnr`, `9` für die übrigen) und die Spaltenüberschrift (*Header*: `Mitarbeiter`, `1. Teil` usw.) fest.

Über *Edit* → *Report Options* wird mit *Box* eine Umrandung erzeugt. Abbildung 6.1 zeigt die mit der Prozedur REPORT erstellte Liste der Datei `biblio.punkte`.

Der Menüpunkt *Locals* → *List REPORT statements* zeigt die Anweisungen der Prozedur an. Speichert man diese über das *File*-Menü ab, kann die Prozedur REPORT zu einem späteren Zeitpunkt ohne die Option PROMPT aufgerufen werden. Für das Beispiel sehen die Anweisungen wie folgt aus:

```
/*--- KA06-09.SAS ---*/
PROC REPORT DATA=BIBLIO.PUNKTE LS=64  PS=40  SPLIT="/"
            NOCENTER BOX ;
     COLUMN  PNR TEIL1 TEIL3 TEIL4;
     DEFINE  PNR / DISPLAY FORMAT= BEST9. WIDTH=11
                   SPACING=2 RIGHT "Mitarbeiter" ;
     DEFINE  TEIL1 / DISPLAY FORMAT= BEST9. WIDTH=9
                   SPACING=2 RIGHT "1. Teil" ;
     DEFINE  TEIL3 / DISPLAY FORMAT= BEST9. WIDTH=9
                   SPACING=2 RIGHT "3. Teil" ;
     DEFINE  TEIL4 / DISPLAY FORMAT= BEST9. WIDTH=9
                   SPACING=2 RIGHT "4. Teil" ;
RUN;
```

Zum weiteren Studium dieser Prozedur wird auf die Online-Hilfe und den ausführlichen REPORT-Guide [7] verwiesen.

6.3 Aufgaben

❶ Schreiben Sie ein Programm, in dem die temporäre, sortierte SAS-Datei `punkte` und die permanente, unsortierte Datei `biblio.punkte` im Ausgabefenster angezeigt werden.

❷ Am Ende von Kapitel 5 haben Sie die Datei `Iris` angelegt. Sortieren Sie diese Datei nun nach der Länge der Kelchblätter und geben Sie die sortierte Datei aus. Beschränken Sie sich auf die Iris-Art und Länge und Breite der Kelchblätter.

❸ Bestimmen Sie die jeweils drei größten Städte pro Bundesland. Sortieren Sie dazu die SAS-Datei `staedte` nach Bundesland und Einwohnerzahl. (Achten Sie darauf, daß die SAS-Datei auf der Begleitdiskette unverändert bleibt.)

❹ Erstellen Sie eine Ergebnisliste der Teilnehmer der Tour de France 1997, aus der Sie die besten Bergfahrer ablesen können.

Die SAS-Datei `tour97` befindet sich auf der Begleitdiskette im Unterverzeichnis AUFGABEN.

7

Alles ist im Fluß – Dateien bearbeiten

In Kapitel 5 wurden die prinzipiellen Varianten vorgestellt, wie SAS-Dateien im Datenschritt neu angelegt werden können und wie man diese Dateien permanent abspeichern kann. Mit der Prozedur SORT tauchte im vergangenen Kapitel eine weitere Möglichkeit auf, um neue Dateien zu erzeugen. Aber gleichgültig, auf welche Art und Weise eine Datei angelegt wurde, kommt irgendwann der Punkt, an dem die bestehende Datei verändert werden soll, etwa weil einzelne Datenwerte falsch eingetragen wurden, neue Variablen aus bestehenden gebildet werden sollen oder nur eine Teilmenge der Beobachtungen weiterverarbeitet und analysiert werden soll. Neben diesen Veränderungsmöglichkeiten erfahren Sie in diesem Kapitel, wie das SAS-System Datumsvariablen behandelt, wie Dateien verknüpft werden können und wie man für Variablen und deren Werte aussagekräftigere Bezeichnungen vergibt.

7.1 Zugriff auf Bestehendes – DATA= und SET

Zur Darstellung der Werte einer Datei oder zur Ausführung von Analysen genügt es, beim Prozeduraufruf in der Anweisung PROC ... über die Option DATA= den Namen der vorhandenen Datei festzulegen.

```
PROC PRINT DATA=biblio.adressen;
RUN;
```

Mit diesem Prozedurschritt werden die Variablenwerte im Ausgabefenster angezeigt. Die Datei `biblio.adressen` wird nicht verändert. Auch wenn einzelne Variablen für die Darstellung ausgewählt wurden, befinden sich weiterhin alle Variablen in der Datei. Prozeduren verarbeiten die Werte einer Datei, ohne diese zu verändern[1].

Eine Veränderung muß folglich an anderer Stelle erfolgen. Da die SAS-Programme aber nur aus Datenschritten, Prozedurschritten und globalen Anweisungen bestehen (vgl. Abschnitt 3.1), muß die Veränderung von Dateien im Datenschritt durchgeführt werden. Entsprechend dem generellen Prinzip beginnt dieser Datenschritt mit einer DATA-Anweisung. Dem Schlüsselwort DATA folgt der Name der neuen Datei (vgl. Kapitel 5). Anschließend teilt man dem System über die SET-Anweisung den Namen der bestehenden Datei mit, deren Werte verändert werden sollen. Zum Schluß folgt noch die RUN-Anweisung zum Beenden des Datenschritts. Fertig! Es wird keine INPUT-Anweisung benötigt, da die Variablen für die existierende Datei bereits deklariert wurden, und es sind keine Rohwerte und damit auch keine Anweisung zu deren Lokalisation notwendig.

```
DATA datei;
     SET datei;
RUN;
```

Je nachdem, welche Dateien in den Anweisungen DATA und SET gewählt werden, wird eine neue Datei angelegt oder eine bestehende überschrieben. Soll die Datei `neu` neu angelegt werden und aus den Beobachtungen der Datei `alt` bestehen, sieht der dazu notwendige Datenschritt wie folgt aus:

```
DATA neu;
     SET alt;
RUN;
```

Mit der SET-Anweisung kann auf diesem Weg z. B. aus einer temporären Datei eine permanente Datei erzeugt werden:

```
DATA biblio.adressen;
     SET adressen;
RUN;
```

[1] Ausnahmen gibt es natürlich auch: Die Prozedur SORT verändert die Reihenfolge der Beobachtungen, wenn nicht über die Option OUT= eine andere Datei für das Sortierergebnis angegeben wurde. Die Prozedur FSEDIT wurde zum Editieren von SAS-Dateien und damit für die Veränderung von Datenwerten und Beobachtungen geschaffen.

Wird in den beiden Anweisungen allerdings der gleiche Dateiname, beispielsweise `test`, angegeben, wird die (alte) Datei `test` durch die (neue) Datei `test` überschrieben.

```
DATA test;
     SET test;
RUN;
```

Das Überschreiben mit sich selbst erscheint zunächst sinnlos, denn dadurch ändert sich die Datei nicht. Aber dieser Datenschritt schafft die Basis für weitere Datenschritt-Anweisungen zur Veränderung der Datei.

Damit mit einem Datenschritt eine Datei angelegt werden kann, genügt es nicht, mit der Anweisung DATA den Namen der Datei festzulegen. Sie müssen auch angeben, auf welche Datei Sie zugreifen möchten: SET, wenn auf eine bereits bestehende Datei zugegriffen wird, INFILE (bzw. LINES) und INPUT, wenn die Datei aus Rohwerten erzeugt wird.

7.2 Konzentration auf Spalten – Variablen erzeugen, löschen, umsortieren

Die für eine Untersuchung erhobenen Datenwerte sind die Grundlage für eine spätere Auswertung. Bevor komplexere Auswertungsverfahren verwendet werden können, müssen die Variablenwerte häufig erst bearbeitet werden oder es müssen neue Variablen angelegt werden. Das SAS-System stellt dazu die *Zuweisung* („=“, engl. *assignment*) zur Verfügung. Mit ihr werden im Datenschritt die Namen der neuen Variablen und die Vorschrift festgelegt, wie die neuen Werte berechnet werden.

Der Personalleiter der Übungsfirma möchte feststellen, wie seine Mitarbeiter beim Einstellungstest abgeschnitten haben. Dazu ermittelt er die Gesamtpunktzahl und die mittlere Punktzahl in den vier Prüfungsabschnitten pro Mitarbeiter. Da sich das Ergebnis des 2. Abschnitts aus zwei Teilen zusammensetzt, berechnet er zunächst die Summe der Einzelergebnisse: `teil2a+teil2b`. Diese Summe wird als Variable `teil2` eingefügt. In die Berechnung der Gesamtpunktzahl gehen die Punkte des 2. und 4. Prüfungsabschnitts mit einem Faktor 1/2 ein[2]:

```
teil1+teil2/2+teil3+teil4/2
```

[2] In diesen beiden Abschnitten konnten maximal 20 Punkte erreicht werden, in den anderen maximal 10. Mit den Faktoren 0.5 erhalten alle vier Abschnitte gleiches Gewicht.

Symbol	Funktion	Beispiel und Ergebnis für a=4 und b=2	
+	Addition	c=a+b;	6
-	Subtraktion	c=a-b;	2
*	Multiplikation	c=a*b;	8
/	Division	c=a/b;	2
**	Potenzierung	c=a**b;	16

Tab. 7.1: Arithmetische Operatoren

Für die Berechnung der durchschnittlichen Punktzahl dividiert man diese Summe durch die Anzahl der Summanden: `summe/4`. Diese Berechnungen führt man für jeden Mitarbeiter durch.

Die Umsetzung in eine Zuweisung für den Datenschritt erfordert nun nicht viele Veränderungen: Zuerst vergibt man einen Namen für die neue Variable, etwa `summe` oder `mittel`, danach folgt ein Gleichheitszeichen (=), das die Zuweisung symbolisiert, und anschließend kommt die obige Verarbeitungsvorschrift. Der komplette Datenschritt hat schließlich die folgenden Form:

```
/*--- KA07-01.SAS ---*/
DATA biblio.punkte2;
     SET biblio.punkte;
     teil2=teil2a+teil2b;
     summe=teil1+teil2/2+teil3+teil4/2;
     mittel=summe/4;
RUN;
```

Der Variablentyp wird bei der Zuweisung implizit bestimmt. Im Unterschied zu einem Datenschritt für die Erzeugung einer SAS-Datei aus Rohwerten, bei dem die Textvariablen mit dem $-Zeichen gekennzeichnet werden müssen, ist dies bei der Zuweisung nicht notwendig. Hier entscheidet die Verarbeitungsvorschrift, ob es sich bei der neuen Variablen um eine numerische oder eine Textvariable handelt. Für die Grundrechenarten Addition, Subtraktion, Multiplikation und Division, sowie für das Potenzieren genügt es, die Zeichen `+`, `-`, `*`, `/` und `**` einzusetzen (vgl. Tabelle 7.1).

Im idealen Fall konnte ein Mitarbeiter 40 Punkte (bzw. 60 Punkte ohne die Gewichtung) erzielen. Um die Differenz zwischen diesem Optimum und der tatsächlich erreichten Punktzahl zu ermitteln, muß in die Datei eine Variable eingefügt werden, die konstant den Wert 40 hat.

```
/*--- KA07-02.SAS ---*/
DATA biblio.punkte2;
     SET biblio.punkte2;
     optimum=40;
     diff=optimum-summe;
     firma='Standort Deutschland';
RUN;
```

Für die Zuweisung des Textes `Standort Deutschland` zu der Textvariablen `firma` muß der Text in Anführungszeichen eingeschlossen werden.

Neben diesen arithmetischen Operatoren und der Zuweisungsmöglichkeit von konstanten Werten stellt das SAS-System eine Fülle von Funktionen zur Verfügung, mit denen man komplexere Berechnungen, wie beispielsweise eine Quadratwurzel oder den Wert der Exponentialfunktion an einer bestimmten Stelle, berechnen kann.

7.2.1 Funktionen

SAS-Funktionen sind Routinen, die im Datenschritt eingesetzt werden. Je nach Definition übergibt man ein oder mehrere Argumente an die Funktion und erhält einen Rückgabewert. Soll z. B. die Quadratwurzel berechnet werden, übergibt man den (konstanten) Wert oder die Variable, aus deren Werte die Wurzel gezogen werden soll, an die Funktion und erhält den Wert bzw. die Variablenwerte.

Der Aufruf einer Funktion erfolgt durch Angabe ihres Namens und Übergabe des oder der Argumente in Klammern. Werden mehrere Argumente an die Funktion übergeben, trennt man diese innerhalb der Klammer durch Kommata ab. Eine Variablenliste kann mit OF ... aufgeführt werden. Funktionen erlauben eine Berechnung auch im Fall einzelner fehlender Werte (siehe unten).

```
Funktion();
Funktion(a);
Funktion(a,b);
Funktion(a,b,c);
Funktion(OF a,b,c);
Funktion(OF a--z);
Funktion(OF x1-x10);
```

Hat eine Funktion per Definition kein Argument, muß das Klammernpaar dennoch angegeben werden, wie im Beispiel der Funktion `Today()` zur Ermittlung

des aktuellen Datums. Bei der Aufzählung einer Variablenliste kann man alle gewünschten Variablen aufzählen (`SUM(OF teil1,teil2)`) oder einen Variablenbereich (analog zu der Variablenliste in der VAR-Anweisung der Prozedur PRINT, vgl. Abschnitt 6.1) angeben. Mit `SUM(OF teil1--teil4)` werden die Variablen aufsummiert, die in der Datei zwischen `teil1` und `teil4` positioniert sind: `teil1`, `teil2a`, `teil2b`, `teil3`, `teil4`. Mit `SUM(OF item1-item3)` dagegen werden die Werte der Variablen aufsummiert, deren Namen sich aus dem Prefix `item` und einer laufenden Nummer zusammensetzen: `item1`, `item2` und `item3`.

Das Ergebnis einer Funktion, der Rückgabewert, kann numerisch oder eine Zeichenkette sein. Eine vollständige Liste der Funktionen mit Beispielen für ihren Aufruf und einer ausführlichen Erläuterung ihrer Funktionalität findet man im SAS Language Reference Guide [9] oder in der Online-Hilfe (*Help → Extended Help → SAS Language → SAS Functions*). Im folgenden werden ausgewählte Funktionen aus verschiedenen Kategorien vorgestellt (siehe auch Tabelle 7.2).

Zu den *Arithmetischen Funktionen* gehören die Berechnung des Absolutbetrags einer Zahl, des Funktionswertes von Exponential- und Logarithmusfunktion und die Quadratwurzel. Mit den *Stichprobenfunktionen* können statistische Kennwerte wie Summe, Mittelwert, größter und kleinster Wert, Standardabweichung und Varianz berechnet werden.

Diese statistischen Kennwerte können auch durch die Umsetzung der entsprechenden Formel berechnet werden, so wie es in obigem Beispiel der Personalleiter gemacht hat. Um den Durchschnitt der Einzelergebnisse zu berechnen, hat er nicht die Funktion MEAN verwendet, sondern die Formel für das arithmetische Mittel in eine Zuweisung übersetzt. Um die Varianz zu berechnen, hätte er sich mehr anstrengen müssen: Zuerst hätte er den Mittelwert berechnen müssen, danach die Abweichungen zum Mittelwert, deren Quadrate, die Summe dieser Werte und schließlich noch durch die um 1 verminderte Anzahl der Beobachtungen teilen. In diesem Fall greift man doch lieber auf eine Funktion zu. Und die Funktionen haben noch einen weiteren Vorteil: Sie berücksichtigen fehlende Werte.

Im folgenden Datenschritt wird eine Datei `mittel` aus Rohwerten angelegt. Ein Großteil der Rohwerte ist unbekannt und wegen dem listengesteuerten Einlesevorgang durch einen Punkt gekennzeichnet. Die vier Variablen werden mit `a`, `b`, `c` und `d` bezeichnet. Der Mittelwert dieser vier Variablen wird nun auf zwei Arten berechnet: durch Umsetzung der Formel (`mittel1`) und durch Verwendung der Funktion MEAN (`mittel2`). Die beiden Zuweisungen erfolgen vor der LINES-Anweisung im gleichen Datenschritt, in dem die Datei erzeugt wird. Es ist kein separater Datenschritt dafür notwendig.

Syntax[1]	Bedeutung	Beispiel	RW[2]
	Arithmetische Funktionen		
ABS(V.)	Absolutbetrag $\|x\|$	`ABS(x)`	n
EXP(V.)	Exponentialfunktion e^x	`EXP(x)`	n
SQRT(V.)	Quadratwurzel $\sqrt{x}$	`SQRT(x)`	n
	Stichprobenfunktionen		
MEAN(Vl.)	Mittelwert	`MEAN(OF t1-t4)`	n
SUM(Vl.)	Summe	`SUM(OF t1-t4)`	n
VAR(Vl.)	Varianz	`VAR(OF t1-t4)`	n
STD(Vl.)	Standardabweichung	`STD(OF t1-t4)`	n
MIN(Vl.)	Minimum	`MIN(OF t1-t4)`	n
MAX(Vl.)	Maximum	`MAX(OF t1-t4)`	n
	Wahrscheinlichkeitsdichte- und Quantilfunktionen		
PROBNORM(Q.)	Normalverteilung	`PROBNORM(q)`	n
PROBT(Q.,FG)	t-Verteilung	`PROBT(q,df)`	n
PROBCHI(Q.,FG)	χ^2-Verteilung	`PROBCHI(q,df)`	n
PROBIT(p)	Normalverteilung	`PROBIT(p)`	n
TINV(p, FG)	t-Verteilung	`TINV(p,df)`	n
CINV(p, FG)	χ^2-Verteilung	`CINV(p,df)`	n
	Zufallszahlenfunktionen		
RANNOR(Start)	Normalverteilte ZZ	`RANNOR(-1)`	n
RANBIN(Start,n,p)	Binomialverteilte ZZ	`RANBIN(-1,10,0.4)`	n
	Datums- und Zeitfunktionen		
TODAY()	Aktuelles Datum	`TODAY()`	d
WEEKDAY(Tag)	Wochentag	`WEEKDAY(TODAY())`	d
	Textfunktionen		
LEFT(Text)	Entnahme von links	`LEFT(' N')`	t
SUBSTR(Text,Pos,n)	Entnahme ab `Pos`	`SUBSTR('abcd',2,3)`	t

[1] V.=Variable, Vl.=Variablen-Liste, Q=Quantil, FG=Freiheitsgrad, p=p-Wert, ZZ=Zufallszahlen
[2] Rückgabewerte: n=numerisch, t=Text, d=Datum

Tab. 7.2: Ausgewählte SAS*-Funktionen*

```
/*--- KA07-03.SAS ---*/
DATA mittel;
     INPUT a b c d;
     mittel1=(a+b+c+d)/4;
     mittel2=MEAN(OF a--d);
     LINES;
1 2 3 4
1 2 3 .
1 2 . .
1 . . .
. . . .
RUN;
PROC PRINT DATA=mittel;
RUN;
```

Mit dem an den Datenschritt anschließenden Prozedurschritt wird das Ergebnis der Zuweisungen im Ausgabefenster sichtbar:

OBS	A	B	C	D	MITTEL1	MITTEL2
1	1	2	3	4	2.5	2.5
2	1	2	3	.	.	2.0
3	1	2	.	.	.	1.5
4	1	.	.	.	.	1.0
5	.	.	.	.	.	.

Nur für die erste Beobachtung, in der alle vier Werte vorliegen, kann mit der direkten Formel der Mittelwert berechnet werden. Bei den anderen Beobachtungen liegt jeweils mindestens ein fehlender Wert vor, wodurch das SAS-System nicht in der Lage ist, die Rechenvorschrift (engl. *operation*) auszuführen. Im Protokollfenster erscheint ein entsprechender Hinweis:

```
NOTE: Missing values were generated as a result of
      performing an operation on missing values.
      Each place is given by:
      (Number of times) at (Line):(Column).
      2 at 7:16   3 at 7:18   4 at 7:20   4 at 7:23
      1 at 8:14
```

Anstelle des Ergebnisses wird ein fehlender Wert zurückgegeben. In der Meldung erscheint die Anzahl der ersetzten fehlenden Werte sowie die Programm-

zeile und die Spalte der Operation, die nicht korrekt ausgeführt werden konnte. Betrachten Sie dazu nochmals die protokollierten Programmzeilen! (Das Zeilenlineal wurde nachträglich des besseren Verständnisses wegen eingefügt.)

```
5    DATA mittel;
6         INPUT a b c d;
     ----|----10---|----20---|----30--
7         mittel1=(a+b+c+d)/4;
8         mittel2=MEAN(OF a--d);
9         LINES;
```

Die erste Operation dieses Datenschritts ist die Addition der Variablen `a` und `b` in der 7. Programmzeile und 16. Programmspalte (`7:16`). Diese Operation konnte zweimal (`2 at`) nicht ausgeführt werden: bei der 4. und bei der 5. Beobachtung (fehlender Wert bei `b`). Die zweite Operation in der 18. Spalte der 7. Zeile (`7:18`), die Addition mit `c`, konnte 3 mal nicht ausgeführt werden (zusätzlich bei der 2. Beobachtung). Die dritte und vierte Operation (Addition in Spalte 20, Division in Spalte 23) konnten überhaupt nur bei der ersten Beobachtung vollzogen werden.

Die Funktion MEAN dagegen scheitert nur bei der letzten Beobachtung, bei der kein einziger Wert bekannt ist. In den anderen Fällen werden die Mittelwerte aus den verbleibenden nichtfehlenden Werten berechnet, d. h., in der 2. Beobachtung wird der Mittelwert aus drei Werten ermittelt, in der 3. Beobachtung aus zwei Werten usw.

Die *Quantil- und Wahrscheinlichkeitsfunktionen* ermöglichen (wie der Name bereits andeutet) die Berechnung von Quantilen und Wahrscheinlichkeiten verschiedener Verteilungsfunktionen: Standardnormalverteilung, t-Verteilung, F-Verteilung, Chiquadrat-, Gamma-Verteilung u.a.m. Damit kann man eigene, nicht im SAS-System enthaltene, statistische Tests umsetzen sowie zugehörige p-Werte oder Konfidenzbereiche ermitteln.

```
/*--- KA07-04.SAS ---*/
DATA ws;
     alfa=PROBNORM(1.645);
     x=TINV(alfa,9);
RUN;
```

Die Funktion PROBNORM berechnet zu dem Quantil 1.645 die Fläche unter der Wahrscheinlichkeitsdichtefunktion der Standardnormalverteilung (genauer: die Fläche links vom Quantil). Der Variablen `alfa` wird der Wert `0.95001` zugewiesen. Die Quantilfunktion TINV berechnet das umgekehrte: Zu der Wahr-

scheinlichkeit `alfa` wird das Quantil, in diesem Fall das der t-Verteilung mit 9 Freiheitsgraden, berechnet.

Mit dem Datenschritt können nicht nur Dateien eingelesen, sondern auch Daten neu erzeugt werden, wie sie beispielsweise für Simulationen benötigt werden. Mit Simulationen untersucht man Verfahren, die nicht „richtig“ durchgeführt werden können, weil kein oder nicht genügend Datenmaterial vorliegt. Man behilft sich mit „künstlichen“ Zufallszahlen, die annähernd die gleichen Eigenschaften wie die realen Daten haben. Das SAS-System stellt mehrere *Zufallszahlenfunktionen* zur Verfügung, die Zufallszahlen mit bestimmten Verteilungseigenschaften erzeugen, etwa normalverteilte Zufallszahlen mit der Funktion RANNOR.

```
/*--- KA07-05.SAS ---*/
DATA zufall;
     DO i=1 to 10;
        x=RANNOR(-1);
        OUTPUT;
     END;
RUN;
```

Mit diesem Datenschritt wird die Datei `zufall` erzeugt, die 10 Beobachtungen und die beiden Variablen `i` und `x` enthält. Die Variablenwerte werden innerhalb der DO/END-Schleife mit der Funktion RANNOR erzeugt. Die so gebildeten Zufallszahlen sind standardnormalverteilt, d. h., ihr Erwartungswert liegt bei 0, ihre Varianz bei 1. Für die 10 simulierten Zahlenwerte kann man natürlich nicht erwarten, daß der Mittelwert bei 0 liegt, aber wenn man hinreichend viele Zahlen zufällig zieht, nähert man sich dem erwarteten Wert immer weiter an. Die Zufallszahlenfunktion RANNOR hat ein Argument: den Startwert des Zufallszahlengenerators. Ist dieser, wie im vorliegenden Fall, kleiner Null, wird der Startwert vom System aus der Uhrzeit bestimmt. Ist der Startwert größer Null, wird dieser Wert verwendet und es werden jedesmal, wenn der Datenschritt ausgeführt wird, die gleichen Zufallszahlen generiert. Damit lassen sich Simulationsexperimente replizieren und verifizieren. Sollen die normalverteilten Zufallszahlen einen bestimmten Erwartungswert und eine Varianz verschieden von 1 aufweisen, multipliziert man die Zufallszahlen `x` mit der erwarteten Standardabweichung und addiert den gewünschten Erwartungswert.

```
x=mittel+std*RANNOR(-1);
```

Die *Datumsfunktionen* gestatten den Umgang mit Datumsangaben im SAS-System. Sie liefern das heutige Datum, teilen dem Anwender den Wochentag zu einem bestimmten Datum mit oder machen aus den Angaben von Tag, Monat und Jahr ein korrektes Datum, um nur ein paar Beispiele zu nennen. Weiterhin gibt es auch *Textfunktionen* zur Verarbeitung von Textvariablen. Mit ihnen

können Teile aus Textvariablen entnommen, die Länge eines Textes bestimmt und ein Text in einzelne Worte zerlegt werden. Mit den Systemfunktionen erhält man Informationen zur installierten SAS-Version, zum Betriebssystem, die aktuelle Uhrzeit etc.

Schließlich wird noch die Funktion INPUT vorgestellt, mit der man numerische Variablen in Textvariablen und Textvariablen in numerische Variablen verwandeln kann.

```
DATA num2text;
     INPUT a $ x @@;
     LINES;
1 1  10 10  100 100
RUN;
```

Die Datei num2text besteht aus zwei Variablen, der Textvariablen a und der numerischen Variablen x. Beide erhalten in den drei Beobachtun die gleichen Werte 1, 10, 100. Im Ausgabefenster bemerkt man den Unterschied zwischen numerischen und Textvariablen bereits durch die Ausrichtung der Werte (vgl. nächstes Ausgabefenster): Textvariablen werden linksbündig, numerische Variablen rechtsbündig ausgerichtet. Mit der Funktion INPUT wird nun in zwei Zuweisungen die numerische Variable x in eine Textvariable tx verwandelt und die Textvariable a in die numerische Variable na. Die Funktion INPUT benötigt zwei Argumente: zum einen die zu verwandelnde Variable, zum anderen das Format, mit dem die Variablenwerte gelesen werden:

```
/*--- KA07-06.SAS ---*/
DATA num2text;
     INPUT a $ x @@;
     na=INPUT(a,10.);
     tx=INPUT(x,$12.);
     LINES;
1 1  10 10  100 100
RUN;
PROC PRINT DATA=num2text;
RUN;
```

10. ist das Einleseformat für numerische Variablen, $12. das Einleseformat für Textvariablen (vgl. Abschnitt 5.3.3). Im Ausgabefenster erscheinen die folgenden vier Variablen:

```
OBS     A         X      NA     TX

 1      1         1       1     1
 2      10       10      10     10
 3      100     100     100     100
```

7.2.2 Ballast abwerfen – Variablen löschen

Bei der Berechnung neuer Variablen werden Zwischenergebnisse häufig in Variablen abgelegt, die im weiteren Verlauf nicht mehr benötigt werden. Werden diese Hilfsvariablen nicht gelöscht, benötigen anschließende Sortier- und Auswerteschritte mehr Zeit als ohne diese Variablen. Ein ähnlicher Wunsch nach dem Löschen von Variablen besteht, wenn Sie Ihre Daten an einen interessierten Kollegen weitergeben möchten, aus Datenschutz-rechtlichen Gründen aber alle personenbezogenen Angaben, die zur Identifizierung der untersuchten Personen führen würden, vor der Weitergabe aus der Datei entfernen müssen.

Dem Anwender stehen zum Löschen von Variablen zwei Möglichkeiten zur Verfügung: Er teilt dem System die Namen der Variablen mit, die gelöscht werden (DROP), oder er benennt die Variablen, die in der Datei verbleiben (KEEP). KEEP und DROP können als Anweisungen in einem Datenschritt oder als Dateioptionen KEEP= und DROP= in Daten- und Prozedurschritten ausgeführt werden.

Mit der Anweisung DROP werden nicht mehr benötigte Variablen einer Datei gelöscht. Bei der Berechnung der durchschnittlich erzielten Punktzahl der Firmenmitarbeiter wurde zuvor die Summe der beiden Teilabschnitte `teil2a` und `teil2b` gebildet. Da die Einzelergebnisse anschließend nicht mehr von Bedeutung sind, können sie mit der Anweisung DROP gelöscht werden, nachdem die neue Variable `teil2` gebildet wurde.

```
/*--- KA07-07.SAS ---*/
DATA biblio.punkte2;
     SET biblio.punkte;
     teil2=teil2a+teil2b;
     DROP teil2a teil2b;
     summe=teil1+teil2/2+teil3+teil4/2;
     mittel=summe/4;
RUN;
```

Alle Variablen, die gelöscht werden sollen, werden im Anschluß an das Schlüsselwort DROP einfach durch Leerzeichen getrennt aufgeführt. Führt man das

Programm aus, erkennt man aus der Meldung im Protokollfenster, daß die Datei `biblio.punkte2` nur noch 6 Variablen hat.

Möchte der Firmenchef die Testergebnisse einsehen, übergibt der Personalleiter ihm eine Liste, die nur die Personalnummern sowie die Summen der Einzelprüfungen enthält. Anstatt nun die Variablen aufzuzählen, die gelöscht werden sollen, benennt er diejenigen, die am Ende in der Datei verbleiben sollen:

```
DATA biblio.punkte3;
     SET biblio.punkte2;
     KEEP pnr summe;
RUN;
```

Die Syntax der Anweisung KEEP ist identisch zu DROP: Nach dem Schlüsselwort werden die Variablen, durch Leerzeichen getrennt, aufgeführt, die in die neue Datei übernommen werden sollen. Alle anderen Variablen werden gelöscht. Die Datei `biblio.punkte3` enthält nur noch zwei Variablen.

```
DROP variablen-liste;
KEEP variablen-liste;
```

Wie bereits oben angedeutet, können die beiden Anweisungen KEEP und DROP nur im Datenschritt eingesetzt werden, im Unterschied zu den Dateioptionen KEEP= und DROP=.

Dateioptionen sind optionale Angaben, die in Klammern eingeschlossen vor dem Semikolon an den Namen von SAS-Dateien angehängt werden.

```
    SAS-Datei (Dateioption=Wert)
```

An jeder Stelle innerhalb eines gültigen Programms, an dem der Name einer SAS-Datei steht, kann auch eine Dateioption folgen, beispielsweise in den Anweisungen SET und DATA:

```
DATA biblio.punkte3;
     SET biblio.punkte2(KEEP=pnr summe);
RUN;

DATA biblio.punkte3(KEEP=pnr summe);
     SET biblio.punkte2;
RUN;
```

Sie können aber auch in jedem Prozedurschritt, in dem die Option DATA= erscheint, eingesetzt werden, beispielsweise als Alternative zur VAR-Anweisung bei der Prozedur PRINT:

```
PROC PRINT DATA=biblio.punkte3(KEEP=pnr summe);
RUN;
```

Die Dateioptionen unterscheiden sich von den Anweisungen dadurch, daß nach dem Wort ein Gleichheitszeichen folgt und die gesamte Option in Klammern eingeschlossen ist, während die Anweisungen mit dem Schlüsselwort beginnen und mit Semikolon enden. Im Fall der Dateioptionen beendet das Semikolon nicht die Option sondern z. B. die Anweisung PROC PRINT. In den Beispielen wurde jeweils nur die Dateioption KEEP= verwendet, analog kann man natürlich auch die Dateioption DROP= einsetzen. Bei der Angabe der zu löschenden oder verbleibenden Variablen können Variablenbereiche (vgl. Abschnitt 6.1) angegeben werden: `KEEP=teil1-teil3` oder `KEEP=pnr- -teil2`.

7.2.3 Kunterbuntes Treiben? – Ändern der Variablenreihenfolge

Variablen, die nachträglich gebildet werden, erscheinen im Variablenfenster als unterster Eintrag, bzw. bei der Tabellenansicht einer Datei im Libraries-Fenster in der rechten Spalte. Das SAS-System ordnet die Variablen nicht alphabetisch oder der Größe nach an sondern chronologisch in der Reihenfolge ihrer Entstehung. Bei größeren Dateien mit vielen Variablen sollen die wichtigen Variablen möglichst weit links in der Tabelle stehen, damit sie nach dem Aufruf ohne viel Blättern sofort sichtbar werden. Dieses Umordnen der Variablen erledigt die RETAIN-Anweisung.

Im vorherigen Abschnitt wurde die Datei `num2text` mit den Variablen `a, x, na` und `tx` angelegt. Damit die Variablen `a` und `na` sowie `x` und `tx` in der Liste nebeneinander stehen, könnte man zum einen die Anweisung VAR im Prozedurschritt angeben. Um die Reihenfolge jedoch dauerhaft zu ändern, muß man dem System in einem neuen Datenschritt mit der Anweisung RETAIN die gewünschte Reihenfolge übergeben.

```
/*--- KA07-08.SAS ---*/
DATA num2text;
     RETAIN a na x tx;
     SET num2text;
RUN;
```

Die RETAIN-Anweisung muß im Datenschritt vor der Anweisung SET erscheinen, da sie die Reihenfolge der Variablen definiert. Würde zuerst die Anweisung SET erscheinen, wären die Variablen und deren Reihenfolge festgelegt und RETAIN würde keinerlei Auswirkungen haben.

```
RETAIN variablen-liste;
```

7.2.4 Namensgebung – Variablen umbenennen

Die Variablennamen werden im Datenschritt entweder in der Anweisung INPUT oder in der Zuweisung festgelegt. Stellt sich im Nachhinein heraus, daß ein Name nicht glücklich gewählt wurde oder die Datei nicht zu einer bestimmten Vorgabe paßt, möchte man den Variablennamen ändern, ohne die Datei neu erstellen zu müssen. Dazu bieten sich eine interaktive und eine prozedurale Variante an.

Über das Libraries-Fenster  kann man interaktiv Variablennamen ändern. Nach dem Aufruf selektiert man zunächst die Datei und öffnet über das Popup-Menü das Variablenfenster. In die Selektionsspalte trägt man nun das Kommando R für *Rename* ein, woraufhin der betreffende Variablenname farbig unterlegt wird und überschrieben werden kann. (Mit C für Cancel bricht man diesen Vorgang ab.)

Der prozedurale Weg geht (wen verwundert es!) über einen Datenschritt. Die Anweisung zum Umbenennen heißt RENAME. In der zuletzt erzeugten Datei num2text sollen die Variablen na und tx klingendere Namen erhalten: a_num und x_text.

```
/*--- KA07-09.SAS ---*/
DATA num2text;
     SET num2text;
     RENAME na=a_num tx=x_text;
RUN;
```

Der RENAME-Anweisung wird zunächst der alte Variablenname übergeben und nach dem Gleichheitszeichen der neue Variablenname definiert. Mehrere Variablen können gleichzeitig umbenannt werden.

```
RENAME altvar1=neuvar1 altvar2=neuvar2 ...;
```

Die Dateioption RENAME= bietet eine Alternative, die auch im Prozedurschritt eingesetzt werden kann. Sie hat sie den Vorteil, daß die Dateiänderungen nur

in diesem Prozedurschritt gelten und die Datei selbst unverändert bleibt. Die Variablen a und x werden für die Ausgabe in folgendem PROC PRINT-Schritt umbenannt in text und num:

```
PROC PRINT DATA=num2text(RENAME=(a=text x=num));
RUN;
```

```
OBS    TEXT    NUM    A_NUM    X_TEXT

 1     1         1       1       1
 2     10       10      10       10
 3     100     100     100       100
```

Die Dateioption RENAME= muß wie die anderen Dateioptionen auch in Klammern gesetzt werden. Zusätzlich werden hier noch die zu ersetzenden Variablen eingeklammert:

```
.... DATA=datei(RENAME=(alt1=neu1 alt2=neu2 ...));
```

7.3 Die Mischung macht's – Dateien verknüpfen

In den Prozedurschritten kann über die Option DATA= für die Darstellung oder Auswertung auf eine Datei zugegriffen werden. Da die Daten aber nicht immer von vornherein in einer Datei vorliegen, etwa weil mehrere Personen gleichzeitig die Beobachtungen erfassen und zunächst getrennt abspeichern oder weil verschiedene Informationen zu einzelnen Beobachtungen in mehreren Dateien abgelegt sind, muß es eine Möglichkeit geben, zwei oder mehrere Dateien zu einer gemeinsamen Datei zusammenzufügen. Die Verknüpfung erfolgt im Datenschritt. Man unterscheidet zwei Varianten:

❶ Die Dateien A und B enthalten verschiedene Beobachtungen zu den gleichen Merkmalen (Abbildung 7.1, oben, dünne Pfeile).

❷ Die Dateien A und B enthalten unterschiedliche Merkmale zu gleichen Beobachtungen (Abbildung 7.1, unten, dicke Pfeile).

Der Fall ❶ ist einfach umzusetzen: Nach der DATA-Anweisung folgt die SET-Anweisung, in der durch Leerzeichen getrennt die Namen der Dateien folgen, die aneinandergefügt werden sollen. Hierbei können nicht nur zwei Dateien verknüpft werden, sondern (fast) beliebig viele.

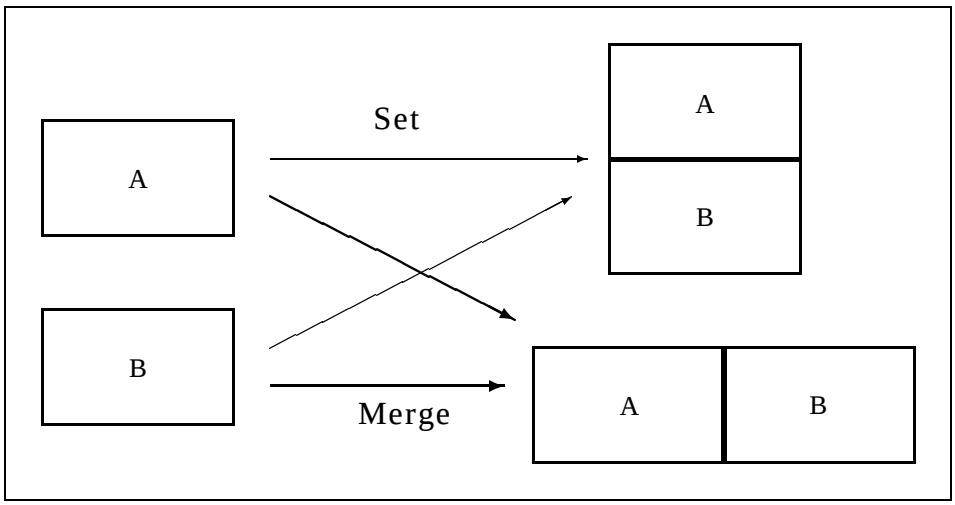


Abb. 7.1: Verknüpfen von Dateien

```
DATA neu;
     SET alt1 alt2 alt3 ...;
RUN;
```

Die Anzahl der Variablen verändert sich nicht, sofern die Dateien, die verknüpft werden, die gleiche Struktur haben. Die Anzahl der Beobachtungen der Datei `neu` besteht dagegen aus der Summe der Beobachtungen aus den Dateien `alt1`, `alt2` usw.

Enthalten die Dateien unterschiedliche Variablen, werden die Variablen in der neuen Datei bei den Beobachtungen, bei denen sie nicht definiert sind, durch fehlende Werte aufgefüllt.

```
/*--- KA07-10.SAS ---*/
DATA alt1;
     INPUT a b c;
     LINES;
1 2 3
4 5 6
RUN;
DATA alt2;
     INPUT a b d;
     LINES;
7 8 9
RUN;
DATA neu;
     SET alt1 alt2;
RUN;
```

Die Datei `alt1` enthält die Variablen a, b, c, die Datei `alt2` dagegen die Variablen a, b, d. Werden die Dateien verknüpft, ist die Variable c für `alt2` unbekannt, entsprechend d für `alt2`.

```
PROC PRINT DATA=neu;
RUN;
```

Die verknüpfte Datei neu enthält vier Variablen. Die Datenwerte der Variablen d werden für die ersten beiden Beobachtungen auf fehlend gesetzt, der Datenwert der Variablen c der dritten Beobachtung ebenfalls.

```
OBS    A    B    C    D

 1     1    2    3    .
 2     4    5    6    .
 3     7    8    .    9
```

Problematisch wird es, wenn die Variablen in beiden Dateien unterschiedlich kodiert sind, d. h., wenn der Variablentyp in der einen Datei numerisch ist, in der anderen Datei alphanumerisch, wie im folgenden Beispiel die Variable a.

```
/*--- KA07-11.SAS ---*/
DATA alt1;
     INPUT a $ @@;
     LINES;
1 2 3 4
RUN;
DATA alt2;
     INPUT a @@;
     LINES;
5 6 7 8
RUN;
DATA neu;
     SET alt1 alt2;
RUN;
```

Beim Versuch, im dritten Datenschritt die beiden Dateien zu verknüpfen, erscheint folgende Fehlermeldung im Protokollfenster:

```
ERROR: Variable A has been defined as both character and
       numeric.
NOTE: The SAS System stopped processing this step because
      of errors.
```

Das System ist erst in der Lage, die beiden Dateien zu verknüpfen, wenn die Variable a in der Datei alt1 in eine numerische Variablen verwandelt wird (oder a von Datei alt2 in eine Textvariable). In dem vorliegenden Miniaturbeispiel würde man einfach in der Anweisung INPUT das Dollarzeichen entfernen und den Einlesevorgang wiederholen. Aber wenn das Einleseprogramm nicht zur Verfügung steht, müssen Sie in die „Trickkiste“ greifen. Stichworte dazu sind INPUT-Funktion und RENAME.

Mit der Funktion INPUT verwandelt man die Textvariable über die Hilfsvariable b in eine numerische Variable. Danach löscht man die Textvariable a und benennt b in a um.

```
/*--- KA07-12.SAS ---*/
DATA alt1(DROP=a RENAME=(b=a));
     SET alt1;
     b=INPUT(a,3.);
RUN;
```

Die neue Datei `alt1` mit der numerischen Variable `a` kann anschließend ohne Fehlermeldung mit `alt2` verknüpft werden.

Enthalten die beiden Dateien verschiedene Merkmale der gleichen Beobachtungen (Fall ❷, Abbildung 7.1), muß man die MERGE-Anweisung einsetzen.

```
DATA neu;
     MERGE alt1 alt2;
     BY variable;
RUN;
```

Beim Einsatz von MERGE empfiehlt es sich nicht, mehr als zwei Dateien auf einmal zu verknüpfen, da man leicht die Kontrolle verlieren kann. Im SAS Language Reference Guide [9], in dem alle Datenschrittanweisungen beschrieben werden, wird zwischen verschiedenen Varianten des Merging unterschieden. In diesem Kurs wird jedoch nur die sicherste Methode vorgestellt: Das *Match Merging*. Hierbei werden die beiden Dateien hinsichtlich einer eindeutigen Schlüsselvariablen verknüpft, die in beiden Dateien vorliegen muß. Die Schlüsselvariable wird mit einer BY-Anweisung vereinbart. Wie bei der Prozedur PRINT auch, müssen die Dateien zuvor nach dieser Variablen sortiert werden. Ohne die Schlüsselvariable werden die beiden Dateien so, wie sie sind, einfach nebeneinander gestellt, ob die Beobachtungen passen oder nicht.

```
/*--- KA07-13.SAS ---*/
DATA alt1;
     INPUT id zeit1 @@;
     LINES;
4 110  1 125  3 200
RUN;
PROC SORT DATA=alt1;
     BY id;
RUN;
DATA alt2;
     INPUT id zeit2 @@;
     LINES;
4 130  1 210  2 145  3 180
RUN;
PROC SORT DATA=alt2;
     BY id;
RUN;
```

```
/*--- KA07-13.SAS ---*/ * Fortsetzung;
DATA neu;
     MERGE alt1 alt2;
     BY id;
RUN;
```

Die beiden Dateien `alt1` und `alt2` enthalten die zu zwei Zeitpunkten (`zeit1`, `zeit2`) erhobenen Meßwerte mehrerer Probanden, wobei in der 1. Datei die Beobachtung für die Person Nr. 2 fehlt. In der verknüpften Datei wird der Meßwert durch einen fehlenden Wert ergänzt.

```
OBS    ID    ZEIT1    ZEIT2

 1      1     125      210
 2      2       .      145
 3      3     200      180
 4      4     110      130
```

Sind die Beobachtungen mit einer einzigen BY-Variablen nicht eindeutig identifizierbar, müssen weitere Variablen in der Anweisung BY angegeben werden. Komplexere Varianten des Verknüpfens können mit der Prozedur SQL realisiert werden. Beispiele und Erläuterungen hierzu finden Sie im SAS Guide zur Prozedur SQL [8].)

Wurden die Schlüsselvariablen in den Dateien unterschiedlich bezeichnet, muß vor dem Verknüpfen mittels RENAME eine Angleichung erfolgen. Ein weiteres Problem tritt auf, wenn Variablen unterschiedlichen Inhalts unter dem gleichen Namen abgelegt sind. Beim Verknüpfen kann die Variable nur einmal angelegt werden. Zunächst wird der Inhalt der Datei eingetragen, die in der MERGE-Anweisung an erster Stelle genannt wird. Danach werden diese Werte mit den Werten der zweiten Datei überschrieben, sofern sie nicht fehlen.

7.4 Die Guten ins Töpfchen . . . – Teilmengen bilden

Die bislang ausgeführten Daten- oder Prozedurschritte haben jeweils alle Beobachtungen, seien sie in einer Rohwertedatei oder in einer SAS-Datei abgelegt worden, in die neue Datei oder in die Analyse aufgenommen. Bei größeren Dateien oder inhomogenem Datenmaterial ist es jedoch ratsam, bestimmte Analysen nur für einen Teil der Beobachtungen durchzuführen oder in Listen und Grafiken nur bestimmte Untergruppen darzustellen. Denken Sie jetzt an

die BY-Anweisung, die in Abschnitt 6.1 eingeführt wurde? Sie erlaubt eine getrennte Ausführung eines Prozedurschritts für die durch die Ausprägungen der BY-Variablen definierten Gruppen. Dies ist hier jedoch nicht gemeint. Anstatt eine getrennte Liste für Frauen und Männer mit der PRINT-Prozedur zu erstellen, besteht z. B. der Wunsch, die Liste nur für die Frauen oder nur für Frauen, die älter als 30 Jahre sind, zu erstellen. Oder ein Abteilungsleiter benötigt eine Personalliste „seiner“ Mitarbeiter.

Diese Einschränkung kann sowohl im Datenschritt, als auch im Prozedurschritt mit der WHERE-Anweisung oder der WHERE=-Dateioption erfolgen oder, allerdings ausschließlich im Datenschritt, mit der IF-Anweisung (Tabelle 7.3).

	Datenschritt	Prozedurschritt
Anweisung IF	ja	nein
Anweisung WHERE	nein[1]	ja
Dateioption WHERE=	ja	ja

[1] nur in Verbindung mit SET

Tab. 7.3: Einsatz von IF *und* WHERE

Die Einschränkung wird über eine logische Bedingung formuliert: Alle Beobachtungen, die diese Bedingung erfüllen, gehören zur gesuchten Teilmenge. Die anderen Beobachtungen werden nicht berücksichtigt. Genau wie bei der Bildung von neuen Variablen wird die Datei beobachtungsweise bearbeitet und für jede einzelne Beobachtung entschieden, ob sie ausgewählt wurde oder nicht. Logische Bedingungen können nur wahr oder falsch sein, daher werden Beobachtungen, für die die Bedingung wahr ist, aufgenommen, die anderen, falschen, dagegen nicht.

Um beispielsweise eine Liste der weiblichen Firmenmitarbeiter zu erstellen, müssen die Frauen ausgewählt werden. Wurde die Variable zum Merkmal Geschlecht mit `sex` benannt und `w` für weibliche, `m` für männliche Mitarbeiter als Wert erfaßt, werden alle Beobachtungen gesucht, bei denen die Variable `sex` den Wert `w` hat: `sex='w'`.

Ausgangsdatei `stammdat`	Logische Bedingung `sex='w'` erfüllt ?	Zieldatei `frauen`
`1001 Hans m`	nein	
`1002 Karin w`	ja	`1002 Karin w`
`1003 Melanie w`	ja	`1003 Melanie w`
`1004 Manfred m`	nein	
`1005 Annerose w`	ja	`1005 Annerose w`

Der Datenschritt zur Erzeugung der Datei `frauen` kann verschiedene Formen annehmen, je nachdem, welche Anweisung (IF oder WHERE) verwendet oder an welcher Stelle die Dateioption WHERE= gesetzt wird.

```
/*--- Anweisung IF      ---*/
DATA frauen;
     SET biblio.stammdat;
     IF sex='w';
RUN;
```

```
/*--- Anweisung WHERE   ---*/
DATA frauen;
     SET biblio.stammdat;
     WHERE sex='w';
RUN;

/*--- Dateioption WHERE (a) ---*/
DATA frauen(WHERE=(sex='w'));
     SET biblio.stammdat;
RUN;
/*--- Dateioption WHERE (b) ---*/
DATA frauen;
     SET biblio.stammdat(WHERE=(sex='w'));
RUN;
```

Die Anweisungen bzw. Dateioptionen wurden in einem Datenschritt eingefügt, wodurch eine neue Datei entstanden ist. Dieses Vorgehen ist empfehlenswert, wenn die Gesamtdatei (hier `biblio.stammdat`) nicht weiter verändert und die Teildatei (`frauen`) für mehrere weiterführende Analysen benötigt wird. Ginge es bei der Fragestellung aber einzig und allein darum, eine Liste der weiblichen Mitarbeiterinnen für den Betriebsrat zu erstellen, würde es genügen, die logische Bedingung mittels WHERE im Prozedurschritt zu formulieren.

```
/*--- Anweisung WHERE   ---*/
PROC PRINT DATA=biblio.stammdat;
     WHERE sex='w';
RUN;
/*--- Dateioption WHERE ---*/
PROC PRINT DATA=biblio.stammdat(WHERE=(sex='w'));
RUN;
```

Ob die Ausprägungen der Textvariablen, nach denen ausgewählt wird, in einfachen oder doppelten Anführungszeichen stehen, spielt dabei keine Rolle. Wichtig ist nur, daß man mit demjenigen Zeichen endet, mit dem man auch begonnen hat. Es ist jedoch relevant, ob die Ausprägung klein oder groß geschrieben wird. Im Unterschied zu den Anweisungen, Funktionen, Variablen- und Dateinamen unterscheidet das SAS-System an dieser Stelle zwischen Groß- und Kleinschreibung.

Werden die beiden folgenden Prozedurschritte ausgeführt, wird nur für den ersten PROC PRINT-Aufruf eine Liste erstellt. Die zweite Liste unterbleibt, da keine Beobachtung die logische Bedingung erfüllt.

```
PROC PRINT DATA=biblio.stammdat(WHERE=(sex='w'));
RUN;
PROC PRINT DATA=biblio.stammdat(WHERE=(sex='W'));
RUN;
```

Man umgeht dieses Problem mit der Groß- und Kleinschreibung, indem man die UPCASE- oder LOWCASE-Funktion einsetzt. Vor der Überprüfung der logischen Bedingung werden die Kleinbuchstaben dabei zuerst in Großbuchstaben verwandelt.

```
PROC PRINT DATA=biblio.stammdat(WHERE=(UPCASE(sex)='W'));
RUN;
```

In den bisherigen Beispielen wurden die Untergruppen durch Angabe von Bedingungen formuliert, in denen eine exakte Gleichheit gefordert wurde. Es gibt weitere *Vergleichsoperatoren*, die entweder als Symbol (wie „=“) oder als Wort (z. B. EQ als Abkürzung für *equal* (= gleich)) angegeben werden können (siehe Tabelle 7.4) oder den IN-, BETWEEN/AND-, CONTAINS- und LIKE-Operator.

Die Vergleichsoperatoren können bei numerischen Variablen und bei Textvariablen eingesetzt werden. Der „>“-Vergleich wird dabei auf die Reihenfolge im Alphabet verlagert: B ist größer als A, Meier größer als Maier.

Ein Problem taucht beim Vergleich numerischer Werte auf, wenn fehlende Werte vorkommen, wie im folgenden Beispiel. Die Altersangaben von fünf Personen konnten nur unvollständig erfaßt werden. Bei zwei Personen steht daher ein Punkt. Erzeugt man eine Liste der Personen, die jünger als 20 sind, würde man erwarten, daß nur `Hans` angezeigt wird.

Vergleichsoperatoren			
Symbol	Wort	Bedeutung	Beispiel
=	EQ	gleich	`alter=25`
!=	NE	ungleich	`alter NE .`
>	GT	größer als	`alter>20`
≥	GE	größer oder gleich	`alter GE 20`
<	LT	kleiner als	`alter<60`
≤	LE	kleiner oder gleich	`alter LE 65`
Andere Operatoren			
	IN	enthalten in	`sex IN ('w','m')`
	BETWEEN /AND	innerhalb von ... und ...	`alter BETWEEN 20 AND 40`
	CONTAINS	enthalten in	`ort CONTAINS 'berg'`
	LIKE	beginnt mit	`nname LIKE 'Sch%z'`[1]
			`nname LIKE 'Sch_lz'`[2]

Zwischen `Sch` und `z` [1] kann jede beliebige Zeichenfolge, im Fall [2] genau ein Zeichen vorkommen.

Tab. 7.4: Vergleichs- und andere Operatoren

```
/*--- KA07-14.SAS ---*/
DATA gruppe;
     INPUT name $ alter @@;
     LINES;
Hans 19 Jutta . Jens 20 Kerstin 22 Uwe .
RUN;
PROC PRINT DATA=gruppe;
     WHERE alter<20;
RUN;
```

Stattdessen erscheinen auch `Jutta` und `Uwe` in der Liste:

```
OBS    NAME      ALTER

 1     Hans        19
 2     Jutta        .
 5     Uwe          .
```

Die Ursache liegt in der Interpretation des fehlenden Werts: Der fehlende Wert wird als kleinster möglicher Wert betrachtet. Daher sind die beiden fehlenden Altersangaben kleiner als 20 und erfüllen die logische Bedingung.

Die einzelnen Bedingungen können mit AND und OR verbunden werden. Sollen z. B. in einer Liste, die für den Betriebsrat erstellt werden soll, nur die Frauen aufgezeigt werden, die nicht in Heidelberg wohnen, wird die Anweisung WHERE um eine weitere Komponente erweitert:

```
PROC PRINT DATA=biblio.stammdat;
     WHERE sex='w' AND wohnort NE 'Heidelberg';
RUN;
```

Zuerst werden die Frauen ausgewählt und danach, in einem weiteren Schritt, diejenigen Mitarbeiterinnen, deren Wohnort nicht Heidelberg ist. Man könnte aber auch erst diejenigen Mitarbeiter auswählen, die außerhalb Heidelbergs wohnen und davon die Frauen bestimmen. Im Endergebnis macht dies keinen Unterschied. Wichtig ist nur, daß beide logischen Bedingungen erfüllt sein müssen. Hätte man dagegen die beiden Bedingungen mit OR verknüpft,

```
PROC PRINT DATA=biblio.stammdat;
     WHERE sex='w' OR wohnort NE 'Heidelberg';
RUN;
```

erhielte man eine Liste aller Frauen und aller Mitarbeiter, die nicht in Heidelberg wohnen. Bei OR reicht es, wenn eine der Bedingungen erfüllt ist, bei AND müssen beide erfüllt sein. Die folgende Tabelle 7.5 zeigt die verschiedenen Möglichkeiten auf, in denen eine kombinierte Bedingung wahr ist (d. h. die Beobachtung wird ausgewählt) bzw. wann sie falsch ist (d. h. sie wird nicht ausgewählt).

Bedingung		Verknüpfung mit	
1	2	AND	OR
wahr	wahr	wahr	wahr
wahr	falsch	falsch	wahr
falsch	wahr	falsch	wahr
falsch	falsch	falsch	falsch

Tab. 7.5: Logische Verknüpfungen mit AND *und* OR

Werden mehr als zwei Einzelbedingungen verknüpft, empfiehlt es sich, Klammern zu setzen, die die Reihenfolge der Ausführung der Einzelbedingungen eindeutig festlegen:

```
WHERE (sex='w' AND (wohnort='Heidelberg' OR name='Maier');
```

7.4.1 Wenn/Dann – Bedingte Anweisungen

Mit den bislang vorgestellten Methoden können bildlich nur ganze Zeilen der Tabelle ausgewählt werden (WHERE) oder komplette Spalten ergänzt, verändert (Zuweisung) oder gelöscht (DROP) werden. Einzelne Zeilen gezielt löschen, einzelne Zellen (= Werte) verändern oder Variablenwerte in Abhängigkeit eines anderen Variablenwerts ändern, kann man erst, wenn man in der Lage ist, Anweisungen unter bestimmten Bedingungen im Datenschritt ausführen zu lassen. Man muß eine Bedingung formulieren und sagen können: Wenn die Bedingung erfüllt ist, dann tue dies und jenes. Übersetzt in die Sprache des SAS-Systems heißt diese Anweisung IF/THEN. Und zusätzlich gibt es noch ein „andernfalls“: ELSE. Die Bedingung wird wieder in Form einer logischen Bedingung formuliert (vgl. WHERE). Bei Erfüllung dieser Bedingung können vollständige Anweisungen ausgeführt werden: Zuweisungen (`ort='HD'`), Löschanweisungen für Beobachtungen (DELETE) etc. Die Anweisung hat die Form[3]

```
IF Logische Bedingung THEN Anweisung;
<ELSE Anweisung;>
```

Betrachten Sie nochmals das Beispiel der fünf Personen, deren Altersangaben erfaßt werden sollten. Ein Problem waren die fehlenden Werte. Konnte letztendlich doch noch das Alter von `Jutta` in Erfahrung gebracht werden, soll es in der bereits bestehenden Datei ergänzt werden. Dazu bedarf es einer Zuweisung: `alter=23`. Würde man diese Zuweisung ohne Bedingung in einem Datenschritt ausführen, wären die Altersangaben aller Personen auf 23 gesetzt. Daher wird folgende Bedingung formuliert: Wenn der Name gleich Jutta ist, dann wird das Alter auf 23 Jahre gesetzt.

```
DATA gruppe;
     SET gruppe;
     IF name="Jutta" THEN alter=23;
RUN;
```

[3] Die beiden Semikolons in der Syntaxbeschreibung der IF/THEN-ELSE-Anweisungen sind strenggenommen überflüssig, da jede Anweisung mit einem Semikolon endet.

Die einzige Voraussetzung ist hierbei, daß die Bedingung eindeutig formuliert wird. Gibt es mehrere Personen namens Jutta in der Datei, würden nach der Ausführung obigen Datenschritts bei allen das gleiche Alter eingetragen sein.

Die bedingte Ausführung von Anweisungen kann nur im Datenschritt erfolgen. Es ist aber nicht unbedingt erforderlich, daß in diesem Datenschritt mit SET auf eine bereits bestehende Datei zugriffen wird. Bereits beim Anlegen einer neuen Datei aus Rohwerten können Anweisungen bedingt ausgeführt werden. In obigem Beispiel könnte man etwa von vornherein alle Beobachtungen ausschließen, bei denen die Altersangabe fehlt.

```
DATA gruppe;
     INPUT name $ alter @@;
     IF alter=. THEN DELETE;
     LINES;
Hans 19 Jutta . Jens 20 Kerstin 22 Uwe .
;
```

Sobald während der Ausführung des Datenschritts festgestellt wird, daß die Altersangabe einer Beobachtung fehlt, wird diese mit der Anweisung DELETE gelöscht. In obigem Beispiel betrifft dies die 2. und letzte Beobachtung. In die Datei `gruppe` werden daher nur 3 Beobachtungen eingetragen.

Fügt man allerdings vor der bedingten Löschanweisung eine Zuweisung ein, wird zumindest die Eintragung von Jutta übernommen:

```
/*--- KA07-15.SAS ---*/
DATA gruppe;
     INPUT name $ alter @@;
     IF name="Jutta" THEN alter=23;
     IF alter=. THEN DELETE;
     LINES;
Hans 19 Jutta . Jens 20 Kerstin 22 Uwe .
RUN;
```

Mit der bedingten IF-Anweisung können auch neue Variablen gebildet werden, die je nach Bedingung verschiedene Werte tragen. In der Datei `gruppe` soll zusätzlich noch ein Variable `sex` gebildet werden mit den möglichen Ausprägungen `w` und `m`.

```
/*--- KA07-16.SAS ---*/
DATA gruppe;
     INPUT name $ alter @@;
     IF name IN ('Hans','Jens','Uwe','Kai','Ulrich')
        THEN sex='m';
     ELSE sex='w';
     LINES;
Hans 19 Jutta . Jens 20 Kerstin 22 Uwe .
RUN;
```

Die weiblichen Namen brauchen nicht aufgezählt zu werden und die Bedingung muß nicht explizit formuliert werden, da IF/THEN nach dem Ausschlußprinzip arbeitet: Für alle Beobachtungen, die die logische Bedingung in der IF/THEN-Anweisung nicht erfüllen, wird automatisch die ELSE-Anweisung ausgeführt. Männer mit ausgefallenen Namen, die in obiger Vergleichsliste nicht erfaßt wurden, oder Personen mit fehlenden Namensangaben durchlaufen die ELSE-Anweisung und werden bei der Variable `sex` mit `w` belegt.

Analog zur WHERE-Anweisung und der unbedingten IF-Anweisung können mehrere logische Bedingungen mittels AND und OR verknüpft werden. Außerdem lassen sich bedingte IF-Anweisungen beliebig tief ineinander verschachteln:

```
IF wohnort='Heidelberg' THEN ort='Bad.Wue.';
ELSE IF wohnort='Ludwigshafen' THEN ort='Rheinl.Pfalz';
     ELSE ...;
```

Wenn mehr als eine Anweisung für die gleiche Bedingung ausgeführt werden soll, kann man durch zusätzliche Einführung eines DO/END-Blocks diese Anweisungen auf einmal ausführen. Jede Anweisung innerhalb des Blocks muß dabei durch ein eigenes Semikolon abgeschlossen werden.

```
IF wohnort='HD' THEN
   DO;
      entfernung=0;
      ort='Stadt';
      ...;
   END;
```

Die bedingte IF-Anweisung können Sie (genauso wie die unbedingte) nur im Datenschritt einsetzen, da sie stets eine Veränderung der Datei nach sich zieht:

Veränderung einzelner Datenwerte, Hinzunahme neuer Variablen oder Löschen von Beobachtungen.

7.5 Aussagekräftige Bezeichnungen für Variablen und ihre Datenwerte

Die Beschränkung auf acht Zeichen kann für den Außenstehenden zu unverständlichen Variablennamen führen. `sex` für die Variable Geschlecht ist eine gute Wahl. Unter `pnr` kann man sowohl Personalnummer als auch Patientennummer verstehen. Und `item1` bis `item125` versteht nur der Eingeweihte. Bei der Eingabe der Datenwerte bemüht man sich, mit wenigen Zeichen auszukommen und, wenn machbar, die Antwortkategorien numerisch, d. h. mit Zahlenwerten zu kodieren. Der Schreibaufwand und die Fehleranfälligkeit werden dadurch beträchtlich minimiert. Für Außenstehende ist in diesem Fall aber nicht klar, welche Frage sich hinter `item7` verbirgt oder ob die Ausprägung `1` bei Ja/Nein-Fragen nun „Ja“ oder „Nein“ bedeuten soll. Der Anwender, der die Dateneingabe und die Auswertung durchführt, will auf der einen Seite effizient mit geringer Tipparbeit arbeiten, derjenige, dem die Ergebnisse präsentiert werden, wünscht sich auf der anderen Seite ausreichende Erläuterungen und verständliche Beschriftungen. Mit der LABEL-Anweisung zur Definition aussagekräftiger Bezeichnungen (*Etiketten*) für Variablennamen und der FORMAT-Anweisung für die Verknüfung der Datenwerte mit Formaten bzw. der FORMAT-Prozedur zur Erstellung eigener Formate trägt das SAS-System diesen Wünschen Rechnung.

7.5.1 Die Anweisung LABEL

Mit der LABEL-Anweisung werden Etiketten für Variablen definiert. Diese Etiketten können bis zu 40 Zeichen lang sein und Sonderzeichen und Leerzeichen enthalten. Außerdem wird die Groß- und Kleinschreibung berücksichtigt.

Die Zuordnung eines solchen Etiketts zu einer Variablen erfolgt im Datenschritt oder im Prozedurschritt. Im Datenschritt wird das Etikett fest mit der Variablen und der Datei verankert. Handelt es sich zudem noch um eine permanente Datei, wird das Etikett mitsamt der Datei abgespeichert und Sie können es beispielsweise im Variablenfenster kontrollieren. Erfolgt die Definition im Prozedurschritt, gilt sie nur für diesen speziellen Programmschritt. Die Datei bleibt unverändert.

Die Syntax der Anweisung ist einfach: Nach dem Schlüsselwort LABEL folgt der Variablenname, ein Gleichheitszeichen und in Anführungszeichen das Etikett:

```
LABEL variable='etikett' variable2='etikett2' ...;
```

Mehrere Etiketten können gleichzeitig in einer LABEL-Anweisung definiert werden. Außerdem können die Etiketten durch neue LABEL-Anweisungen beliebig häufig überschrieben werden.

Im folgenden Beispiel werden von den Mitarbeiter-Stammdaten die Namen und das Geschlecht in einer Liste mit Variablenetiketten ausgegeben:

```
/*--- KA07-17.SAS ---*/
PROC PRINT DATA=biblio.stammdat LABEL;
     VAR name vorname sex;
     LABEL name='Nachname' vorname='Vorname'
           sex='Geschlecht';
RUN;
```

Die LABEL-Anweisung legt die Etiketten fest, die in diesem Prozedurschritt verwendet werden. Die Option LABEL in der PROC PRINT-Anweisung ist eine Besonderheit dieser Prozedur. Ohne die Option würden keine Etiketten ausgegeben, obwohl sie definiert wurden. Bei den anderen Prozduren ist diese Option nicht notwendig. Sie werden dazu noch Beispiele kennenlernen.

```
OBS    Nachname    Vorname      Geschlecht

  1    Meier       Hans             m
  2    Schulz      Karin            w
  3    Frisch      Melanie          w
  ...
 29    Bauer       Albert           m
 30    Weber       Manfred          m
```

Damit die Etiketten fest mit der Datei verknüpft werden und immer zur Verfügung stehen, müssen sie im Datenschritt definiert werden. Im Variablenfenster werden sie dann, wie in Abbildung 7.2 wiedergegeben, angezeigt.

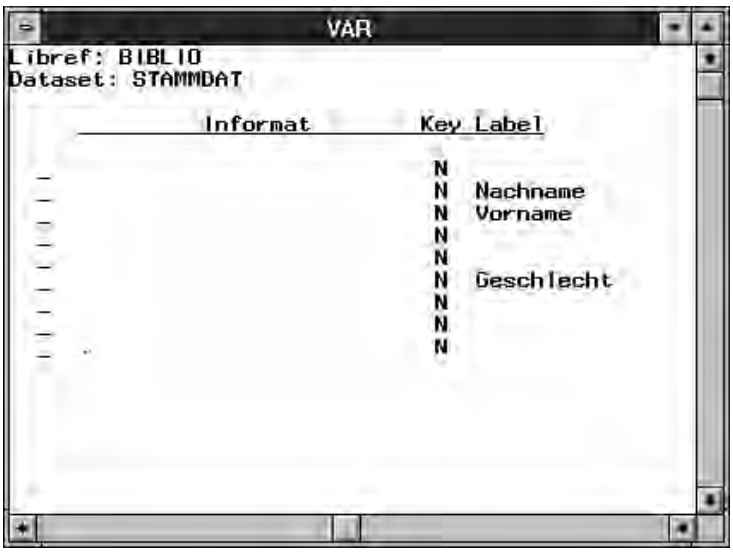


Abb. 7.2: Etiketten (Labels) in der Datei `stammdat`

```
/*--- KA07-18.SAS ---*/
DATA biblio.stammdat;
     SET biblio.stammdat;
     LABEL name='Nachname' vorname='Vorname'
           sex='Geschlecht';
RUN;
```

Etiketten, die im Datenschritt definiert wurden, löscht man über das Variablenfenster (Kommando R für Rename in der Selektionsspalte eintragen und das Etikett mit ENTF löschen) oder man führt einen weiteren Datenschritt aus, in dem leere Etiketten definiert werden:

```
DATA biblio.stammdat;
     SET biblio.stammdat;
     LABEL name= vorname= sex=;
RUN;
```

7.5.2 Die Anweisung FORMAT und die gleichnamige Prozedur

Mit der Anweisung FORMAT werden Etiketten für Datenwerte definiert. Diese Etiketten werden als *Formate* bezeichnet, da sie den Datenwerten eine Form bei der Darstellung geben. In Kapitel 5 wurden Einleseformate vorgestellt, die für das formatgesteuerte Einlesen notwendig sind. Sie geben den Werten die Form zum Einlesen. Die Ausgabeformate, oder einfach kurz Formate, liefern die Form, in der die Werte im Ausgabefenster erscheinen.

Die Formate können wie die Variablenetiketten sowohl im Datenschritt (dann sind sie fest mit der Datei verankert) oder im Prozedurschritt (nur für diesen Programmschritt) definiert werden. Die Syntax der Anweisung FORMAT unterscheidet sich von der LABEL-Anweisung. Nach dem Schlüsselwort folgt ein Variablenname und anschließend ein Formatname. Sie definieren an dieser Stelle nicht, wie die einzelnen Werte dargestellt werden, sondern benennen nur das Format.

```
FORMAT variable formatname. variable2 formatname2. ...;
```

Das Format muß zu diesem Zeitpunkt bereits vorhanden sein, andernfalls erscheint eine entsprechende Fehlermeldung. Dem Anwender werden eine Reihe von systemeigenen Formaten zur Verfügung gestellt: Formate für Textvariablen, mit denen die Breite der Darstellung gesteuert wird ($W.), Formate für numerische Variablen zur Bestimmung der darzustellenden Nachkommastellen (W.D) oder Formate für Datumsvariablen (DDMMYY8., vgl. Abschnitt 7.6). Die Formate enthalten wie die Einleseformate stets einen Punkt.

Im folgenden Beispiel werden die Punktzahlen aus den Teilabschnitten 3 und 4 mit einer und mit 2 Dezimalstellen ausgegeben:

```
PROC PRINT DATA=biblio.stammdat;
     VAR pnr teil3 teil4;
     FORMAT teil3 4.1 teil4 5.2;
RUN;
```

Das definierte Format muß breit genug gewählt werden, damit die Zahlenwerte damit dargestellt werden können. Während Textvariablen gegebenenfalls abgeschnitten werden, werden numerische Variablenwerte nicht ausgegeben. Bei der Gesamtbreite zählt der Dezimalpunkt mit!

```
OBS      PNR    TEIL3    TEIL4
  1     1001     8.0      6.00
  2     1002     5.0     15.00
  3     1003     5.0     13.00
 ..     ....     ...     ....
 29     1029     8.0     12.00
 30     1030     4.0      6.00
 31     1031     6.0     10.00
```

Neben diesen SAS-eigenen Formaten kann sich der Anwender eigene Formate mit der Prozedur FORMAT und der VALUE-Anweisung definieren. Die Definition des Formats erfolgt zunächst unabhängig von der Datei. In einem zweiten Schritt wird dieses Format der Variable zugewiesen, womit die Darstellungsform der Ausprägungen festgelegt ist. Der Formatname muß den üblichen Namenskonventionen des Systems genügen.

In den Stammdaten wurden die Frauen mit `w` gekennzeichnet, die Männer mit `m`. Damit anstelle dieser Abkürzungen `Frau` und `Herr` erscheint, wird das Format `$fsex` definiert. Da es sich bei dem Format um ein Textformat handelt, das nur einer Textvariablen zugewiesen werden kann, muß der Name mit einem $ -Zeichen beginnen (weshalb für den eigentlichen Namen nur maximal sieben Zeichen zur Verfügung stehen!).

```
/*--- KA07-19.SAS ---*/
PROC FORMAT;
     VALUE $fsex w=Frau
                 m=Herr
     ;
RUN;
```

Nachdem dieses Programm ausgeführt wurde, erscheint im Protokollfenster folgende Meldung:

```
NOTE: Format $FSEX has been output.
```

Wenn der Prozedurschritt ein weiteres Mal ausgeführt wird, erscheint die Warnung, daß dieses Format bereits existierte (und überschrieben wurde):

```
WARNING: Format $FSEX is already on the library.
NOTE: Format $FSEX has been output.
```

Nachdem das neue Format mit der Prozedur FORMAT angelegt wurde, kann es wie die systemeigenen Formate verwendet werden:

```
PROC PRINT DATA=biblio.stammdat;
     VAR sex vname nname;
     FORMAT sex $fsex.;
RUN;
```

In der FORMAT-Anweisung ist wichtig, daß Sie auch bei den selbstdefinierten Formaten den Punkt am Ende nicht vergessen. Denn ohne den Punkt würde `$fsex` als Variablenname interpretiert, aber ein Dollarzeichen zu Beginn des Namens ist nicht erlaubt ...

```
OBS    SEX     VNAME        NNAME

  1    Herr    Hans         Meier
  2    Frau    Karin        Schulz
  3    Frau    Melanie      Frisch
 ...
 29    Herr    Albert       Bauer
 30    Herr    Manfred      Weber
 31    Herr    Stefan       Scholz
```

Die Prozedur FORMAT gestattet die Zuweisung bestimmter Formen zu einzelnen Werten, zu Bereichen (von/bis) oder zu allen Werten, denen bislang keine Form zugewiesen wurde (OTHER).

Als Beispiel werden die Punktzahlen des Einstellungstests herangezogen. Das Abschneiden im 4. Teilabschnitt soll folgendermaßen bewertet werden: 16-20 Punkte = sehr gut, 11-15 = gut, 6-10 = in Ordnung und 0-5 = wiederholen. Das Format für eine numerische Variable erhält den Namen `bwertung`. Ein zweites Format, `extreme`, soll die mittleren Punkte (6-15) von den Extremwerten abgrenzen.

```
/*--- KA07-20.SAS ---*/
PROC FORMAT;
     VALUE bwertung LOW-5=wiederholen
                     6-10=in Ordnung
                    11-15=gut
                    16-HIGH=ausgezeichnet;
     VALUE extreme  6-15=im Mittel
                    OTHER=extrem;
RUN;
```

Werden selbstdefinierte Formate in einem Datenschritt mit einer permanenten Datei verknüpft, muß man das Format ebenfalls permanent speichern, um sicherzustellen, daß es auch in späteren Sitzungen zur Verfügung steht. Dazu benötigt man einen speziellen Bibliotheksnamen, LIBRARY, und die Option LIB= in der PROC FORMAT-Anweisung.

```
LIBNAME LIBRARY 'C:\kurs';
PROC FORMAT LIB=LIBRARY;
     VALUE ...;
RUN;
```

Die Bibliothek LIBRARY referenziert das Übungsverzeichnis `kurs` auf Laufwerk C:. In diesem Verzeichnis wird mit dem Prozedurschritt der SAS-Katalog FORMATS.SC2 angelegt, in dem die permanenten Formate gespeichert werden. Ohne die Option LIB= werden die Formate im temporären Verzeichnis abgelegt.

Um in einer späteren Sitzung auf eines der permanenten Formate zugreifen zu können, genügt es, den Bibliotheksnamen LIBRARY mit dem entsprechenden Verzeichnis zu verknüpfen. Die Prozedur FORMAT muß nicht erneut ausgeführt werden.

```
LIBNAME LIBRARY 'C:\kurs';
PROC PRINT DATA=biblio.punkte2;
     FORMAT teil2 bwertung. teil4 extreme.;
RUN;
```

Ist der permanente Formatkatalog einmal nicht verfügbar, oder hat ein Kollege Ihnen eine Datei ohne den erforderlichen Formatkatalog überlassen, erlaubt Ihnen das System nicht, diese Datei zu kopieren und auszudrucken, solange die Formate fehlen. Mit der FORMAT-Anweisung

```
FORMAT _ALL_ ;
```

werden die Formatzuweisungen rückgängig gemacht, leider auch die der SAS-eigenen Formate, die Ihnen immer zur Verfügung stehen. Wenn Sie genauer lokalisieren können, welche Formate fehlen, dann geben Sie nur die betroffenen Variablen an (`FORMAT variable1 variable2;`).

7.6 Datumsvariablen

„Bei Datumsangaben muß es sich um besondere Variablen handeln.“ Dieser Eindruck ist bei Ihnen vielleicht entstanden. Obwohl zu Beginn betont wurde, daß

das SAS-System nur numerische und Textvariablen unterscheidet, wurden die Datumsvariablen bereits in Kapitel 5 im Zusammenhang mit Nichtstandard-Daten genannt und der Bedarf entsprechender Einlese- sowie Ausgabeformate erwähnt. Also was nun: numerisch oder Nichtstandard? Die Antwort ist ein klares „Sowohl als auch“. Die Datumsvariablen werden als numerische Variablen behandelt. Damit das System aber „unsere“ Schreibweise für Datumsangaben versteht und die Werte in dieser Form auch ausgeben kann, müssen spezielle Einlese- und Ausgabeformate verwendet werden.

Intern werden die Datumsangaben in die Anzahl der Tage seit (oder vor) dem 1. Januar 1960 umgerechnet. Diese Werte werden in der Datei gespeichert. Differenzen zwischen zwei Datumsangaben, etwa die Liegedauer eines Patienten im Krankenhaus oder die Anzahl der Tage oder Wochen seit Ausgang der Bestellung einer Ware, können somit einfach berechnet werden, und auch der Übergang ins Jahr 2000 bereitet keine Mühe.

Für das nächste Beispiel werden die Stammdaten der Firma erneut in eine Datei eingelesen. Das Eintrittsdatum der Mitarbeiter ist in der Form Tag (2-stellig), Monat (2-stellig, numerisch) und Jahr (2-stellig) in der Rohwerte-Datei `stammdat.dat` eingetragen.

```
/*--- KA07-21.SAS ---*/
DATA stammdat;
     INFILE 'A:\firma\stammdat.dat';
     INPUT @6 nname $ 6-12 vname $ 14-22
           @46 eintritt DDMMYY8.;
     FORMAT eintritt DDMMYY8.;
RUN;
```

Die beiden Variablen `nname` und `vname` werden spaltengesteuert eingelesen, das Eintrittsdatum formatgebunden. Der absolute Spaltenzeiger beginnt bei Spalte 46. Das Einleseformat entspricht der Form, in der die Datumsangaben vorliegen. Gleichzeitig wird mit der Anweisung FORMAT das Ausgabeformat festgelegt. Wählt man als Ausgabeformat DDMMYY10. wird die Jahreszahl 4-stellig angezeigt. Mit dem internationalen Format DEUDFWDX. erhält man ein Format, das man vom Briefeschreiben kennt: `1. Januar 1997`. Ohne Format wird die Anzahl der Tage (vor/seit dem 01.01.1960) ausgegeben.

Zum Schluß wird noch berechnet, wie lange die Mitarbeiter bereits in der Firma beschäftigt sind. Dazu benötigt man eine Konstante, die das aktuelle Datum widerspiegelt. Eine Möglichkeit bietet die Funktion TODAY(). Mit der Funktion INPUT, der eine Zeichenkette und das Datumsformat übergeben werden, oder der Funktion MDY, die aus den Angaben für Monat, Tag und Jahr ein richtiges Datum zusammensetzt, könnte man das Problem ebenfalls lösen.

```
/*--- KA07-22.SAS ---*/
DATA stammdat;
     SET stammdat;
     heute=TODAY();
       * heute=INPUT('21.06.97',DDMMYY8.);
       * heute=MDY(6,21,97);
       * heute='21JUN97'D;
     FORMAT heute DDMMYY8.;
     diff_t=heute-eintritt;
     diff_j=INTCK('year',eintritt,heute);
RUN;
PROC PRINT DATA=stammdat;
RUN;
```

Damit die Variable heute in lesbarer Form ausgegeben wird, wird ihr im gleichen Datenschritt ein Format zugewiesen. Neben der Funktion TODAY() wurden auch die anderen Möglichkeiten, konstante Datumsangaben zu erzeugen, als Kommentar aufgeführt. Bei der letzten, der vierten aufgezeigten Variante, der einzigen direkten Methode, wird das US-amerikanische Format DATE7. verwendet, bei dem für den Monat die ersten drei Buchstaben der englischsprachigen Bezeichnung eingetragen werden.

Die Differenz zwischen heute und dem Eintrittsdatum liefert die Anzahl der Tage, seit der die Mitarbeiter beschäftigt sind: diff_t. Mit der Funktion INTCK wird die Anzahl der Jahre (das Intervall ist das erste Argument dieser Funktion, hier: 'year') zwischen den beiden Datumsangaben eintritt und heute berechnet.

```
OBS  NNAME   VNAME       EINTRITT  HEUTE   DIFF_T DIFF_J

  1  Meier   Hans      15/01/1991 21/06/97  2349      6
  2  Schulz  Karin     01/06/1990 21/06/97  2577      7
  3  Frisch  Melanie   01/06/1990 21/06/97  2577      7
 ...
 29  Bauer   Albert    01/06/1993 21/06/97  1481      4
 30  Weber   Manfred   01/08/1994 21/06/97  1055      3
 31  Scholz  Stefan    01/04/1995 21/06/97   812      2
```

Manche der in diesem Kapitel vorgestellen Anweisungen können auch mit den Prozeduren DATASETS und COPY ausgeführt werden. Im SAS-System führen mehrere Wege ans gleiche Ziel. (Vgl. dazu auch die Online-Hilfe oder [11].)

7.7 Aufgaben

❶ Führen Sie den Datenschritt `DATA neu;RUN;` aus und überprüfen Sie im Libraries-Fenster, ob die Datei angelegt wurde, welche Variablen sie enthält und wie die Datenwerte aussehen. Welche Meldungen werden im Protokollfenster ausgegeben?

❷ Führen Sie den folgenden Datenschritt, der weder SET noch INPUT, INFILE oder LINES enthält, aus.

```
DATA ws;
     alfa=PROBNORM(1.645);
     x=TINV(alfa,9);
RUN;
```

Überprüfen Sie, wieviele Variablen und Beobachtungen die Datei `ws` hat und von welchem Typ die Variablen sind.

❸ Öffnen Sie das Programm `KA07-05.SAS` im Programmfenster und führen Sie den Datenschritt ohne die Anweisung OUTPUT aus. Wie unterscheiden sich die Ergebnisse? Setzen Sie die obere Grenze des Schleifenzählers `i` von 10 auf 100 hoch. Wie verändert sich die Datei?

❹ Öffnen Sie das Programm `KA07-14.SAS` und ändern Sie die WHERE-Anweisung so ab, daß nur `Hans` in der Liste erscheint.

❺ In den beiden Labors (`A` und `B`) wurden identische Gewebeproben analysiert und die darin gefundenen Mengen dreier Substanzen `x`, `y` und `z` protokolliert.

```
Labor A      Labor B
x 100.00     x  90.00
x 110.00     x 105.00
x 105.00     x  98.00
y 110.00     y 120.00
y 100.00     y 130.00
y  95.00     y 110.00
z 200.00     z 180.00
z 170.00     z 170.00
z 150.00     z 140.00
```

Übertragen Sie die Werte in eine SAS-Datei `labor` mit den drei Variablen `labor`, `substanz` und `menge`.

Bilden Sie eine neue Datei sub_x, die nur die Meßwerte der Substanz x enthält.

Geben Sie alle Beobachtungen der Datei sub_x im Ausgabefenster aus, deren Meßwerte kleiner als 100 sind.

Übertragen Sie die Laborwerte sämtlicher Substanzen in die neue Datei pruefen, deren Menge bei den Sustanzen x und y größer als 100 ist und bei z größer als 160.

❻ Untersuchen Sie, was geschieht, wenn Sie in dem Beispielprogramm KA07-16.SAS die Anweisung ELSE weglassen. Welche Ausprägung hat die Variable sex dann bei den Frauen?

❼ Definieren Sie ein Format, mit dem sich die Variable sex der Firmendaten in schönerer Form präsentieren läßt. Geben Sie anstelle von m männlich und anstelle von w weiblich aus.

❽ Verknüpfen Sie die Dateien stammdat und punkte aus Ihrem Übungsverzeichnis C:\kurs zu einer gemeinsamen Datei firma.

❾ Definieren Sie ein permanentes Format $fstand, mit dem sich die Variable famstand der Firmendaten ansprechend darstellen läßt (l-edig, ver-h-eiratet, ver-w-itwet und g-eschieden).

8

Zurück zu den Wurzeln – Statistische Analysen

Die drei Buchstaben „S A S“ erinnern an die ursprüngliche Intention dieses Programmsystems: **S**tatistical **A**nalysis **S**ystem. Für viele Anwender an Universitäten und Fachhochschulen ist die statistische Auswertung auch heute noch das wichtigste Ziel ihrer Beschäftigung mit dem SAS-System. In anderen Einsatzbereichen spielt die Statistik neben dem Datenmanagement ebenfalls eine bedeutende Rolle: Prognosen für das Wirtschaftswachstum eines Betriebs in den nächsten zehn Jahren sollen erstellt werden, die Prämien einer Lebensversicherung an die veränderten Überlebensraten der Bevölkerung angepaßt werden, um nur zwei Beispiele zu nennen.

Die statistische Analyse umfaßt u.a. die drei Bereiche Deskription des vorliegenden Datenmaterials durch die Ermittlung von Kennwerten oder die Darstellung in Form von Grafiken (siehe dazu Kapitel 9), Berechnung von Punkt- und Intervallschätzern für einzelne Merkmale oder Merkmalskombinationen und die Durchführung von statistischen Hypothesentests. Die Auswahl des geeigneten Verfahrens und die dieses Verfahren umsetzende SAS-Prozedur hängt auch vom Meß- oder Skalenniveau der Variablen ab. Bei kategoriellen (oder nominalen) Variablen, wie z. B. Geschlecht oder Wohnort, die nicht in Zahlenwerten dargestellt werden können und auch nicht in eine inhaltlich begründete Reihenfolge zu bringen sind, möchte man Häufigkeiten berechnen, während bei stetigen, intervallskalierten Variablen eher der Mittelwert, der Median und entsprechende Streuungsmaße gefragt sind. Zum Vergleich zweier Gruppen zieht man im ersten Fall den χ^2-Test heran, während bei normalverteilten Grundgesamtheiten der t-Test das geeignete Verfahren ist. Werden mehr als zwei Gruppen verglichen, muß eine Varianzanalyse durchgeführt werden.

Aus der Fülle der Statistikprozeduren, die das SAS-System bietet und die in Abbildung 8.1 thematisch gruppiert angeordnet wurden, können nur einige wenige Prozeduren vorgestellt werden. Mit der Prozedur FREQ werden Kreuztabellen oder Kontingenztafeln aufgestellt, aus denen die absoluten und relativen Häufigkeiten kategorieller Variablen abgelesen sowie die darauf aufbauenden statistischen Tests (χ^2- und Fishers exakten Test) durchgeführt werden. Darüber hinaus lassen sich verschiedene Zusammenhangsmaße, wie etwa der Phi-Koeffizient, bestimmen. Mit den beiden Prozeduren MEANS und UNIVARIATE werden Kennwerte berechnet, Konfidenzintervalle für den Mittelwert bestimmt, Tests von Mittelwerten gegen einen festen Wert bzw. Tests von Mittelwerten zweier abhängiger Stichproben und der Test auf Normalverteilung durchgeführt sowie grafische Darstellungen der empirischen Verteilung einer Stichprobe erzeugt. Die Prozedur TTEST führt den gleichnamigen Hypothesentest zum Vergleich zweier unabhängiger Stichproben durch, kann aber auch die Homogenität zweier Varianzen testen. Die äquivalenten nichtparametrischen Verfahren werden mit der Prozedur NPAR1WAY umgesetzt. Die Prozedur CORR berechnet Zusammenhangsmaße (Pearson, Spearman). Im Anhang erfolgt eine kurze Einführung in die Prozedur INSIGHT.

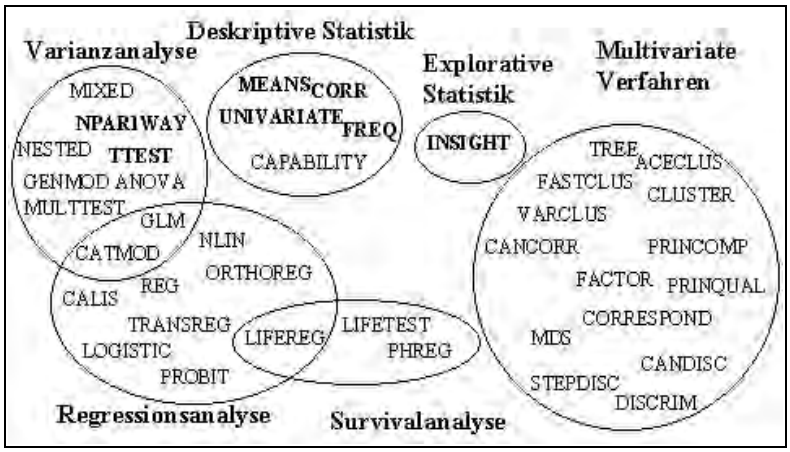


Abb. 8.1: Statistische Prozeduren im SAS-*System*

Während MEANS und FREQ Prozeduren des Basismoduls SAS/BASE sind, muß für die anderen Prozeduren das Modul SAS/STAT oder SAS/INSIGHT lizensiert und installiert sein. Entsprechend finden Sie die Beschreibungen der Prozeduren im Procedures Guide [11], in den SAS/STAT-Handbüchern [16] und [17] und im SAS/INSIGHT-Handbuch [15]. Eine ausführliche Einleitung zu den Statistikprozeduren finden Sie u. a. im SAS/STAT-Handbuch [16, Volume 1, Kap. 1-8].

Bei den folgenden Erläuterungen wird vorausgesetzt, daß dem Leser die statistischen Grundbegriffe wie Schätzer, Konfidenzintervall oder Hypothesentest zumindest ansatzweise vertraut sind. Ist dies nicht der Fall, wird empfohlen, bei Bedarf in einem einschlägigen Statistiklehrbuch (z. B. Bortz [1] oder Hartung [3]) nachzuschlagen.

8.1 Kontingenztafelanalyse – Die Prozedur FREQ

Kategorielle Variablen wie z. B. Geschlecht, Familienstand und Abteilungszugehörigkeit analysiert man im SAS-System mit der Prozedur FREQ. Mit dieser Prozedur werden Kreuztabellen erstellt, die die Häufigkeiten einzelner und kreuzklassifizierter kategorieller Merkmale aufzeigen, und der χ^2-Test (zur Überprüfung, ob die Auftrittshäufigkeiten eines Merkmals der Erwartung entsprechen oder ob zwei Merkmale voneinander abhängig sind bzw. ob sich die Häufigkeiten eines Merkmals zwischen zwei Gruppen unterscheiden) durchgeführt.

Natürlich können auch für ordinale und intervallskalierte Merkmale Häufigkeiten berechnet und die entsprechenden Tests durchgeführt werden. Haben die Merkmale jedoch viele Ausprägungen, sind die einzelnen Zellbesetzungszahlen der Kreuztabelle gering und der statistische Test nicht aussagekräftig. In solch einem Fall können die Ausprägungen, z. B. mit der Prozedur FORMAT (vgl. Abschnitt 7.5.2), klassifiziert werden.

Das folgende Beispiel greift auf die Stammdaten der Firmenmitarbeiter zu. Der Personalleiter möchte für weitere Personaleinstellungen wissen, ob bisher ungefähr gleich viele Frauen wie Männer in der Firma beschäftigt sind und ob es bedeutsame Unterschiede im Familienstand der Frauen und Männer gibt. Dazu wird zunächst mit der Prozedur FREQ ausgezählt, wieviele weibliche und männliche Mitarbeiter in der Firma arbeiten.

```
/*--- KA08-01.SAS ---*/
LIBNAME biblio 'C:\kurs';
PROC FREQ DATA=biblio.stammdat;
     TABLES sex;
RUN;
```

Mit der LIBNAME-Anweisung wird die Bibliothek referenziert, die die Daten enthält, so daß in der Anweisung PROC FREQ die Spezifikation der Option DATA= genügt. Die Anweisung TABLES der Prozedur FREQ legt die Variable fest, hier sex, für die die Häufigkeiten berechnet werden.

```
                                Cumulative Cumulative
SEX   Frequency   Percent   Frequency    Percent
-----------------------------------------------
m           19      61.3          19       61.3
w           12      38.7          31      100.0
```

Im Ausgabefenster erscheint keine Kreuztabelle (denn es wurden keine zwei Merkmale gekreuzt), sondern lediglich eine Tabelle der beiden einzigen Ausprägungen der Variable sex und die jeweils zugehörigen absoluten (Frequency) und relativen (Percent), sowie die kumulativen (Cumulative) absoluten und relativen Häufigkeiten der beiden Ausprägungen. In der Firma herrscht ein leichtes Ungleichgewicht: 12 Frauen gegenüber 19 Männern. Damit sind ca. 1/3 mehr Männer als Frauen beschäftigt.

Um zu überprüfen, ob sich diese These auch statistisch halten läßt, muß ein Test durchgeführt werden, der die Hypothese prüft, daß die erwarteten Häufigkeiten p gleich 0.5 sind: $H_0 : p_w = p_m (= 0.5)$. Bei insgesamt 31 Mitarbeitern würde man bei einer gleichmäßigen Verteilung der beiden Geschlechter erwarten, daß 15.5 weiblich und 15.5 männlich sind. Der Test wird mit dem χ^2-Test durchgeführt, der die erwarteten Häufigkeiten (E) mit den beobachteten Häufigkeiten (B) vergleicht. In Formeln ausgedrückt:

$$\chi^2 = \sum_{i=1}^{R} \frac{(B-E)^2}{E}$$

Summiert wird über die beiden Ausprägungen der Variablen sex ($R = 2$). Um den Test mit der Prozedur FREQ zu rechnen, erweitert man die Anweisung TABLES nach dem Variablennamen um einen Schrägstrich und die Option CHISQ.

```
PROC FREQ DATA=biblio.stammdat;
     TABLES sex / CHISQ;
RUN;
```

Im Ausgabefenster erscheint nun zusätzlich zur bereits dargestellten Tabelle das Ergebnis des χ^2-Tests auf gleiche Proportionen.

```
Chi-Square Test for Equal Proportions
-------------------------------------
Statistic = 1.581     DF = 1     Prob = 0.209
```

Die Teststatistik (Statistic) errechnet sich nach obiger Formel. Der Freiheitsgrad (DF=$R-1$) beträgt in diesem Fall eins, da die Variable sex zwei Ausprä-

gungen hat. (Würde man eine der beiden Häufigkeiten festlegen, etwa die der Frauen, würde sich die andere aus der gegebenen Anzahl der Mitarbeiter insgesamt ermitteln lassen: 31-12. Es bleibt daher nur ein Freiheitsgrad übrig.)

Die Teststatistik χ^2 ist unter der Nullhypothese approximativ (d. h. für hinreichend große Stichproben) χ^2-verteilt. Mit `Prob` wird das zur ermittelten Teststatistik gehörende Quantil der χ^2-Verteilung berechnet. Sie können dies mit der Funktion PROBCHI (siehe Tabelle 7.2) nachrechnen:

```
DATA _NULL_;
      p=1-PROBCHI(1.581,1);
      PUT p;
RUN;
```

Dieser p-Wert gibt die Wahrscheinlichkeit der Teststatistik unter der Nullhypothese an. Mit 0.2 wie im vorliegenden Fall kann die Nullhypothese, daß gleichviele Frauen wie Männer in der Firma arbeiten, auf dem 5%-Signifikanzniveau nicht abgelehnt werden.

Zum Überprüfen der Nullhypothese, daß in der Firma 2/3 Männer und 1/3 Frauen beschäftigt sind, müssen über eine zusätzliche Option in der TABLES-Anweisung diese erwarteten Häufigkeiten übergeben werden.

```
PROC FREQ DATA=biblio.stammdat;
     TABLES sex / TESTP=(0.67,0.33);
RUN;
```

Die Option CHISQ braucht nicht angegeben zu werden, da sie durch TESTP= implizit eingeschlossen ist. Die erwarteten relativen Häufigkeiten, „P“ für Proportion, werden in der Reihenfolge ihres Auftretens in der Tabelle in Klammern, durch Komma getrennt, aufgeführt. Sie müssen sich zu 1 addieren lassen. Die Ausgabe unterscheidet sich leicht von obiger Form:

```
                                    Test     Cumulative Cumulative
SEX    Frequency     Percent     Percent    Frequency     Percent
-------------------------------------------------------------------
m             19        61.3        67.0           19        61.3
w             12        38.7        33.0           31       100.0
```

```
Chi-Square Test for Specified Proportions
-----------------------------------------
Statistic = 0.457     DF = 1     Prob = 0.499
```

Als zusätzliche Spalte werden die vorgegebenen Häufigkeiten (`Test Percent`) ausgegeben. Das Testergebnis hat sich deutlich in Richtung Nullhypothese verschoben: Der p-Wert liegt mit 0.499 aber dennoch jenseits des 5%-Signifikanzbereichs. (Mit der Option TESTF= könnten auch absolute erwartete Häufigkeiten vorgegeben werden. Diese müßten sich zur Gesamtanzahl der Beobachtungen addieren.)

Im nächsten Schritt zieht der Personalchef den Familienstand in seine Untersuchung hinzu. Zunächst zählt er beide Variablen getrennt aus. Er erweitert dazu die Anweisung TABLES um die zweite Variable `famstand` und, damit diese in ansprechender Form erscheint, verknüpft er diese Variable mit dem im vorangegangenen Kapitel definierten permanenten Format `$fstand.` über die Anweisung FORMAT (vgl. Aufgabe ❾, Kapitel 7).

```
/*--- KA08-02.SAS ---*/
LIBNAME LIBRARY 'C:\kurs';
PROC FREQ DATA=biblio.stammdat ORDER=DATA;
     TABLES famstand sex;
     FORMAT famstand $fstand.;
RUN;
```

Die Option ORDER= der Anweisung PROC FREQ bestimmt die Reihenfolge, in der die Ausprägungen der zu tabellierenden Variablen aufgeführt werden. `ORDER=DATA` besagt, daß die Ausprägungen in der Reihenfolge aufgeführt werden, in der sie in der Datei auftreten, wenn diese von oben nach unten abgearbeitet wird. (Die Option `ORDER=FREQ` gibt die Werte nach Häufigkeiten sortiert aus und würde an dieser Stelle zufällig zum gleichen Resultat führen.)

FAMSTAND	Frequency	Percent	Cumulative Frequency	Cumulative Percent
ledig	16	51.6	16	51.6
verheiratet	11	35.5	27	87.1
geschieden	3	9.7	30	96.8
verwitwet	1	3.2	31	100.0

Der Personalleiter kann an dieser Auszählung ablesen, daß über 80% der Belegschaft ledig oder verheiratet ist. Nur insgesamt vier Mitarbeiter sind geschieden

oder verwitwet. (Die Tabelle für die Variable sex unterscheidet sich nicht von obiger Darstellung und wurde hier nicht wiedergegeben.)

Um die beiden Variablen famstand und sex nun zu kreuzklassifizieren, was bedeutet, daß ausgezählt wird, wie häufig die einzelnen Ausprägungskombinationen auftreten, müssen die Variablen in der Anweisung TABLES mit einem Stern „*“ verbunden werden:

```
/*--- KA08-03.SAS ---*/
PROC FREQ DATA=biblio.stammdat ORDER=DATA;
     TABLES famstand*sex;
     FORMAT famstand $fstand.;
RUN;
```

Nach der Ausführung dieses Prozedurschritts erscheint im Ausgabefenster die auf der nächsten Seite wiedergegebene Kreuztabelle mit den beiden gekreuzten Merkmalen.

In den 2x4=8 Feldern (oder Zellen) der Kontingenztafel erscheinen jeweils vier Zahlen: Die oberste, erste Zahl gibt die absolute Häufigkeit an, die darunterfolgende die relative Häufigkeit, also den Anteil- oder Prozentwert, den diese Zelle an der Gesamtzahl der Beobachtungen hat. Anhand dieser Werte kann der Personalleiter nun beispielsweise leicht abgelesen, wieviele der weiblichen Mitarbeiter verheiratet sind (7) und wieviele der männlichen Kollegen (4). Die weiteren beiden Werte geben die relativen Häufigkeiten der Zelle an der jeweiligen Zeile bzw. an der jeweiligen Spalte an (Row Pct, Col Pct). Aus der zweiten Zelle in der zweiten Zeile kann man etwa ablesen, daß 58.33% der Frauen in der Firma verheiratet sind, im Unterschied zu den männlichen Kollegen, wo die Mehrheit ledig ist (63.16%). Die Gruppe der ledigen Männer macht ein gutes Drittel der gesamten Belegschaft aus (38.71%). Aus der Tabelle lassen sich anhand der Randsummen aber auch die absoluten und relativen Häufigkeiten der beiden beteiligten Variablen famstand und sex ablesen, die zuvor in Form einer Liste ausgegeben wurden.

Um die Spalten und Zeilen der Tabelle zu tauschen, d. h. die Ausprägungen der Variablen sex in den Zeilen und die von famstand in den Spalten einzutragen, tauscht man die beiden Variablen in der TABLES-Anweisung:

```
    TABLES sex*famstand;
```

TABLE OF FAMSTAND BY SEX

FAMSTAND / Frequency / Percent / Row Pct / Col Pct	SEX: m	SEX: w	Total
ledig	12	4	16
	38.71	12.90	51.61
	75.00	25.00	
	63.16	33.33	
verheiratet	4	7	11
	12.90	22.58	35.48
	36.36	63.64	
	21.05	58.33	
geschieden	2	1	3
	6.45	3.23	9.68
	66.67	33.33	
	10.53	8.33	
verwitwet	1	0	1
	3.23	0.00	3.23
	100.00	0.00	
	5.26	0.00	
Total	19	12	31
	61.29	38.71	100.00

Daß sich das Verhältnis von ledigen zu verheirateten Männern deutlich von dem der Frauen unterscheidet, kann aus der Tabelle abgelesen werden. Soll nun auch statistisch gesichert werden, ob es einen signifikanten Unterschied zwischen Frauen und Männern hinsichtlich der Verteilung von Ledigen und Verheirateten gibt oder aus einem anderen Blickwinkel betrachtet, ob es einen Zusammenhang zwischen den Merkmalen Geschlecht und Familienstand in dem Betrieb gibt, muß ein Hypothesentest durchgeführt werden.

Der bekannteste Test für kreuzklassifizierte Merkmale ist der bereits oben behandelte χ^2-Test. Er ist auch dann angezeigt, wenn zwei Gruppen hinsichtlich eines Merkmals klassifiziert werden oder eine Gruppe hinsichtlich zweier Merk-

male kreuzklassifiziert wird, aber auch, wenn mehr als zwei Gruppen oder mehr als zwei Merkmale kreuzklassifiziert werden.

Der Test wird mit der Prozedur FREQ durchgeführt, indem die Option CHISQ in der Anweisung TABLES gesetzt wird.

```
PROC FREQ DATA=biblio.stammdat ORDER=DATA;
     TABLES famstand*sex / CHISQ;
     FORMAT famstand $fstand.;
RUN;
```

Zusätzlich zur Kreuztabelle erscheint im Ausgabefenster eine Tabelle, in der die statistischen Tests aufgeführt werden.

```
STATISTICS FOR TABLE OF SEX BY FAMSTAND

Statistic                     DF     Value        Prob
------------------------------------------------------
Chi-Square                     3     4.816       0.186
Likelihood Ratio Chi-Square    3     5.146       0.161
Mantel-Haenszel Chi-Square     1     0.338       0.561
Phi Coefficient                      0.394
Contingency Coefficient              0.367
Cramer's V                           0.394

Sample Size = 31
WARNING: 63 of the cells have expected counts less
         than 5. Chi-Square may not be a valid test.
```

In der ersten Zeile wird das Ergebnis des χ^2-Tests aufgeführt. Neben dem Freiheitsgrad (DF, engl. *degree of freedom*) wird der Testwert (Value) und der p-Wert (Prob) angegeben, der die Wahrscheinlichkeit angibt, diesen Testwert unter der Nullhypothese zu erhalten. Er besagt, daß die Nullhypothese auf dem 5%-Signifikanzniveau nicht abgelehnt werden kann, da 0.186>0.05. Neben dem χ^2-Test werden zwei weitere Tests (der Likelihood-Ratio Test und der Mantel-Haenszel Test) und drei Zusammenhangsmaße (der Phi-Koeffizient, der Kontingenzkoeffizient und Cramers V) berechnet.

Für den Testwert des oben durchgeführten χ^2-Tests zum Vergleich der Häufigkeiten der Variable sex wurden die Differenzen zu den erwarteten Häufigkeiten berechnet. Dieses Prinzip läßt sich auch auf die Kontingenztafel übertragen: Für jede Zelle wird zunächst die Differenz zwischen der unter der Nullhypothese erwarteten Häufigkeit (E) und der beobachteten Häufigkeit (B) berechnet, quadriert und durch die erwartete Häufigkeit dividiert. Anschließend werden die

Werte aller Zellen aufsummiert. In Formeln ausgedrückt berechnet sich dieser Testwert wie folgt:

$$\chi^2 = \sum_{i=1}^{R} \sum_{j=1}^{S} \frac{(E_{ij} - B_{ij})^2}{E_{ij}}$$

wobei R die Anzahl der Zeilen der Kreuztabelle ist, d. h. die Anzahl der Ausprägungen der ersten Variable der TABLES-Anweisung und S die Anzahl der Spalten (Ausprägungen) der zweiten Variablen. Diese Teststatistik ist approximativ χ^2-verteilt mit $(R-1)(S-1)$ Freiheitsgraden. Im obigen Fall hatte das Merkmal `famstand` vier Ausprägungen, das Merkmal `sex` zwei. Der Freiheitsgrad berechnete sich dadurch zu DF=(4-1)*(2-1)=3*1=3.

Mit den zusätzlichen Optionen EXPECTED, DEVIATION und CELLCHI2 der Anweisung TABLES werden in der Kontingenztafel auch die erwarteten Häufigkeiten (`Expected`), die Differenzen zwischen beobachteten und erwarteten Häufigkeiten (`Deviation`) sowie die Beiträge ausgegeben, die jede einzelne Zelle zum Gesamt-χ^2-Wert beiträgt (`Cell Chi-Square`). Damit die Kreuztabelle nicht zu groß und unübersichtlich wird, kann man gleichzeitig mit den Optionen NOPERCENT, NOROW und NOCOL die Ausgabe der relativen Häufigkeiten sowie der Zeilen- und Spaltenprozentwerte unterdrücken. Natürlich könnte jede Option auch für sich gesetzt werden, da keine in ihrer Funktion von einer der anderen abhängig ist.

```
/*--- KA08-04.SAS ---*/
PROC FREQ DATA=biblio.stammdat ORDER=DATA;
     TABLES famsstand*sex / EXPECTED DEVIATION CELLCHI2
                            NOCOL NOROW NOPERCENT;
     FORMAT famstand $fstand.;
RUN;
```

```
TABLE OF FAMSTAND BY SEX

FAMSTAND        SEX

Frequency      |
Expected       |
Deviation      |
Cell Chi-Square|m       |w       |  Total
---------------+--------+--------+
ledig          |     12 |      4 |     16
               | 9.8065 | 6.1935 |
               | 2.1935 | -2.194 |
               | 0.4907 | 0.7769 |
---------------+--------+--------+
verheiratet    |      4 |      7 |     11
               | 6.7419 | 4.2581 |
               | -2.742 | 2.7419 |
               | 1.1151 | 1.7656 |
---------------+--------+--------+
geschieden     |      2 |      1 |      3
               | 1.8387 | 1.1613 |
               | 0.1613 | -0.161 |
               | 0.0141 | 0.0224 |
---------------+--------+--------+
verwitwet      |      1 |      0 |      1
               | 0.6129 | 0.3871 |
               | 0.3871 | -0.387 |
               | 0.2445 | 0.3871 |
---------------+--------+--------+
Total               19       12       31
```

Die χ^2-Einzelwerte zeigen auf, in welcher Zelle die beobachteten Häufigkeiten besonders stark von den erwarteten abweichen. Bei signifikanten Testergebnissen geben sie Aufschluß darüber, welche Kategorien die Unterschiede zwischen den Gruppen, die Heterogenitäten, verursachen.

Die Beurteilung der erwarteten Werte garantiert, daß der χ^2-Test nur dann eingesetzt wird, wenn er zulässig ist: Der zur Teststatistik gehörende p-Wert, in obigem Beispiel 0.186, würde die Ablehnung der Nullhypothese auf dem 5%-Niveau nicht erlauben, wenn nicht am Ende der Ausgabe die folgende Warnung erscheinen würde:

```
Statistic                     DF     Value        Prob
------------------------------------------------------
Chi-Square                     3     4.816       0.186
...
WARNING: 63 of the cells have expected counts less
         than 5. Chi-Square may not be a valid test.
```

Der χ^2-Test basiert auf der Approximation der Verteilung der Teststatistik durch die χ^2-Verteilung. Sind die erwarteten Häufigkeiten in den Zellen sehr klein oder liegen sehr viele Kategorien vor, von denen manche mit hoher Wahrscheinlichkeit gar nicht besetzt werden können, sollte der Aussage des χ^2-Tests nicht zuviel Gewicht beigemessen werden. Eine Faustregel, nach der das SAS-System urteilt, besagt, daß die Erwartungswerte größer als 5 sein müssen und es keine leeren Zellen geben darf. In obigem Beispiel sind bei 63% der Zellen (5 von 8) die erwarteten Häufigkeiten kleiner als 5. Dem Personalleiter bleiben nun folgende Auswege:

❶ Er läßt die kleine Gruppe der Geschiedenen und Verwitweten außer Acht und vergleicht nur die Ledigen mit den Verheirateten: (`WHERE famstand IN ('l' 'h');`). Aber: Auch in diesem Fall bliebe noch eine Zelle mit erwarteter Häufigkeit kleiner als 5 übrig.

❷ Er zählt die vier Geschiedenen und Verwitweten zu den Ledigen oder zu den Verheirateten dazu. (Am einfachsten geht dies, indem er ein zweites Format für die Variable `famstand` definiert und die Option OTHER= in der Anweisung VALUE verwendet.)

❸ Er rechnet einen alternativen Test, der auch mit leeren Zellen und kleinen erwarteten Häufigkeiten umgehen kann: Fishers exakten Test.

Wenn, wie im zweiten Fall, der χ^2-Test für eine Vierfeldertafel angefordert wird, wird Fishers exakter Test automatisch mit ausgegeben. Bei größeren Kontingenztafeln muß die Anweisung TABLES um die Option EXACT erweitert werden:

```
PROC FREQ DATA=biblio.stammdat;
     TABLES famstand*sex / EXACT;
RUN;
```

Haben die Merkmale mehr als zwei Ausprägungen, wird nur das zweiseitige Testergebnis ausgegeben.

```
STATISTICS FOR TABLE OF FAMSTAND BY SEX

Statistic                     DF      Value        Prob
------------------------------------------------------
Chi-Square                     3      4.816       0.186
Likelihood Ratio Chi-Square    3      5.146       0.161
Mantel-Haenszel Chi-Square     1      0.338       0.561
Fisher's Exact Test (2-Tail)                      0.116
Phi Coefficient                       0.394
Contingency Coefficient               0.367
Cramer's V                            0.394

Sample Size = 31
WARNING: 63 of the cells have expected counts less
         than 5. Chi-Square may not be a valid test.
```

Der Test ist, wie der Name bereits andeutet, ein exaktes Verfahren. Für seine Durchführung wird keine Teststatistik berechnet, sondern auf Basis der hypergeometrischen Verteilung die Wahrscheinlichkeiten der vorliegenden und daraus abgeleiteten Tafeln zu den gegebenen Randverteilungen berechnet. Im Ausgabefenster erscheint somit weder eine Teststatistik noch ein Freiheitsgrad, sondern lediglich der p-Wert. Auf dem 5%-Signifikanzniveau kann die Nullhypothese, daß sich die Frauen und Männer in der Firma hinsichtlich des Familienstandes unterscheiden, somit nicht abgelehnt werden (p-Wert `0.116` ist größer als 0.05.)

Bislang lag den Beispielen die Datei `stammdat` mit ihren 31 Einzelbeobachtungen zugrunde, und die Prozedur FREQ mußte die Häufigkeiten der einzelnen Zellen jeweils neu bestimmen. Eine andere Situation liegt vor, wenn die Zellhäufigkeiten bereits außerhalb des Systems ausgezählt wurden und ein χ^2-Test für diese aggregierten Daten gerechnet werden soll.

Mit einer anonymen Befragung möchte der Personalchef die Motivation seiner Mitarbeiter erfragen hinsichtlich eines Wechsels des Firmenstandorts in die USA. Dazu erhebt er, ob die Mitarbeiter im derzeitigen Einzugsgebiet der Firma geboren wurden. Nach Auszählung der Stimmen hat er folgendes Bild vor Augen:

		Zu Wechsel bereit		Insgesamt
		Ja	Nein	
Geboren im	Ja	3	17	20
Einzugsgebiet	Nein	7	4	11
Insgesamt		10	21	31

Den Personalleiter wundert das Ergebnis nicht, hat er doch erwartet, daß die Mitarbeiter, die bereits mindestens einmal umgezogen sind, um in der Firma am gegenwärtigen Standort zu arbeiten, auch eher bereit sind, in die USA mitzuwechseln, als die bodenständigen, die schon immer am gleichen Ort leben. Um die Aussage statistisch abzusichern, möchte er einen χ^2-Test rechnen. Er definiert zunächst ein Format für die beiden Ja/Nein-Antworten. Anschließend trägt er die Daten in die SAS-Datei `umfrage` ein und weist den Variablen das Format und Etiketten zu.

```
/*--- KA08-05.SAS ---*/
PROC FORMAT;
     VALUE jn 1=Ja
              0=Nein;
RUN;
DATA umfrage;
     INPUT wechsel hier anzahl @@;
     LABEL wechsel='Bereit zum Wechsel ?'
           hier='Hier geboren ?';
     FORMAT wechsel hier jn.;
     LINES;
1 1 3 1 0 17 0 1 7 0 0 4
RUN;
```

Die Datei `umfrage` enthält drei Variablen: `wechsel` für die Bereitschaft eines Arbeitsplatzwechsels in die USA, `hier` für die Frage nach der Geburt im Einzugsgebiet der Firma und die Variable `anzahl`, die angibt, wie häufig die einzelnen Merkmalskombinationen von `wechsel` und `hier` aufgetreten sind.

```
/*--- KA08-06.SAS ---*/
PROC FREQ DATA=umfrage;
     TABLES hier*wechsel / NOROW NOCOL;
     WEIGHT anzahl;
RUN;
```

Die WEIGHT-Anweisung im Prozedur FREQ-Schritt benennt die Variable, `anzahl`, und zeigt dem System an, daß bereits aggregierte Daten vorliegen. In der Kreuztabelle werden somit die Werte der Variable `anzahl` eingetragen.

```
TABLE OF WECHSEL BY HIER
WECHSEL(Bereit zum Wechsel ?)
          HIER(Hier geboren ?)
Frequency|
Percent  |Nein    |Ja      |  Total
---------+--------+--------+
Nein     |      4 |      7 |     11
         |  12.90 |  22.58 |  35.48
---------+--------+--------+
Ja       |     17 |      3 |     20
         |  54.84 |   9.68 |  64.52
---------+--------+--------+
Total          21       10       31
            67.74    32.26   100.00
```

Würde die Anweisung WEIGHT vergessen werden, wären in der Darstellung alle Zellen mit Häufigkeit 1 besetzt. Als nächstes folgt die Durchführung des Tests. Da die erwartete Häufigkeit der Zelle rechts oben kleiner als 5 ist (11*10/31= 3.55), kann das Ergebnis des χ^2-Tests nicht berücksichtigt werden, sondern es muß der exakte Test von Fisher herangezogen werden, indem die Option EXACT in der Anweisung TABLES ergänzt wird. Im Ausgabefenster erscheint daraufhin:

```
STATISTICS FOR TABLE OF WECHSEL BY HIER

Statistic                     DF      Value        Prob
------------------------------------------------------
Chi-Square                     1      7.682       0.006
Likelihood Ratio Chi-Square    1      7.657       0.006
Continuity Adj. Chi-Square     1      5.618       0.018
Mantel-Haenszel Chi-Square     1      7.434       0.006
Fisher's Exact Test (Left)                    9.21E-03
                    (Right)                      0.999
                    (2-Tail)                     0.013
Phi Coefficient                      -0.498
Contingency Coefficient               0.446
Cramer's V                           -0.498

Sample Size = 31
```

```
WARNING: 25 of the cells have expected counts less
         than 5. Chi-Square may not be a valid test.
```

Der zweiseitige p-Wert von 0.013 unterstützt die These des Personalleiters, so daß dieser auf dem 5%-Signifikanzniveau die Nullhypothese verwerfen kann. Er kann mit hoher Wahrscheinlichkeit davon ausgegehen, daß es einen Zusammenhang zwischen Geburtsort und Wechselfreudigkeit gibt. (Der einseitige p-Wert des exakten Fisher Tests wird in der Form `9.21E-03` dargestellt, was nichts anderes bedeutet als $9.21 \cdot 10^{-3}$, also 0.00921.)

Werden mehrere Variablen gegeneinander kreuzklassifiziert, kann man in der Anweisung TABLES diese Anforderungen gleichzeitig angeben:

```
TABLES sex*famstand sex*wohnort;
```

oder kürzer:

```
TABLES sex*(famstand wohnort);.
```

Die Dreifachkombination `TABLES sex*famstand*wohnort;` erzeugt Kreuztabellen für die Variablen `famstand` und `wohnort`, getrennt für die Ausprägungen der Variablen `sex`.

Die Ausprägungen werden standardmäßig, d. h. ohne Zutun des Anwenders, bei alphanumerischen Variablen in alphabetischer Reihenfolge aufgelistet, bei numerischen Variablen in aufsteigender Folge. In obigem Beispiel erscheint `Nein` vor `Ja`, da die Zahlen mit `0` und `1` kodiert wurden.

Mit der Option ORDER= der Anweisung PROC FREQ kann diese Standardreihenfolge verändert werden. ORDER=DATA beläßt die Ausprägungen in ihrer Originalreihenfolge, also in der Reihenfolge, in der sie in der Datei auftreten. ORDER=FORMATTED ordnet die Ausprägungen entsprechend dem verwendeten Format. ORDER=FREQ schließlich ordnet die Ausprägungen nach der absoluten Häufigkeit ihres Auftretens.

Die Syntax der Prozedur FREQ hat die folgende Form:

```
PROC FREQ DATA=Datei <ORDER=DATA|FORMATTED|FREQ>;
     TABLES <(>Variablen-Liste <<)>*<(>Variablen-Liste<)>>
                  </ <NOCOL> <NOROW> <NOPERCENT>
                     <EXPECTED> <DEVIATION> <CELLCHI2>
                     <CHISQ> <EXACT>>;
```

8.2 Mittelwerte und anderes – Die Prozedur MEANS

Die Prozedur MEANS berechnet Kennwerte, die in Zusammenhang mit dem Mittelwert stehen: Mittelwert, Standardabweichung, Varianz, Konfidenzintervalle für den Mittelwert und den verbundenen t-Test zum Vergleich eines Mittelwerts mit einem festen Wert.

Mit dem PROC MEANS-Aufruf ohne weitere Anweisungen, nur der Option DATA=, werden die Standardkennwerte aller numerischen Variablen (hier für die Firmendaten `biblio.firma`) im Ausgabefenster aufgelistet.

```
/*--- KA08-07.SAS ---*/
LIBNAME biblio 'D:\kurs';
PROC MEANS DATA=biblio.firma;
RUN;
```

```
Variable  N        Mean     Std Dev    Minimum    Maximum
-----------------------------------------------------------
NR       31    1016.00   9.0921211    1001.00    1031.00
ALTER    31 37.1290323  11.3892989 19.0000000 58.0000000
KINDER    9  1.6666667   0.7071068  1.0000000  3.0000000
EINTRITT 31   12269.97 722.9415137   11109.00   13484.00
TEIL1    31  6.0645161   2.3228644  1.0000000 10.0000000
TEIL2A   31  5.8387097   2.8413233  1.0000000 10.0000000
TEIL2B   31  6.7419355   2.0325311  1.0000000 10.0000000
TEIL3    31  6.3870968   1.6467629  3.0000000 10.0000000
TEIL4    31 11.8709677   4.0966811  2.0000000 19.0000000
```

Da die Variable `nr` numerisch kodiert wurde, werden auch für sie die Kennwerte berechnet:

- `N`, die Anzahl der nichtfehlenden Werte in der Datei;
- `Mean`, der Mittelwert, und `Std Dev`, die Standardabweichung; sowie
- `Minimum` und `Maximum`, der kleinste und der größte Wert.

Die Anzahl der fehlenden Werte entspricht in fast allen Fällen der Anzahl der Beobachtungen in der Datei `firma`. Nur bei `KINDER` erscheint die Zahl 9. Das heißt, daß bei den restlichen 31-9=22 Mitarbeitern ein fehlender Wert eingetragen ist, was so viel bedeutet wie „keine Kinder“ oder fehlende Angabe. Die durchschnittliche Kinderzahl, 1.66, wird dabei auch nur für diese 9 Mitarbeiter berechnet. Die fehlenden Werte gehen nicht in die Berechnung ein.

Das Durchschnittsalter der Mitarbeiter liegt bei 37 Jahren. Der jüngste Mitarbeiter ist 19 Jahre alt, der älteste 58. Wie bei der Variablen NR macht die Kennwertberechnung auch beim Eintrittsdatum wenig Sinn, zumindest solange die Durchschnittswerte nicht in Monate oder Jahre umgerechnet werden.

Mit der Anweisung VAR kann die Liste der Variablen eingeschränkt werden. Im folgenden Beispiel werden nur die Kennwerte für die einzelnen Fortbildungsabschnitte `teil1, teil2a ...` berechnet.

```
/*--- KA08-08.SAS ---*/
LIBNAME biblio 'D:\kurs';
PROC MEANS DATA=biblio.firma;
     VAR teil1--teil4;
RUN;
```

Diese Einschränkung ist dann sinnvoll, wenn numerisch kodierte Variablen vorliegen, die eigentlich nominalskaliert sind, und die Berechnung der Kennwerte sinnlos wäre. Oder wenn in der Datei sehr viele Variablen vorhanden sind und nur von einzelnen die Kennwerte berechnet werden müssen. Mit der VAR-Anweisung reduziert sich die Zeit, die das SAS-System braucht, um die Kennwerte zu berechnen und für das Ausgabefenster aufzubereiten.

Mit der CLASS-Anweisung werden die Kennwerte getrennt für die Ausprägungen der CLASS-Variablen berechnet. Im folgenden Beispiel wird das Alter und die Kinderzahl getrennt nach dem Geschlecht berechnet. Die CLASS-Variable ist `sex`, die Variablen, für die die Kennwerte berechnet werden sollen, `alter` und `kinder`.

```
/*--- KA08-09.SAS ---*/
PROC MEANS DATA=biblio.firma;
     CLASS sex;
     VAR alter kinder;
RUN;
```

Die Ergebnistabelle wird um drei Spalten erweitert: `SEX` zeigt die beiden Ausprägungen `m` und `w` der CLASS-Variablen, `N Obs` gibt die Anzahl der Beobachtungen an und `Variable` gibt die Variablen an, für die die Kennwerte gelten.

```
SEX   N Obs  Variable   N          Mean       Std Dev
-------------------------------------------------------
m        19  ALTER     19    36.5789474    11.5485246
             KINDER     4     1.5000000     0.5773503

w        12  ALTER     12    38.0000000    11.5836876
             KINDER     5     1.8000000     0.8366600
-------------------------------------------------------

SEX   N Obs  Variable       Minimum       Maximum
-----------------------------------------------
m        19  ALTER       21.0000000    58.0000000
             KINDER       1.0000000     2.0000000

w        12  ALTER       19.0000000    56.0000000
             KINDER       1.0000000     3.0000000
-----------------------------------------------
```

Mit der BY-Anweisung werden die Kennwerte ebenfalls getrennt für die Ausprägungen der BY-Variablen berechnet. Die Datei muß zuvor, analog zur Prozedur PRINT, nach der Variablen sortiert werden.

Um das durchschnittliche Alter der Mitarbeiter getrennt nach Geschlecht zu berechnen, wird die Datei `biblio.firma` zunächst nach der Variablen `sex` sortiert und unter dem Namen `firma` als temporäre Datei abgespeichert. Anschließend wird die Prozedur MEANS mit der BY-Anweisung aufgerufen.

```
/*--- KA08-10.SAS ---*/
PROC SORT DATA=biblio.firma out=firma;
     BY sex;
RUN;
PROC MEANS DATA=firma;
     VAR alter;
     BY sex;
RUN;
```

Im Ausgabefenster erscheint das folgende Ergebnis:

```
Analysis Variable : ALTER

 SEX=m

 N          Mean       Std Dev       Minimum       Maximum
-------------------------------------------------------------
19    36.5789474    11.5485246    21.0000000    58.0000000
-------------------------------------------------------------

 SEX=w

 N          Mean       Std Dev       Minimum       Maximum
-------------------------------------------------------------
12    38.0000000    11.5836876    19.0000000    56.0000000
-------------------------------------------------------------
```

Getrennt nach Männern (SEX=m) und Frauen (SEX=w) werden die Mittelwerte und die anderen Kennwerte für die Analysevariable ALTER in separaten Tabellen dargestellt. Die Reihenfolge der BY-Variablen hängt von der Sortierreihenfolge ab. Mit der Option DESCENDING könnte man die Reihenfolge umkehren.

Ist Ihnen aufgefallen, daß sowohl die Anweisung CLASS als auch BY für eine getrennte Kennwertberechnung sorgen? Während die Anweisung BY allerdings voraussetzt, daß die Datei entsprechend der BY-Variablen sortiert ist, muß dies bei CLASS nicht gegeben sein. Außerdem ist die Darstellung mit CLASS kompakter, wenn man davon absieht, daß Minimum und Maximum aufgrund der Randbegrenzung für dieses Buchlayout nicht mehr in die gleiche Zeile wie die anderen Kennwerte paßten. Die Reihenfolge der Ausprägungen läßt sich jedoch mit CLASS nicht umkehren, das kann nur die Anweisung BY.

Wenn die Kennwerte für weitere Berechnungen benötigt werden, kann man die Anweisung OUTPUT einsetzen, um die Kennwerte in eine neue SAS-Datei umzuleiten.

```
/*--- KA08-11.SAS ---*/
PROC MEANS DATA=biblio.firma NOPRINT;
     VAR teil1--teil3;
     OUTPUT OUT=fortbild;
RUN;
```

Dieser Prozedurschritt erzeugt wegen der Option NOPRINT in der Anweisung PROC MEANS keine Ausgabe im Ausgabefenster. Stattdessen wird aufgrund der

Anweisung OUTPUT eine neue Datei `fortbild` erzeugt, welche die Kennwerte der Variablen `teil1` bis `teil4` enthält. Im Protokollfenster erscheint eine entsprechende Meldung:

```
NOTE: The data set WORK.FORTBILD has 5 observations and 7
      variables.
```

Die neue Datei hat fünf Beobachtungen und sieben Variablen. Um nicht Rätselraten zu müssen, ob sich die fünf Beobachtungen auf die ursprünglichen Variablen beziehen oder auf die Kennwerte, sollten Sie die Struktur studieren, entweder über das Libraries-Fenster oder mithilfe eines PROC PRINT-Schritts:

```
OBS _TYPE_ _FREQ_ _STAT_  TEIL1   TEIL2A  TEIL2B  TEIL3

 1     0     31    N     31.0000 31.0000 31.0000 31.0000
 2     0     31    MIN    1.0000  1.0000  1.0000  3.0000
 3     0     31    MAX   10.0000 10.0000 10.0000 10.0000
 4     0     31    MEAN   6.0645  5.8387  6.7419  6.3871
 5     0     31    STD    2.3229  2.8413  2.0325  1.6468
```

Aha! Die fünf Zeilen beziehen sich auf die fünf Kennwerte, die in der Variablen `_STAT_` genannt werden: `N, MIN, MAX, MEAN` und `STD`. `_FREQ_` gibt die Anzahl der Beobachtungen aus und die Variable `_TYPE_` verändert sich erst, wenn die Anweisung CLASS eingesetzt wird. Diese Datei kann nun verwendet werden, um Mittelwerte über allen fünf Teilabschnitten zu bilden oder diese Mittelwerte mit ähnlichen Veranstaltungen anderer Firmen zu vergleichen.

Haben Sie sich schon gefragt, ob sich die vielen Nachkommastellen auf ein erträgliches Maß reduzieren lassen? Bei der Darstellung von Meßgrößen im Millimeterbereich mag es notwendig sein, sechs und mehr Nachkommastellen anzugeben. Aber beim Alter? Natürlich läßt sich das ändern: Mit den beiden Optionen FW und MAXDEX in der Anweisung PROC PRINT wird die Darstellung der Zahlenwerte gesteuert. Mit FW (für *Field Width*) wird die Spaltenbreite festgelegt, mit MAXDEX die Anzahl der Dezimalstellen. Im folgenden Beispiel werden die Kennwerte für das Alter erneut berechnet, aber nur sechs Spalten breit und mit zwei Nachkommastellen im Ausgabefenster angezeigt.

```
PROC MEANS DATA=biblio.firma FW=6 MAXDEC=2;
     VAR alter;
RUN;
```

Die Spaltenbreite FW beeinflußt nur die Darstellungsbreite der Kennwerte, die Überschrift `Maximum` mit sieben Buchstaben wird unverändert ausgegeben.

```
Analysis Variable : ALTER

 N    Mean  Std Dev  Minimum  Maximum
-------------------------------------
31   37.13    11.39    19.00    58.00
-------------------------------------
```

Neben den fünf Standardkennwerten (N, Mittelwert, Standardabweichung, Minimun und Maximum) können weitere statistische Größen berechnet werden. Dazu ist eine Erweiterung der Anweisung PROC MEANS um eines oder mehrere der folgenden Statistikschlüsselworte erforderlich:

Schlüsselwort	Bedeutung
NMISS	Anzahl der fehlenden Werte
RANGE	Spannweite
VAR	Varianz
SKEWNESS	Schiefe
KURTOSIS	Wölbung
CV	Variationskoeffizient

Durch weitere Schlüsselworte der PROC MEANS-Anweisung können zusätzlich Konfidenzintervalle für den Erwartungswert und der Einstichproben- oder verbundene t-Test berechnet werden.

Um das 95%-Konfidenzintervall für den Erwartungswert der Punktzahl der Mitarbeiter beim vierten Fortbildungsabschnitt zu ermitteln, muß als Schlüsselwort CLM angegeben werden.

```
/*--- KA08-12.SAS ---*/
PROC MEANS DATA=biblio.stammdat MEAN CLM;
     VAR teil4;
RUN;
```

Zusätzlich wurde noch MEAN angegeben, damit neben dem Konfidenzintervall auch der Mittelwert im Ausgabefenster erscheint:

```
Analysis Variable : TEIL4

        Mean  Lower 95.0 CLM  Upper 95.0 CLM
-----------------------------------------------
  11.8709677       10.3682930       13.3736425
-----------------------------------------------
```

Ohne zusätzliche Angabe wird das 95%-Konfidenzintervall berechnet und dessen untere und obere Grenze ausgegeben. Der Bereich um den Mittelwert des 4. Abschnitts, in welchem der Erwartungswert der Grundgesamtheit mit hoher (nämlich 95%-iger) Wahrscheinlichkeit liegt, bewegt sich zwischen 10.36 und 13.38 Punkten. Andere Konfidenzbereiche erhält man durch Vorgabe der Option ALPHA=:

```
/*--- KA08-13.SAS ---*/
PROC MEANS DATA=biblio.firma MEAN CLM ALPHA=0.01;
     VAR teil4;
RUN;
```

Das mit ALPHA=0.01 berechnete 99%-Konfidenzintervall ist breiter als das zuvor berechnete 95%-Konfidenzintervall:

```
Analysis Variable : TEIL4

        Mean  Lower 99.0 CLM  Upper 99.0 CLM
-----------------------------------------------
  11.8709677        9.8475604       13.8943751
-----------------------------------------------
```

Einseitige Konfidenzgrenzen erhält man über die Optionen LCLM (L für *lower*, unteres Konfidenzintervall) und UCLM (für *upper*, oberes Konfidenzintervall).

```
PROC MEANS DATA=biblio.noten2 MEAN LCLM;
     VAR gesamt;
RUN;
```

Die einseitigen unteren bzw. oberen Konfidenzintervalle geben Auskunft darüber, mit welcher Wahrscheinlichkeit der Erwartungswert größer bzw. kleiner als die errechnete Grenze sein wird.

```
Analysis Variable : TEIL4
        Mean  Lower 95.0 CLM
----------------------------
  11.8709677       10.6221476
----------------------------
```

Aus diesem Ergebnis kann geschlossen werden, daß der Erwartungswert der Grundgesamtheit mit 95%-iger Wahrscheinlichkeit größer als 10.62 ist. Das heißt, im Durchschnitt wurden mehr als die Hälfte aller Punkte erzielt.

Über die Statistikschlüsselworte T und PRT der PROC MEANS-Anweisung wird der zweiseitige Einstichproben t-Test zum Testen der Nullhypothese „Der Erwartungswert der Grundgesamtheit ist gleich 0“ gegen die Alternativhypothese „Der Erwartungswert ist ungleich 0“ berechnet.

```
/*--- KA08-14.SAS ---*/
PROC MEANS DATA=biblio.firma T PRT;
     VAR teil4;
RUN;
```

Mit der Option T wird die Teststatistik angefordert, mit der Option PRT der zugehörige p-Wert.

```
Analysis Variable : TEIL4

           T  Prob>|T|
----------------------
  16.1337312    0.0001
----------------------
```

Die Teststatistik T, auch Testwert genannt, ist die Größe, die aus den Stichprobenwerten berechnet wird. Im Fall des vorliegenden Einstichproben t-Tests ist dies der Mittelwert geteilt durch den Standardfehler oder in Formeln:

$$T = \frac{\bar{x}}{\sigma/\sqrt{n}}$$

($\bar{x}$ ist der Mittelwert, σ die Standardabweichung und n der Umfang der Stichprobe.) In obigem Beispiel beträgt der Testwert für die Variable `teil4` 16.13.

Üblicherweise wird beim Hypothesentesten dieser Testwert mit dem kritischen Wert der Verteilung der Teststatistik verglichen. Im vorliegenden Fall wäre der

kritische Wert gerade das 1-α-Quantil der t-Verteilung mit $n-1$ Freiheitsgraden. Diesen kritischen Wert sucht man jedoch vergebens im Ausgabefenster. Stattdessen liefert das System aufgrund der Option PRT die zur Teststatistik unter der Nullhypothese gehörende Wahrscheinlichkeit (im Ausgabefenster mit `Prob>|T|` überschrieben). Diese Wahrscheinlichkeit oder auch kurz p-Wert (p für engl. *probability*) muß der Anwender mit dem von ihm zuvor gewählten Signifikanzniveau α des Tests vergleichen:

- Liegt der p-Wert unterhalb der Signifikanzschranke ($p \leq \alpha$), wird die Nullhypothese abgelehnt.
- Liegt der p-Wert oberhalb der Signifikanzschranke ($p > \alpha$), kann die Nullhypothese nicht verworfen werden.

Im vorliegenden Beispiel beträgt der p-Wert 0.0001, d. h., die Nullhypothese, daß die Punktzahl der Mitarbeiter im 4. Testabschnitt gleich 0 ist, kann auf dem 1%-Signifikanzniveau verworfen werden. Diese Aussage ist aber nicht sonderlich verwunderlich, da jeder eingestellte Mitarbeiter zumindest ein paar Punkte erzielt hat.

Wie in obigem Beispiel ist es nicht immer sinnvoll, gegen den Standardwert 0 zu testen. Im Fall der Punktzahlen wäre es viel interessanter zu überprüfen, ob sich der Erwartungswert von 10, also der Hälfte der zu erzielenden Punkte, unterscheidet. Dazu testet man die Nullhypothese „Der Erwartungswert liegt bei 10“ gegen die Alternativhypothese „Der Erwartungswert ist von 10 verschieden“.

Da die Prozedur MEANS nur gegen den Standardwert 0 testen kann, muß ein Trick angewandt werden. Der zu prüfende Sollwert von 10 wird von den eigentlichen Beobachtungswerten subtrahiert (`neu=teil4-10`) und anschließend wird getestet, ob dieser neue Wert gleich 0 ist[1].

```
/*--- KA08-15.SAS ---*/
DATA firma;
     SET biblio.firma;
     neu=teil4-10;
RUN;
PROC MEANS DATA=firma PRT T;
     VAR neu;
RUN;
```

Die Variable `neu` wird in einem Datenschritt angelegt. Anschließend kann die Prozedur MEANS aufgerufen werden, um den t-Test durchzuführen.

[1] Hier benutzt man die Invarianzeigenschaft der Verteilungsfunktionen gegen Translation.

```
Analysis Variable : NEU

           T  Prob>|T|
----------------------
   2.5428163    0.0164
----------------------
```

Die Teststatistik beträgt in diesem Fall 2.5 und die Wahrscheinlichkeit dieser Teststatistik unter der Nullhypothese 0.0164, womit die Nullhypothese, daß der Erwartungswert gleich 0 ist, auf dem 5%-Signifikanzniveau ebenfalls abgelehnt werden kann. Damit kann aber auch die Nullhypothese, daß der Erwartungswert der Gesamtpunktzahl gleich 10 ist, abgelehnt werden.

Ein weiteres Einsatzgebiet dieses Hypothesentests ist der Vergleich zweier verbundener Stichproben. Verbundene Stichproben treten auf, wenn Merkmale an Probanden mehrfach gemessen werden. Ein typisches Beispiel für solche Verbundenheit ist die Messung des Gewichts von Teilnehmerinnen und Teilnehmern an einer Frühjahrskur zu Beginn und am Ende der Kur. Die alle interessierende Frage lautet: Hat die Kur einen Erfolg gehabt? Oder präziser: Kam es zu einer Gewichtsreduktion? Aber auch die fünf verschiedenen Punktzahlen der Mitarbeiter sind verbundene Stichproben. Der Chef kann sich beispielsweise die Frage stellen, ob sich die mittlere Punktzahl bei Aufgabenteil A des 2. Abschnitts von der von Teil B unterscheidet.

Übertragen auf einen Hypothesentest wird die Nullhypothese wie folgt formuliert: Die mittlere Punktzahl im Teil A unterscheidet sich nicht von der mittleren Punktzahl im Teil B. Und die Alternativhypothese lautet: Die mittleren Punktzahlen von Teil A und Teil B unterscheiden sich. Getestet werden sollen die Hypothesen auf dem 5%-Niveau.

Um den Test mit der Prozedur MEANS durchführen zu können, wird zunächst die Differenz der beiden Punktangaben in einem Datenschritt ermittelt und im Anschluß getestet, ob diese Differenz gleich 0 ist:

```
/*--- KA08-16.SAS ---*/
DATA firma;
     SET biblio.firma;
     diff=teil2a-teil2b;
RUN;
PROC MEANS DATA=firma T PRT;
     VAR diff;
RUN;
```

```
Analysis Variable : DIFF

           T  Prob>|T|
----------------------
  -1.3091165    0.2004
----------------------
```

Die Nullhypothese kann auf dem 5%-Signifikanzniveau nicht abgelehnt werden (0.2004>0.05). Die erzielten Punkte bei den beiden Abschnitten des 2. Teils unterscheiden sich daher nicht.

Die Syntax der Prozedur MEANS hat die Form:

```
PROC MEANS <DATA=Datei> <FW=n> <MAXDEC=n> <NOPRINT>
           <N> <NMISS> <MIN> <MAX> <RANGE>
           <MEAN> <STD> <VAR> <SKEWNESS> <KURTOSIS> <CV>
           <CLM> <LCLM> <UCLM> <ALPHA=>
           <T> <PRT>;
     <CLASS Variablen-Liste;>
     <VAR Variablen-Liste;>
     <OUTPUT OUT=Datei;>
```

8.3 Für Singles – Die Prozedur UNIVARIATE

Die Prozedur MEANS berechnet Konfidenzintervalle und Hypothesentests, die nur für normalverteilte Variablen sinnvoll interpretiert werden können. Mit der Prozedur UNIVARIATE erhält man zusätzliche Kennwerte, die auch für nichtnormalverteilte und quantitative Variablen von Bedeutung sind. So können Median und Modalwert bestimmt werden, sowie Quartile, Quantile und Perzentile.

Neben dem Einstichproben t-Test wird auch der Vorzeichentest und der Wilcoxon Vorzeichen-Rangtest gerechnet. Weiterhin steht mit dem Shapiro-Wilks Test bzw. bei größeren Stichprobenumfängen mit dem Kolmogoroff-Smirnov Test ein Test zur Überprüfung der Normalverteilung der Variablen zur Verfügung. Mit dem Stamm und Blatt Diagramm, dem Boxplot und dem Normal Probability Plot kann die empirische Verteilung der Daten grafisch dargestellt werden. Konfidenzintervalle werden im Unterschied zur Prozedur MEANS nicht unterstützt.

Für das folgende Beispiel wird wieder auf die Stammdaten der Firmenmitarbeiter und damit auf die Datei `biblio.stammdat` zurückgegriffen.

```
/*--- KA08-17.SAS ---*/
PROC UNIVARIATE DATA=biblio.stammdat;
     VAR alter;
RUN;
```

Bei der Prozedur UNIVARIATE empfiehlt sich immer die Angabe der VAR-Anweisung zur Festlegung der Variablen, für die Kennwerte berechnet werden, da die Ergebnisdarstellung weit weniger kompakt ist als die der Prozedur MEANS.

Die Prozedurausgabe gliedert sich in die drei Abschnitte `Moments`, `Quantiles` und `Extremes`, die im folgenden getrennt dargestellt und besprochen werden.

```
Univariate Procedure
Variable=ALTER
                  Moments
 N                  31  Sum Wgts          31
 Mean          37.12903  Sum             1151
 Std Dev        11.3893  Variance     129.7161
 Skewness      0.323226  Kurtosis     -1.11428
 USS              46627  CSS          3891.484
 CV            30.67491  Std Mean     2.045578
 T:Mean=0      18.15087  Pr>|T|         0.0001
 Num ^= 0            31  Num > 0            31
 M(Sign)           15.5  Pr>=|M|        0.0001
 Sgn Rank           248  Pr>=|S|        0.0001
```

Unter der Überschrift `Moments` finden sich die bereits im Zusammenhang mit der Prozedur MEANS vorgestellten Kenngrößen Mittelwert (`Mean`), Standardabweichung (`Std Dev`), Varianz (`Variance`) und die Anzahl der nichtfehlenden Beobachtungen (`N`). An weiteren wichtigen Kenngrößen werden noch ausgegeben: die Schiefe (`Skewness`), die Wölbung (`Kurtosis`), der Variationskoeffizient

(CV, engl. *coefficient of variation*), die Anzahl der von Null verschiedenen Beobachtungen (Num^=0) und die Anzahl der Beobachtungen, die größer als Null sind (Num>0).

Auch die Ergebnisse dreier Einstichproben-Hypothesentests werden in diesem Abschnitt Moments aufgeführt: Der verbundene t-Test (T), der Vorzeichentest (M(Sign)) und der Wilcoxon Vorzeichen-Rangtest (Sgn Rank).

Der t-Test entspricht genau dem im vorherigen Abschnitt für die Prozedur MEANS besprochenen Test. Beim Vorzeichentest und dem Wilcoxon Vorzeichen-Rangtest handelt es sich dagegen um nichtparametrische Hypothesentests, die die Normalverteilung der zugrundeliegenden Population nicht voraussetzen. Getestet wird mit beiden nichtparametrischen Tests, ob die Beobachtungen im Mittel gleich Null sind.

Für die Berechnung des Testwerts M des Vorzeichentests werden die positiven Beobachtungen ausgezählt und anschließend der unter der Nullhypothese erwartete Wert, das ist gerade die Hälfte aller Beobachtungen, davon subtrahiert. In Formeln[2] ausgedrückt:

$$M = \sum_{x_i > 0} 1 - n/2$$

Für den Wilcoxon Vorzeichen-Rangtest werden die beobachteten Werte zunächst durch ihre Rangzahlen ersetzt. Anschließend werden die Rangzahlen der positiven Beobachtungen aufaddiert. Von dieser Summe S wird schließlich wie beim Vorzeichentest die unter der Nullhypothese erwartete Anzahl subtrahiert[2].

$$S = \sum_{x_i > 0} x_{(i)} - \frac{n(n+1)}{4}$$

Die Prozedur UNIVARIATE liefert in der linken Spalte der Ausgabe den Testwert und in der rechten Spalte die zugehörige Wahrscheinlichkeit dieses Testwerts unter der Nullhypothese, den p-Wert.

```
T:Mean=0   18.15087  Pr>|T|       0.0001
M(Sign)         15.5  Pr>=|M|      0.0001
Sgn Rank         248  Pr>=|S|      0.0001
```

Alle drei Hypothesentests werden zweiseitig ausgeführt, was durch die Betragsstriche um die Teststatistiken in der rechten Spalte angedeutet wird. Auf dem 5%-Signifkanzniveau lehnen alle drei Tests die Nullhypothese ab, daß das mittlere Alter gleich 0 ist, da $0.0001 < 0.05$, was natürlich nicht verblüfft. Sollte gegen einen anderen Wert getestet werden, etwa 25, müßten die Daten zunächst

[2] x_i sind die positiven Beobachtungen, n bezeichnet die Anzahl aller Beobachtungen.

wie bei der Prozedur MEANS beschrieben, um diesen Wert verschoben und der Test anschließend wiederholt werden.

```
          Quantiles(Def=5)

100% Max        58       99%        58
 75% Q3         47       95%        56
 50% Med        35       90%        53
 25% Q1         28       10%        24
  0% Min        19        5%        21
                          1%        19
Range           39
Q3-Q1           19
Mode            29
```

Im Anschluß an die Kennwerte und Hypothesentests erscheinen unter der Überschrift `Quantiles` die fünf Quartile (0%, 25%, 50%, 75% und 100%) sowie die 1%, 5%, 10%, 90%, 95% und 99% Perzentile. Die Bezeichnung `Range` weist die Spannweite aus, den gesamten Wertebereich von der kleinsten bis zur größten Beobachtung, und `Q3-Q1` verdeutlicht den Quartilsabstand, die Differenz zwischen dem 75%- und dem 25%-Quartil. `Mode` liefert den Modalwert, der häufigste Wert in der Stichprobe.

Die Werte entsprechen den Ausprägungen. Der älteste Mitarbeiter in der Firma ist beispielsweise 58 Jahre (`100%` = `Max`) alt, der jüngste 19 Jahre (`0%`, `Min`). Der Median (`50%`) liegt bei 35 Jahren.

```
              Extremes

  Lowest    Obs     Highest    Obs
      19(     17)       51(     12)
      21(     31)       53(      5)
      24(     29)       56(     15)
      24(      7)       56(     26)
      25(      1)       58(      4)
```

Mit einem Blick auf die Extremwerte wird in vielen Fällen schon eine Fehleingabe erkannt, oder eventuelle Ausreißer, die gesondert behandelt werden müssen, werden entdeckt. Im letzten Abschnitt, `Extremes`, werden hierfür die fünf kleinsten (`Lowest`) und die fünf größten (`Highest`) Beobachtungen mit den zugehörigen Beobachtungsnummern (`Obs`) ausgegeben. Im Firmenbeispiel ist der jüngste Mitarbeiter die Nr. 17, der älteste die Nr. 4.

Neben den Kennwerten und Tests kann über die Option PLOT in der Anweisung PROC UNIVARIATE die empirische Verteilung der Stichprobenwerte grafisch dargestellt werden.

```
/*--- KA08-18.SAS ---*/
PROC UNIVARIATE DATA=biblio.stammdat PLOT;
     VAR alter;
RUN;
```

Mit der Option PLOT werden ein Box and Whiskers Plot erzeugt, ein Stem and Leaf Diagramm und ein Normal Probability Plot.

```
Univariate Procedure

Variable=ALTER

   Stem Leaf                     #  Boxplot
      5 668                      3     |
      5 113                      3     |
      4 578                      3  +-----+
      4 1233                     4  |     |
      3 588                      3  *--+--*
      3 112                      3  |     |
      2 57889999                 8  +-----+
      2 144                      3     |
      1 9                        1     |
        ----+----+----+----+
    Multiply Stem.Leaf by 10**+1
```

Das Stem and Leaf Diagramm (= Stamm und Blatt) auf der linken Seite zeigt die empirische Häufigkeitsverteilung der Beobachtungen in Form eines vertikalen Histogramms, wozu die Beobachtungen in nicht zu beeinflußender Weise klassifiziert werden. Auf der x-Achse werden die absoluten Häufigkeiten aufgetragen, auf der y-Achse die klassifizierten Beobachtungen. Der *Stamm* wird von der linken Säule gebildet, rechts schließen sich die *Blätter* an. Die Kombination von Stamm und Blatt gibt zusammen mit dem unterhalb der x-Achse angegebenen Multiplikationsfaktor die tatsächlichen Beobachtungswerte wieder. So ergibt etwa $1.9 \cdot 10^{1} = 19$ das kleinste Alter. Die Anzahl der Blätter pro Stammabschnitt entspricht im vorliegenden Beispiel gerade der Anzahl der Beobachtungen pro Klasse. Diese Anzahl wird in der mittleren, mit # überschriebenen, Spalte zusätzlich explizit angegeben. Das Stamm und Blatt Diagramm liefert somit einen Eindruck von der Form der Verteilung der Stichprobenwerte.

Der Box and Whiskers Plot, oder kurz Boxplot, vervollständigt dieses Bild. In ihm werden die Quartile grafisch veranschaulicht. Die Box verläuft vom 25%- bis zum 75%-Quartil. In der Mitte, durch ein Pluszeichen (+) gekennzeichnet, wird der Median eingetragen. Die quer verlaufende Linie kennzeichnet die Lage des Mittelwerts in der Stichprobe. Die Whiskers, die dünnen vertikalen Striche, verlaufen von der Box bis zum kleinsten bzw. größten Wert in der Stichprobe. Lägen Werte weiter als das 1.5 fache der Höhe der Box von der Box entfernt, würden die durchgezogenen Whiskers nicht bis zum äußersten Rand gezogen, sondern die weiter entfernt liegenden Werte durch Einzelpunkte dargestellt.

Der Boxplot vermittelt einen guten Eindruck von der Symmetrie der empirischen Stichprobenverteilung, der Lage von Mittelwert und Median, und er gibt Hinweise auf Ausreißer, also auf Werte, die außerhalb des „normalen“ Bereichs liegen.

```
Univariate Procedure

Variable=ALTER
                    Normal Probability Plot
  57.5+                                                  * *++++*
      |                                              ** *+++
      |                                          ***+++
      |                                      ***++
  37.5+                                  +**+
      |                              ++++**
      |                          ***+* *
      |                  *+*+++
      |
  17.5+     * ++++
       +----+----+----+----+----+----+----+----+----+----+----+
            -2        -1         0        +1        +2
```

Für den Normal Probability Plot werden in einem Koordinatensystem die Quantile der empirischen Verteilung der Beobachtungen als Sterne (*) gekennzeichnet, und die Quantile der Normalverteilung (+) eingetragen. Der Normal Probability Plot erlaubt Aussagen über die Normalverteilung der Stichprobe: Liegen die Beobachtungen auf der „+“-Geraden, kann von einer normalverteilten Population ausgegangen werden. Weichen die Beobachtungen dagegen von der Geraden ab, liegt der Grundpopulation mit hoher Wahrscheinlichkeit eine andere Verteilung zugrunde.

Aufgrund der schlechten Auflösung im Ausgabefenster sind die Grafiken für einen ersten Eindruck gut geeignet, für die Präsentation sollte man jedoch besser eines der zahlreichen Makros zur Erzeugung eines „schönen“ Boxplots verwenden oder im SAS-Modul SAS/QC die Prozedur SHEWHART verwenden.

Wenn die Kennwerte und die mit PLOT erzeugten Grafiken Sie noch nicht endgültig von der Normalverteilung der Daten überzeugen konnten, läßt sich mit der Option NORMAL schließlich noch ein Signifikanztest auf Normalverteilung durchführen.

```
/*--- KA08-19.SAS ---*/
PROC UNIVARIATE DATA=biblio.stammdat NORMAL;
     VAR alter;
RUN;
```

Im Abschnitt `Moments` erscheint damit als letzte Zeile das Ergebnis dieses Tests auf Normalverteilung. Die Prozedur UNIVARIATE verwendet dazu bei Stichproben, die mehr als 2000 Beobachtungen haben, den Kolmogoroff Smirnov Test, bei kleineren Stichprobenumfängen den Shapiro Wilks Test. Je nachdem, welcher Test gerechnet wird, erscheint im Ergebnis der Buchstabe `K` oder `W` zur Bezeichnung der Teststatistik. Die Nullhypothese beider Tests lautet: Die den Daten zugrundeliegende Population ist normalverteilt. Die Alternative: Die Population ist nicht normalverteilt.

```
 W:Normal   0.935736  Pr<W         0.0751
```

Wie bei den drei anderen Hypothesentests erscheint in der linken Spalte der Testwert und in der rechten Spalte der zugehörige p-Wert. Der Testwert `W` kann nur dann sinnvoll interpretiert werden, wenn der Test und dessen Berechnung bekannt sind. Der p-Wert dagegen ist immer interpretierbar: Er nimmt Werte zwischen 0 und 1 an. Kleine Werte sprechen gegen die Nullhypothese und für die Alternative. Im vorliegenden Fall würde man nicht davon ausgehen können, daß die Variable `alter` normalverteilt ist, da der Shapiro Wilks Test auf dem 10%-Signifikanzniveau abgelehnt werden kann[3].

Ähnlich wie bei der Prozedur MEANS können die Kennwerte in eine neue Datei übertragen werden. Dazu wird die Anweisung OUTPUT formuliert.

```
/*--- KA08-20.SAS ---*/
PROC UNIVARIATE DATA=biblio.stamm NOPRINT;
     VAR alter kinder;
     OUTPUT OUT=stamm
            MEAN=m_alt m_kin MEDIAN=med_alt med_kin;
RUN;
```

[3] Die Signifikanzschranke wurde nicht willkürlich niedriger angesetzt, um ein signifikantes Ergebnis zu erzielen, sondern bewußt, um einen Test mit mehr Güte zu erhalten.

Die Option NOPRINT der Anweisung PROC UNIVARIATE unterdrückt die Ausgabe der Kennwerte und Testergebnisse. Mit der Anweisung OUTPUT und der Option OUT= wird die Datei `stamm` erzeugt und die beiden Kennwerte MEAN= und MEDIAN= eingetragen. Da in der Anweisung VAR zwei Variablen angegeben wurden, müssen neben den Schlüsselworten für die Kennwerte auch zwei Variablennamen angegeben werden, unter denen das System die ermittelten Kennwerte in der Datei `stamm` ablegt. Natürlich können nicht nur Mittelwert und Median in die Datei übertragen werden, sondern alle Kennwerte und Testgrößen, die von der Prozedur UNIVARIATE berechnet werden. Über die Online-Hilfe erhalten Sie die komplette Liste aller verfügbaren Werte und die dafür notwendigen Schlüsselworte.

Abschließend wird die Syntax der Prozedur UNIVARIATE zusammengestellt:

```
PROC UNIVARIATE DATA=Datei <NOPRINT> <PLOT> <NORMAL>;
     <VAR Variablen-Liste;>
     <OUTPUT OUT=Datei Kennwert=Variablen-Liste <...>;>
```

8.4 Zusammenhänge – Die Prozedur CORR

Die Prozeduren MEANS und UNIVARIATE berechnen Kennwerte für jeweils ein Merkmal. Ist man dagegen am Zusammenspiel zweier oder mehrerer Merkmale interessiert, möchte man den Grad des Zusammenhangs der Merkmale messen und eventuell die Form des Zusammenhangs festlegen. Mit der Prozedur CORR können dazu verschiedene Zusammenhangsmaße berechnet und diese Korrelationen genauer untersucht werden.

Der Firmenchef stellt sich beispielsweise die Frage, ob es einen Zusammenhang zwischen den Ergebnissen der verschiedenen Abschnitte des Einstellungstests gibt, z. B. ob der Mitarbeiter, der am Anfang gut war, dies auch noch am Ende ist oder ob es Veränderungen gibt.

Das bekannteste Zusammenhangsmaß für normalverteilte Merkmale ist der Produktmomenten-Korrelationskoeffizient nach Pearson[4]. Er berechnet die Kovarianz beider Merkmale unter Berücksichtigung der jeweiligen Einzelvarianzen. Der daraus resultierende Wert liegt zwischen -1 und +1. Ist er in der Nähe von 0, liegt sicher kein Zusammenhang zwischen den beiden Variablen vor. Ist er größer Null, liegt ein konkordanter Zusammenhang vor, d. h., mit Anstieg der einen Variablen, steigt auch die andere Variable an. Ist der Koeffizient kleiner Null, ist der Zusammenhang diskordant, d. h., mit Anstieg der einen

[4] kurz: Pearsonscher Korrelationskoeffizient oder, wenn der Zusammenhang eindeutig ist: Korrelationskoeffizient

Variablen fällt die andere Variable ab. Beträgt der Wert genau +1 oder -1, liegen die Punktepaare genau auf einer Geraden.

Neben dem Pearsonschen Korrelationskoeffizienten berechnet die Prozedur CORR ein nichtparametrisches Pendant, das bei nichtnormalverteilten metrischen Daten eingesetzt werden kann: der Korrelationskoeffizient nach Spearman. Zu seiner Bestimmung werden zunächst die Rangzahlen der beobachteten Werte ermittelt und für diese Rangzahlen anschließend der Pearsonsche Korrelationskoeffizient berechnet.

Standardmäßig, d. h. ohne weitere Angaben, berechnet die Prozedur CORR den Korrelationskoeffizienten nach Pearson.

```
/*--- KA08-21.SAS ---*/
PROC CORR DATA=biblio.firma;
     VAR teil1--teil4;
RUN;
```

In der VAR-Anweisung werden die Variablen benannt, für die Korrelationsmaße berechnet werden sollen. Die Prozedur bildet aus den angegebenen Variablen alle möglichen Paare und berechnet für diese den Zusammenhang.

Im Ausgabefenster werden allerdings zunächst die ausgewählten Variablen aufgeführt und ihre wichtigsten Kennwerte unter der Überschrift `Simple Statistics` tabelliert: Anzahl der nichtfehlenden Beobachtungen (`N`), Mittelwert (`Mean`), Standardabweichung (`Std Dev`), Summe (`Sum`) und kleinster und größter Wert (`Minimum`, `Maximum`). Mit der Option NOSIMPLE läßt sich dieser Teil unterdrücken. Im Anschluß an diese Kennwerte erscheint die Korrelationsmatrix.

Correlation Analysis
5 'VAR' Variables: TEIL1 TEIL2A TEIL2B TEIL3 TEIL4

Simple Statistics

Variable	N	Mean	Std Dev	Sum
TEIL1	31	6.064516	2.322864	188.000000
TEIL2A	31	5.838710	2.841323	181.000000
TEIL2B	31	6.741935	2.032531	209.000000
TEIL3	31	6.387097	1.646763	198.000000
TEIL4	31	11.870968	4.096681	368.000000

Variable	Minimum	Maximum
TEIL1	1.000000	10.000000
TEIL2A	1.000000	10.000000
TEIL2B	1.000000	10.000000
TEIL3	3.000000	10.000000
TEIL4	2.000000	19.000000

Pearson Correlation Coefficients /
Prob > |R| under Ho: Rho=0 / N = 31

	TEIL1	TEIL2A	TEIL2B	TEIL3	TEIL4
TEIL1	1.00000	0.20870	0.06013	0.42025	-0.14972
	0.0	0.2599	0.7480	0.0186	0.4215
TEIL2A	0.20870	1.00000	-0.22101	0.20614	0.10697
	0.2599	0.0	0.2322	0.2659	0.5668
TEIL2B	0.06013	-0.22101	1.00000	-0.02891	-0.18828
	0.7480	0.2322	0.0	0.8773	0.3104
TEIL3	0.42025	0.20614	-0.02891	1.00000	0.08177
	0.0186	0.2659	0.8773	0.0	0.6619
TEIL4	-0.14972	0.10697	-0.18828	0.08177	1.00000
	0.4215	0.5668	0.3104	0.6619	0.0

Diese Matrix besteht aus 5x5 Feldern. In den jeweils 5 Spalten und Zeilen stehen die 5 ausgewählten Variablen `teil1`, `teil2a` ... `teil4`. In jedem Feld findet man das jeweils zugehörige Zusammenhangsmaß. Die Matrix ist bezüglich der Gegendiagonalen symmetrisch aufgebaut, da die Korrelation von X und Y genau der Korrelation von Y und X entspricht. Auf der Gegendiagonalen stehen lauter Einsen, da die Korrelation von einer Variablen mit sich selbst 1 ergibt.

Die Korrelationen zwischen den 5 Teilabschnitten sind ausgesprochen gering. Einzig und allein zwischen `teil1` und `teil3` beträgt der Korrelationskoeffizient 0.42 und liegt somit in einem Bereich, in dem man dem Wert zumindest Beachtung schenkt. Alle anderen sind unbedeutend. Ab wann ein Zusammenhangsmaß „groß“ ist und man von einem bedeutsamen Zusammenhang sprechen kann, ist nicht allgemein festzulegen. Vielmehr muß hierbei der Kontext gesehen werden, in dem die Frage nach der Beziehung zweier Variablen aufgeworfen wurde. Im Bereich der Medizin, wo beispielsweise aus den Meßwerten eines Geräts auf das Vorhandensein eines bestimmten Stoffes gefolgert werden soll, muß der Korrelationskoeffizient sehr hoch sein. Werte unterhalb von 0.9 werden nicht in Betracht gezogen. Bei Erhebungen im sozialwissenschaftlichen Bereich dagegen, in denen „weiche“ Parameter wie Vorlieben, Empfindungen oder Eigenschaften untersucht werden sollen, ist man dagegen oft schon mit Zusammenhangsmaßen zwischen 0.4 und 0.6 zufrieden.

In der Matrix werden unterhalb der Korrelationen p-Werte für den Test mit der Nullhypothese „Der Korrelationskoeffizient ist gleich 0“ ausgegeben (`Prob > |R| under Ho: Rho=0`).

Dieser p-Wert sollte nicht überbewertet werden. Kann die Nullhypothese abgelehnt werden, bedeutet das zunächst nur, daß die Korrelation mit großer Wahrscheinlichkeit von Null verschieden ist. Aber wie groß sie tatsächlich ist oder ob sie nahe bei 1 liegt, kann daraus nicht geschlossen werden. Außerdem kann für hinreichend große Stichproben jeder beliebige von Null verschiedene Wert als signifikant verschieden von Null getestet werden!

In der obigen Ausgabe erscheint im Kopf der Matrix `N=31`. Dadurch signalisiert das System, daß alle Korrelationsberechnungen für jeweils 31 Beobachtungen durchgeführt wurden. Würden bei einer der Variablen fehlende Werte vorliegen, würde in den einzelnen Feldern als dritter Wert die Anzahl der Beobachtungen aufgeführt werden, die zur Berechnung beitragen.

Liegen keine normalverteilten Beobachtungen vor, wird mit Hilfe der Option SPEARMAN in der Anweisung PROC CORR der Korrelationskoeffizient nach Spearman berechnet.

```
PROC CORR DATA=biblio.firma SPEARMAN;
     VAR teil1 teil3;
RUN;
```

Die Ausgabe unterscheidet sich von der obigen Matrix bezüglich der äußeren Form nur durch das Wort Spearman.

```
Spearman Correlation Coefficients /
Prob > |R| under Ho: Rho=0 / N = 31
                  TEIL1             TEIL3
TEIL1           1.00000           0.46778
                 0.0               0.0080
TEIL3           0.46778           1.00000
                 0.0080            0.0
```

Die Korrelationskoeffizienten können in eine neue Datei übertragen werden. Dazu gibt man die Optionen OUTP= bzw. OUTS= in der Anweisung PROC CORR an.

```
/*--- KA08-22.SAS ---*/
PROC CORR DATA=biblio.firma OUTP=firma_p NOPRINT;
     VAR teil1--teil4;
RUN;
```

Neben den vorgestellten Korrelationskoeffizienten nach Pearson und Spearman gibt es weitere Zusammenhangsmaße, den Kendallschen Koeffizienten, Cronbachs Alpha, den Phi-Koeffizienten, Kappa-Koeffizienten usw. Die ersten beiden Koeffizienten können ebenfalls mit der Prozedur CORR berechnet werden, indem die Optionen KENDALL bzw. ALPHA gesetzt werden. Die anderen können mit der Prozedur FREQ berechnet werden (Optionen AGREE und MEASURES der Anweisung TABLES). Die allgemeine Syntax der Prozedur CORR hat die Form:

```
PROC CORR DATA=Datei <SPEARMAN> <KENDALL> <ALPHA>
           <NOPRINT> <OUTP=Datei> <OUTS=Datei>;
     VAR Variablen-Liste;
     <WITH Variablen-Liste;>
RUN;
```

8.5 Vergleich von zwei Gruppen – Die Prozedur TTEST

Einer der bekanntesten Hypothesentests dürfte der Student t-Test (kurz: t-Test) sein. Er vergleicht die Erwartungswerte zweier unabhängiger Stichproben, denen normalverteilte Populationen mit gleicher Varianz zugrundeliegen. Seine Nullhypothese lautet: Die Erwartungswerte der beiden Gruppen sind gleich. Die

zweiseitige Alternativhypothese: Die beiden Erwartungswerte unterscheiden sich. Für die Teststatistik werden die Mittelwertdifferenz und der Standardfehler – analog zum Einstichproben t-Test – ins Verhältnis gesetzt[5].

$$T = \frac{\bar{x}_1 + \bar{x}_2}{s_p \sqrt{\frac{1}{n_1} + \frac{1}{n_2}}}$$

Die Teststatistik ist t-verteilt mit $n_1 + n_2 - 2$ Freiheitsgraden. Der t-Test wird innerhalb des SAS-Systems mit der Prozedur TTEST durchgeführt.

Der Personalleiter möchte mit Hilfe des t-Tests feststellen, ob sich die weiblichen und männlichen Mitarbeiter im ersten Abschnitt des Einstellungstests signifikant unterscheiden. Dazu formuliert er die Nullhypothese: Frauen und Männer unterscheiden sich im Mittel nicht hinsichtlich der Punktzahl im 1. Abschnitt. Die Alternativhypothese „Die Punktzahlen unterscheiden sich“ soll mit einem zweiseitigen Test auf dem 5%-Signifikanzniveau gesichert werden.

```
/*--- KA08-23.SAS ---*/
PROC TTEST DATA=biblio.firma;
     CLASS sex;
     VAR teil1;
RUN;
```

Die Anweisung CLASS legt die Gruppenvariable fest, nach deren Ausprägungen die Gruppen gebildet werden. Da der t-Test nur zwei Gruppen miteinander vergleichen kann, darf die CLASS-Variable auch nur zwei Ausprägungen haben. Mit der Anweisung VAR wird die Zielvariable festgelegt, hinsichtlich derer die beiden Gruppen verglichen werden. Im Ausgabefenster erscheint nach Ausführung des Programms folgendes Ergebnis:

```
TTEST PROCEDURE

Variable: TEIL1

SEX     N          Mean        Std Dev      Std Error
-----------------------------------------------------
m      19    6.05263158     2.57063944     0.58974513
w      12    6.08333333     1.97522534     0.57019844
```

Neben den üblichen Kennwerten `N`, `Mean` und `Std Dev` wird im Ausgabefenster auch der Standardfehler `Std Error` (=$\sqrt{\texttt{N}}$ `Mean` / `Std Dev`) ausgegeben. In der

[5] $\bar{x}_i$ bezeichnet den jeweiligen Gruppenmittelwert und s_p die gepoolte Stichprobenvarianz.

Tabelle darunter liefert die Prozedur TTEST die Ergebnisse zweier Hypothesentests: der t-Test für gleiche Varianzen (`Equal`), wie er oben erläutert wurde, und der modifizierte t-Test, auch Welch, oder Sattherswaite t-Test genannt, für ungleiche Varianzen (`Unequal`).

```
Variances        T       DF    Prob>|T|
----------------------------------------
Unequal    -0.0374     27.7      0.9704
Equal      -0.0352     29.0      0.9721

For H0: Variances are equal, F' = 1.69    DF = (18,11)
                              Prob>F' = 0.3744
```

Die Teststatistiken `T` betragen `-0.03`, die zugehörigen p-Werte `0.97`. Die Testergebnisse bestätigen das, was man den Kennwerten eigentlich auch schon ansieht: Die Nullhypothese kann nicht verworfen werden. Es gibt somit keinen Anhaltspunkt, daß sich Frauen und Männer im Mittel hinsichtlich der Variable `teil1` unterscheiden ($0.97 > 0.05$).

Im letzten Abschnitt der Ausgabe liefert die Prozedur TTEST einen F-Test auf Varianzhomogenität. Damit kann die Nullhypothese „Die beiden Gruppenvarianzen sind gleich“ gegen die Alternativhypothese „Die Gruppenvarianzen unterscheiden sich“ getestet werden, wenn etwa in einer Voruntersuchung die Frage geklärt werden muß, ob sich zwei Gruppen hinsichtlich ihrer Varianz unterscheiden. Die Teststatistik `F'` des F-Tests berechnet sich aus dem Quotienten der beiden Stichprobenvarianzen σ_i^2: `F'`=σ_1^2/σ_2^2. Sie ist F-verteilt mit $n_1 - 1$ und $n_2 - 1$ Freiheitsgraden. Aus dem Testwert und den Freiheitsgraden, in obigem Beispiel `1.69` und `18, 11` läßt sich zur Kontrolle mit der Funktion PROBF der zugehörige p-Wert ermitteln: `0.37`. Die Nullhypothese kann somit auf dem 5%-Signifikanzniveau nicht abgelehnt werden.

Als Anwender sollten Sie nicht versucht sein, zuerst diesen Test auf Varianzhomogenität zu Rate zu ziehen, um danach, je nach Aussage, im gleichen Anlauf den t-Test für gleiche oder den für ungleiche Varianzen zu interpretieren. Sie könnten mit diesem multiplen Testproblem eventuell das gewünschte Signifikanzniveau nicht mehr einhalten.

Zum Schluß folgt noch eine Zusammenfassung der Syntax der Prozedur TTEST:

```
PROC TTEST <DATA=Datei>;
     CLASS Variable;
     <VAR Variablen-Liste;>
RUN;
```

8.6 Nichtparametrik – Die Prozedur NPAR1WAY

Ist die Normalverteilungsvoraussetzung nicht erfüllt, haben aber die den Stichproben zugrundeliegenden Verteilungen eine annähernd gleiche Form, kann als Alternative zum t-Test der Wilcoxon Mann-Whitney (oder kurz Wilcoxon) Rangsummentest mit der Prozedur NPAR1WAY durchgeführt werden. Der Aufruf erfolgt analog zur Prozedur TTEST. Da diese Prozedur aber auch für den Vergleich mehrerer Gruppen ausgelegt ist, wird im folgenden Beispiel die Option WILCOXON gesetzt, damit in der Ausgabe auch nur dieser nichtparametrische Zweigruppenvergleich durchgeführt wird.

```
/*--- KA08-24.SAS ---*/
PROC NPAR1WAY DATA=biblio.firma WILCOXON;
     CLASS sex;
     VAR teil1;
RUN;
```

Der Wilcoxon Rangsummentest basiert, wie der Name schon andeutet, auf der Berechnung der Summe der Rangzahlen. Unter der Nullhypothese der gleichmäßigen Verteilung der Meßwerte sind die Ränge der zu vergleichenden Gruppen gleichverteilt, d. h., die Rangsummen der beiden Gruppen sollten ungefähr gleich sein. Als Teststatistik des Wilcoxon Rangsummentests dient daher die Rangsumme der beiden Gruppen, bzw. die Rangsumme einer Gruppe, da sich aus der Anzahl der Beobachtungen die Gesamtsumme der Rangzahlen berechnen läßt. Im SAS-System wird die Rangsumme der Gruppe mit den wenigsten Beobachtungen herangezogen.

```
N P A R 1 W A Y  P R O C E D U R E
Wilcoxon Scores (Rank Sums) for Variable TEIL1
Classified by Variable SEX

              Sum of    Expected     Std Dev        Mean
  SEX    N    Scores    Under H0    Under H0       Score

  m     19     302.0       304.0  24.1731656  15.8947368
  w     12     194.0       192.0  24.1731656  16.1666667
Average Scores Were Used for Ties
```

Im Ausgabefenster erscheinen zunächst für die beiden Gruppen getrennt die ermittelten Rangsummen (`Sum of Scores`), die unter der Nullhypothese erwarteten Rangsummen (`Expected Under H0`), die Standardabweichung unter der Nullhypothese sowie der mittlere Rang (`Mean Score`, Rangsumme geteilt durch Beobachtungszahl). Im Anschluß erscheint der oder genauer gesagt erscheinen die Testergebnisse des Wilcoxon Rangsummentests.

```
Wilcoxon 2-Sample Test (Normal Approximation)
(with Continuity Correction of .5)
S =  194.000   Z = 0.062052   Prob > |Z| = 0.9505
T-Test Approx. Significance = 0.9509
Kruskal-Wallis Test (Chi-Square Approximation)
CHISQ = 0.00685   DF =  1   Prob > CHISQ = 0.9341
```

Als erstes wird die Normalapproximation der Teststatistik S angezeigt. Die normalverteilte Teststatistik S kann aus Z errechnet werden durch Subtraktion der erwarteten Rangsumme, anschließender Division mit dem Standardfehler und Anführen der Stetigkeitskorrektur. Der zu dieser normalverteilten Größe gehörige p-Wert `0.95` könnte auch mit der Funktion PROBNORM ermittelt werden. Zusätzlich werden noch zwei weitere Approximationen ausgegeben: die t-Test Approximation und die χ^2-Approximation, wobei letztere allerdings erst beim Vergleich mehrerer Gruppen interessant wird.

Während bei größeren Stichproben die Approximationen völlig ausreichend sind, sollte bei kleinen, schwach besetzten Gruppen der Wilcoxon Rangsummentest exakt durchgeführt werden. Das heißt, der zur Teststatistik gehörige p-Wert wird mit der exakten Verteilung der Teststatistik ermittelt. Dazu erweitert man den Prozedurschritt um die Anweisung EXACT.

```
/*--- KA08-25.SAS ---*/
PROC NPAR1WAY DATA=biblio.firma WILCOXON;
     CLASS sex;
     VAR teil1;
     EXACT WILCOXON;
RUN;
```

Der Zusatz WILCOXON ist erforderlich, da mit der Prozedur noch weitere exakte nichtparametrische Verfahren berechnet werden können. Im Ausgabefenster wird das Ergebnis um folgende Zeilen erweitert:

```
Wilcoxon 2-Sample Test      S =  194.000

   Exact P-Values
     (One-sided)  Prob >= S          = 0.4693
     (Two-sided)  Prob >= |S - Mean| = 0.9415
```

Zur Teststatistik `S=194` werden die ein- und zweiseitigen Überschreitungswahrscheinlichkeiten (`Exact P-Values`) ausgegeben. Für das vorliegende Beispiel unterscheidet sich der exakte zweiseitige p-Wert (`0.94`) allerdings aufgrund der Stichprobengröße kaum vom approximativ ermittelten p-Wert (`0.95`).

Bei größeren Stichprobenumfängen (N>50) kann die Berechnung von Fishers exaktem Test sehr lange dauern. Unter Umständen müssen Sie den Prozedurschritt mit STRG+UNTBR unterbrechen.

Die allgemeine Syntax der Prozedur NPAR1WAY hat die Form:

```
PROC NPAR1WAY DATA=datei <WILCOXON>;
     CLASS Variable;
     <VAR Variablen-Liste;>
     <EXACT WILCOXON;>
RUN;
```

8.7 Aufgaben

Auf der Begleitdiskette befindet sich die Textdatei `kur.dat`, die neben einer Identifikationsnummer das Gewicht (in Kilogramm) zu Beginn und am Ende einer Kurmaßnahme von 30 Teilnehmerinnen enthält. Die Teilnehmerinnen wurden in zwei Gruppen eingeteilt. Ein Teil (Gruppe R) erhielt anstelle des normalen Essens eine kalorienverminderte Reduktionsdiät, die andere Gruppe (L) mußte zweimal täglich je 30 Minuten dauerlaufen, erhielt aber normales Essen.

❶ Übertragen Sie die Werte in eine SAS-Datei.

❷ Untersuchen Sie, ob in beiden Gruppen gleich viele Teilnehmer waren.

❸ Berechnen Sie die wichtigsten Kennwerte für das Körpergewicht am Anfang und am Ende der Kur für alle Beobachtungen und getrennt für die beiden Gruppen.

❹ Berechnen Sie die Gewichtsdifferenz und prüfen Sie mit einem statistischen Test, ob sich das Gewicht im Laufe der Kur signifikant verändert hat.

❺ Berechnen Sie das 95%-Konfidenzintervall für die Gewichtsdifferenz für alle Beobachtungen und getrennt für die beiden Gruppen.

❻ Berechnen Sie den Zusammenhang zwischen dem Gewicht zu Beginn und zum Ende der Kur.

❼ Untersuchen Sie, ob die beiden Kurmaßnahmen (Reduktionsdiät und Sport) unterschiedliche Effekte auf die Gewichtsabnahme zeigen.

9

Eine andere Art der Darstellung – Grafik

Häufigkeiten und Zusammenhänge lassen sich nicht nur in Zahlenwerten ausdrücken, sondern können in grafischer Form oftmals verständlicher präsentiert werden. Im folgenden Kapitel werden zwei Grafikprozeduren, GCHART und GPLOT, vorgestellt, mit denen im SAS-System Häufigkeitsdiagramme zur Darstellung der empirischen Verteilung einer Variablen und Streudiagramme zum Aufzeigen eines Zusammenhangs zwischen zwei Merkmalen erzeugt werden können.

Im Zusammenhang mit der Prozedur UNIVARIATE haben Sie bereits erste Grafiken im SAS-System kennengelernt. Diese Grafiken erscheinen jedoch wie jede andere Ausgabe auch im Ausgabefenster, und sie setzen sich aus Punkten, Strichen und anderen Symbolen zusammen. Von dieser Art Grafik wird im folgenden nicht die Rede sein. Die Grafikprozeduren des SAS-Systems, deren Namen stets mit einem „G“ beginnt, erzeugen sogenannte hochauflösende Grafiken, die im Grafikfenster angezeigt und in SAS-Katalogen gespeichert werden.

Mit dem Modul SAS/ASSIST gestaltet sich der Zugang zu den Grafikprozeduren besonders einfach. Nach Aufruf der Menüfolge *Globals* →SAS/ASSIST (oder der Schaltfläche ) gelangt man über *Graphics* und *High Resolution* zu einem Menü, das die wichtigsten Möglichkeiten auflistet. In Kapitel 11 wird an einem konkreten Beispiel gezeigt, wie man mit dem Assistenten eine Grafik erstellen und das zugehörige Programm abspeichern kann. Dieses Kapitel wird dazu beitragen, daß Sie als Anwender das vom SAS-Assistent aufgebaute Grundgerüst verstehen. Es vermittelt Ihnen einen ersten Eindruck von den vielfältigen Einsatzmöglichkeiten der beiden wichtigsten Grafikprozeduren. Für den vertiefenden Einstieg werden die Handbücher zum Grafikmodul [12] empfohlen, in denen

neben der Syntax auch zahlreiche Beispiele für den Gebrauch einzelner Anweisungen und Optionen angegeben werden. Die meisten Beispiele sind mit dem Modul SAS/GRAPH installiert und können über die Online-Hilfe und *Sample Programs* geöffnet und ausgeführt werden.

9.1 Balken und Torten – Die Prozedur GCHART

Häufigkeiten werden grafisch in Form von Balken oder Kreissegmenten dargestellt. Mit der Prozedur GCHART können solche Balken- und Kreisdiagramme im SAS-System umgesetzt werden. Daneben gibt es auch eine Prozedur CHART aus dem Basis-Modul SAS/BASE, das ähnliche Darstellungsmöglichkeiten im Ausgabefenster bietet.

In den folgenden Beispielen wird die Mitarbeiterdatei der Firma unter verschiedenen Gesichtspunkten dargestellt. Zunächst werden die Anteile von Frauen und Männern unter den Mitarbeitern in Form eines Balkendiagramms mit vertikalen, d. h. senkrecht verlaufenden, Balken dargestellt. Dazu wird der folgende Prozedurschritt ausgeführt:

```
/*--- KA09-01.SAS ---*/
LIBNAME biblio 'C:\kurs';
PATTERN1 V=SOLID;
PROC GCHART DATA=biblio.stammdat;
     VBAR sex;
RUN;QUIT;
```

Die Anweisung PROC GCHART leitet den Prozedurschritt ein und benennt mit der Option DATA= die Datei, die die Stammdaten der Mitarbeiter enthält. Die nächste Anweisung VBAR weist das System an, die Balken vertikal aufzutragen. Das heißt, sowohl die Balken selbst als auch die Achse, an der die Häufigkeiten abgelesen werden können, verlaufen senkrecht. Der im Anschluß an das Schlüsselwort VBAR folgende Variablenname bezeichnet die Variable, in obigem Beispiel `sex`, für die die Häufigkeiten berechnet werden. Auf der horizontalen, waagrechten Achse werden die Ausprägungen der Variablen `sex` (`m` und `w`) aufgetragen. Führt man nun diesen Prozedurschritt aus, öffnet sich das Grafikfenster und zeigt das Histogramm an (Abbildung 9.1). An der Grafik läßt sich erkennen, daß deutlich mehr Männer als Frauen in der Firma beschäftigt sind, nämlich 19 gegenüber 12, wie sich der Achsenbeschriftung entnehmen läßt. Die Zahlenwerte, die im vorangegangen Kapitel mit der Prozedur FREQ numerisch bestimmt wurden, werden hier nun in Form einer Grafik wiedergegeben.

Im nächsten Schritt wird die Altersstruktur der Mitarbeiter aufgetragen. Dazu wird in obigem Prozedurschritt lediglich die Variable `sex` durch `alter` ersetzt.

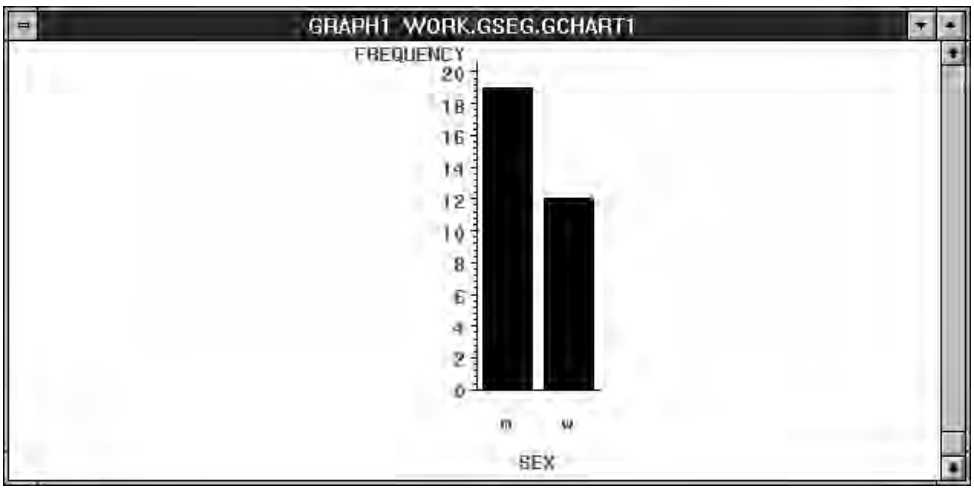


Abb. 9.1: Vertikales Balkendiagramm der Variable sex

```
PROC GCHART DATA=biblio.stammdat;
     VBAR alter;
RUN;QUIT;
```

Im Unterschied zur Textvariablen sex, bei der für jede Ausprägung ein eigener Balken erzeugt wird, werden bei der numerischen Variablen alter mehrere Einzelwerte zu Klassen zusammengefaßt und die Häufigkeiten pro Klasse ausgezählt und wiedergegeben (siehe Abbildung 9.2), ähnlich wie beim Stamm und Blatt-Diagramm der Prozedur UNIVARIATE. Auf der vertikalen Achse können die Häufigkeiten abgelesen werden, auf der horizontalen Achse die Mittelpunkte der Altersklassen (ALTER MIDPOINTS). Die eigentlichen Klassengrenzen müssen selbst bestimmt werden. Die Klassengrenzen liegen in der Mitte zwischen zwei Mittelpunkten. Die Altersklassen verlaufen demnach von 16 bis 24 Jahren, 24 bis 32, 32 bis 40 usw., wobei die obere Grenze nicht mehr zur Klasse dazugehört. (In der Mathematik spricht man von halboffenen Intervallen $[a, b)$.) Das heißt, Mitarbeiter, die beispielsweise 32 Jahre alt sind, werden zur dritten Altersklasse 32-40 gezählt.

Die Einteilung der Klassen erfolgt nach einem internen Algorithmus. Mit der Option MIDPOINTS= kann man dieses Schema abschalten und eigene Klassengrenzen festlegen, etwa 10-Jahres-Altersklassen. Um die Altersstruktur der Firmenmitarbeiter in diesen 10-Jahresabständen zu klassifizieren, muß man die

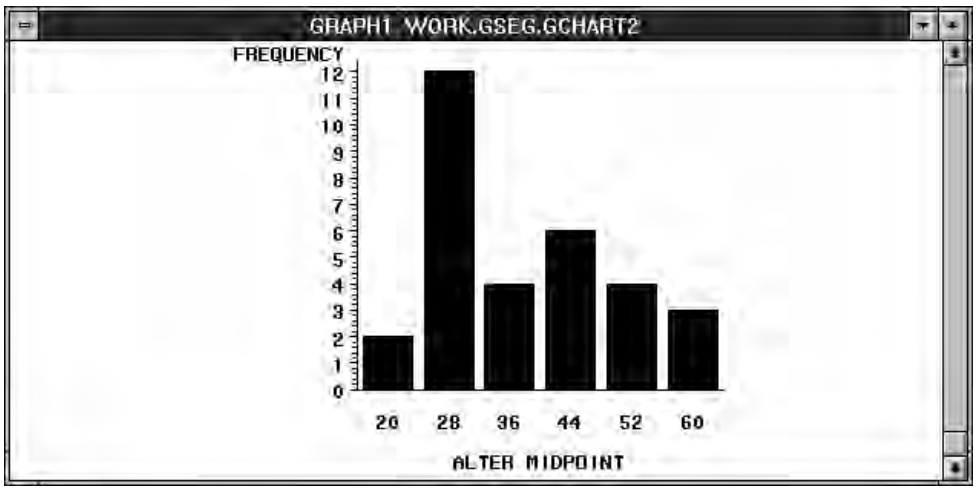


Abb. 9.2: Balkendiagramm der Variable `alter`

Klassenmittelpunkte auf 15, 25 etc. setzen, damit die Klassen korrekt gewählt werden.

```
/*--- KA09-02.SAS ---*/
PROC GCHART DATA=biblio.stammdat;
     VBAR alter / MIDPOINTS=15 25 35 45 55;
RUN;QUIT;
```

Aus der Abbildung 9.3 (siehe S. 189) geht deutlich hervor, daß ein Großteil der Mitarbeiter zwischen 20 und 30 Jahren alt ist.

Durch eine Veränderung der Klasseneinteilung können sehr unterschiedliche Resultate erzielt werden. Während die empirische Verteilung bei zu vielen Klassen sehr flach werden kann, verliert sie bei zu wenigen Klassen häufig die Struktur. Als Alternative zur Wahl der Klassenmittelpunkte kann mit der Option LEVEL die Anzahl der zu bildenden Klassen festgelegt werden. Mit der Anweisung `VBAR alter / LEVELS=6;` würden beispielsweise sechs Klassen gebildet.

Die Balken können nicht nur vertikal, sondern auch horizontal aufgetragen werden. Dazu ersetzt man die Anweisung VBAR durch HBAR. Zusätzlich zu den Balken werden am rechten Grafikrand die absoluten und relativen Häufigkeiten als Zahlenwerte angegeben (Abbildung 9.4, Seite 190).

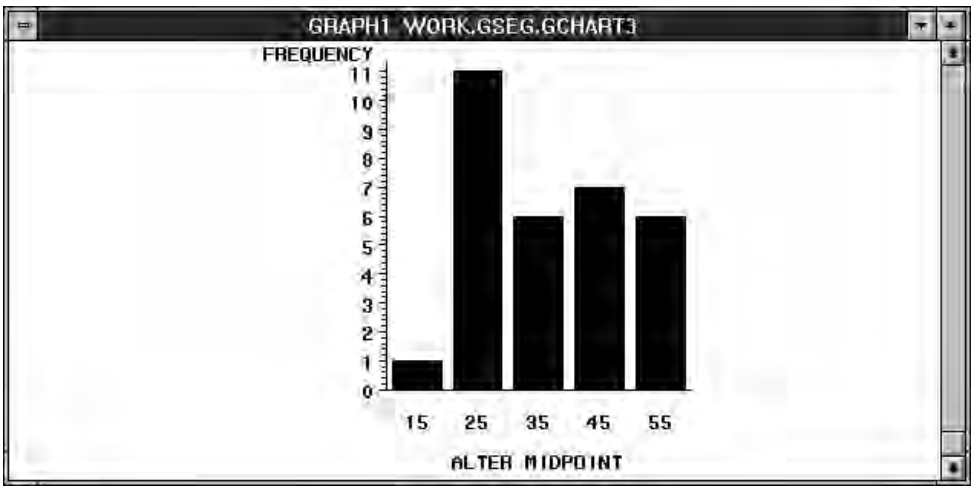


Abb. 9.3: Balkendiagramm der Variable `alter` *mit der Option* MIDPOINTS=

```
PROC GCHART DATA=biblio.stammdat;
     HBAR sex;
RUN;QUIT;
```

Mit der Option NOSTATS in der HBAR-Anweisung läßt sich die Ausgabe dieser Häufigkeiten am rechten Rand allerdings unterdrücken. Diese Option wird wie MIDPOINTS= nach dem Variablennamen und einem Schrägstrich angeführt.

Um Altersstrukturen mehrerer Firmen miteinander vergleichen zu können, werden mit `TYPE=PERCENT` anstelle der absoluten Häufigkeiten die Prozentwerte in dem Balkendiagramm aufgetragen. Standardmäßig ist die TYPE=-Option auf `FREQ` eingestellt zur Darstellung der absoluten Häufigkeiten.

```
PROC GCHART DATA=biblio.stammdat;
     HBAR sex / TYPE=PERCENT;
RUN;QUIT;
```

Es können auch Mittelwerte und Summen dargestellt werden. Dazu setzt man `TYPE=MEAN` oder `TYPE=SUM` und gibt gleichzeitig über die Option SUMVAR= den Namen der Variablen an, über die gemittelt bzw. summiert wird. Im folgenden Beispiel wird das mittlere Alter der Frauen und Männer mit der GCHART-Prozedur dargestellt.

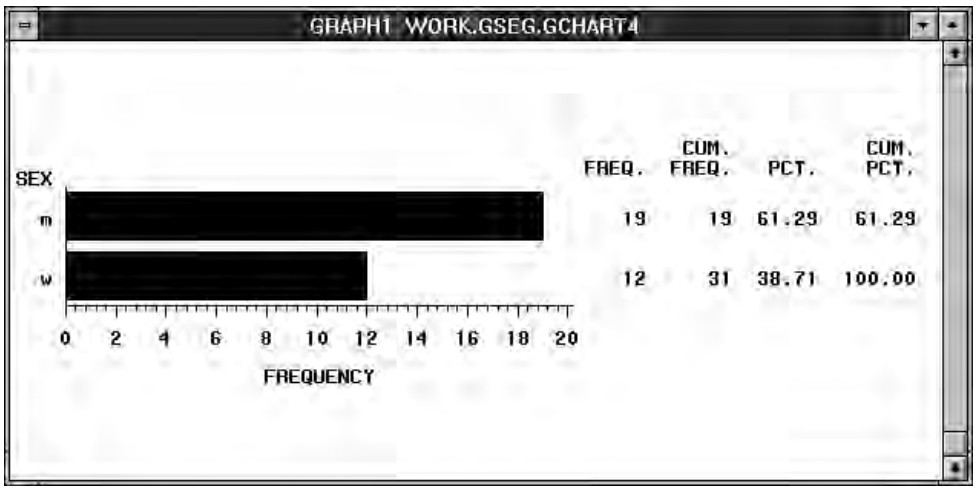


Abb. 9.4: Horizontales Balkendiagramm der Variable sex

```
/*--- KA09-03.SAS ---*/
PROC GCHART DATA=biblio.stammdat;
     HBAR sex / TYPE=MEAN SUMVAR=alter ERRORBAR=TOP NOSTATS;
RUN;QUIT;
```

Die Option ERRORBAR= sorgt dafür, daß zusätzlich zu den Mittelwerten auch noch der Standardfehler in die Grafik eingetragen wird (Abbildung 9.5). In der Beispielfirma sind die weiblichen Mitarbeiter nur minimal älter als die Männer. Der Durchschnitt liegt bei etwas mehr als 35 Jahren.

Neben den Balkendiagrammen unterstützt die Prozedur GCHART auch die Erzeugung von Kreisdiagrammen (Anweisung PIE), wie man sie von Wahlergebnissen kennt. Als Beispiel wird das Einzugsgebiet der Firmenmitarbeiter grafisch dargestellt.

```
/*--- KA09-04.SAS ---*/
PROC GCHART DATA=biblio.stammdat;
     PIE ort;
RUN;QUIT;
```

Wenn Sie diesen Prozedurschritt am Bildschirm ausführen, erhalten Sie eine farbige Grafik. Für den Schwarz-Weiß-Druck können über die globale Anwei-

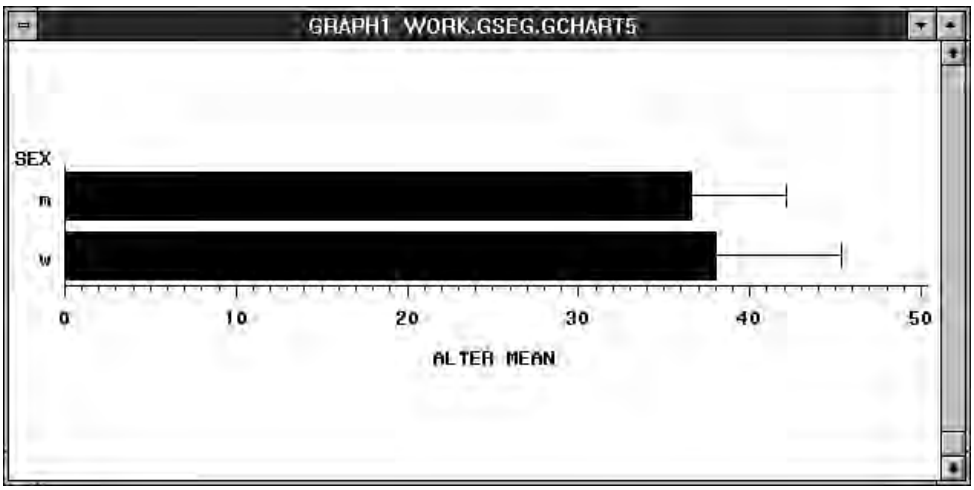


Abb. 9.5: Mittleres Alter als Balkendiagramm getrennt nach Geschlecht

sung PATTERN Graustufen zum Ausfüllen der einzelnen Kreissegmente gewählt werden. (C= wählt die Graustufe, V= das Füllmuster.)

```
/*--- KA09-05.SAS ---*/
PATTERN1 C=GRAY88 V=SOLID;
PATTERN2 C=GRAYEE V=SOLID;
PATTERN3 C=GRAY22 V=SOLID;
PATTERN4 C=GRAYAA V=SOLID;
PATTERN5 C=GRAYCC V=SOLID;
PATTERN6 C=GRAY44 V=SOLID;
```

Die Wohnorte wurden in Abbildung 9.6 alphabetisch angeordnet. Neben den Namen wurden die absoluten Häufigkeiten mit ausgegeben. Während bei Histogrammen die Häufigkeit in der Länge des Balkens wiedergegeben wurde, drückt bei den Kreisdiagrammen die Größe des Kreissegments den Anteil aus. Die meisten Mitarbeiter wohnen in den beiden Städten Heidelberg und Mannheim (11 und 9). Aus der weiter entfernt liegenden Stadt Karlsruhe dagegen kommen nur 3 Mitarbeiter.

Wurde ein kategorielles Merkmal aus Gründen des leichteren Erfassens numerisch kodiert (z. B. Geschlecht mit 1 für weiblich und 2 für männlich), werden ohne zusätzliche Angaben im PROC GCHART-Schritt zur Erzeugung eines Balkendiagramms wie bei dem numerischen Merkmal `alter` auch Klassen gebildet. Die Grafik mutet dann sehr verwunderlich an, wenn für das Geschlecht

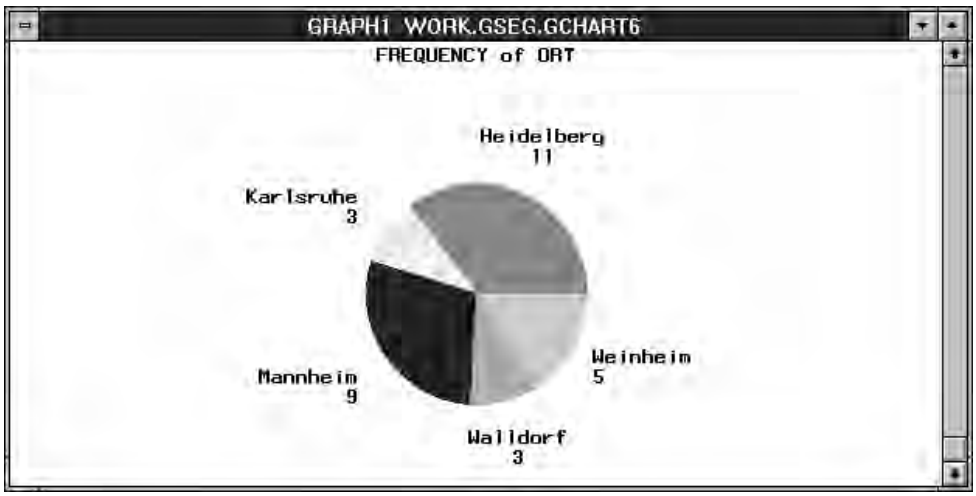


Abb. 9.6: Kreisdiagramm

Klassenmittelpunkte von 0.8 oder 1.3 ausgewiesen werden. Mit der Option DISCRETE in der Anweisung VBAR (bzw. HBAR oder PIE) verhindert man dieses Vorgehen. Die Option DISCRETE weist das System an, die numerische Variable wie eine Textvariable zu behandeln und für jede Ausprägung eine eigene Klasse zu bilden.

Über die beiden Optionen SUBGROUP= und GROUP= können die Häufigkeiten auch getrennt nach Gruppen in einem Balken- oder Kreisdiagramm dargestellt werden. Während mit GROUP= die Balken oder Kreise für die verschiedenen Ausprägungen der Gruppenvariablen nebeneinander dargestellt werden, unterteilt die Option SUBGROUP= die Balken nach der Untergruppenvariablen. (Kreissegmente lassen sich nicht unterteilen.) Zur Demonstration wird in folgendem Beispiel gruppiert nach Männern und Frauen die Altersstruktur der Mitarbeiter aus Heidelberg und Mannheim in einem Histogramm dargestellt.

```
/*--- KA09-06.SAS ---*/
PROC GCHART DATA=biblio.stammdat;
     VBAR alter / MIDPOINTS=15 to 55 by 10
                  GROUP=sex SUBGROUP=ort;
     WHERE ort IN ('Heidelberg' 'Mannheim');
RUN;QUIT;
```

Auf einer Achse, aber nebeneinander, erscheinen die beiden Balkendiagramme der Altersstruktur von Männern und Frauen (Abbildung 9.7). Die einzelnen Bal-

ken sind unterteilt in die Mitarbeiter aus Heidelberg und Mannheim. (Aus Platzgründen wurden die anderen Orte mit der Anweisung WHERE ausgegrenzt.)

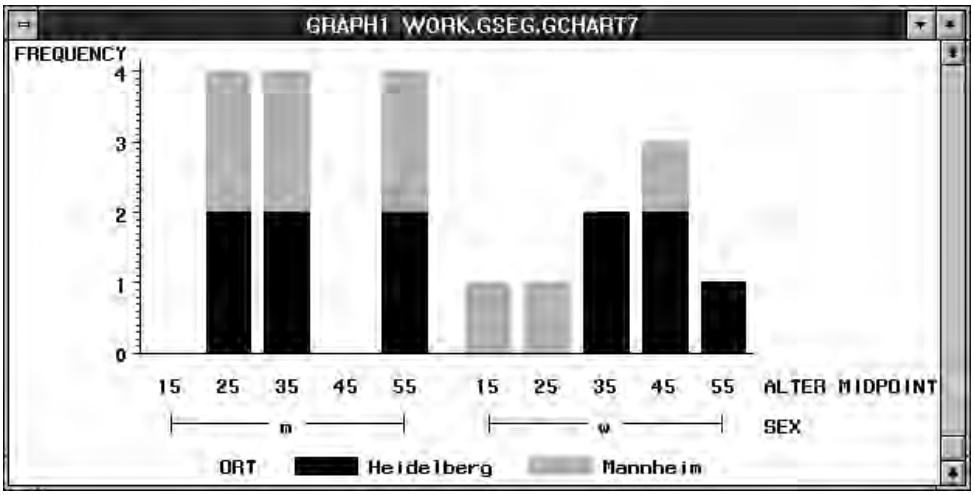


Abb. 9.7: Gruppiertes Balkendiagramm

Obwohl es noch viel zu der Prozedur GCHART zu schreiben gäbe, sollen diese Beispiele und der Hinweis auf die online verfügbaren Beispielprogramme genügen. Hier folgt noch eine Zusammenfassung der besprochenen Syntaxelemente.

```
PROC GCHART DATA=SAS-Datei;
     VBAR Variablen-Liste / <DISCRETE>
                    <MIDPOINTS=n1 n2 ...> <LEVELS=n>
                    <TYPE=FREQ|PERCENT|MEAN|SUM>
                    <SUMVAR=Variable> <ERRORBAR=TOP>
                    <GROUP=Variable> <SUBGROUP=Variable>;
     HBAR ...;
     PIE ...;
```

9.2 Punktwolken und Kurven – Die Prozedur GPLOT

Mit der Prozedur GPLOT können Zusammenhänge zwischen stetigen (oder zumindest ordinal skalierten) Merkmalen grafisch veranschaulicht werden, indem die Daten zweier Merkmale in ein Koordinatensystem eingetragen werden. Jede Beobachtung wird durch einen Punkt repräsentiert, wobei ein Merkmal auf der

horizontalen x-Achse, das andere Merkmal auf der vertikalen y-Achse aufgetragen wird. Die auf diese Weise gebildete Punktwolke zeigt mögliche Zusammenhänge zwischen den beiden Variablen auf. So kann etwa mit dem Ansteigen der x-Variable auch die y-Variable ansteigen oder abfallen, oder die Punkte liegen zerstreut im Koordinatensystem und zeigen keinerlei Zusammenhänge. Werden auf der x-Achse Kalendertage aufgetragen und auf der y-Achse Gewichtszu- oder -abnahmen, können die Punkte auch mit einer Linie verbunden werden, um die Entwicklung des Gewichts deutlich aufzuzeigen. Die Prozedur PLOT aus dem Modul SAS/BASE erzeugt ebenfalls Streu- und Liniendiagramme, allerdings im Ausgabefenster.

In Kapitel 8 wurde der Zusammenhang zwischen den Ergebnissen der vier Einzelstufen des Eingangstests bereits untersucht und mit Hilfe der Prozedur CORR in Form von Korrelationskoeffizienten als Maßzahl ausgedrückt. Im folgenden wird der Zusammenhang zwischen den Ergebnissen der 1. und 3. Stufe grafisch dargestellt.

```
/*--- KA09-07.SAS ---*/
SYMBOL1 I=NONE V=DOT H=0.5;
PROC GPLOT DATA=biblio.firma;
     PLOT teil3*teil1;
RUN;QUIT;
```

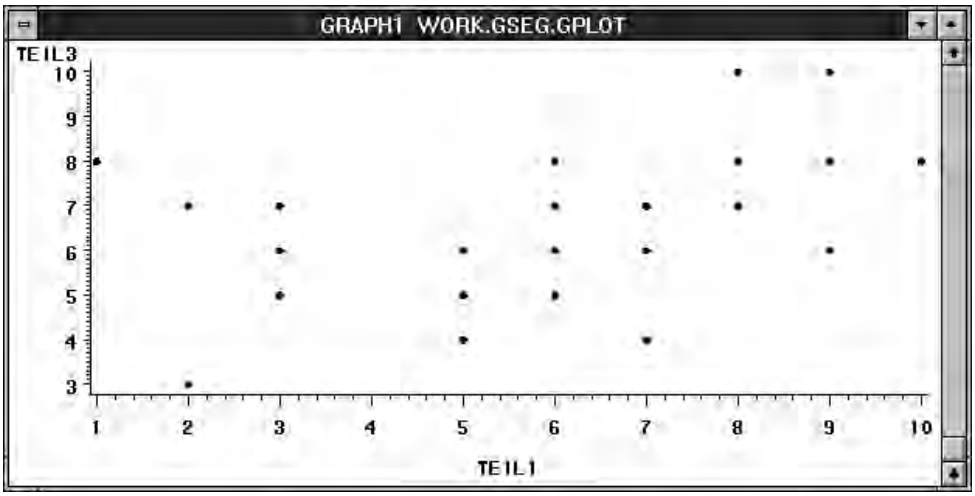


Abb. 9.8: Streudiagramm zur Darstellung von Zusammenhängen

In der Anweisung PROC GPLOT wird mit der Option DATA= die Datei übergeben. Die Anweisung PLOT bezeichnet die Variablen, die gegeneinander aufgetragen werden; die Variable links vom Stern auf der y-Achse, die Variable rechts vom Stern auf der x-Achse. In Abbildung 9.8 wird die Variable `teil1` auf die x-Achse aufgetragen, `teil3` auf die y-Achse. Die Achsenlänge wird automatisch festgelegt, so daß alle Beobachtungen im Diagramm dargestellt werden können. Mit der globalen Anweisung SYMBOL wurde das Symbol zur Darstellung der Beobachtungen festgelegt. `DOT` steht für einen Punkt. Jeder Punkt im Streudiagramm repräsentiert eine Beobachtung aus der Datei `biblio.firma`, also einen Mitarbeiter.

Sollen mehrere y-Variablen gegen ein- und dieselbe x-Variable in ein Koordinatensystem eingetragen werden, schreibt man in der Anweisung PLOT alle y-Variablen links vom Stern in Klammern[1] und fügt nach dem Namen der x-Variablen und einem Schrägstrich die Option OVERLAY zur Überlagerung der beiden Grafiken ein (siehe Abbildung 9.9). Ohne diese Option würde für jede x/y-Kombination eine eigene Grafik erstellt werden.

```
/*--- KA09-08.SAS ---*/
SYMBOL1 I=NONE C=BLACK V=DOT  H=0.5;
SYMBOL2 I=NONE C=BLACK V=STAR H=1.0;
PROC GPLOT DATA=biblio.firma;
     PLOT (teil3 teil4)*teil1 / OVERLAY LEGEND;
RUN;QUIT;
```

Über die zweite SYMBOL-Anweisung wird das zweite Plotsymbol definiert. Zusätzlich wird in der Anweisung PLOT die Option LEGEND gesetzt, damit am Fuß der Grafik eine Legende zur Erläuterung der beiden Plotsymbole erzeugt wird. Beachten sollten Sie, daß die Plotpunkte tatsächlich überlagert werden (z. B. bei (6,5) und (6,6)), die Beschriftung der y-Achse aber nicht. Hier steht `TEIL3`, obwohl auch die Werte der Variablen `teil4` eingetragen wurden. Mit der globalen Anweisung AXIS und der Option VAXIS=AXIS ließe sich dieses Problem umgehen:

```
AXIS LABEL=NONE;
PROC GPLOT DATA=biblio.firma;
     PLOT (teil3 teil4)*teil1 / OVERLAY LEGEND VAXIS=AXIS;
RUN;QUIT;
```

[1] analog zur Anweisung TABLES der Prozedur FREQ

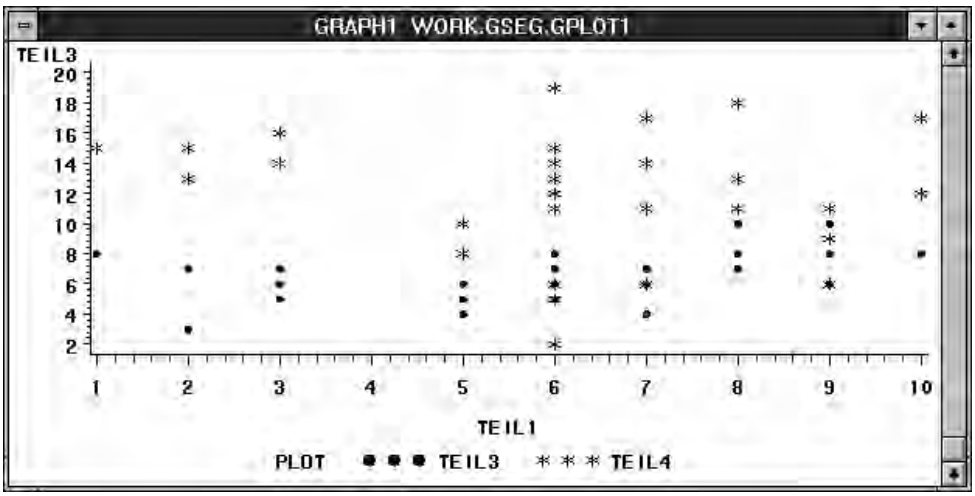


Abb. 9.9: Überlagertes Streudiagramm

Die Anweisung AXIS definiert eine Achse, bei der die Ausgabe der Beschriftung (LABEL) unterdrückt wird. Mit der Option VAXIS= in der PLOT-Anweisung wird der vertikalen Achse diese Definition zugewiesen.

Die SYMBOL-Anweisung kann zusätzlich steuern, ob die einzelnen Punkte durch Geradenstücke verbunden werden (Standard, `I=JOIN`), ob eine Regressionsgerade hinzugefügt werden soll (`I=RL`) oder ob eine glatte Kurve angepaßt werden soll (`I=SPLINE`). Für das letzte Beispiel zur Prozedur GPLOT wird eine Regressionsgerade in das Streudiagramm eingefügt.

```
/*--- KA09-09.SAS ---*/
SYMBOL1 I=NONE V=DOT H=0.5 C=BLACK;
SYMBOL2 I=RL C=BLACK;
PROC GPLOT DATA=biblio.firma;
     PLOT teil3*teil1 teil3*teil1 / OVERLAY REQEQN;
RUN;QUIT;
```

In der PLOT-Anweisung wird zweimal die Anforderung `teil3*teil1` gestellt. Die erste Grafik wird mit der ersten SYMBOL-Definition erstellt: die normale Punktwolke wie in der Grafik zuvor. Die zweite Grafik wird mit der Definition SYMBOL2 erstellt, in der über die Option `I=RL` eine Regressionsgerade angefordert wird. Wegen OVERLAY erscheinen beide Darstellungen in einer gemeinsamen Grafik (Abbildung 9.10). Die Option REQEQN in der Anweisung PLOT sorgt

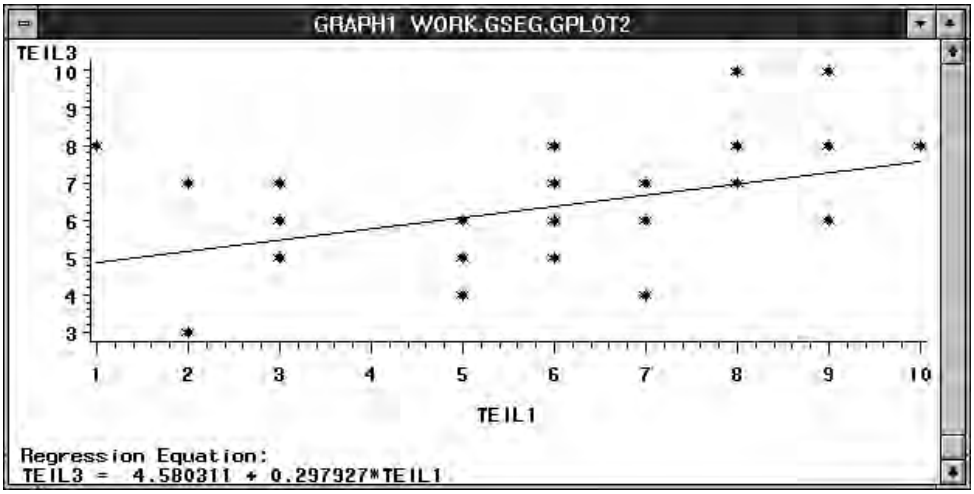


Abb. 9.10: Streudiagramm mit Regressionsgerade

schließlich noch dafür, daß am Fuß der Grafik die Gleichung der Regressionsgeraden erscheint, `TEIL3 = 4.58 + 0.30*TEIL1`.

```
SYMBOL<1|2|...> <I=NONE|JOIN|RL> <V=DOT|STAR|...>
                <H=n> <C=Farbe>;
AXIS<1|2|...> <LABEL=NONE>;
PROC GPLOT DATA=SAS-Datei;
     PLOT y-Variablen-Liste*x-Variablen-Liste
               </ <OVERLAY> <LEGEND> <REQEQN>>;
```

9.3 Schwarz auf Weiß oder „Wie bringe ich die Grafik zu Papier?“

Ausdrucken von SAS-Grafiken sieht auf den ersten Blick nicht nach einem größeren Problem aus, zumal wenn man unter Windows arbeitet: Sobald eine Grafik angezeigt wird, klickt man auf die Schaltfläche und die Grafik wird am aktuell eingestellten Drucker ausgedruckt. Doch bereits bei dieser Aktion kann es vorkommen, daß, bevor die Grafik ausgedruckt wird, das System den Anwender nach einem *Device name* fragt. Mit `winprtm` geht es anschließend zwar weiter, aber das Resultat ist bescheiden.

Bevor eine Grafik „richtig schön“ ausgedruckt werden kann, muß dem SAS-System der Gerätetreiber mitgeteilt werden: der *Device*. Der Anwender kann dabei wählen, ob er direkt den unter Windows aktiven Drucker ansteuern möchte oder ob ein Drucker aus der Liste der SAS-Gerätetreiber verwendet werden soll. Für die Windows-Drucker muß nicht das spezielle Druckermodell gewählt werden, denn diese Information kennt das Betriebssystem, sondern man wählt nur aus, ob der Drucker schwarz-weiß (WINPRTM), in Graustufen (WINPRTG) oder farbig (WINPRTC) drucken kann oder ob es sich um einen Plotter handelt (WINPLOT, vgl. auch Tabelle 9.1). Dem System wird der Name mittels der Option DEVICE= in der GOPTIONS-Anweisung vor Aufruf des Prozedurschritts übermittelt:

```
/*--- KA09-10.SAS ---*/
GOPTIONS DEVICE=WINPRTG;
PROC GPLOT DATA=biblio.firma;
     PLOT teil4*alter;
RUN;QUIT;
```

Die Grafik wird hierdurch nicht im Grafikfenster angezeigt, sondern direkt zum Drucker weitergeleitet. Soll die Grafik vor dem Ausdrucken zu Kontrollzwecken in einem Grafikfenster angezeigt werden (*Preview*), muß die Option DEVICE= durch TARGETDEVICE= (oder kurz TARGET=) ersetzt werden.

```
GOPTIONS TARGET=WINPRTG;
PROC GPLOT DATA=biblio.firma;
     PLOT teil4*alter;
RUN;QUIT;
```

Das Streudiagramm wird zunächst im Grafikfenster angezeigt. Zum Ausdruck klickt man auf die Schaltfläche  . Die Anweisung GOPTIONS ist eine globale Anweisung wie LIBNAME und FILENAME auch, d. h., sie muß nur einmal ausgeführt werden, damit für die gesamte SAS-Sitzung der Drucker korrekt eingestellt ist.

Neben diesen Windows-Druckern unterstützt das SAS-System durch entsprechende Gerätetreiber weitere Drucker. HPLJ300 etwa ist der Gerätetreiber für die Hewlett Packard LaserJet Series II Drucker. Mit der Anweisung `GOPTIONS TARGET=HPLJ300;` bzw. `GOPTIONS DEVICE=HPLJ300;` wird die Grafik für den HP LaserJet entsprechend aufbereitet. Das SAS-System bietet außerdem Treiber für andere Textverarbeitungs- und Grafikprogramme an. Die SAS-Grafiken können damit in einem geeigneten Format ausgegeben werden, so daß sie von den anderen Programmen eingelesen und weiterverarbeitet werden können. Die

Device Name	für
WIN	Bildschirm unter Windows
WINPRTM	monochromer Drucker
WINPRTG	Graustufendrucker
WINPRTC	Farbdrucker
WINPLOT	Farbplotter
CGMMW6C	Word für Windows 6.0
CGMMWWC	Word für Windows 2.0
WCGMWPWA	Word Perfect für Windows
PSEPSF	Encapsulated PostScript
PSLEPSF	Encapsulated PostScript, dünne Linien
PSCLRA4	Farbiges PostScript im DinA4-Format

Tab. 9.1: Wichtige Windows- und Gerätetreiber

Zeiten, in denen SAS-Grafiken mittels Schere, Papier und Klebstoff in Textdokumente eingebunden wurden, gehören der Vergangenheit an.

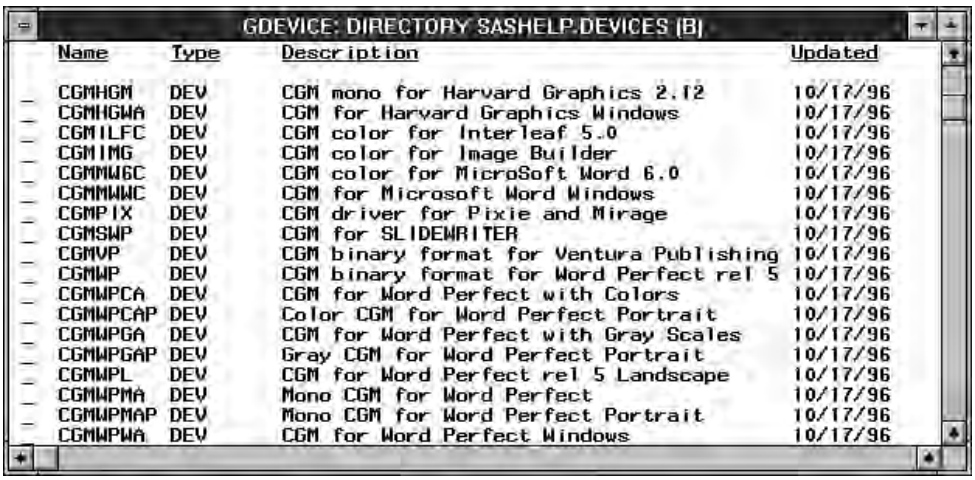
Die mit der SAS-Software ausgelieferten Gerätetreiber befinden sich alle in dem Katalog SASHELP.DEVICES, dessen Inhalt über das Libraries-Fenster aufgelistet oder mit der Prozedur GDEVICE geöffnet werden kann (Abbildung 9.11).

```
/*--- KA09-11.SAS ---*/
PROC GDEVICE CAT=SASHELP.DEVICES;
RUN;
```

Da die Namen der Treiber wie alle Namen im SAS-System auf acht Zeichen beschränkt sind, erscheinen manche Treibernamen recht kryptisch. Der Treiber für Word for Windows 6.0 beispielsweise heißt CGMMW6C. CGM steht als Abkürzung für *Computer Graphics Metafile*, eines der unter Windows üblichen Grafikformate. Die beiden nächsten Buchstaben, MW, stehen für *Microsoft Word für Windows*; die Zahl 6 für die Programmversion 6.0 und C für die Variante *c* des Treibers. Eigentlich ganz logisch, oder?

Um eine SAS-Grafik nach Word for Windows 6.0 zu übertragen, muß der Treiber CGMMW6C in der Anweisung GOPTIONS verwendet werden:

```
GOPTIONS TARGET=CGMMW6C;
PROC GPLOT DATA=biblio.firma;
     PLOT teil4*alter;
RUN;QUIT;
```



GDEVICE: DIRECTORY SASHELP.DEVICES (B)

Name	Type	Description	Updated
CGMHGM	DEV	CGM mono for Harvard Graphics 2.12	10/17/96
CGMHGWA	DEV	CGM for Harvard Graphics Windows	10/17/96
CGMILFC	DEV	CGM color for Interleaf 5.0	10/17/96
CGMIMG	DEV	CGM color for Image Builder	10/17/96
CGMMW6C	DEV	CGM color for MicroSoft Word 6.0	10/17/96
CGMMWWC	DEV	CGM for Microsoft Word Windows	10/17/96
CGMPIX	DEV	CGM driver for Pixie and Mirage	10/17/96
CGMSWP	DEV	CGM for SLIDEWRITER	10/17/96
CGMVP	DEV	CGM binary format for Ventura Publishing	10/17/96
CGMWP	DEV	CGM binary format for Word Perfect rel 5	10/17/96
CGMWPCA	DEV	CGM for Word Perfect with Colors	10/17/96
CGMWPCAP	DEV	Color CGM for Word Perfect Portrait	10/17/96
CGMWPGA	DEV	CGM for Word Perfect with Gray Scales	10/17/96
CGMWPGAP	DEV	Gray CGM for Word Perfect Portrait	10/17/96
CGMWPL	DEV	CGM for Word Perfect rel 5 Landscape	10/17/96
CGMWPMA	DEV	Mono CGM for Word Perfect	10/17/96
CGMWPMAP	DEV	Mono CGM for Word Perfect Portrait	10/17/96
CGMWPWA	DEV	CGM for Word Perfect Windows	10/17/96

Abb. 9.11: Der Katalog SASHELP.DEVICES

Wegen der Option TARGET= wird die Grafik zuerst am Bildschirm angezeigt und anschließend, wenn gewünscht, mit zum Ausgabegerät geschickt. In diesem Fall ist dies nicht der Drucker, sondern standardmäßig die Datei GRAPH.GSF im aktuellen SAS-Verzeichnis (vgl. Abbildung 2.2). Die Grafik wird in einer Art Metasprache in dieser Datei abgelegt, so daß sie anschließend in das aktuelle Worddokument mittels der Menüfolge *Einfügen* → *Grafik* eingebunden werden kann.

Werden in einem Programm mehrere Grafiken erzeugt, werden alle in die eine Datei GRAPH.GSF eingetragen, was äußerst unpraktisch ist, da kaum eine Anwendungssoftware in der Lage ist, mehrere Grafiken aus einer Metadatei auszulesen. Dieses Problem läßt sich nun dadurch umgehen, daß man die Grafikdatei anders benennt. Dazu sind zwei Schritte notwendig:

❶ Mit der Anweisung FILENAME wird ein Referenzname für die bestehende oder neu zu bildende Grafikdatei definiert.

❷ Über die Option GSFNAME= der Anweisung GOPTIONS wird dieser Referenzname als Metadatei für die SAS-Grafik gesetzt.

Als komplettes Programm sieht das ganze wie folgt aus:

```
/*--- KA09-12.SAS ---*/
FILENAME grafik 'C:\kurs\grafik1.cgm';
GOPTIONS TARGET=CGMMW6C GSFNAME=grafik GSFMODE=REPLACE;
PROC GPLOT DATA=biblio.firma;
     PLOT teil1*alter;
RUN;QUIT;
```

Die Grafikdatei wird auf der Festplatte im Übungsverzeichnis `C:\kurs` unter dem Namen `grafik1.cgm` abgelegt. Dieser Name ist frei wählbar, er sollte jedoch so eindeutig sein, daß sein Wiedererkennungswert hoch ist. Die Erweiterung `cgm` weist beispielsweise daraufhin, daß es sich um eine Grafikdatei im CGM-Format handelt, im Gegensatz zu PostScript-, Gif- und Tiff-Formaten, die mit PS oder EPS, GIF und TIF bezeichnet werden könnten. Die Anweisung GOPTIONS legt den Treiber fest, `TARGET=CGMMW6C`.

Die Option GSFNAME= definiert den Referenznamen für die Grafikdatei, wobei die Grafikdatei, wenn sie bereits existiert hat, aufgrund der Option GSFMODE=REPLACE ersetzt wird[2].

Führt man das Programm aus, erscheint die Grafik wie erwartet zunächst im Grafikfenster und wenn sie gefällt, klickt man auf die Schaltfläche  . Nach einer kurzen Weile erscheint in der Statuszeile links unten am Bildschirmrand *Done* .... Verläßt man das Grafikfenster, sollte im Protokollfenster als Hinweis eine Vollzugsmeldung der folgenden Art stehen:

```
NOTE:  44 RECORDS WRITTEN TO GRAFIK.
```

Die Anzahl der `RECORDS`, d. h. Zeilen, die in die Grafikdatei eingetragen wurden, hängt von der Größe der Grafik ab. Verwendet man anstelle der Option TARGET= die Option DEVICE=, wird die Grafik direkt in die Grafikdatei eingetragen. Auch in diesem Fall sollte auf die Meldung im Protokollfenster geachtet werden.

Werden mehrere Grafiken in ein Dokument eingebunden und müssen die Grafiken eventuell mehrfach erstellt werden, empfiehlt es sich, die Größe der Grafik mit den Optionen HSIZE= und VSIZE= der Anweisung GOPTIONS festzulegen.

[2] Die Alternative dazu wäre GSFMODE=APPEND, aber dann stehen mehrere Grafiken in der Metadatei.

```
FILENAME grafik 'C:\kurs\grafik1.cgm';
GOPTIONS TARGET=CGMMW6C GSFNAME=grafik GSFMODE=REPLACE
         HSIZE=10 CM VSIZE=7 CM;
PROC GPLOT DATA=biblio.firma;
     PLOT teil1*alter;
RUN;QUIT;
```

Die Größen HSIZE= und VSIZE= können wie im Beispiel oben in Zentimetern angegeben werden. Das Streudiagramm erhält somit eine Breite (HSIZE=) von 10 cm und eine Höhe (VSIZE=) von 7 cm. Die Einheiten können aber auch in Inches (INCH), Prozentwerten relativ zum Gesamtbild (PERCENT/PCT) und in Bildzellen (CELLS) angegeben werden. Letztere hängen von der Auflösung des Treibers ab und sind schwer abzuschätzen. Dennoch ist dies die Standardeinstellung, wenn keine Einheit mit dem Zahlenwert angegeben wurde.

Die Schriftarten (engl. *font*), die das SAS-System für die Metagrafiken verwendet, hängen von dem jeweiligen Gerätetreiber ab. Bei CGMMW6C sind dies beispielsweise für normalen Text ARIAL und für Überschriften ARIAL BOLD. Wird eine solche SAS-Grafik in ein Worddokument eingebunden, das in z. B. TIMES ROMAN geschrieben ist, kann der Schriftartenwechsel den Leser des fertigen Dokuments erheblich stören.

GDEVICE: Chartype

Catalog: SASHELP.DEVICES Entry: CGMMW6C

Chartype	Rows	Cols	Font Name	Scalable
1	50	80	Arial	Y
2	50	80	ArialBold	Y
3	50	80	ArialItalic	Y
4	50	80	ArialBoldItalic	Y
5	50	80	TimesRoman	Y
6	50	80	TimesRomanBold	Y
7	50	80	TimesRomanItalic	Y
8	50	80	TimesRomanBoldItalic	Y
9	50	80	Script	Y
10	50	80	ScriptBold	Y
11	50	80	ScriptItalic	Y
12	50	80	ScriptBoldItalic	Y
0	0	0		
0	0	0		
0	0	0		
0	0	0		
0	0	0		
0	0	0		

Abb. 9.12: Unterstützte Schriftarten für Word for Windows 6.0

Das SAS-System bietet neben der Schriftart ARIAL in den Varianten NORMAL, BOLD, ITALIC und BOLD ITALIC zwei weitere häufig verwendete Schriftarten

an: TIMES ROMAN und SCRIPT mit den jeweiligen vier Varianten. Leider kann man nun nicht einfach den Namen der Schriftart angeben, sondern muß eine Nummer aus dem Schriftartenverzeichnis auswählen. Dazu ruft man die Prozedur GDEVICE wie oben auf, sucht den gewünschten Treiber, im vorliegenden Fall CGMMW6C, und öffnet diesen, indem man ein `s` (für *select*) in die Selektionsspalte links vom Treibernamen einträgt. Danach aktiviert man die Menüfolge *Locals* → *Go to Chartype Window*, woraufhin sich ein weiteres Fenster öffnet, das die unterstützten Schriftarten auflistet. In Abbildung 9.12 sehen Sie die Schriftarten (`Font Name`), die für Word for Windows 6.0 zur Verfügung stehen: ARIAL, TIMES ROMAN und SCRIPT in den Varianten Normal, Fett, Kursiv und Fett plus Kursiv. Für den Anwender ist nun die erste Spalte `Chartype` mit den Zahlen von Bedeutung, denn er muß zur Festlegung der Schriftart neben HWCGM (für *Hardware CGM*) diese Nummer als dreistellige Zahl, etwa 005 für TIMES ROMAN angeben.

Um in obigem Beispiel etwa alle Schriftzüge in TIMES ROMAN auszugeben, erweitert man die Anweisung GOPTIONS um die Option FTEXT=:

```
FILENAME grafik 'C:\kurs\grafik1.cgm';
GOPTIONS TARGET=CGMMW6C GSFNAME=grafik GSFMODE=REPLACE
         HSIZE=10 CM VSIZE=7 CM FTEXT=HWCGM005;
PROC GPLOT DATA=biblio.firma;
     PLOT teil1*alter;
RUN;QUIT;
```

Das SAS-System selbst kann Hardware-Schriftarten leider nicht anzeigen, weshalb die Option TARGETDEVICE= in diesem Zusammenhang nur von mäßiger Bedeutung ist. Letztendlich entscheidet erst die Erscheinung im Textverarbeitungsprogramm darüber, ob die Grafik gelungen ist oder nicht.

Bindet man die mit obigem Programm (allerdings in Schriftart ARIAL) erstellte Grafik in Word for Windows 6.0 ein, werden, wie in Abbildung 9.13 zu sehen ist, die Beschriftung der y-Achse und ein Teil der Achseneinteilungen nicht angezeigt.

Dieser Fehler läßt sich nur mit einem Griff in die Trickkiste umgehen: Man überlagert die Grafik mit einem unsichtbaren Rahmen, der dafür sorgt, daß die Außenmaße der Grafik eingehalten werden. Den Rahmen erzeugt man mit speziellen Makros, *Annotate-Makros*, in einem Datenschritt. Im Prozedurschritt zur Erzeugung der eigentlichen Grafik wird über die Option ANNO= auf den Rahmen, genauer auf die Datei `rahmen`, zugegriffen.

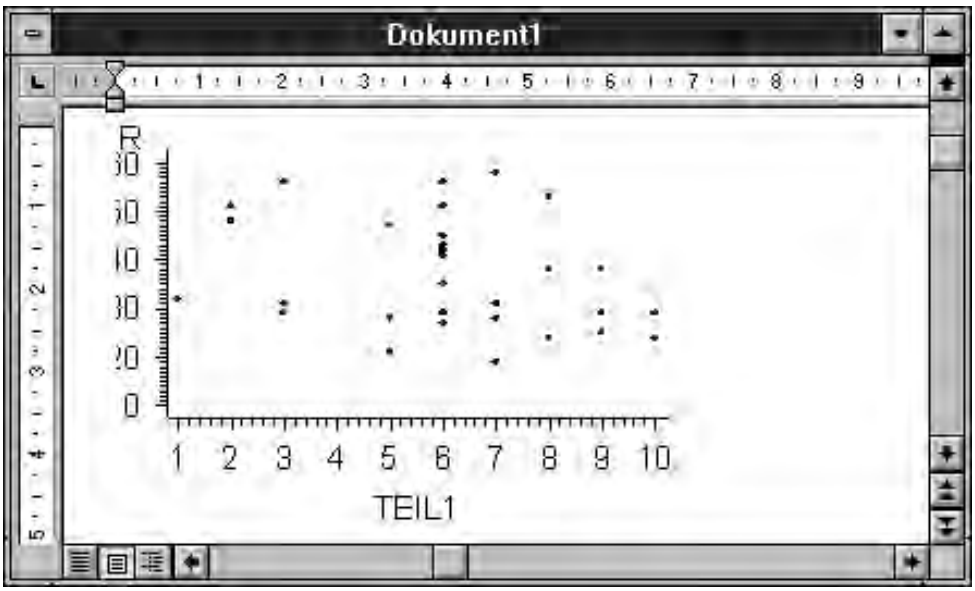


Abb. 9.13: Streudiagramm in Word for Windows 6.0 angezeigt

```
/*--- KA09-13.SAS ---*/
%ANNOMAC;
DATA biblio.rahmen;
     %DCLANNO;
     %SYSTEM(3,3,3);
     %FRAME(WHITE,1,1,EMPTY);
RUN;
PROC GPLOT DATA=biblio.firma ANNO=biblio.rahmen;
     PLOT teil1*alter;
RUN;QUIT;
```

Die Datei `biblio.rahmen` besteht, nachdem der Datenschritt ausgeführt wurde, aus einer Beobachtung und 12 Variablen[3]. Die globalen Anweisungen FILENAME und GOPTIONS müssen nicht wiederholt werden. In der Anweisung PROC GPLOT wurde die Option ANNO= ergänzt. Mit ihr wird auf die Datei `biblio.rahmen` zugegriffen. Führt man dieses Programm aus und erzeugt die

[3] Sie finden diese Datei auch auf der Begleitdiskette im Unterverzeichnis `A:\kurs`.

Metadatei durch Klicken auf die Schaltfläche , kann die Grafik komplett mit allen Beschriftungen in das Worddokument eingefügt werden, wie in Abbildung 9.14 wiedergegeben. Bei Grafiken, die sehr viel Begleittext enthalten, kann es zu Problemen bei der Positionierung der Texte kommen. Hier muß der Anwender für sich entscheiden, ob er mehr Zeit für die Erstellung der Grafiken aufwendet oder besser auf die Hardware-Schriftarten verzichtet.

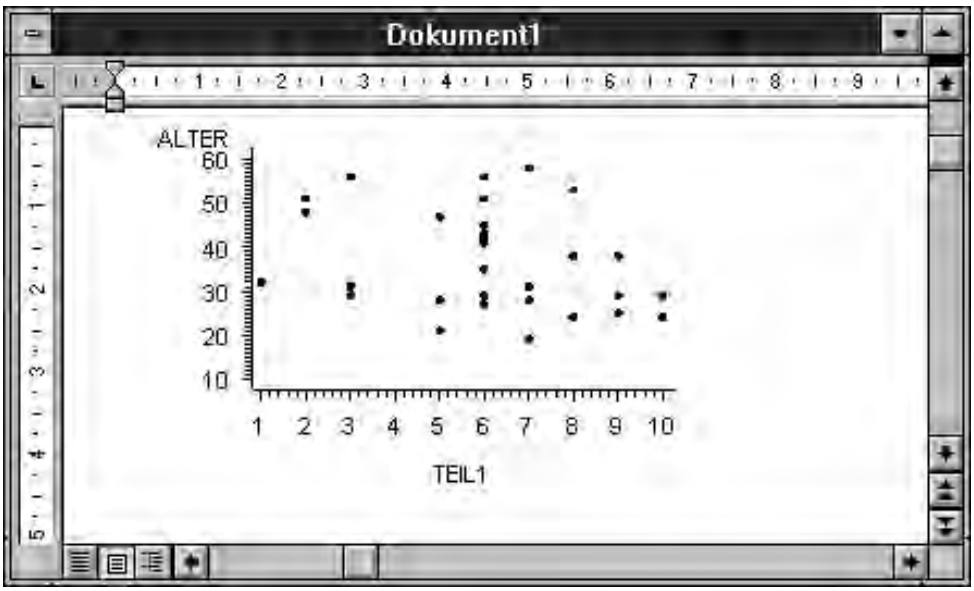


Abb. 9.14: Streudiagramm mit zusätzlichem Rahmen

Zum Schluß werden nochmals alle in diesem Abschnitt zum Einsatz gekommenen Optionen der Anweisung GOPTIONS zusammengestellt:

```
GOPTIONS <DEVICE=treiber> <TARGETDEVICE=treiber>
         <GSFNAME=fileref> <GSFMODE=REPLACE|APPEND>
         <HSIZE=n CM|INCH|PERCENT|CELLS>
         <VSIZE=n CM|INCH|PERCENT|CELLS>
         <FTEXT=schriftart>;
```

Die weiteren Optionen für die Anweisung GOPTIONS erhalten Sie, wenn Sie sich die aktuellen Einstellungen durch Ausführung eines PROC GOPTIONS Prozedurschritts ohne weitere Optionen und Anweisungen ansehen. Wer genauer wissen möchte, was die eine oder andere Option bedeutet, sei auf das SAS/GRAPH Handbuch [12, Vol. 1] verwiesen.

Beim Experimentieren mit der Anweisung GOPTIONS kann es schon mal vorkommen, daß am Ende gar nichts mehr richtig geht. In diesem Fall beendet man entweder die komplette Sitzung und fängt von neuem an oder man führt die Anweisung `GOPTIONS RESET=ALL;` aus, damit alle Optionen wieder auf ihre Standardeinstellung zurückgesetzt werden.

Das SAS-System bietet noch eine weitere Möglichkeit, Grafiken in Metadateien zu übertragen, um auf sie an anderer Stelle zugreifen zu können: den *Export* über das *File*-Menü (vgl. Abbildung 9.15). Diese Methode geht wesentlich schneller als die Festlegung der Anweisung GOPTIONS mit ihren vielen Optionen. Allerdings unterstützt die Exportfunktion nicht so viele verschiedene Treiber, sondern im wesentlichen die Grafikformate BMP, WMF, GIF, TIF und JPG. Die Qualität der Grafiken ist gegenüber CGM-Grafiken nicht ganz so gut, da sie keine Hardware-Schriftarten unterstützen und nachträglich schlecht verändert werden können. Sollen jedoch gerade diese Grafikformate erzeugt werden, etwa um Grafiken fürs World Wide Web aufzubereiten, ist der Weg über *File* → *Export* eine effektive Variante.

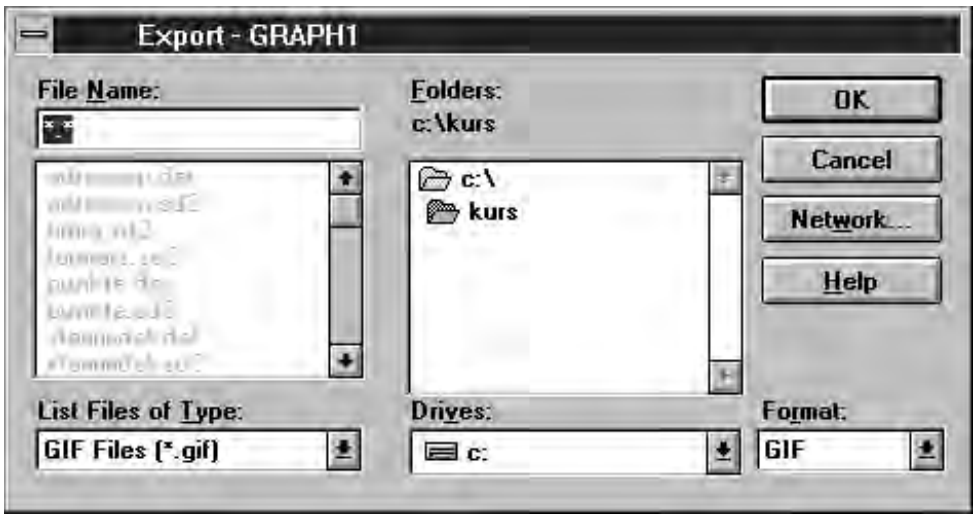


Abb. 9.15: Grafikexport

9.4 Aufgaben

❶ Stellen Sie dar, aus welchem Einzugsgebiet die Mitarbeiter der Beispielfirma jeden Tag zur Arbeit anreisen. Erzeugen Sie dazu ein vertikales Balkendiagramm, aus dem sich ablesen läßt, wieviele Mitarbeiter aus welchen Wohnorten kommen.

❷ Greifen Sie nochmals auf die Kur-Daten zurück, die am Ende des letzten Kapitels angelegt wurden.

Erzeugen Sie ein Kreisdiagramm, anhand dessen Sie ablesen können, wieviele Frauen jeweils in den beiden Gruppen teilnahmen.

❸ Erzeugen Sie ein horizontales Balkendiagramm der Gewichtsangaben zu Beginn der Kur. Unterteilen Sie die Balken nach der Gruppenzugehörigkeit.

❹ Tragen Sie die Gewichtsdifferenzen getrennt für die beiden Gruppen in einem vertikalen Balkendiagramm auf.

❺ Tragen Sie die Gewichtsangaben zu Beginn und zum Ende der Kur in ein Streudiagramm ein. Unterscheiden Sie die Punkte für die beiden Gruppen.

❻ Ergänzen Sie das Streudiagramm um die Regressionsgerade.

10

Ganz global – Optionen, Titel, Tools und Keys

Die im Zusammenhang mit den Grafiken behandelten Anweisungen SYMBOL, PATTERN und AXIS gehören genauso wie die Anweisungen LIBNAME, FILENAME und GOPTIONS zu den globalen Anweisungen. Diese globalen Anweisungen werden unabhängig von Daten- und Prozedurschritten definiert (vgl. Abschnitt 3.1), und ihre Einstellungen gelten so lange, bis eine neue Definition erfolgt oder die Definition zurückgesetzt wird. Neben den grafischen Optionen, die über die Anweisung GOPTIONS festgelegt werden, gibt es auch Systemeinstellungen, die das Ausgabefenster und das SAS-System im allgemeinen betreffen. Die Festlegungen werden mit der globalen Anweisung OPTIONS getroffen. Titel- und Fußzeilen zur Beschriftung von Grafiken und anderen Prozedurergebnissen im Ausgabefenster werden über die globalen Anweisungen TITLE und FOOTNOTE gesteuert.

Alle globalen Einstellungen werden beim SAS-Systemstart mit bestimmten Standardwerten, den *Defaults*, belegt. Der Anwender kann diese ändern und an die eigenen Bedürfnisse anpassen oder sie einfach ausschalten. Wenn Sie in einer größeren Einrichtung mit dem SAS-System arbeiten, gibt es wahrscheinlich einen Systemverwalter, der sich um die Installation und Programmpflege des SAS-Systems kümmert. Dieser kann in den beiden Konfigurationsdateien AUTOEXEC.SAS und CONFIG.SAS Systemoptionen ändern und Anweisungen eintragen, die zu Beginn jeder SAS-Sitzung ausgeführt werden und für alle Anwender gültig sind.

Als Anwender müssen Sie diese Einstellungen aber nicht als gegeben hinnehmen, sondern können die meisten Einstellungen nachträglich, d. h. im Verlauf Ihrer Arbeitssitzung, wieder ändern. Nur einige Systemoptionen, wie z. B. AUTOEXEC, MACRO und NEWS, können nur vor Aufruf der Sitzung festgelegt

werden, nachträglich lassen sie sich nicht ändern. Zu den wichtigsten globalen Anweisungen gehören die TITLE- und FOOTNOTE-Anweisungen zur Festlegung von Überschriften und Fußzeilen sowie die OPTIONS-Anweisung, mit der die Darstellungen im Ausgabefenster gesteuert werden.

10.1 Titel- und Fußzeilen

Im einführenden Beispiel des Abschnitts 3.3 wurde die globale Anweisung TITLE genutzt, um die Ausgabe der Prozedur PRINT mit der Überschrift `Adressen` zu versehen. Titel und Fußzeilen helfen, die Darstellung im Ausgabefenster für Außenstehende verständlicher und leichter lesbar zu machen. Analog dazu kann man mit der Anweisung FOOTNOTE Fußzeilen definieren, die ebenfalls auf jeder Seite des Ausgabefensters ausgegeben werden. Die Titel und Fußzeilen erscheinen auch im Grafikfenster am oberen bzw. unteren Rand, wie das folgende Beispiel zeigt. In das Programm, mit dem Abbildung 9.2 (Seite 188) erzeugt wurde, werden vor den Prozedurschritt folgende Anweisungen eingefügt:

```
TITLE 'Altersstruktur in der Firma';
FOOTNOTE 'Stand: 31. Dezember 1996';
```

Das Resultat wird in Abbildung 10.1 dargestellt.

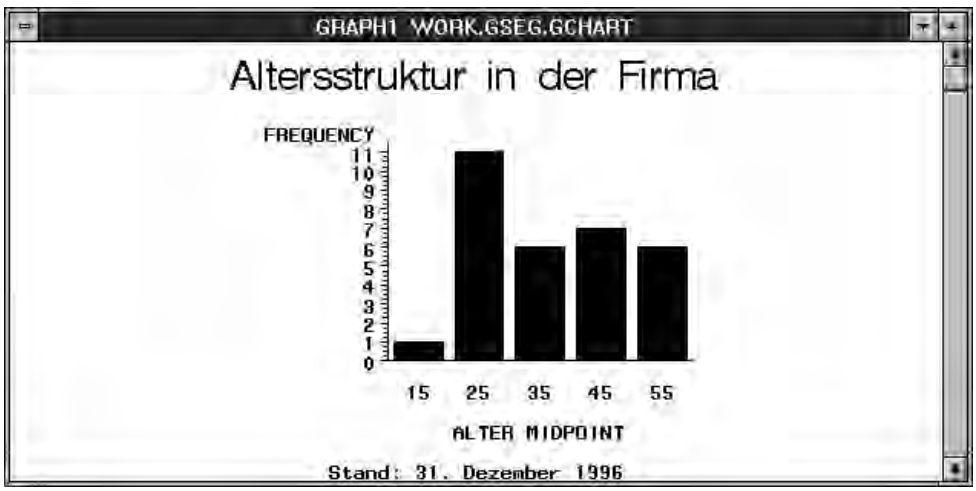


Abb. 10.1: Histogramm mit Titel- und Fußzeile

Im Ausgabefenster erscheint die Fußzeile allerdings selten direkt unterhalb einer Tabelle oder Liste. Sie wird in Abhängigkeit von der Seitenlänge, die wiederum vom aktuell eingestellten Drucker abhängt, am unteren Rand der Seite angebracht. Der Standardtitel, d. h. der Titel, den das SAS-System ausgibt, ohne daß die Anweisung TITLE gesetzt wurde, lautet: `The SAS System`. Die Standardfußzeile ist leer. Sobald eine eigene TITLE-Anweisung ausgeführt wurde, wird der Standardtitel überschrieben. Es können bis zu zehn Titel- und Fußzeilen definiert werden.

```
/*--- KA10-01.SAS ---*/
TITLE1 'Firma: Standort Deutschland';
TITLE2 'Mitarbeiterverzeichnis';
FOOTNOTE1 'Stand: 31. Dezember 1996';
FOOTNOTE2 'Erstellt von Herrn Meier';
PROC PRINT DATA=biblio.firma;
     VAR vorname name wohnort;
RUN;
```

Im obigen Beispiel wurden je zwei Titel- und zwei Fußzeilen definiert, die im Ausgabefenster neben den Datenwerten der Mitarbeiter ausgegeben werden: die Titelzeilen untereinander, die Fußzeilen übereinander.

```
            Firma: Standort Deutschland
               Mitarbeiterverzeichnis

      OBS    VNAME        NNAME         ORT

        1    Hans         Meier      Walldorf
        2    Karin        Schulz     Mannheim
        3    Melanie      Frisch     Heidelberg
       ..    ...          ...        ...
       29    Albert       Bauer      Weinheim
       30    Manfred      Weber      Mannheim
       31    Stefan       Scholz     Heidelberg

              Stand: 31. Dezember 1996
              Erstellt von Herrn Meier
```

Als globale Anweisungen gelten die definitierten Fuß- und Titelzeilen so lange, bis eine neue Definition erfolgt oder bis alle Definitionen mit den folgenden Anweisungen zurückgesetzt werden:

```
TITLE;
FOOTNOTE;

* oder alternativ:
GOPTIONS RESET TITLES FOOTNOTES;
```

Mit TITLE3 dagegen würden die dritte Titelzeile und alle Titelzeilen mit höherer Nummer zurückgesetzt. Die erste und zweite Titelzeile blieben verändert.

Die Titel- und Fußzeilen im Ausgabefenster werden je nach Systemeinstellung (Option CENTER|NOCENTER) zentriert oder linksbündig ausgegeben. Weitere Gestaltungsmöglichkeiten gibt es hierbei nicht. Ganz anders verhält es sich dagegen bei Grafiken. Im Grafikfenster stehen Farben zur Verfügung, verschiedene Schriftarten und die einzelnen Zeichen können in unterschiedlicher Größe dargestellt werden. Bei der Altersverteilung in Abbildung 10.1 beispielsweise erscheint die Titelzeile doppelt so groß wie der übrige Text und in einer anderen Schriftart (SWISS). Die Fußzeile wird in der gleichen Größe und Schrift wie der übrige Text ausgegeben.

Über Optionen können in den TITLE- und FOOTNOTE-Anweisungen andere Farben, Größen und Schriftarten gewählt werden.

- Die Farbe wird über die Option COLOR= (oder kurz C=) festgelegt. Die Farben, die konkret zur Verfügung stehen, hängen allerdings vom gewählten Gerätetreiber ab. So kann ein Bildschirm (DEV=WIN) bis zu 256 Farben darstellen, während ein HP LaserJet Series II nur Schwarz/Weiß drucken kann. Die Farbnamen werden auf Englisch angegeben: BLACK, YELLOW, RED ...

- Mit der Option HEIGHT= (oder H=) wird die Größe der Buchstaben festgelegt. Als Einheiten gibt man, analog zur Größe einer Grafik, CM für Zentimeter oder PCT für Prozent an, oder man läßt die Einheit weg, womit die Standardeinheit Zeichenzellen (CELLS, PIXELS) verwendet wird.

- Die Schriftart wird mit FONT= (oder F=) festgelegt. Das SAS-System stellt verschiedene Schriftarten zur Verfügung, z. B. SWISS, CENT, CENTB, aber auch Hardware-Schriftarten wie HWCGM001 (vgl. Abschnitt 9.3).

- Die Option JUSTIFY= (J=) erlaubt eine linksbündige (J=L), zentrierte (J=C) oder rechtsbündige (J=R) Ausrichtung des Textes.

Eine Titel- oder Fußzeile kann in mehreren Farben, Größen und Schriftarten dargestellt werden. Dazu wird der gesamte Text in Teilstücke zerlegt, die jedes

für sich in Hochkommata eingeschlossen sind. Vor dem jeweiligen Teilstück werden die Optionen zu dessen Darstellung gesetzt. Mit der Anweisung

```
TITLE C=BLACK  H=4 CM 'Schwarz'
      C=RED    H=3 CM 'Rot'
      C=YELLOW H=2 CM 'Gelb';
```

wird beispielsweise der Text *Schwarz Rot Gelb* in den entsprechenden Farben ausgegeben, und die Größe der Buchstaben der drei Worte zwischen 2 und 4 Zentimetern variiert. Im ersten Band des SAS/GRAPH-Handbuchs [12] finden Sie weitere Beispiele für den Einsatz von Titel- und Fußzeilen in Grafiken.

Die Befehlssyntax der Anweisungen TITLE und FOOTNOTE hat folgende Form, wobei die Gestaltungsmöglichkeiten hinsichtlich Farbe, Schriftart und Größe nur in Grafiken umgesetzt werden können:

```
TITLE<1|2|...10>    <<C=Farbe> <H=n <Einheiten>>
                     <J=L|C|R> 'Text'> ... ;
FOOTNOTE<1|2|...10> <<C=Farbe> <H=n <Einheiten>>
                     <J=L|C|R> 'Text'> ...;
```

10.2 Systemoptionen

Unter den Systemoptionen, die mit der Anweisung OPTIONS geändert werden können, gibt es Schalter, bei denen der Anwender zwischen den beiden Stellungen „Ein“ und „Aus“ wählen kann, und freie Einstellungen, denen explizit Werte zugewiesen werden müssen. Ein Teil der Optionen steuert beispielsweise die Darstellung der Prozedurresultate im Ausgabefenster. Der Schalter DATE gibt das aktuelle Datum rechtsbündig, unterhalb der Titelzeile aus, während die Ausgabe mit NODATE unterbleibt. Auch CENTER zur zentrierten Positionierung der Resultate im Ausgabefenster ist ein Schalter. Mit NOCENTER wird das Resultat, so wie in fast allen Beispielen in diesem Kurs, linksbündig ausgegeben. Bei der Seitenlänge und -breite handelt es sich um eine Einstellung. Hier muß dem System mitgeteilt werden, wieviele Druckzeilen bzw. -spalten auf eine Seite passen. Normalerweise entnimmt das System diese Information dem aktuell verfügbaren Drucker. Aber manchmal möchte man, wie etwa im Fall des vorliegenden Buchs, die Ergebnisse schmaler ausgeben. Dann muß man die Option LINESIZE= (LS=) setzen und den erforderlichen Wert angeben, `OPTIONS LS=64;`. Die Option für die Länge lautet PAGESIZE=.

Bevor man Veränderungen an den Einstellungen vornimmt, sollte man sich die aktuellen Einstellungen im Options-Fenster betrachten. Dieses Fenster öffnet man mit dem Kommando `options` oder über die Menüfolge *Globals* → *Options* → *Global options*[1]. (Man kann auch die Prozedur OPTIONS ohne weitere Anweisungen und Optionen aufrufen, um die Werte ins Protokollfenster einzutragen.) Wie Sie in Abbildung 10.2 erkennen können, befinden sich in der oberen Hälfte des Options-Fensters die Schalter. Ist das Kästchen neben der Option leer, ist die Option ausgeschaltet, wie etwa DATE; befindet sich ein Kreuz im Kästchen, ist die Option eingeschaltet (CENTER). Darunter befinden sich in alphabetischer Reihenfolge alle Optionen, zu denen der Anwender Zugang hat. Soll die Seitenbreite von 64 Zeichen wieder auf 78 Zeichen verbreitert werden, blättert man so lange nach unten, bis die Option LINESIZE angezeigt wird und trägt dann 78 ein. Mit F3 bzw. *File* → *End* werden die Veränderungen ausgeführt, während man mit *File* → *Cancel* den Änderungsvorgang abbrechen kann. Die Optionen bleiben dann unverändert.

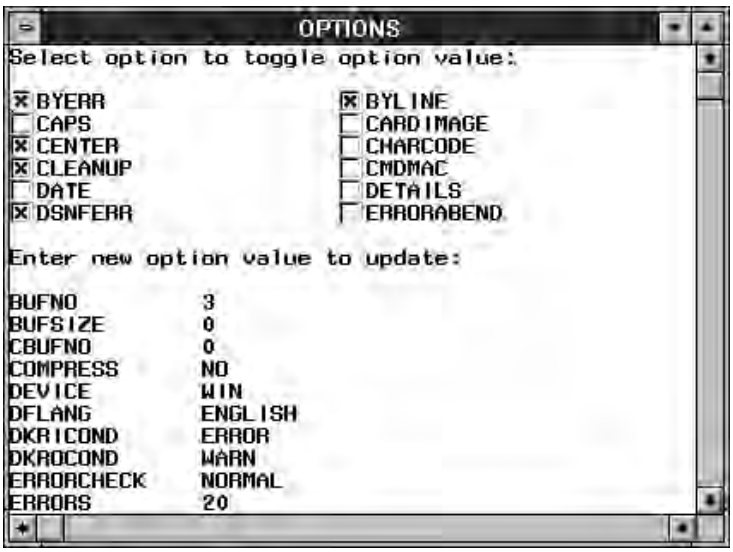


Abb. 10.2: Options-Fenster

Wenn Sie bei der Arbeit mit dem SAS-System stets die gleichen Änderungen vornehmen, können Sie die dazu notwendigen Anweisungen in die Datei AUTOEXEC.SAS eintragen. Diese Datei hat die Form eines „normalen“ SAS-Pro-

[1] Mit *Edit* → *Options* werden die Einstellungen zur Arbeit mit dem Display Manager System geändert, nicht die Systemoptionen.

gramms. Sie kann globale Anweisungen, Daten- und Prozedurschritte enthalten. Der einzige, aber wichtige Unterschied besteht darin, daß dieses Programm automatisch beim Systemstart ausgeführt wird.

Die Syntax der OPTIONS-Anweisung hat die folgende Form:

```
OPTIONS  Option|NO-Option Option=Wert ...;
```

Im Verlauf dieser Einführung wurden dabei folgende Optionen ausführlicher behandelt:

Option	Bedeutung
DATE|NODATE	Angabe des Datums
CENTER|NOCENTER	Zentrierte Ausgabe
LS=	Breite des Ausgabefensters
PS=	Länge des Ausgabefensters

10.3 Eigene Funktionstasten

Die Kommunikation zwischen dem Anwender und dem SAS-System wird durch die Funktionstasten wesentlich vereinfacht, da sie, wie bereits im einführenden Beispiel und in Kapitel 4 erläutert, mit Kommandos belegt sind, die mit einem „einfachen“ Knopfdruck ausgeführt werden können. Da jeder Anwender eigene Schwerpunkte bei der Arbeit mit der SAS-Software hat, sind auch die Anforderungen an die Funktionstastenbelegung recht unterschiedlich. Der Anfänger gibt sich eher mit der Standardbelegung zufrieden, als der professionelle Anwender.

Der einfachste Weg, die Funktionstasten neu zu belegen, bzw. freie Tasten zu belegen, führt über das Keys-Fenster. Man öffnet das Fenster mit F9 oder dem Kommando `keys` (vgl. Abbildung 4.1) und trägt in der gewünschten Zeile das Kommando ein, das per Tastendruck ausgeführt werden soll. Für Programmierer kann es wünschenswert sein, von Zeit zu Zeit sowohl das Protokoll- als auch das Ausgabefenster komplett zu löschen. Das dazu notwendige Kommando lautet `out;clear;log;clear;`. Mit `pgm` am Ende wird das Programmfenster anschließend wieder zum aktiven Fenster. Dieses zusammengesetzte Kommando wird nun auf die erste freie Funktionstaste, F12, eingetragen.

Drückt man nun F12, sollte der gewünschte Effekt eintreten. Gleichzeitig wurden die Kommandos auf den Tasten SHIFT+F7 und SHIFT+F8 gelöscht, was in Abbildung 10.3 dadurch zu erkennen ist, daß die jeweilige Definitionszeile nun leer ist.

Abb. 10.3: Geänderte Funktionstastenbelegung

Im Unterschied zu den Systemoptionen bleiben die Funktionstastenbelegungen dem Anwender erhalten, auch nach Beendigung der SAS-Sitzung. Die Änderungen werden in der Bibliothek SASUSER im Katalog PROFILE.SC2 als Eintrag DMKEYS.KEYS abgelegt. Startet man eine neue SAS-Sitzung, findet das System diesen Eintrag und verwendet die zuvor definierten Tastenbelegungen. Erst wenn der Katalog verloren geht oder gar das komplette Verzeichnis gelöscht wird, muß die Belegung neu definiert werden.

10.4 Eigene Werkzeuge – Ändern der Tools-Leiste

Mit der Tools-Leiste verhält es sich ähnlich wie mit den Funktionstasten. Sie sind mit Kommandos belegt, der Anwender kann ihre Gestalt verändern und die Veränderungen können über eine Sitzung hinaus aufbewahrt werden. Um diese Tools-Leiste zu verändern, ruft man die Menüfolge *Options* → *Edit tools* auf, woraufhin sich das in Abbildung 10.4 dargestellte Tool Editor-Fenster öffnet.

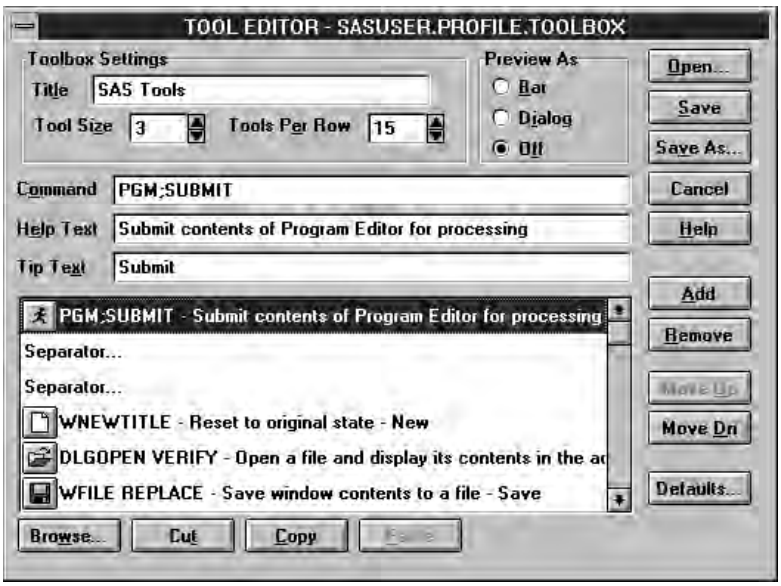


Abb. 10.4: Änderung der Tools-Leiste

Der in der Fensterbeschriftung wiedergegebene Name zeigt an, wo und unter welchem Namen die Informationen zur Tools-Leiste abgelegt sind: in der Bibliothek SASUSER im Katalog PROFILE.SC2 als Eintrag TOOLBOX.TOOLBOX, wobei die zweite „TOOLBOX“, die den Typ des Katalogeintrags ausweist, nicht angezeigt wird.

Der Tool Editor zeigt die Eigenschaften der Schaltflächen auf der Tools-Leiste an. Für die in Abbildung 10.4 aktivierte Submit-Schaltfläche erkennt man beispielsweise:

Command	:	`Pgm;Submit`
Help Text	:	Text für die Hinweiszeile
Tip Text	:	Text für den Tool-Tip

Die Submit-Schaltfläche ist durch zwei Separatoren von der nächsten Schaltfläche abgesetzt.

Um in die Tools-Leiste ein eigenes Werkzeug einzufügen, mit dem z. B. Protokoll- und Ausgabefenster auf einmal gelöscht werden können, klickt man auf *Add* und wählt zunächst über *Browse* ein passendes Bitmap-Symbol aus der angebotenen Liste aus. Anschließend trägt man das Kommando, den Hilfetext und den Tool-Tip ein, wie in Abbildung 10.5 dargestellt. Mittels *Move Dn* bewegt man das

Abb. 10.5: Einfügen eines neuen Tools in die Tools-Leiste

neue Werkzeug an die passende Stelle. Mit *Add* → *Separator* kann man zusätzliche Platzhalter einfügen. Am Ende speichert man die Tools-Leiste mit *Save* ab. Die Tools-Leiste zeigt sich in der veränderten Form (Abbildung 10.6) und sollte nun das geforderte Kommando ausführen und sowohl den Hilfetext als auch den Tool-Tip anzeigen.

Abb. 10.6: Die veränderte Tools-Leiste

Um die Änderungen der Tools-Leiste und der Funktionstastenbelegungen rückgängig zu machen, kann man diese entweder erneut editieren und den Ausgangszustand herstellen, oder man löscht die entsprechenden Einträge aus dem Katalog SASUSER.PROFILE mit der Prozedur CATALOG und der DELETE-Anweisung. Die Option ENTRYTYPE= spezifiziert den jeweiligen Katalogeintragstyp.

```
PROC CATALOG CAT=sasuser.profile;
     DELETE toolbox / ENTRYTYPE=toolbox;
     DELETE dmkeys  / ENTRYTYPE=keys;
RUN;
```

Die beiden Katalogeinträge können auch interaktiv über das Libraries-Fenster gelöscht werden. Man wählt dazu die Bibliothek SASUSER und den Katalog PROFILE aus. In die Selektionsspalte des Katalogfensters trägt man nun vor dem Eintrag TOOLBOX.TOOLBOX bzw. vor DMKEYS.KEYS ein D für *Delete* ein und auf Nachfrage V für *Verify*. (Mit C für *Cancel* könnte man den Löschvorgang auch abbrechen.) Nach dem erneuten Aufruf der SAS-Sitzung ist der ursprüngliche Zustand wieder hergestellt.

10.5 Aufgaben

❶ Ergänzen Sie das horizontale Balkendiagramm in Abbildung 9.5 um eine geeignete Überschrift und eine Fußzeile.

❷ Ersetzen Sie den Titel in Abbildung 9.6.

❸ Verwenden Sie die Systemoptionen LS=, NOCENTER und DATE um eine Liste der Fahrer der Tour de France 1997 anzufertigen. Definieren Sie zusätzlich geeignete Titel und Fußzeilen.

❹ Erweitern Sie Ihre Tools-Leiste um eine Schaltfläche, mit der die Zeilennummern im Programmfenster angezeigt werden.

11

Lust auf mehr? Ein Ausblick

Die Schwerpunkte in diesem einführenden Buch über das SAS-System liegen in der Erstellung von SAS- Dateien und in der Datenanalyse und Präsentation der Ergebnisse. Diese Bereiche wurden in den bisherigen Kapiteln ausführlich behandelt und vorgestellt. Im diesem letzten Kapitel lernen Sie weitere Prozeduren und Module des SAS-Systems kennen, die für Ihre Arbeit hilfreich sein oder bedeutsam werden könnten: die menügesteuerte Benutzerführung über den SAS/ASSIST, den Im- und Export von Daten, die in anderen Systemen abgelegt sind, die Dateneingabe über Masken, die explorative Datenanalyse und die Möglichkeiten der Programmierung. Die einzelnen Themen können nur beispielhaft vorgestellt werden, für eine gründliche Einarbeitung muß auf die Handbücher verwiesen werden, die im Anhang aufgeführt werden.

11.1 SAS/ASSIST – Der Programmierer im Hintergrund

Nicht im typischen Windows-Stil, aber dennoch menügeführt, unterstützt der SAS/ASSIST den Anwender bei der Bedienung des SAS-Systems über Masken, Menüs und Dialoge. Für den reinen Anfänger ist der Assistent ungeeignet, da er sich in der Vielfalt der Begriffe und in den verschlungenen Pfaden leicht verirren kann. Doch wenn ein Grundverständnis vom Aufbau einer SAS-Datei und ihrer Organisation in Bibliotheken verhanden ist und wenn Grundbegriffe wie LIBNAME, Prozedur und Option verstanden wurden, kann der Anfänger das Modul SAS/ASSIST gewinnbringend einsetzen. Auch für fortgeschrittene

Anwender kann der Assistent nützlich sein, da er als „guter Programmierer im Hintergrund“ Protokoll führt und die aufgerufenen Prozeduren mitsamt ihrer Anweisungen und Optionen aufzeichnet. Der Anwender kann diese Programme studieren, abspeichern und zu einem späteren Zeitpunkt auch ohne die Hilfe von SAS/ASSIST ausführen, mit oder ohne zusätzlichen Änderungen oder Anpassungen.

Der Aufruf von SAS/ASSIST erfolgt über die Schaltfläche, das Kommando `assist` oder die Menüfolge *Globals* → *SAS/ASSIST*. Der Assistent präsentiert sich zunächst mit 12 auswählbaren Schaltflächen (Abbildung 11.1). Unter *Tutorial* verbergen sich die sechs SAS-Lernprogramme (vgl. Abschnitt 2.2). Alles, was zur Datenerfassung, -haltung und -verwaltung gehört, ist unter *Data Mgmt* (Data Management) zusammengefaßt. Tabellen und Berichte erstellt man mittels *Report Writing*, Analysen führt man mit *Data Analysis* durch. Am Beispiel einer Grafik soll im folgenden die Vorgehensweise bei der Arbeit mit Unterstützung des Assistenten gezeigt werden.

Abb. 11.1: Die SAS/ASSIST-*Oberfläche*

Wählt man nacheinander *Graphics*, *High resolution*, *Bar Chart*, *Vertical bar charts* und *Stacked* aus, öffnet sich das in Abbildung 11.2 gezeigte Fenster, in dem mehrere Felder auszufüllen sind, bevor das vertikale, unterteilte Balkendiagramm erzeugt wird. Die Felder, die als -*REQUIRED*- gekennzeichnet sind, müssen ausgefüllt werden. Die andere Felder werden vom SAS-System mit Standardeinstellungen belegt.

Die einzigen beiden erforderlichen Angaben sind in diesem Beispiel der Name der die Daten enthaltenden SAS-Datei (*Active data set*), sowie die Zielvariable,

deren Häufigkeiten berechnet und dargestellt werden sollen (*Chart variable*). Die Angabe einer Variablen zur Unterteilung der Balken (*Stacking variable*) ist nicht unbedingt notwendig. Als Ausgabegerät (*Graphics device*) wird der Bildschirm gewählt (*Microsoft Windows Display*), eine Untergruppe wurde nicht ausgewählt und für die aufgetragenen Werte (*Bar values*) wird der Standard (*-Default-*) gewählt.

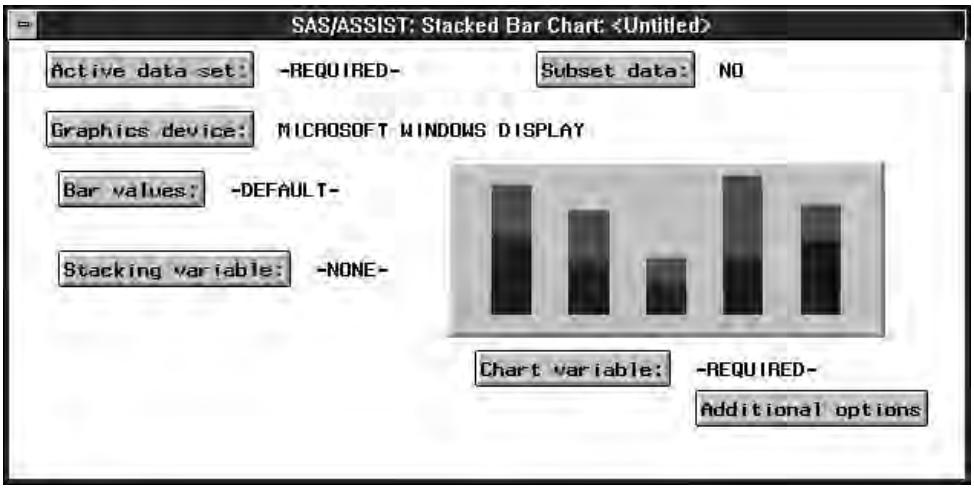


Abb. 11.2: Vertikales Balkendiagramm mit SAS/ASSIST

Zunächst muß nun die Datei ausgewählt werden: Dazu klickt man auf die Schaltfläche *Active data set* und sucht aus allen im Zugriff befindlichen Dateien die gewünschte aus. Eventuell notwendige Bibliotheksreferenzen müssen zu diesem Zeitpunkt bereits gebildet sein. Über *Create* ließen sich an dieser Stelle auch externe Daten einlesen. Als Datei soll `biblio.firma` mit der Zielvariablen `alter` verwendet werden. Die Balken sollen nach `sex` unterteilt werden und anstelle der absoluten die relativen Häufigkeiten aufgetragen werden (*Bar values = Percent*). Damit wurden alle notwendigen Eintragungen vorgenommen (vgl. Abbildung 11.3) und über die Menüfolge *Locals* → *Run* wird die Grafik erzeugt und angezeigt.

Über das *Customize*-Menü können Titel- und Fußzeilen definiert werden, sowie über die Schaltfläche *Additional Options* die Standardeinstellungen der globalen Optionen, Füllmuster, Farben und Achsendarstellungen verändert werden.

Um die Grafik erzeugen zu können, muß der Assistent aus den Angaben des Anwenders ein SAS-Programm zusammenstellen, in dem die grafischen Optionen für den Treiber gesetzt sowie die notwendigen globalen Anweisungen definiert

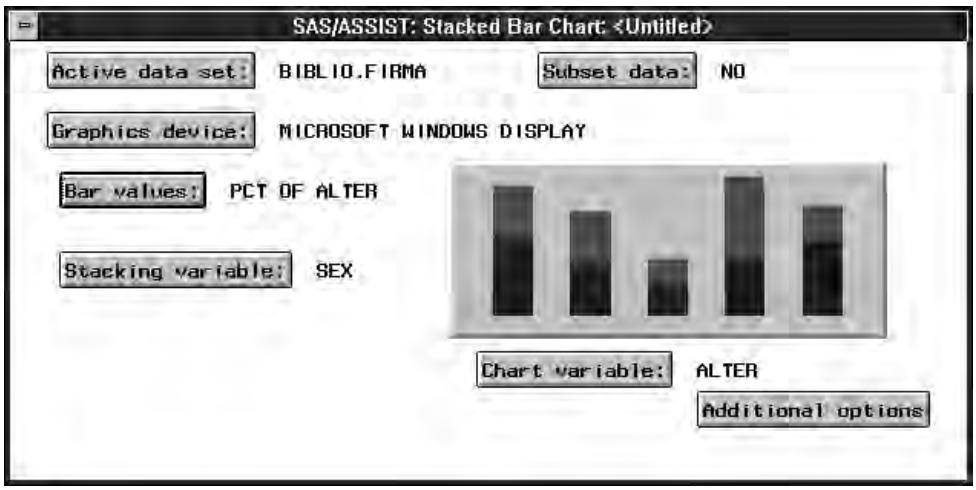


Abb. 11.3: Eingabemaske des SAS/ASSIST

werden und ein PROC GCHART-Schritt ausgeführt wird. Über *Locals* → *View Source* können Sie sich vom Aussehen des Programms überzeugen und das Programm mit *File* → *Save as* → *Write to file* (oder dem Kommando `file`) abspeichern, so daß Sie es zu einem späteren Zeitpunkt ohne Aufruf von SAS/ASSIST im Programmfenster öffnen und ausführen können.

Den Assistenten sollten Sie sich zum Vorbild nehmen, da er zu den wenigen Programmierern gehört, die jeden Programmschritt kommentieren. Dadurch ist es für Sie leicht, die Bedeutung der einzelnen Teile zu verstehen und gezielte Änderungen vornehmen zu können. Das Programm beginnt mit einer kurzen Übersicht, die die Aufgabe des nachfolgenden Programms beschreibt.

```
/*------------------------------------------------------------*
 | Summary:                                                   |
 |    Creating a stacked bar chart using the data set         |
 |    BIBLIO.FIRMA and charting the variable                  |
 |    ALTER along the horizontal axis.                        |
 | Generated: 05JUL97 10:27:15                                |
 *------------------------------------------------------------*/
```

Anschließend folgen in diesem konkreten Beispiel die globalen Anweisungen. In den beiden GOPTIONS-Anweisungen werden die grafischen Einstellungen festgelegt.

```
/*-------------------------------------------------------------*
 | The GOPTIONS statement allows you to have more control over |
 | the final appearance of your output such as fonts, colors,  |
 | text height and so on.  The output device and destination is |
 | also specified in the goptions statement.                   |
 *-------------------------------------------------------------*/

goptions reset=(axis, legend, pattern, symbol, title, footnote)
         norotate hpos=0 vpos=0 htext= ftext= ctext=
         target= gaccess= gsfmode= ;
goptions device=WIN  ctext=blue
         graphrc interpol=join;
```

Die folgenden Anweisungen PATTERN und AXIS legen die Füllmuster für die Balken sowie drei Achsendefinitionen fest, die im anschließenden Prozeduraufruf verwendet werden.

```
/*-------------------------------------------------------------*
 | PATTERN statements allow you to define colors and patterns in |
 | the chart, map or plot that you are creating.  SAS/GRAPH uses |
 | any pattern statements that you specify.  If more are needed, |
 | default PATTERN statements are used.                        |
 *-------------------------------------------------------------*/
pattern1 value=SOLID;
/*-------------------------------------------------------------*
 | AXIS statements allow you to supply information on how your |
 | vertical and horizontal axes will appear on the graph.      |
 *-------------------------------------------------------------*/
axis1
   color=blue
   width=2.0
   ;
axis2
   color=blue
   width=2.0
   ;
axis3
   color=blue
   width=2.0
   ;
```

Schließlich erfolgt der Aufruf der Prozedur GCHART mit einer VBAR-Anweisung und den beiden Optionen SUBGROUP= zur Unterteilung der Balken nach dem Geschlecht und TYPE= für die Darstellung der relativen Häufigkeiten.

```
/*-------------------------------------------------------------*
 | This section produces the actual bar chart and contains the  |
 | options that directly relate to the data and the axis area.  |
 *-------------------------------------------------------------*/
proc gchart data=BIBLIO.FIRMA;
      vbar ALTER /
      maxis=axis1
      raxis=axis2
      subgroup=SEX
      type=PCT
      ;
run; quit;
```

Die Zuweisung der Achsendefinitionen erfolgt über die Optionen MAXIS= (Midpoint-Achse) und RAXIS= (Response-Achse).

Im Unterschied zur üblichen Syntax verwendet der Assistent kleine Buchstaben für feste SAS-Bezeichnungen und Optionen, während Benutzer-eigene Datei- und Variablennamen stets in Großbuchstaben angegeben werden.

Um eine Prozedur mit Hilfe des SAS-Assistenten kennenzulernen, können Sie die verschiedenen Varianten durchspielen und anschließend anhand der Programme die Bedeutung einzelner Anweisungen und Optionen ergründen. Der Assistent erspart Ihnen zwar nicht, bei besonderen Anforderungen in das Handbuch oder die Online-Hilfe zu schauen, aber er garantiert, daß alle von ihm erstellten Programme fehlerfrei laufen. (Von eigenen Programmen kann man das leider nicht immer behaupten.) Aber „Nobody is perfect“, auch der SAS-Assistent nicht. Das Modul SAS/ASSIST deckt leider nur ca. 70-80% der Prozeduren, Anweisungen und Optionen des SAS-Systems ab und gerade die Neuerungen brauchen ihre Zeit, bis sie integriert werden, so daß man nicht immer ums Programmieren herum kommt.

11.2 Keine Angst vor Fremdformaten wie dBASE, Excel und SPSS

Das SAS-System kennt kaum Grenzen, wenn es darum geht, auf Dateien zuzugreifen, die von anderen Anwendungsprogrammen im eigenen Format abgelegt sind. Reine Text- oder ASCII-Dateien werden im Datenschritt mit der Anweisung INFILE (vgl. Kapitel 5) und eventuell erforderlichen Einleseformaten in eine SAS-Datei überführt, bevor Analysen durchgeführt oder Grafiken erstellt werden können.

Daten, die mit einem anderen Anwendungsprogramm erfaßt und dort in einem spezifischen Format abgespeichert wurden, überträgt man ungern in eine Text-

datei, um sie anschließend ins SAS-System zu integrieren, da bei diesem Vorgehen in der Regel Informationen über die Struktur und die Variablentypen verloren gehen. Stattdessen wünscht man sich als Anwender einen direkten Weg zur Übertragung einer Excel-, dBASE- oder SPSS-Datei in eine SAS-Datei.

Wie so oft bei der Arbeit mit dem SAS-System, gibt es auch beim Datenimport nicht *den* goldenen Weg, sondern für jedes Fremdfomat eine individuelle Lösung, für die gewisse Voraussetzungen erfüllt sein müssen. In Tabelle 11.1 wurden für einige Fremdformate die Methoden und das jeweils erforderliche SAS-Modul zusammengestellt. Der Import Wizard, die Prozedur DBF, der dynamische Datenaustausch (DDE, engl. *dynamic data exchange*) und die Engine SPSS werden im folgenden anhand von Beispielen erläutert. Für eingehendere Studien wird allerdings auf die betreffenden SAS-Handbücher ([5] und [6]) verwiesen, sowie auf das über das World Wide Web verfügbare Skript von Geißler und Ortseifen [2] aus dem Rechenzentrum der Universität Heidelberg. Im letzten Abschnitt wird noch gezeigt, wie Sie auf SAS-Dateien zugreifen können, die mit der SAS-Software unter einem anderen Betriebssystem angelegt wurden.

Anwendung	Methode	SAS-Modul
dBASE	Import Wizard	SAS/ACCESS to PC File Formats
	Prozedur DBF	SAS/ACCESS to PC File Formats
Excel	Import Wizard	SAS/ACCESS to PC File Formats
	DDE	SAS/BASE
MS/Access	ODBC	SAS/ODBC
SPSS	Engine SPSS	SAS/BASE
	Prozedur CONVERT	SAS/BASE
BMDP	Engine BMDP	SAS/BASE
Oracle	Prozedur SQL	SAS/ORACLE
SAS[1]	Prozedur CIMPORT	SAS/BASE

[1] SAS-Dateien, die unter einem anderen Betriebssystem erstellt wurden.

Tab. 11.1: Import von Fremdformaten ins SAS-*System*

11.2.1 Der Import Wizard

Sofern man das Modul SAS/ACCESS lizensiert und installiert hat, kann man interaktiv über das Menü *File* → *Import* den Import Wizard aktivieren und einige der gängigsten Datenformate importieren: dBASE, Excel sowie durch Komma, Tabulator oder andere Zeichen separierte ASCII-Dateien. Nach dem Aufruf des Import Wizards wird der Anwender durch zahlreiche Menüs geführt.

Abb. 11.4: Der Import Wizard

Nach der Festlegung des Fremdformattyps (aus Abbildung 11.4 geht hervor, welche unterstützt werden) muß der Anwender den Namen der zu importierenden Datei angeben oder anhand des Dateibaums auswählen (*Browse*) und anschließend die Bibliothek und den Namen bestimmen, unter dem die SAS-Datei abgelegt werden soll. Mit *Finish* wird der Importvorgang dann in Gang gesetzt und im Protokollfenster eine erfolgreiche Vollzugs- oder eine Fehlermeldung ausgegeben.

```
NOTE: BIBLIO.adressen was successfully created.
```

Analog zum Importvorgang funktioniert auch das Exportieren: Man legt den Formattyp fest, nennt die zu exportierende SAS-Datei und definiert den Namen der neuen Datei.

11.2.2 Zugriff auf dBASE-Dateien mit der Prozedur DBF

Datenbanken, die mit dem Anwendungsprogramm dBASE erstellt wurden, können mit dem Import Wizard wie auch mit der Prozedur DBF importiert bzw. SAS-Dateien in dBASE-Datenbanken exportiert werden. Auch diese Prozedur ist Bestandteil des Moduls SAS/ACCESS und daher nur einsetzbar, wenn dieses Modul lizensiert und installiert wurde. Der Aufruf der Prozedur ist bestechend einfach:

```
/*--- KA11-01.SAS ---*/
/* Import */
FILENAME ref1 'A:\kurs\adressen.dbf';
PROC DBF OUT=adressen DB3=ref1;
RUN;

/* Export */
FILENAME ref2 'C:\kurs\firma.dbf';
PROC DBF DATA=biblio.firma DB4=ref2;
RUN;
```

Die bestehende oder zu erzeugende dBASE-Datei wird mit der FILENAME-Anweisung definiert und mit einem logischen Referenznamen verknüpft. Dieser Referenzname wird in der PROC DBF-Anweisung mit den Optionen DB2=, DB3= bzw. DB4= verbunden, je nachdem, ob es sich um dBASE II, III oder IV handelt.

Soll eine neue SAS-Datei aus einer bestehenden dBASE-Datei erstellt werden, wird noch die Option OUT= gesetzt. Wird dagegen eine dBASE-Datei aus einer bestehenden SAS-Datei erstellt, setzt man die Option DATA=.

In obigem Beispiel wird im ersten Prozedurschritt aus der bestehenden dBASE III Datei `adressen.dbf` eine temporäre SAS-Datei `adressen` erzeugt. Im zweiten Prozedurschritt wird die Firmendatei in eine dBASE IV Datei `firma.dbf` überführt.

Probleme treten bei der Prozedur DBF auf, wenn in einer dBASE-Datei sogenannte Memofelder vorliegen, da diese vom SAS-System nicht unterstützt werden. Variablennamen, die bei dBASE länger als 8 Zeichen sein können, werden bei der Übertragung radikal auf 8 Zeichen gekürzt, wodurch es zu Problemen kommen kann, wenn zwei Variablen in den ersten 8 Zeichen übereinstimmen. Vorsichtig sollte man auch sein, wenn bei dBASE-Dateien Felder vollständig mit den Ziffern 9 besetzt sind, denn diese werden vom SAS-System als fehlende Werte interpretiert und als „.“abgelegt. Hier hilft nur, zuvor in dBASE das Feld um eine Spalte zu erweitern.

Über das WWW bieten erfahrene SAS-Programmierer Makros (vgl. Abschnitt 11.5) an, mit denen beispielsweise ohne auf das Modul SAS/ACCESS zurückzugreifen in dBASE eine ASCII-Datei und ein SAS-Programm erstellt wird, das innerhalb des SAS-Systems ausgeführt wird, damit die ASCII-Daten in eine SAS-Datei überführt werden können. Andere Makros können direkt auf die dBASE-Datei zugreifen. Ein guter Einstiegspunkt hierfür ist die WWW-Seite von Donald Peter Cram:
`http://www-leland.stanford.edu/class/gsb/excel2sas.html`

11.2.3 Dynamischer Datenaustausch – Zugriff auf Excel-Tabellen

Neben dem Import Wizard und dem Umweg über eine dBASE-Datei kann der Zugriff auf Excel-Tabellen auch über den dynamischen Datenaustausch erfolgen. Man startet dazu zunächst sowohl eine SAS- als auch eine Excel-Sitzung und öffnet die zu importierende Excel-Tabelle `gruppe.xls` (Abbildung 11.5).

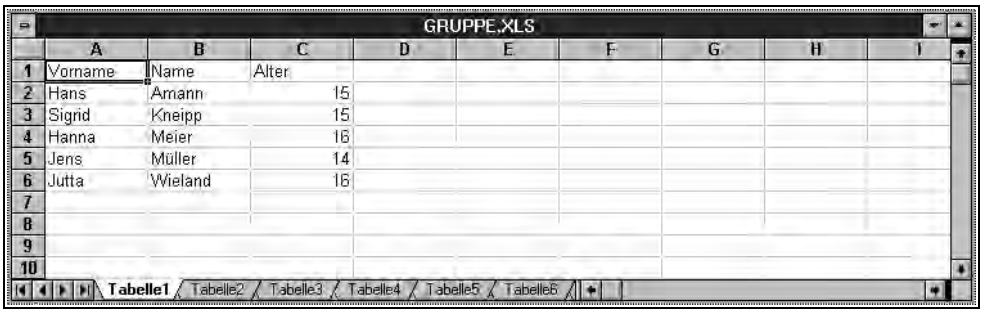


Abb. 11.5: Excel-Tabelle

Über eine spezielle FILENAME-Anweisung wird diese Excel-Tabelle nun im SAS-System referenziert und in einem anschließenden Datenschritt mittels INFILE und INPUT in eine SAS-Datei überführt.

```
/*--- KA11-02.SAS ---*/
FILENAME in DDE 'excel|[gruppe.xls]tabelle1!z2s1:z6s3';
DATA s_gruppe;
     INFILE in;
     INPUT vname $ nname $ alter;
RUN;
```

Der Name der Anwendung, `excel`, wird durch einen senkrechten Strich von dem Namen der Excel-Tabelle, `[gruppe.xls]tabelle1`, getrennt. Anschließend, nach einem Ausrufezeichen, wird der Spalten- und Zeilenbereich der Daten in der Excel-Tabelle beschrieben (`z2s1:z6s3`). Mit `tabelle2` könnte die 2. Tabelle aus der Mappe `gruppe` importiert werden. Würde die Excel-Tabelle in einer der Standardmappen `mappe1`, `mappe2` usw. stehen, könnte man auf die Nennung der Mappe in eckigen Klammern verzichten.

Beim Betrachten des anschließend erforderlichen Datenschritts fällt sofort der Nachteil dieser Methode auf: Die Variablen müssen in der SAS-Umgebung neu definiert werden. Außerdem muß man durch Optionen der Anweisung INFILE (z. B. DSD, NOTAB) sicherstellen, daß leere Excel-Tabellenfelder als fehlende

Werte erkannt werden und daß Datums- und Zeitformate (durch spezielle Picture-Formate) korrekt übernommen werden.

11.2.4 Die SPSS-Engine

Mit der SPSS-Engine des SAS-Systems kann man auf SPSS-Transportdateien, die meist die Erweiterung POR tragen, zugreifen. Dazu führt man eine LIBNAME-Anweisung aus, die erstens auf die Transportdatei zeigt, nicht auf die Bibliothek, in der sie abgelegt ist, und zweitens nach dem Referenznamen die Engine SPSS enthält, das Verzeichnis.

Unter SPSS 6.1 für Windows wurde mit *File* → *Save As* die Transportdatei `gruppe.por` im Verzeichnis `C:\kurs` abgelegt. Um die Datenwerte dieser Datei mit der SAS-Software im Ausgabefenster anzuzeigen, wird folgendes Programm ausgeführt:

```
/*--- KA11-03.SAS ---*/
LIBNAME in SPSS 'A:\kurs\gruppe.por';
PROC PRINT DATA=in.spssdat;
RUN;
```

Mit der LIBNAME-Anweisung und der Engine SPSS wird die Transportdatei mit dem Referenznamen `in` verknüpft. Dieser Referenzname und der Dateiname `datei` werden im anschließenden Prozedur PRINT-Schritt verwendet. Es spielt aber keine Rolle, welchen Dateinamen Sie an dieser Stelle angeben. Hauptsache ist, daß überhaupt ein Name da steht. Wenn die SPSS-Datei weiterverarbeitet werden soll, muß sie in eine SAS-Datei transferiert werden. Dazu führt man am einfachsten einen Datenschritt aus und greift mit der SET-Anweisung auf die SPSS-Datei `in.spssdat` zu:

```
LIBNAME in SPSS 'C:\kurs\gruppe.por';
DATA biblio.sasdat;
     SET in.spssdat;
RUN;
```

11.2.5 Die Prozeduren CPORT und CIMPORT

SAS-Dateien, die unter einem UNIX-System erstellt wurden, können nicht ohne weiteres unter SAS für Windows eingelesen und weiterverarbeitet werden, da die SAS-Software Betriebssystem-spezifische Eigenschaften bei ihrer Dateiverwaltung ausnutzt. Der Anwender, der auf Dateien zugreifen möchte, die unter einem anderen Betriebssystem angelegt wurden, muß diese daher zunächst

in einen Transportfile verwandeln, exportieren, diesen übertragen und importieren.

Der Vorgang wird am Beispiel von SAS für UNIX[1] und SAS für Windows erläutert. Unter UNIX existiert die SAS-Datei `firma` im aktuellen Verzeichnis. In einem ersten Schritt wird diese unter SAS für UNIX in den Transportfile `trans.fer` übertragen.

```
/*   UNIX-Umgebung   */
FILENAME trans './trans.fer';
LIBNAME biblio './';
PROC CPORT DATA=biblio.firma FILE=trans;
RUN;
```

Die FILENAME-Anweisung referenziert den Transportfile `trans.fer`, die LIBNAME-Anweisung die Bibliothek. Die Prozedur CPORT erzeugt den mit der Option FILE= angewiesenen Transportfile aus der mit DATA= benannten Datei.

Der Transportfile `trans.fer` wird anschließend mittels Filetransfer (FTP) binär zum PC ins Verzeichnis `A:\kurs` übertragen und folgendes Programm ausgeführt:

```
/*--- KA11-04.SAS ---*/
/* Windows-Umgebung  */
FILENAME trans 'A:\kurs\trans.fer';
LIBNAME biblio 'C:\kurs';
PROC CIMPORT DATA=biblio.firma INFILE=trans;
RUN;
```

Die beiden Anweisungen FILENAME und LIBNAME weisen hier wieder auf die Transportdatei und die Bibliothek. Anstelle von CPORT wird die Prozedur CIMPORT eingesetzt mit der Option INFILE=. Nachdem das Programm ausgeführt wurde, liegt aufgrund der DATA=-Option im Verzeichnis `C:\kurs` die SAS-Datei `firma` vor.

[1] Es spielt keine Rolle, ob es sich dabei um AIX, HP oder eine andere UNIX-Plattform handelt.

11.3 Versteckte Strukturen – Maskengesteuerte Eingabe mit FSEDIT

Liegen die Daten noch nicht in Form einer SAS-Datei vor und wurden sie auch noch nicht mit einem anderen Anwendungsprogramm erfaßt oder in eine Textdatei eingetragen, dann bleibt mit den bisher vorgestellten Verfahren nur die Möglichkeit, die Datenwerte in einem Datenschritt nach der Anweisung LINES (oder CARDS) einzutragen. Das ist für einige wenige Beobachtungen und ein paar Variablen die gängige Praxis, für größere Datenmengen allerdings unpraktisch und fehleranfällig. Hier wünscht man sich Eingabemasken, die auf die Vorlage abgestimmt sind, auf der die Daten von Hand eingetragen werden, und die gewisse Fehleingaben von vornherein durch Fehlerprüfungen ausschließen. Innerhalb der SAS-Umgebung können Sie solche Eingabemasken mit der Prozedur FSEDIT aus dem Modul SAS/FSP entwickeln und einsetzen. Mit folgendem Prozedurschritt wird die bestehende Datei `biblio.firma` aufgerufen und die erste Beobachtung am Bildschirm mit einer Standardmaske angezeigt (Abbildung 11.6).

```
/*--- KA11-05.SAS ---*/
PROC FSEDIT DATA=biblio.firma;
RUN;
```

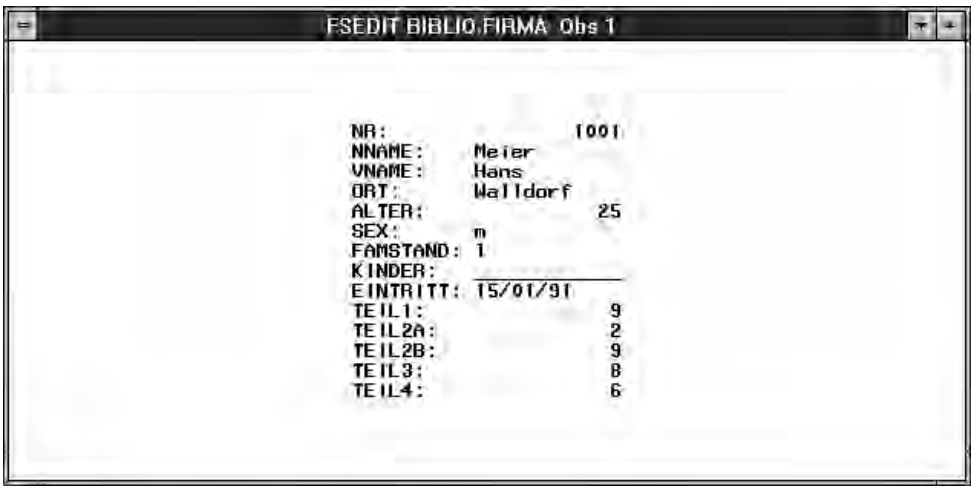


Abb. 11.6: Die FSEDIT*-Standardeingabemaske*

Jede Beobachtung wird auf einer eigenen Seite angezeigt, mit BILD ↑ und BILD ↓ blättert man zu den nächsten Beobachtungen der Datei weiter. Die Variablen erscheinen in der Reihenfolge, in der sie in die Datei eingetragen wurden, in Spalten angeordnet. Rechts daneben stehen die zur jeweiligen Beobachtung gehörenden Datenwerte: numerische Variablen rechts-, Textvariablen linksbündig ausgerichtet. Wenn mehr Variablen in der Datei enthalten sind, so daß der Platz in einer Spalte nicht ausreicht, werden weitere Spalten angelegt. Reicht der gesamte Bildschirm nicht für alle Variablen aus, werden weitere Seiten (*Screens*) aufgebaut, die man über das *View*-Menü erreichen kann.

Das *Search*-Menü unterstützt den Anwender bei der Suche nach bestimmten Beobachtungen: Mit *Where ...* können Untergruppen gebildet werden (Abbildung 11.7). *Where also ...* erlaubt zusätzliche Einschränkungsbedingungen, die mit der ersten Einschränkung über AND verknüpft werden. *Undo last where* hebt die Einschränkungen sukzessive wieder auf. Dagegen wird bei *Find*, *Locate* und *Search* jeweils nur die erste Beobachtung angezeigt, die die vorgegebene Bedingung erfüllt. Blättert man weiter, erscheint die nächste Beobachtung in der Datei ohne Berücksichtigung des Suchkriteriums. Einzelne Eintragungen lassen sich auf diesem Wege einfach ändern, indem die betreffende Beobachtung aufgesucht wird, eventuell der Überschreibemodus ein- bzw. der Einfügemodus ausgeschaltet und die Eintragung korrigiert wird.

Abb. 11.7: Einschränkung mit Where ...

Es können aber auch weitere Beobachtungen in die Datei aufgenommen werden, indem die Menüfolge *Edit* → *Add new record* aktiviert wird (oder das Kommando add ausgeführt bzw. F5 gedrückt wird). Die Prozedur FSEDIT fügt eine leere Beobachtung ans Ende der Datei an und der Anwender trägt die Werte ein. Mit F3 wird der Vorgang beendet und die Beobachtung abgespeichert. Mit *File* → *Cancel* kann der Erweiterungsvorgang abgebrochen werden.

Die Standardeingabemaske, die von der Prozedur FSEDIT erstellt wird, ist nicht besonders ansprechend und die Variablen erscheinen selten in der optimalen Reihenfolge. Als Anwender haben Sie die Möglichkeit, sich eine eigene Einga-

bemaske zu definieren. Rufen Sie die Prozedur FSEDIT erneut auf und übergeben Sie als weitere Option den Namen der Bildschirmmaske mit der Option SCREEN=.

```
PROC FSEDIT DATA=biblio.firma SCREEN=biblio.firma.schirm;
RUN;
```

Es erscheint wieder die in Abbildung 11.6 gezeigte Eingabemaske, die nun über *Locals* → *Modify Screen* verändert wird[2]. Die Eingabemaske wird permanent in der Bibliothek `biblio` im Katalog `firma` unter dem Namen `schirm.screen` abgelegt. Im Anschluß an die Frage nach dem Kennwort für die Eingabemaske erscheint das FSEDIT MENU mit den folgenden sechs Unterpunkten:

❶ Information about screen modification

❷ Screen Modification and Field Identification

❸ Edit Program Statements and Compile

❹ Assign Special Attributs to Fields

❺ Modification of General Attributes

❻ Browse Program Statements

Der Menüpunkt ❷ ist der zunächst entscheidende. Wählt man ihn aus, öffnet sich die Eingabemaske im Änderungsmodus (*Modify*). Nun kann die Eingabemaske frei gestaltet werden: Zusätzlicher Text läßt sich eingeben, die Variablen an andere Stellen plazieren, die Variablennamen durch Kleinbuchstaben ersetzen oder andere Bezeichnungen wählen. Die Felder für die Datenwerte müssen durch Unterstriche gekennzeichnet werden, wobei dem ersten Unterstrich mindestens ein Leerzeichen vorausgehen muß (Abbildung 11.8).

Läßt man in dem Fenster die Zeilennummern anzeigen (`nums on`), kann man die Zeilenkommandos (vgl. Abschnitt 4.2) verwenden. Mit dem Kommando `autosplit` kann man mit ENTER die Zeile an der Stelle des Cursors umbrechen bzw. mit ENTF Zeilen verbinden.

Wurden die Änderungen vorgenommen, verläßt man den Bildschirm mit F3 oder *File* → *End*. Die Frage nach neuen Variablen verneint man. In der Statuszeile erscheint nun für alle betroffenen Variablen die in Abbildung 11.9 angezeigte Meldung. Jene Variablen, deren Felder in der Maske verschoben wurden, müssen neu identifiziert werden. Dazu bewegt man den Cursor zum ersten Unterstrich

[2] Auch die Standardeingabemaske kann verändert werden, allerdings gehen die Änderungen mit Beendigung der SAS-Sitzung verloren, da es sich um eine temporäre Maske handelt.

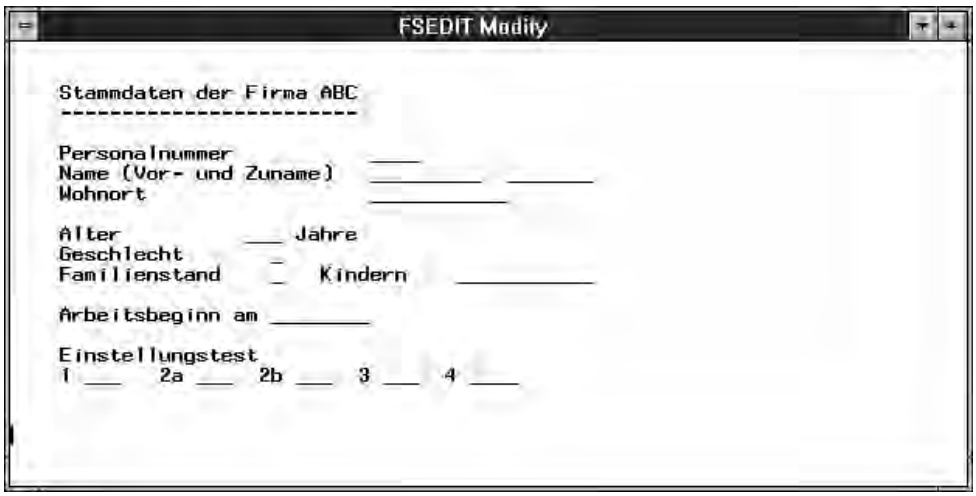


Abb. 11.8: Modifizierte Eingabemaske

des betreffenden Feldes und drückt ENTER. Wird eine Variable nicht mehr in der Maske angezeigt, wählt man *Locals* → *Unwanted*. Sind alle Felder identifiziert, kann man mit F3 wieder ins FSEDIT MENU zurückkehren.

Abb. 11.9: Identifikation der Felder

Mit dem Menüpunkt ❹, Assign Special Attributes to Fields, öffnet man das erste Attributfenster (*FSEDIT Attribute: INITIAL*), in dem Initialwerte definiert werden können, die den Variablen neuer Beobachtungen standardmäßig zugewiesen werden. Über *View* → *Go to attribute* gelangt man zu weiteren Attributfenstern, wie beispielsweise *Minimum* und *Maximum*, in denen untere und obere Grenzen für die Datenwerte vereinbart werden können, und *Required*, womit festgelegt wird, welche Felder für jede Beobachtung ausgefüllt werden müssen. Darüber hinaus gibt es weitere Eigenschaften, mit denen der Anwender bereits einen Teil der Fehleingaben abfangen kann.

Richtige Plausibilitätsprüfungen erfordern jedoch, daß mehrere Variablenwerte gleichzeitig überprüft werden. Wenn der Mitarbeiter etwa in allen fünf Abschnitten des Einstellungstests mit weniger als 5 Punkten abgeschnitten hat, dann sollte die Eingabe nochmals überprüft werden. Solche Aufgaben lassen sich nicht mit den speziellen Feldattributen bewältigen. Hier bietet die Prozedur FSEDIT die Möglichkeit, eine spezielle Maskenabfragesprache, die *Screen*

Control Language (SCL) einzusetzen. Die dazu notwendigen Zeilen trägt man unter ❸ in das FSEDIT Programmfenster (*FSEDIT Program*) ein:

```
/*--- KA11-06.SCL ---*/
INIT:
RETURN;

MAIN:
* Überprüfung der Variablen teil1 - teil4;
IF (teil1 LT 5 AND teil2a LT 5 AND teil2b LT 5 AND
    teil3 LT 5 AND teil4 LT 5) THEN DO;
        ERRORON teil1;     ERRORON teil2a;
        ERRORON teil2b;    ERRORON teil3;
        ERRORON teil4;
        _MSG_ = "Überprüfen Sie die Testergebnisse";
        ALARM;
        teil1  = .;        teil2a = .;
        teil2b = .;        teil1  = .;
        teil3  = .;
    GOTO INIT;
END;
RETURN;

TERM:
RETURN;
```

Das Programm besteht aus drei Abschnitten. Der mittlere Abschnitt, `MAIN`, die Überprüfung der Eingabewerte, wird jedesmal ausgeführt, wenn eine neue Beobachtung eingetragen wurde. Der erste Abschnitt, `INIT`, wird einmal, beim Aufruf der Eingabemaske, ausgeführt und der dritte Abschnitt, `TERM`, nur beim Verlassen am Ende.

Verläßt man das FSEDIT-Programmfenster mit F3, wird das SCL-Programm abgespeichert und kompiliert. Eventuelle Fehler werden im Protokollfenster ausgegeben. Sobald es fehlerfrei kompiliert werden konnte, können Sie es testen, indem Sie das FSEDIT-Menü verlassen, F5 drücken und fünf Werte, die kleiner als 5 sind, als Testergebnisse eintragen. Die Fehleingabe wird durch ein akustisches Signal (`ALARM;`) und folgenden Hinweis in der Statuszeile angezeigt: (`_MSG_='Überprüfen Sie..`). Gleichzeitig werden die fünf Werte gelöscht (`teil1=.;`, `teil2=.;` usw.).

Mit dem vorletzten Unterpunkt, ❺ Modification of General Attributes, werden das allgemeine Aussehen und die generellen Eigenschaften des Bildschirms der Eingabemaske gesteuert: Text- und Hintergrundfarben, ob Einträge eingefügt

Abb. 11.10: Variablendeklaration bei FSEDIT

und gelöscht werden dürfen, Name des Katalogeintrags mit den Funktionstastenbelegungen usw.

Mit der Prozedur FSEDIT können nicht nur bestehende Dateien verändert und erweitert, sondern auch neue Dateien angelegt werden. Dazu wird die Prozedur mit der Option NEW= anstelle von DATA= aufgerufen.

```
PROC FSEDIT NEW=biblio.verkauf;
RUN;
```

Bevor die Datenwerte eintragen werden können, muß, analog zum Datenschritt, die Dateistruktur in dem *FSEDIT New*-Fenster festgelegt werden: Variablennamen, Typ, Länge der Felder, eventuelle Etiketten und Ein- und Ausgabeformate, wie in Abbildung 11.10 dargestellt. Mit *Locals* → *Format/Informat* wechselt man die rechte Spalte gegen die Einleseformate aus. Verläßt man dieses Fenster, wird die Datei erstellt und mit *Locals* → *Add new Record* können die Beobachtungen eingetragen werden.

Wenn bei der Deklaration allerdings eine Variable vergessen wurde, dann kann man dieses Fenster leider nicht mehr öffnen, sondern muß wie in Kapitel 7 gezeigt, einen Datenschritt ausführen, um neue Variablen in bestehende Dateien aufzunehmen. Diese neuen Variablen werden dann mit *Locals* → *Modify* und *Wanted* in eine permanente Eingabemaske aufgenommen.

11.4 „Schau'n wir mal"– Explorative Datenanalyse mit SAS/INSIGHT

SAS/INSIGHT ist ein aus einer einzigen Prozedur bestehendes Modul, das dem Statistiker unter den Anwendern ein Werkzeug in die Hand gibt, mit dem er seine Analysen interaktiv ausführen kann. Neben den bereits in den Kapiteln 8 und 9 behandelten deskriptiven und inferenzstatistischen Verfahren öffnen sich dem Anwender neue Dimensionen bzw. die Möglichkeit, verschiedene Grafiken oder Statistiken gleichzeitig zu betrachten und interaktiv einzelne Beobachtungen oder Teilmengen ein- und auszuschließen.

Der Aufruf erfolgt mit dem Kommando `INSIGHT`, der Menüfolge *Globals* → *Analyze* → *Interactive Data Analysis* oder einem Prozedurschritt:

```
PROC INSIGHT;
RUN;
```

Nach dem Aufruf wählt man entweder eine bestehende SAS-Datei aus einer der vorhandenen Bibliotheken aus oder legt über *New* eine neue Datei an. Die Eingaben können im Gegensatz zu FSEDIT nicht direkt überprüft werden, auch können keine Masken eingerichtet werden, doch das Datenfenster erinnert sicherlich manchen Anwender an andere Tabellenkalkulationsprogramme.

BIBLIO.FIRMA

	NR	NNAME	VNAME	ORT	ALTER	SEX	FAMSTAND
1	1001	Meier	Hans	Walldorf	25	m	l
2	1002	Schulz	Karin	Mannheim	27	w	h
3	1003	Frisch	Melanie	Heidelberg	41	w	h
4	1004	Neuer	Manfred	Heidelberg	58	m	g
5	1005	Neuer	Annerose	Heidelberg	53	w	l
6	1006	Karl	Helmut	Mannheim	51	m	g
7	1007	Manz	Eva	Walldorf	24	w	l
8	1008	Ganz	Ulrich	Mannheim	38	m	h
9	1009	Weber	Joachim	Karlsruhe	42	m	h
10	1010	Ulrich	Matthias	Weinheim	29	m	l
11	1011	Neuer	Heidi	Heidelberg	35	w	l
12	1012	Maier	Horst	Mannheim	51	m	w
13	1013	Schuber	Sabine	Weinheim	29	w	h
14	1014	Sauer	Maria	Heidelberg	48	w	h

Abb. 11.11: Datenfenster in SAS/INSIGHT

Die Daten werden in einem Datenfenster angezeigt (Abbildung 11.11). Über das *Edit*-Menü können Beobachtungen aus Darstellungen oder Berechnungen ein- und ausgeschlossen und bestehende Variablen transformiert werden. *Analyze* öffnet dem Anwender die ganze Palette der verfügbaren statistischen Verfahren: rein deskriptive Methoden wie Histogramme, Box- und Linienplots, Streudiagramme und Rotierende 3-D Plots, sowie Verteilungsuntersuchungen, Anpassungen von linearen Modellen und Hauptkomponentenanalysen. Diese Verfahren sind zum größten Teil auch Bestandteil des Moduls SAS/STAT, allerdings können sie dort nur in sequentieller Abfolge, eins nach dem anderen, ausgeführt werden.

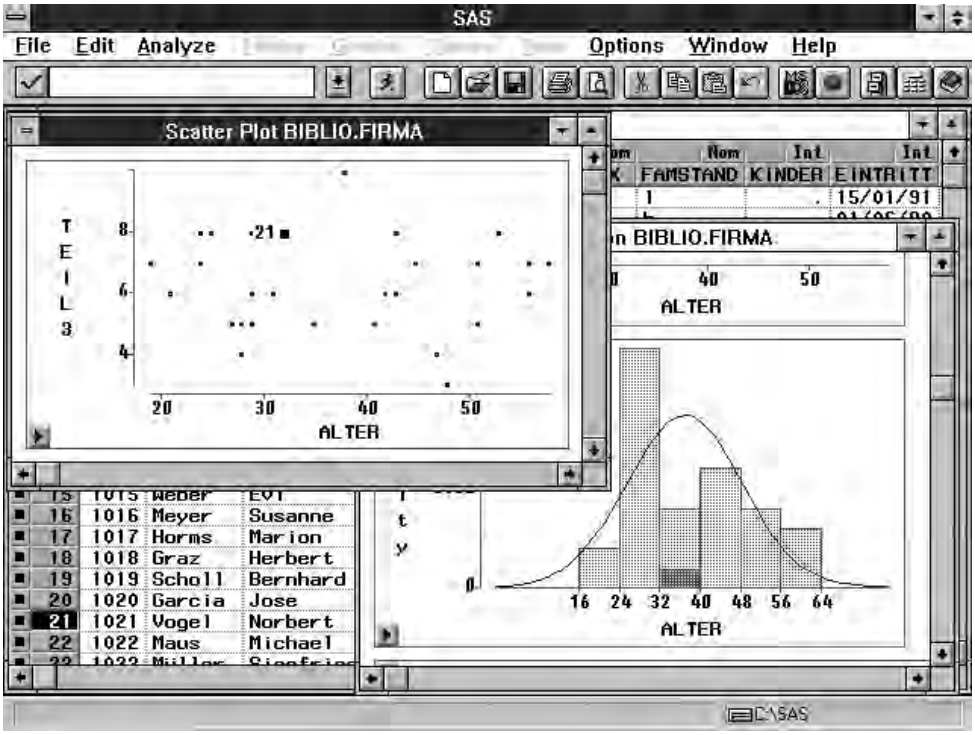


Abb. 11.12: Explorative Analyse mit SAS/INSIGHT

Explorative Datenanalyse bedeutet dagegen, daß verschiedene Darstellungen gleichzeitig angefordert und erstellt werden können. So kann etwa ein Histogramm, das von einer Verteilungsfunktion überlagert wird, neben einem Streudiagramm und dem Datenfenster aufgebaut werden. Klickt man dann im Streudiagramm einen Punkt, d. h. eine Beobachtung an, wird diese Beobachtung im Datenfenster markiert und im Histogramm sieht man die Gruppe, zu der die Beobachtung gehört (Abbildung 11.12). Die Auswahl der Methoden, Va-

riablen und Optionen erfolgt durch Eintragung in entsprechende Fenster oder durch Markierung der Variablen im Datenfenster.

11.5 Selbst programmieren – Makros, IML und AF

Während ein Teil der SAS-Anwender mit dem Assistenten, dessen Menüführung und den unterstützten Verfahren zufrieden ist, reichen einem in der Forschung tätigen Statistiker oder einem für Routineauswertungen zuständigem Mitarbeiter die bisherigen Prozeduren und Methoden nicht aus. Arbeitsabläufe müssen vereinfacht, neue Algorithmen integriert, eigene Oberflächen für den Endanwender der neuen Verfahren entworfen werden. Das SAS-System unterstützt diese vielfältigen Anforderungen durch Makros, die Prozedur IML und die beiden Module SAS/AF und SAS/EIS.

SAS-Makros bestehen aus einer Reihe von Daten- und/oder Prozedurschritten, deren Ablauf durch Parameter, lokale und globale Makrovariablen, sowie verschiedene Steuerungselemente geregelt werden. Um beispielsweise einen Überblick über eine neue Datei zu erhalten, kann man zunächst mit der Prozedur CONTENTS die Variablennamen auflisten, dann mit der Prozedur MEANS die wichtigsten Kennwerte berechnen und mit GCHART Häufigkeitsverteilungen grafisch darstellen. Drei Prozedurschritte, die man gerne mit einem einzigen Aufruf ausführen möchte. Das einzige, was dabei anzugeben ist, ist der Dateiname. Dies ist eine typische Anwendung für ein Makro, wie es im folgenden definiert wird:

```
/*--- KA11-07.SAS ---*/
%MACRO beginn(datei);
   PROC CONTENTS DATA=&datei;
   PROC MEANS DATA=&datei N MEAN STD;
   PROC GCHART DATA=&datei;
        VBAR _ALL_;
   RUN;
%MEND beginn;
```

Die Makrodefinition beginnt mit der Anweisung %MACRO. Diese benennt das Makro, hier `beginn`, und definiert die Aufrufparameter. In obigem Beispiel gibt es nur einen einzigen Parameter, `datei`. Im Anschluß folgen die Prozedurschritte: CONTENTS, MEANS und GCHART. Anstelle einer expliziten Datei wird die Makrovariable &DATEI bei der Option DATA= angegeben. Diese wird bei der Ausführung des Makros durch die Datei ersetzt, die als Argument übergeben wird. Die Anweisung `VBAR _ALL_;` bewirkt, daß für jede Variable ein Balkendiagramm gebildet wird. Die letzte Anweisung %MEND beendet die Makrodefi-

nition. Um das Makro anschließend auszuführen, ist folgende Anweisung notwendig:

```
%beginn(biblio.firma);
```

Vor den Makronamen wird ein %-Zeichen gestellt und das Argument wird in Klammern gesetzt. Die Datei `biblio.firma` wird an das Makro übergeben und die drei Prozedurschritte ausgeführt. Im Ausgabe- bzw. im Grafikfenster erscheinen die gewünschten Resultate.

Weitere Beispiele für Makros finden Sie im SAS-System im Unterverzeichnis SAS\CORE\SASMACRO (vgl. Anhang A), im Handbuch zur Makrosprache [10] und im WWW, beispielsweise unter

```
http://web.urz.uni-heidelberg.de/.Unterstuetzung/
Hinweise/Einzel/SAS/#makro
```

Beim Programmieren über den Datenschritt gibt es Schwierigkeiten, wenn komplexe numerische oder algebraische Algorithmen umgesetzt werden sollen. In diesem Fall bietet sich die Prozedur IML, die *Interactive Matrix Language*, als Alternative an, die die Definition höherdimensionaler Strukturen, Matrizen und Vektoren und Berechnungen mit diesen in einer einfachen Schreibweise erlaubt. Die Prozedur IML ist als ein eigenes Modul implementiert und daher nur verfügbar, wenn dieses lizensiert und installiert ist. Der Aufruf erfolgt klassisch über einen Prozedurschritt, der eine interaktive Umgebung aktiviert, und wird mit der Meldung `IML Ready` im Protokollfenster bestätigt. Die Prozedur wird mit der Anweisung QUIT beendet. Dazwischen erscheinen lokale Anweisungen der Prozedur IML.

```
/*--- KA11-08.SAS ---*/
PROC IML;
     RESET PRINT;
     a={3 -1 2,2 -2 3,4 1 -4};
     y={8,2,9};
     x=INV(a)*c;
QUIT;
```

Die Anweisung RESET PRINT im ersten Beispiel sorgt dafür, daß die im folgenden definierten Vektoren und Matrizen sowie die Ergebnisse der Berechnungen im Ausgabefenster angezeigt werden. Anschließend wird eine Matrix `A` definiert, die drei Zeilen (durch Komma getrennte Werte) und drei Spalten (durch Leerzeichen getrennt) enthält. Zudem wird der Vektor `y` definiert. Der Vektor `x`, der die Bedingung `Ax=y` erfüllt, wird durch Auflösung der Gleichung ermittelt:

$x=A^{-1}y$, wobei die Inverse der Matrix A mit Hilfe der Funktion INV berechnet wird. Im Ausgabefenster erscheinen die drei Elemente:

```
A             3 rows      3 cols     (numeric)
        3         -1          2
        2         -2          3
        4          1         -4

Y             3 rows      1 col      (numeric)
        8
        2
        9

X             3 rows      1 col      (numeric)
        3
        5
        2
```

Im nächsten Beispiel werden die Zahlen von 1 bis n aufaddiert. Ohne Verwendung der einschlägigen Formel wird eine DO-Schleife n mal durchlaufen und die jeweilige Summe gebildet. Am Ende wird das Ergebnis mit PRINT ausgegeben.

```
/*--- KA11-09.SAS ---*/
PROC IML;
    /*-- Unterprogramm/Modul ADDIERE ---*/
     START addiere(n);
        zaehler=1;
        summe=0;
        DO UNTIL (zaehler>n);
           summe=summe+zaehler;
           zaehler=zaehler+1;
        END;
        RETURN(summe);
     FINISH addiere;

   /*--- Hauptprogramm ---*/
     s=addiere(10);
     PRINT s;
QUIT;
```

Innerhalb des Prozedurschritts wird zunächst ein Modul `addiere` für das allgemeine Argument `n` definiert: START und `Finish` legen Anfang und Ende dieser

Definition fest. Der `zaehler` wird zunächst auf Eins gesetzt, die `summe` auf 0. In der DO UNTIL-Schleife wird die `summe` mit dem `zaehler` summiert und dieser um 1 erhöht. Die Schleife wird solange durchlaufen, bis der `zaehler` den Wert `n` annimmt. Am Ende wird die `summe` zurückgegeben. Nach der Definition des Moduls `addiere` wird es mit dem Argument `10` aufgerufen: `addiere(10)`. Das Ergebnis `55` wird der Variablen `s` zugewiesen und im Ausgabefenster angezeigt. Neben diesen beiden Trivialbeispielen können auch komplexe Algorithmen realisiert werden. Die Prozedur kann auf Dateien zugreifen und sogar Grafiken erstellen. Beispiele finden Sie in der Online-Hilfe (*Help* → *Sample Programs*) oder im Handbuch zum Modul SAS/IML ([14]).

Eine weitere Stufe der Programmierung innerhalb des SAS-Systems bietet das Modul SAS/AF mit der Prozedur BUILD. Es erlaubt die Erstellung eigener Anwendungen, die vom Endanwender keine SAS- und Programmierkenntnisse erfordern. Es genügt, daß er die Maus und die Tastatur bedienen kann und inhaltlich mit dem Programm vertraut ist. Das Modul SAS/ASSIST ist eine typische SAS/AF-Anwendung, die Sie bereits kennengelernt haben. Der Endanwender sieht eine schöne, oft auch farbige und mit Symbolen versehene Oberfläche, durch die er sich klickend und Dialog- bzw. Menü-geführt bewegt. Alle erforderlichen Daten- und Prozedurschritte laufen im Hintergrund ab und müssen vom Entwickler geplant und vorbereitet werden.

Mit dem folgenden, abschließenden Beispiel werden Sie eine einfache Anwendung entwickeln, die die Grundprinzipien aufzeigt: Für einen programmierunerfahrenen, aber an der Statistik interessierten Endanwender verpacken Sie den Aufruf der Prozedur TTEST in eine AF-Anwendung, so daß Ihr Anwender nur den Namen der Datei auswählen und die Gruppierungsvariable und die Zielvariable festlegen muß, bevor auf Knopfdruck das Ergebnis des t-Tests und des Tests auf Varianzhomogenität (vgl. Abschnitt 8.5) im Ausgabefenster erscheint.

Die Anwendungen werden mit der Prozedur BUILD in Katalogen an- und abgelegt. Der Aufruf erfolgt über die Kommandozeile (`build`), über das Menü (*Globals* → *Develop* → *Application builder*) oder über einen Prozedurschritt:

```
PROC BUILD;
RUN;
```

Mit dem Aufruf der Prozedur BUILD öffnet sich das *Build*-Fenster, in dem alle im Zugriff befindlichen Bibliotheken aufgelistet werden. Sobald eine Bibliothek selektiert wird (sichtbar durch die farbige Unterlegung), werden auch die darin befindlichen Kataloge angezeigt. In der folgenden Abbildung 11.13 (links oben) erkennt man u.a. die Bibliothek `biblio` mit den drei Katalogen `aftest`, `firma` und `formats`.

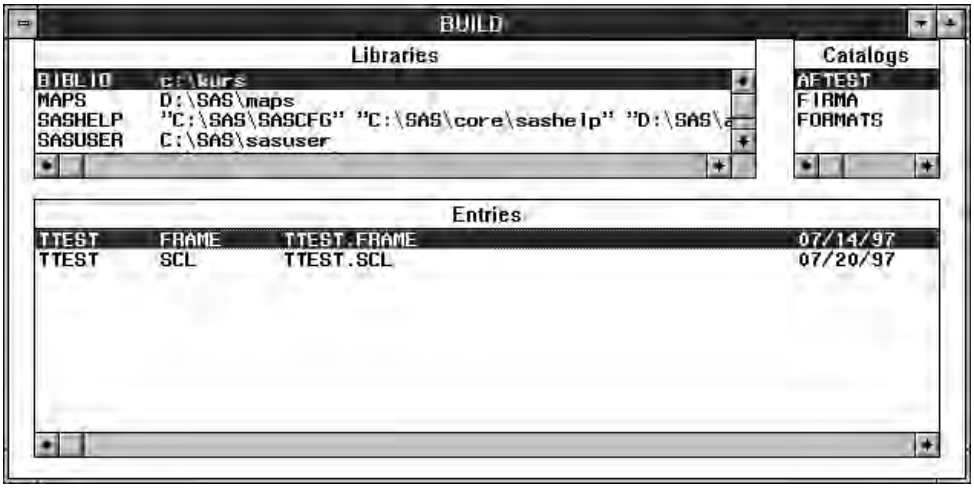


Abb. 11.13: Das Build-Fenster von SAS/AF

Der Katalog `formats` enthält die in Abschnitt 7.5.2 definierten permanenten Formate und `firma` die zu Beginn dieses Kapitels gebildeten Eingabemasken für die FSEDIT-Prozedur. `aftest` ist der Katalog, der die neue Anwendung enthält. Er ist auf der Begleitdiskette zu finden. Sie können den Katalog entweder von der Diskette (Unterverzeichnis KURS) in Ihr Übungsverzeichnis kopieren, oder Sie folgen den weiteren Anweisungen und legen sich einen eigenen Katalog `aftest` an.

Mit der Menüfolge *File* → *New* → *Catalog* wird ein neuer Katalog `aftest` in der aktuellen, d. h. farbig unterlegten Bibliothek angelegt. So, wie `formats` zahlreiche Formate und `firma` verschiedene Eingabemasken enthalten können, kann auch dieser Katalog mehrere Einträge (engl. *Entry*) enthalten. Mit *File* → *New* → *Entry* wird ein neuer Eintrag im aktuell gewählten Katalog angelegt. Als Typ wird FRAME gewählt und für den Namen `ttest` eingesetzt. Durch Bestätigen mit OK öffnet sich ein neuer, leerer Frame (= Rahmen). Mit F3 verläßt man, wie immer in der SAS-Umgebung, den Frame und kehrt zum Build-Fenster zurück. Mit ENTER oder Doppelklick wird der neue Frame erneut geöffnet.

Zunächst ist der Frame völlig leer, aber über die Menüfolge *Actions* → *Make*[3] können verschiedene Elemente eingetragen werden:

- Textfelder (Graphic Text, Text Label)
- Grafiken (Graphics, Image, SAS/GRAPH-Output, Map)

[3] Das Menü wird auch geöffnet, wenn man die rechte Maustaste drückt.

- Schaltflächen (Icon, Push Button, Image Icon, Hotspot)
- Eingabefelder (Input Field, Text Entry)
- Auswahlboxen (Radio Box, Check Box)
- Dateien (Data Form, Data Table)
- Elemente zur OLE-Automation
- und weitere.

Für die t-Test-Anwendung werden benötigt: eine Überschrift, drei Eingabefelder für den Dateinamen sowie die Namen der Gruppierungs- und der Zielvariablen, drei Felder zur Beschriftung der Eingabefelder und die beiden Aktionen „Ausführen“ und „Beenden“.

Abb. 11.14: Attribute eines Graphic Text-Elements

Die Überschrift wird als GRAPHICS TEXT-Element angelegt (*Action* → *Make* → *Graphic Text* → *Ok*), da sich damit größere Schriftzüge mit den SAS-eigenen Schriftarten erzeugen lassen. Nach der Auswahl des Elementtyps erscheint eine mit gestrichelten Linien umrandete Box, die zunächst mit Hilfe der Maus auf dem Bildschirm plaziert werden muß. Die Überschrift wird im oberen Drittel der Gesamtfläche zentriert abgelegt. Diese Plazierung kann später jederzeit wieder korrigiert und verändert werden, indem man das Element durch Anklicken auswählt und mit der Maus an die gewünschte Position verschiebt.

Nach der Plazierung öffnet sich das Attributfenster (Abbildung 11.14). Für den Objektnamen wird `Titel`, für den Schriftzug `Zweigruppenvergleich mit der Prozedur TTEST` und für den Schrifttyp CENTB gewählt. Für die übrigen Eigenschaften werden die Standardeinstellungen übernommen. Nachdem die Eintragungen mit *Ok* bestätigt wurden, erscheint das Element auf dem Bildschirm. Wenn es zu klein oder die Schrift verzerrt angezeigt wird, selektieren Sie es und ziehen solange an dessen Kanten oder Ecken, bis der Text in der gewünschten Form erscheint.

Über den Menüpunkt *Locals* → *Object Attributes* (oder *Object Attributes* im Pop-up-Menü) können die Eigenschaften eines zuvor selektierten Elements nachträglich verändert werden.

Neben der Überschrift werden für die Anwendung drei Elemente zur Auswahl der SAS-Datei, der Gruppierungs- und der Zielvariablen benötigt sowie drei Felder, in denen die ausgewählte Datei und die Variablen angezeigt werden. Im vorliegenden Beispiel wird die Auswahl mit PUSH BUTTON- und die Anzeige mit INPUT FIELD-Elementen realisiert.

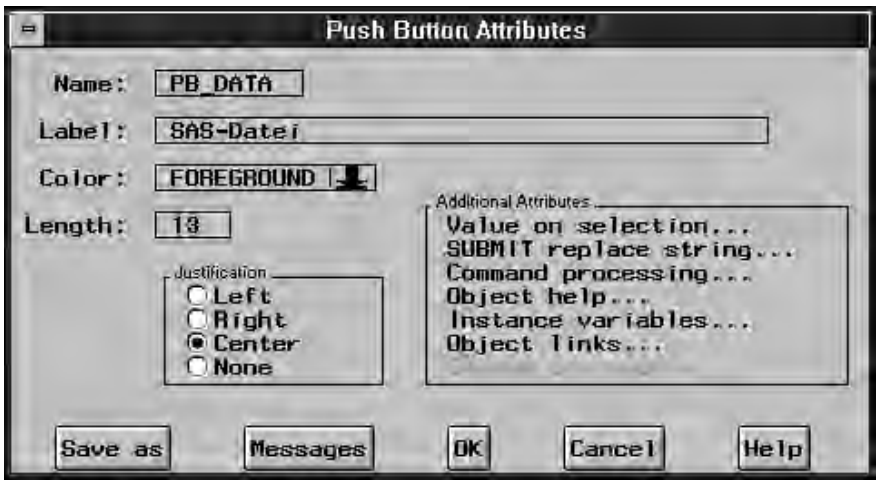


Abb. 11.15: Attribute eines Push Button-Elements

Da die PUSH BUTTON-Elemente im weiteren Verlauf der Anwendungsentwicklung noch benötigt werden, erhalten Sie die Namen `pb_data, pb_class` und `pb_var`, wie in Abbildung 11.15 für das erste Element dargestellt: `pb` für PUSH BUTTOM und `data`, `class` und `var` für deren Rolle im späteren PROC TTEST-Aufruf. Über *Label* wird die Beschriftung des Elements festgelegt, `SAS-Datei`, `Gruppe` und `Zielvariable`. Die übrigen Einstellungen bleiben unverändert. Mit *OK* verläßt man das Fenster.

Die ausgewählte Datei und die Variablen werden in den INPUT FIELD-Elementen angezeigt. Eine Fehleingabe wird unterbunden, indem die Felder mittels *Options*, *Protect* geschützt sind (Abbildung 11.16). Der Endanwender hat somit keine Möglichkeit, in diese Felder über die Tastatur Namen einzutragen, er kann seine Auswahl nur über die PUSH BUTTON-Elemente treffen. Die drei INPUT FIELD-Elemente erhalten die Namen `datei`, `class` und `var`, da sie ebenfalls noch benötigt werden.

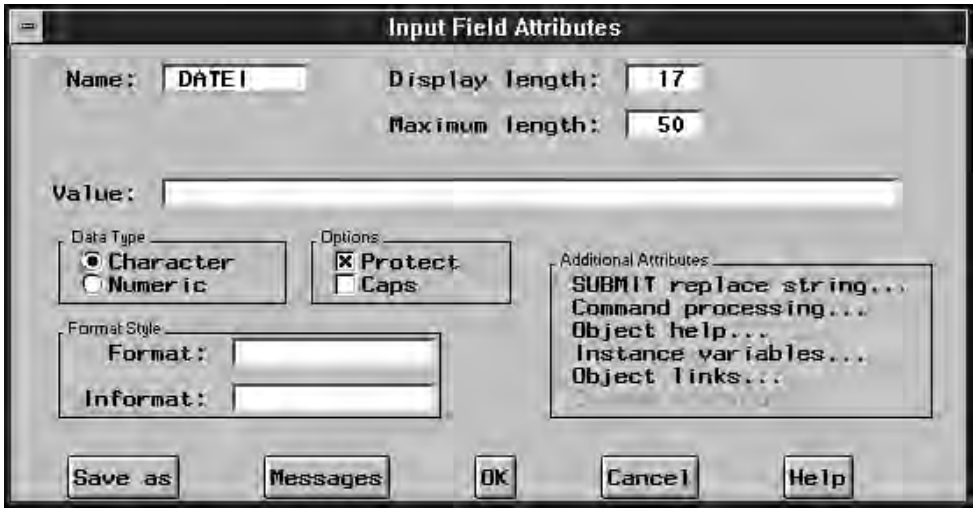


Abb. 11.16: Attribute eines Input Field-Elements

Die beiden Aktionen *Ausführen* und *Ende* werden über IMAGE ICON-Elemente ausgelöst. Neben den Namen `Submit` und `Ende` werden aus dem reichen Angebot, das mit der SAS/AF-Software zur Verfügung gestellt wird, zwei Symbole ausgewählt, die die Aktionen treffend wiedergeben (siehe Abbildung 11.17). Da das Ausführen von Programmen innerhalb der normalen SAS-Umgebung über die Schaltfläche gesteuert wird, wird für *Ausführen* das gleiche Zeichen gewählt. *Ende* wird durch den Pfeil symbolisiert, der auch im Modul SAS/ASSIST verwendet wird.

Die gesamte Anwendung wird, wie jede andere SAS-Anwendung auch, mit dem Kommando `End` beendet. Dieses Kommando wird unter *Additional Attributes* → *Command Processing* im Attributfenster des IMAGE ICON-Elements eingetragen, wie in Abbildung 11.18 wiedergegeben. Sobald man auf das Icon *Ende* klickt, wird die t-Test Anwendung beendet.

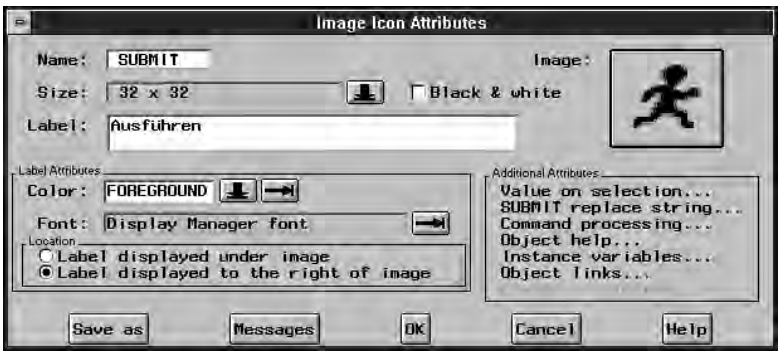


Abb. 11.17: Attribute eines Image Icon-Elements

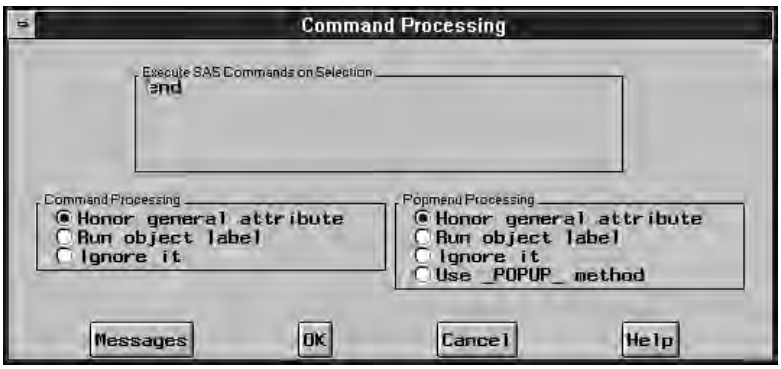


Abb. 11.18: Das Kommando zum Beenden

Soweit, so gut: Die Oberfläche ist fertig und sieht bei Ihnen vielleicht ähnlich aus wie in Abbildung 11.19. Was die ganze Anwendung jedoch erst zum Leben erweckt, ist die Verbindung der einzelnen Frame-Elemente. Diese Verknüpfung wird (wen wundert es) über ein Programm gesteuert: ein SCL-Programm (Screen Control Language). Vielleicht erinnern Sie sich noch an das SCL-Programm, das im Zusammenhang mit der Prozedur FSEDIT besprochen wurde (vgl. Abschnitt 11.3). Ähnliche Bildschirm-steuernde Programme werden auch bei SAS/AF-Anwendungen eingesetzt. Um das SCL-Programm zum Frame `ttest` neu anzulegen, oder ein bereits vorhandenes zu verändern, ruft man *Locals → Edit SCL source* auf.

Das Programm besteht nun im Gegensatz zu FSEDIT nicht aus den drei Abschnitten `INIT`, `MAIN` und `END`, sondern aus dem Abschnitt `INIT` sowie Abschnitten für die drei PUSH BUTTON-Elemente und für das Icon *Ausführen*. Deren Na-

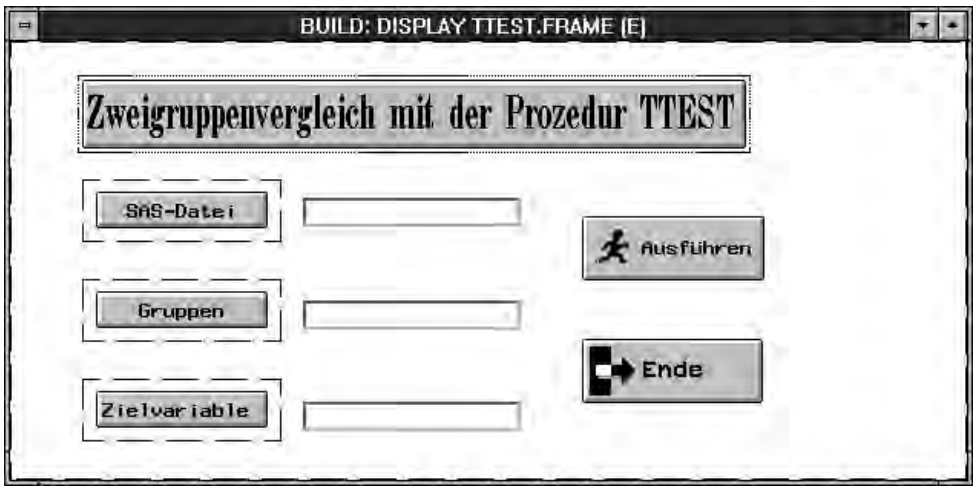


Abb. 11.19: Der Frame `ttest`

men werden entsprechend der Namen der Frame-Elemente gewählt: `PB_DATA`, `PB_CLASS`, `PB_VAR` und `SUBMIT`.

Die SCL-Funktionen OPEN und CLOSE öffnen bzw. schließen eine SAS-Datei, DIRLIST und VARLIST zeigen Auswahllisten mit Dateien bzw. Variablen an. Über Parameter können diese Listen eingeschränkt werden, z. B. in dem nur Dateien vom Type `DATA` oder nur numerische Variablen angezeigt werden.

Mit Auswahl der Datei im Abschnitt `PB_DATA` wird die Variable `nvars` angelegt, die die Anzahl der Variablen widerspiegelt. Gleichzeitig werden die beiden Variablen `class` und `var`, die in den INPUT FIELD-Elementen angezeigt werden, auf `_BLANK_` gesetzt, d. h., eventuelle Eintragungen in diesen Feldern werden gelöscht. In den Abschnitten `PB_CLASS` und `PB_VAR` werden die Gruppen- und die Analysevariablen abgefragt. Im letzten Abschnitt, `Submit`, wird dann das eigentliche SAS-Programm aufgebaut. Innerhalb von SUBMIT/ENDSUBMIT-Blöcken wird der Prozedurschritt für TTEST zusammengesetzt. Der letzte Block, SUBMIT CONTINUE/ENDSUBMIT, beendet die Zusammenstellung und übergibt das fertige Programm an das SAS-System. Das SCL-Fenster wird mit F3 verlassen.

Bevor das Programm getestet werden kann, muß es über die Menüfolge *Locals* → *Compile* kompiliert werden. Fehler, die während des Kompilierungsvorgangs auftreten, werden in einem speziellen Mitteilungsfenster (*Message window*) angezeigt. Achten Sie auf diesbezügliche Hinweise in der Statuszeile.

```
/*--- KA11-10.SCL ---*/
INIT:
RETURN;
PB_DATA:
       datei=DIRLIST('*','DATA',1,'Y');
       dsid=OPEN(datei);
       nvars=ATTRN(dsid,'NVARS');
       dsid=CLOSE(dsid);
       class=_BLANK_;
       var=_BLANK_;
RETURN;
PB_CLASS:
       dd=OPEN(datei,'I');
       class=VARLIST(dd,'A',nvars,'Gruppe (CLASS)');
       dd=CLOSE(dd);
RETURN;

PB_VAR:
       dd=OPEN(datei,'I');
       var=VARLIST(dd,'N',nvars,'Analyse-Variablen (VAR)');
       dd=CLOSE(dd);
RETURN;
SUBMIT:
      SUBMIT;
             PROC TTEST DATA = &datei;
      ENDSUBMIT;

      IF class NE ' ' THEN DO;
             SUBMIT;
                  CLASS &class;
             ENDSUBMIT;
      END;
      IF var NE ' ' THEN DO;
             SUBMIT;
                  VAR &var;
             ENDSUBMIT;
      END;
      SUBMIT CONTINUE;
             RUN;
      ENDSUBMIT;
RETURN;
```

Konnte das Programm fehlerfrei kompiliert werden, kann die Anwendung über *Locals* → *Testaf* getestet werden. Allerdings nur mit Einschränkungen: Die Aus-

wahl läßt sich prüfen, nicht jedoch die Ausführung des PROC TTEST-Schritts, da die SUBMIT-Blöcke in diesem Testmodus nicht ausgeführt werden.

Um die fertige Anwendung außerhalb der PROC BUILD-Umgebung aufzurufen, trägt man das Kommando `AF C=biblio.aftest.ttest.frame` in der Kommandozeile ein oder führt im Programmfenster das folgende Programm aus. (Die Anweisung DM ist die Abkürzung für DISPLAY MANAGER und eine Möglichkeit, innerhalb eines SAS-Programms Kommandos aufzurufen.)

```
DM 'AF C=biblio.aftest.ttest.frame';
```

Nach dem Aufruf erscheint der neu gebildete Frame (Abbildung 11.20) im Anwendungsmodus, d. h. ohne die Linien um die einzelnen Elemente. Nachdem die drei PUSH BUTTON-Elemente angeklickt und die Auswahl getroffen wurde, sind alle Felder ausgefüllt. Wählt man nun noch *Ausführen*, wird der PROC TTEST-Schritt zusammengebaut und ausgeführt und das Ergebnis im Ausgabefenster angezeigt. Verläßt man dieses, gelangt man zur Anwendung zurück. Mit *Ende* (oder F3) wird diese beendet.

Abb. 11.20: Test der Anwendung

In der Anwendung könnten nun noch Hilfetexte und Fehlerprüfungen (z. B. ob die Gruppierungs- und Analysevariablen identisch sind) sowie eigene Funktionstastenbelegungen eingebaut werden. Der Aufruf dieser Anwendung könnte direkt über die Betriebssystemebene erfolgen, so daß der Anwender nicht merkt, daß er mit einem SAS-Programm arbeitet.

A

Zur Installation der SAS-Software

Die Installation der SAS-Software läuft standardisiert ab und kann ohne Probleme mit Hilfe der Installationsunterlagen vorgenommen werden. Sie orientiert sich im PC-Bereich am Microsoft-Windows-Standard. Auf der Installations-CD befindet sich die ausführbare Datei *setup.exe*, die den Anwender durch den Installationsprozeß führt. Je nach Kapazität kann das komplette SAS-System oder nur einzelne Module installiert werden. Der mit den Installationsunterlagen ausgelieferte Lizenzcode, das sogenannte *Setinit*, muß im Verlauf der Installation oder am Ende eingespielt werden. Bei dem Setinit handelt es sich um „Ihren" Schlüssel zur SAS-Software. Erst wenn er richtig paßt, können Sie die neue Welt betreten. Das Setinit enthält Informationen über die lizensierten Module und den Lizenznehmer. Das Setinit muß buchstabengetreu übernommen werden. Es dürfen keine Anführungszeichen vergessen, kein Semikolon ausgelassen oder Groß- und Kleinschreibung vertauscht werden. Wenn das Setinit korrekt eingetragen und während des Installationsvorgangs eingespielt wurde, kann das SAS-System gestartet werden. Auf dem Installationsmedium befinden sich auch Testprogramme, mit denen die Installation überprüft werden kann. Einzelne Module können jederzeit nachträglich installiert werden. Der Lizenzcode muß jedes Jahr erneut eingespielt werden.

Das SAS-System belegt auf der Festplatte eines Rechners mehrere Hundert MB Plattenplatz, wenn man es für den, in den Augen von SAS Institute, typischen Anwender (*Typical User*) installiert: Ohne Rücksicht auf die tatsächlichen Lizenzen werden fast alle Module kopiert.

Um sich bei einer bestehenden Installation einen Überblick über die lizensierten Module zu verschaffen, ruft man nach dem Systemstart im Menü *Globals* → *Accessories* → *Setinit* auf, worauf das `Setinit-Siteval` Fenster erscheint, das die Informationen zum Lizenznehmer enthält. Wählt man nun im *Locals*-Menü *Windows* → *Prod*, erhält man in dem `Setinit-Prod` Fenster eine mehrseitige Liste der lizensierten Produkte (Abbildung A.1). Diese sind mit einem Stern (*) gekennzeichnet. Bei nichtlizensierten Produkten bleibt das Feld leer. Gleichzeitig kann man in der oberen Hälfte des Bildschirms noch das *Expiration Date* ablesen, das Aufschluß darüber gibt, wie lange der Lizenzcode derzeit noch gültig ist. Im vorliegenden Beispiel in Abbildung A.1 endete der Gültigkeitszeitraum der SAS-Lizenz am 30. April 1997.

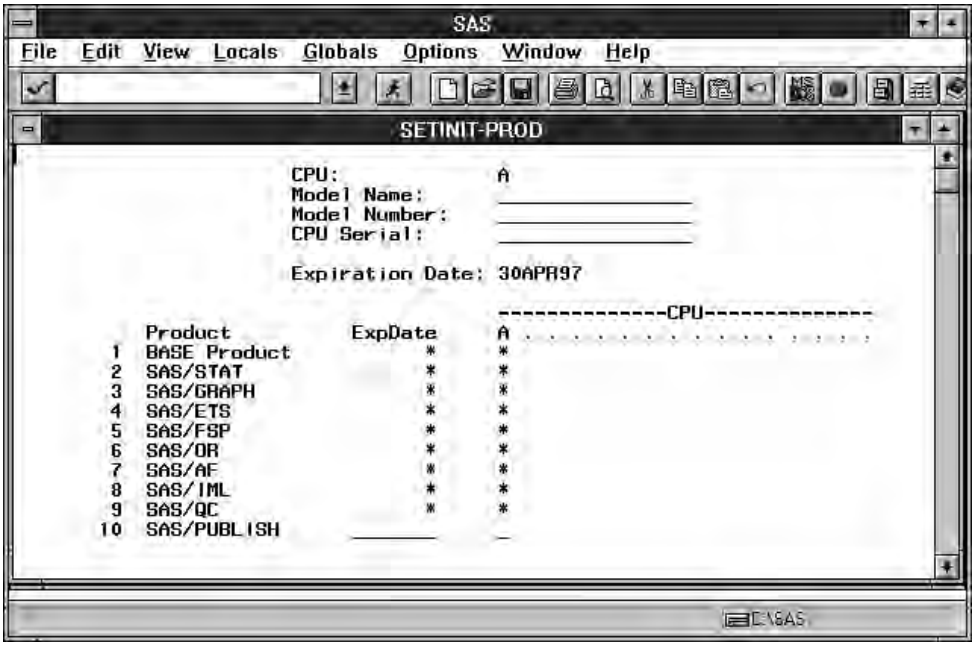
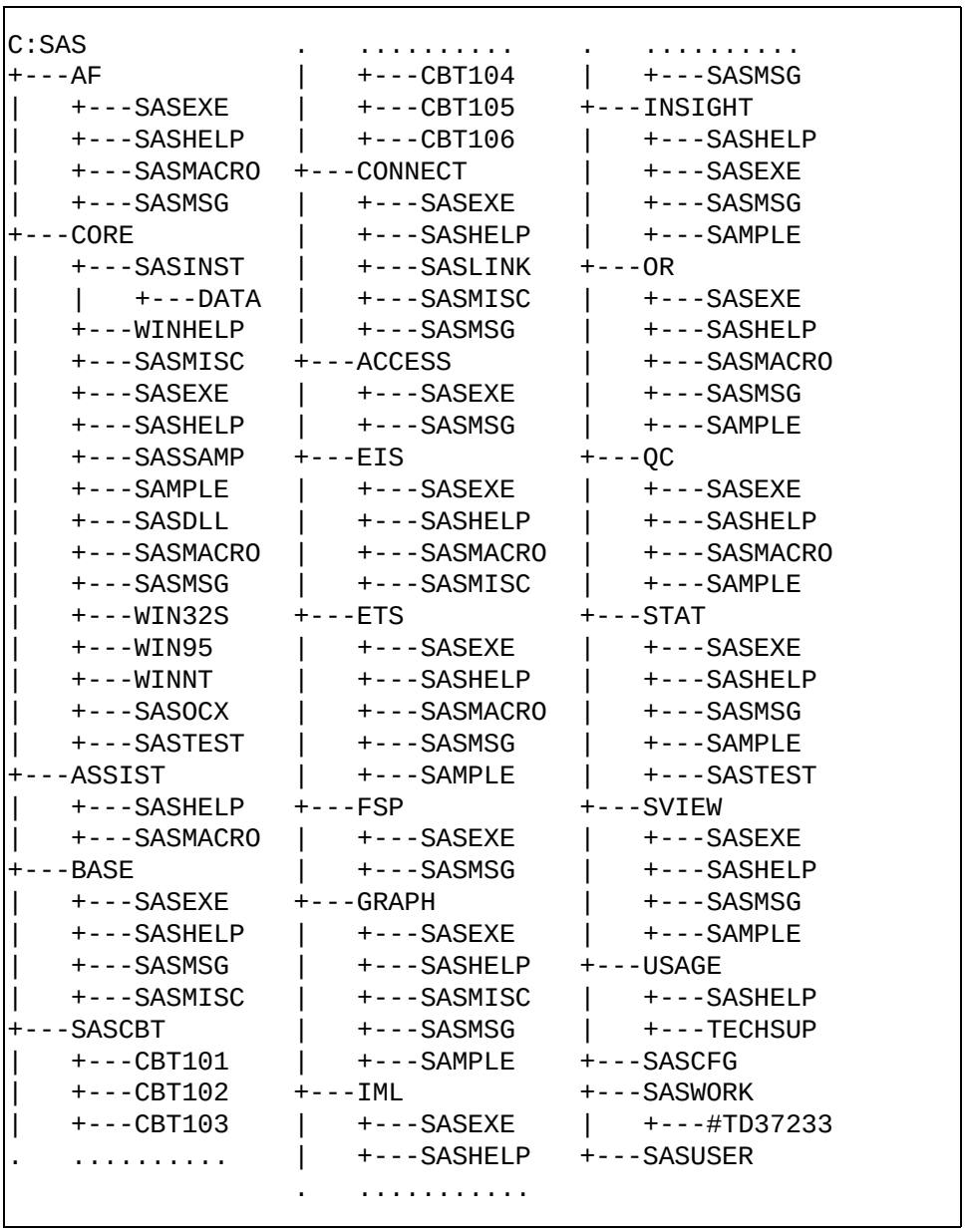


Abb. A.1: Das Setinit-Prod Fenster

Nach der Installation finden Sie in etwa den im folgenden gezeigten Verzeichnisbaum.



Für jedes Modul wurde ein eigenes Verzeichnis angelegt. Die ausführbaren DLL-Dateien (engl. *dynamic load library*) der Module befinden sich im Unterverzeichnis SASEXE, die Help-Files in SASHELP und die Beispielprogramme in SAMPLE.

Im Verzeichnis SASWORK werden die temporären Dateien abgelegt (`DATA neu;` bzw. `DATA WORK.neu;`). Für jede Sitzung wird wiederum ein Unterverzeichnis (hier `#TD37233`) angelegt, das nach Beendigung der SAS-Sitzung gelöscht wird. Die Lokalisation dieses Verzeichnisses wird über die Systemoption WORK= beim Aufruf der SAS-Sitzung festgelegt (in der Konfigurationsdatei CONFIG.SAS, im AUTOEXEC.SAS-Startprogramm oder als Aufrufparameter).

Das Verzeichnis SASUSER enthält Benutzer-spezifische Informationen wie etwa den PROFILE-Katalog oder die beim Aufruf von SAS/ASSIST angelegten Beispieldateien. Seine Lokation wird beim Aufruf über die Systemoption SASUSER= definiert.

Sofern dieses Verzeichnis an seinem Originalplatz verbleibt, wird es mit einer neuen Installation der SAS-Software überschrieben. Legen Sie daher keine Ihrer Dateien in diesem Verzeichnis ab, bevor Sie die Lokalisation geändert haben. Es besteht sonst die Gefahr, daß sie gelöscht werden.

Während des Installationsprozesses oder beim Einspielen eines neuen Lizenzcodes werden Sie noch nach dem SASROOT- und dem SASFOLDER-Verzeichnis gefragt. Bei SASROOT handelt es sich um das Verzeichnis, unter dem die SAS-Software installiert und die ausführbare Datei SAS.EXE abgelegt ist. Im vorliegenden Fall ist dies `C:\SAS`. Das SASFOLDER-Verzeichnis ist das Verzeichnis, das die Unterverzeichnisse SASWORK und SASUSER enthält. Bei der Standardinstallation sind SASROOT und SASFOLDER identisch. Doch wenn die Software teilweise von CD geladen wird oder auf mehreren Laufwerken verteilt installiert wurde, können sich die beiden Angaben unterscheiden.

B

Zur Begleitdiskette

Zu diesem Buch gehört eine Begleitdiskette, auf der sich nicht nur die Beispielprogramme und Beispieldaten befinden, auf die im Verlauf des Textes Bezug genommen wird, sondern auch Dateien, die für die Übungsaufgaben benötigt werden. Der besseren Übersicht wegen wurden diese Dateien und Programme auf die folgenden Unterverzeichnisse verteilt:

- PROGRAMS,
- AUFGABEN,
- FIRMA und
- KURS

Das Unterverzeichnis PROGRAMS enthält die Beispielprogramme, die im laufenden Text durch das Diskettensymbol gekennzeichnet sind (Tabelle B.1). Der Name der Beispielprogramme beginnt mit den Buchstaben KA, es folgt die zweistellige Kapitelnummer und, nach einem Bindestrich, die Nummer des Beispielprogramms. Die Endung lautet *.SAS, damit das jeweilige Programm beim Öffnen über das Menü *File* → *Open* sofort nach Auswahl des Unterverzeichnisses angezeigt wird. Die in Kapitel 11 verwendeten SCL-Programme enden mit *.SCL.

Im Verzeichnis AUFGABEN befinden sich SAS- und andere Dateien, die für die Lösung der Übungsaufgaben erforderlich sind (Tabelle B.2). Die Dateien liegen als SAS 6.12 für Windows-Dateien (V 6.12-Engine) vor und sind an der Endung *.SD2 zu erkennen. Für die Anwender anderer SAS-Versionen wurden die Dateien zusätzlich im ASCII-Format (Endung *.DAT) mit den zugehörigen SAS-

KA03-01.SAS	KA05-01.SAS	KA05-02.SAS	KA05-03.SAS
KA05-04.SAS	KA05-05.SAS	KA05-06.SAS	KA05-07.SAS
KA05-08.SAS	KA05-09.SAS	KA05-10.SAS	KA05-11.SAS
KA05-12.SAS	KA06-01.SAS	KA06-02.SAS	KA06-03.SAS
KA06-04.SAS	KA06-05.SAS	KA06-06.SAS	KA06-07.SAS
KA06-08.SAS	KA06-09.SAS	KA07-01.SAS	KA07-02.SAS
KA07-03.SAS	KA07-04.SAS	KA07-05.SAS	KA07-06.SAS
KA07-07.SAS	KA07-08.SAS	KA07-09.SAS	KA07-10.SAS
KA07-11.SAS	KA07-12.SAS	KA07-13.SAS	KA07-14.SAS
KA07-15.SAS	KA07-16.SAS	KA07-17.SAS	KA07-18.SAS
KA07-19.SAS	KA07-20.SAS	KA07-21.SAS	KA07-22.SAS
KA08-01.SAS	KA08-02.SAS	KA08-03.SAS	KA08-04.SAS
KA08-05.SAS	KA08-05.SAS	KA08-06.SAS	KA08-07.SAS
KA08-08.SAS	KA08-09.SAS	KA08-10.SAS	KA08-11.SAS
KA08-12.SAS	KA08-13.SAS	KA08-14.SAS	KA08-15.SAS
KA08-16.SAS	KA08-17.SAS	KA08-18.SAS	KA08-19.SAS
KA08-20.SAS	KA08-21.SAS	KA08-22.SAS	KA08-23.SAS
KA08-24.SAS	KA08-25.SAS	KA09-01.SAS	KA09-02.SAS
KA09-03.SAS	KA09-04.SAS	KA09-05.SAS	KA09-06.SAS
KA09-07.SAS	KA09-08.SAS	KA09-09.SAS	KA09-10.SAS
KA09-11.SAS	KA09-12.SAS	KA09-13.SAS	KA10-01.SAS
KA11-01.SAS	KA11-02.SAS	KA11-03.SAS	KA11-04.SAS
KA11-05.SAS	KA11-06.SCL	KA11-07.SAS	KA11-08.SAS
KA11-09.SAS	KA11-10.SCL		

Tab. B.1: Programme auf der Begleitdiskette

Einleseprogrammen (`*.SAS`) abgelegt. Außerdem wurden die SAS-Dateien in ein Transportformat (`*.tra`) übertragen, das von (fast[1]) jeder SAS-Version mit Hilfe der Prozedur CIMPORT importiert werden kann. Die Transportdatei `staedte.tra` wird beispielsweise mit folgendem Programm importiert:

```
FILENAME transfer 'A:\aufgaben\staedte.tra';
LIBNAME biblio 'C:\kurs';
PROC CIMPORT DATA=biblio.staedte INFILE=transfer;
RUN;
```

Die FILENAME-Anweisung referenziert die Transportdatei mit dem logischen Namen `transfer`. Mit der Option DATA= der Prozedur CIMPORT wird der Na-

[1] Ausnahme: SAS für DOS 6.04

me der Datei festgelegt, unter dem die neugebildete Datei abgelegt werden soll. Mit INFILE= wird der Referenzname der Transportdatei benannt. Mit der Option CAT= anstelle von DATA= würden Sie einen SAS-Katalog importieren.

Im Unterverzeichnis FIRMA befinden sich die zum Nachvollziehen der Beispielprogramme erforderlichen SAS- und sonstigen Dateien (Tabelle B.2). Die beiden Dateien `stammdat` und `punkte` werden in den folgenden Abschnitten kurz erläutert. Daneben finden Sie hier auch die aus der Verknüpfung dieser beiden Dateien entstandene SAS-Datei `firma`. Wurden die Einleseprogramme für die ASCII-Dateien im Text oder den Lösungen zu den Aufgaben behandelt, erfolgt in der Tabelle B.2 lediglich ein Hinweis auf das Beispielprogramm (vgl. Tabelle B.1) oder das Kapitel und die Nummer der Aufgabe. Mit 7-❽ wird beispielsweise die ❽. Aufgabe aus Kapitel 7 bezeichnet.

SAS für Windows	Einleseprogramm	Rohwerte	Transportdatei
	Unterverzeichnis AUFGABEN		
KUR.SD2	8-❶	KUR.DAT	KUR.TRA
IRIS.SD2	5-❹	IRIS.DAT	IRIS.TRA
STAEDTE.SD2	STAEDTE.SAS	STAEDTE.DAT	STAEDTE.TRA
TOUR97.SD2	TOUR97.SAS	TOUR97.DAT	TOUR97.TRA
	Unterverzeichnis FIRMA		
STAMMDAT.SD2	STAMMDAT.SAS	STAMMDAT.DAT	STAMMDAT.TRA
PUNKTE.SD2	KA06-01.SAS	PUNKTE.DAT	PUNKTE.TRA
FIRMA.SD2	7-❽		FIRMA.TRA
FIRMA.SC2			FIRMA_K.TRA
	Unterverzeichnis KURS		
		ADRESSEN.DAT	
		ZAHL.DAT	
RAHMEN.SD2	KA09-13.SAS		RAHMEN.TRA
AFTEST.SC2			AFTEST.TRA
ADRESSEN.DBF	(dBASE IV-Datei)		
GRUPPE.XLS	(Excel-Tabelle)		
GRUPPE.POR	(SPSS-Transportdatei)		
TRANS.FER	(SAS für UNIX-Transportdatei)		

Tab. B.2: Dateien auf der Begleitdiskette

Im Unterverzeichnis KURS befinden sich Textdateien, die für die Beispielprogramme verwendet werden, sowie Hilfsdateien und Kataloge. Sie können diese Dateien entweder in Ihr persönliches Unterverzeichnis `C:\kurs` kopieren oder direkt von der Begleitdiskette verwenden.

B.1 Die Stammdaten – stammdat

Von den 31 (fiktiven) Mitarbeitern der Übungsfirma werden neben der Personalnummer noch Name, Vorname, Wohnort, Alter, Geschlecht, Familienstand und Beschäftigungsbeginn in den Stammdaten (Tabelle B.3) notiert. Die beiden Merkmale Geschlecht und Familienstand wurden des leichteren Erfassens wegen nach folgendem Schema kodiert:

Merkmal	Kodierung	Bedeutung
Geschlecht	m	männlich
	w	weiblich
Familienstand	l	ledig
	h	verheiratet
	g	geschieden
	w	verwitwet

B.2 Der Einstellungstest – punkte

Bei der Einstellung mußten die Mitarbeiter der Beispielfirma einen aus vier Abschnitten zusammengesetzten Eignungstest absolvieren. Der Personalleiter archiviert die Ergebnisse in den Personalakten in der Datei punkte (Tabelle B.4).

Da sich der 2. Testabschnitt aus zwei Teilen zusammensetzt, sind in der Datei neben der Personalnummer fünf Variablen enthalten. Die Skala für die Testabschnittte 1 bis 3 reichte von 0 bis 10 Punkte und für den 4. Teil von 0 bis 20.

Nr.	N-Name name	V-Name	Ort	A.	G.	Fam. stand	K.	Eintritt
1001	Meier	Hans	Walldorf	25	m	l	.	15/01/91
1002	Schulz	Karin	Mannheim	27	w	h	.	01/06/90
1003	Frisch	Melanie	Heidelberg	41	w	h	2	01/06/90
1004	Neuer	Manfred	Heidelberg	58	m	g	.	01/07/91
1005	Neuer	Annerose	Heidelberg	53	w	l	.	15/08/94
1006	Karl	Helmut	Mannheim	51	m	g	1	01/09/93
1007	Manz	Eva	Walldorf	24	w	l	1	01/09/91
1008	Ganz	Ulrich	Mannheim	38	m	h	.	01/07/91
1009	Weber	Joachim	Karlsruhe	42	m	h	2	01/12/95
1010	Ulrich	Matthias	Weinheim	29	m	l	.	15/06/94
1011	Heuer	Heidi	Heidelberg	35	w	l	.	01/12/90
1012	Maier	Horst	Mannheim	51	m	w	.	01/10/92
1013	Schuber	Sabine	Weinheim	29	w	h	1	01/09/91
1014	Sauer	Maria	Heidelberg	48	w	h	2	01/10/91
1015	Weber	Evi	Walldorf	56	w	g	.	15/11/96
1016	Meyer	Susanne	Karlsruhe	43	w	h	.	15/06/93
1017	Horms	Marion	Mannheim	19	w	l	.	15/03/96
1018	Graz	Herbert	Heidelberg	28	m	l	.	01/05/94
1019	Scholl	Bernhard	Mannheim	31	m	h	1	15/03/95
1020	Garcia	Jose	Weinheim	45	m	l	2	01/02/95
1021	Vogel	Norbert	Heidelberg	32	m	l	.	01/04/92
1022	Maus	Michael	Mannheim	29	m	l	.	01/01/96
1023	Müller	Siegfried	Karlsruhe	47	m	l	.	15/02/95
1024	Schmidt	Johann	Weinheim	29	m	h	.	01/05/91
1025	Karl	Max	Heidelberg	31	m	l	.	01/12/96
1026	Maier	Johannes	Heidelberg	56	m	l	.	01/09/93
1027	Schmid	Sigrun	Mannheim	43	w	h	3	01/05/95
1028	Maier	Marta	Heidelberg	38	w	h	.	15/02/94
1029	Bauer	Albert	Weinheim	24	m	l	.	01/06/93
1030	Weber	Manfred	Mannheim	28	m	l	.	01/08/94
1031	Scholz	Stefan	Heidelberg	21	m	l	.	01/04/95

Tab. B.3: Die Stammdaten `stammdat`

Personal-	Testabschnitt				
nummer	1	2a	2b	3	4
1001	9	2	9	8	6
1002	6	9	6	5	15
1003	6	7	7	5	13
1004	7	9	7	7	14
1005	8	10	8	8	18
1006	2	6	7	7	13
1007	8	2	10	7	13
1008	8	9	9	10	11
1009	6	8	6	6	2
1010	10	7	9	8	17
1011	6	3	6	5	12
1012	6	3	7	5	14
1013	3	9	6	5	14
1014	2	5	4	3	15
1015	6	2	6	6	19
1016	6	2	7	6	11
1017	7	10	7	7	11
1018	5	3	6	5	8
1019	3	5	6	6	14
1020	6	1	8	7	5
1021	1	5	6	8	15
1022	9	7	7	6	11
1023	5	5	8	4	8
1024	6	6	9	5	6
1025	7	8	4	6	17
1026	3	4	6	7	16
1027	6	2	7	8	13
1028	9	10	2	10	9
1029	10	8	1	8	12
1030	7	6	8	4	6
1031	5	8	10	6	10

Tab. B.4: Der Einstellungstest `punkte`

C

Lösungen zu den Übungsaufgaben

Am Ende der Kapitel befinden sich zumeist mehrere Aufgabenstellungen, die Ihnen bei der Umsetzung der Anweisungen und Kommandos helfen und wichtige Aspekte verdeutlichen sollen. Zu diesen Aufgaben werden hier Lösungsvorschläge und Erläuterungen gegeben. Seien Sie bitte nicht enttäuscht, wenn Ihre Lösung nicht genau mit diesem Vorschlag übereinstimmt. Es gibt in komplexeren Systemen, zu denen auch das SAS-System gehört, oft mehrere Wege, die zum Ziel führen. Für die Aufgaben ab Kapitel 5 wird vorausgesetzt, daß das Kursverzeichnis `C:\kurs` mit dem logischen Namen `biblio` über das Libraries-Fenster oder eine entsprechende LIBNAME-Anweisung referenziert wurde.

C.1 Kapitel 3

❶ Das Beispielprogramm `KA03-01.SAS` besteht aus zwei globalen Anweisungen, einem Daten- und einem Prozedurschritt.

`/*--- KA03-01.SAS ---*/` ist ein Kommentar.

`OPTIONS` ist das Schlüsselwort der Anweisung OPTIONS.

`NODATE` ist die Option zur Unterdrückung der Ausgabe des aktuellen Datums.

`;` beendet die OPTIONS-Anweisung.

`*-- Globale Anweisungen;` ist eine kommentierte Anweisung.

`TITLE` ist das Schlüsselwort der Anweisung TITLE.

`'Adressen'` ist der Text, der als Titelzeile im Ausgabefenster ausgegeben wird.

`;` beendet die TITLE-Anweisung.

`DATA` kennzeichnet den Beginn des Datenschritts und der DATA-Anweisung.

`adressen` ist der Name der SAS-Datei, die erzeugt wird.

`;` beendet die DATA-Anweisung.

`*-- Datenschritt;` ist eine kommentierte Anweisung.

`INPUT` ist das Schlüsselwort der Anweisung INPUT, mit der die Variablen deklariert werden.

`name` ist der Name der ersten Variablen.

`$` ist eine Option, mit der die Variable `name` als Zeichenkettenvariable festgelegt wird.

`vorname` ist der Name der zweiten Variablen.

`$` ist eine Option, mit der die Variable `vorname` als Zeichenkettenvariable festgelegt wird.

`wohnort` ist der Name der dritten Variablen.

`$` ist eine Option, mit der die Variable `wohnort` als Zeichenkettenvariable festgelegt wird.

`alter` ist der Name der vierten Variablen.

`;` beendet die INPUT-Anweisung.

`LINES` Die LINES-Anweisung teilt dem System mit, daß die Datensammlung folgt.

`;` beendet die LINES-Anweisung.

`*-- Datensammlung;` ist eine kommentierte Anweisung.

`Fritz Petra Heidelberg 31 ...` sind die Datenwerte.

`RUN` ist das Schlüsselwort der Anweisung RUN.

`;` beendet die RUN-Anweisung und den Datenschritt.

`PROC` kennzeichnet den Beginn des Prozedurschritts und ist das Schlüsselwort der PROC-Anweisung.

`PRINT` ist der Name der gewählten Prozedur.

`DATA=` Die DATA=-Option wählt die zu verarbeitende Datei.

`adressen` ist der Name der darzustellenden Datei.

`;` beendet die PROC-Anweisung.

`*-- Prozedurschritt;` ist eine kommentierte Anweisung.

`RUN` ist das Schlüsselwort der Anweisung RUN.

`;` beendet die RUN-Anweisung und den Prozedurschritt.

❷ Das Beispielprogramm `KA03-01.SAS` von der Diskette wird über das Menü *File* → *Open* geöffnet.

Der Sortiervorgang erfolgt mit Hilfe der Prozedur SORT. Die Anweisung BY mit der Option DESCENDING gibt die Sortierreihenfolge vor. Das Programm (ohne die Kommentare) hat die folgende Form:

```
OPTIONS NODATE;
TITLE 'Adressen';
DATA adressen;
     INPUT name $ vorname $ wohnort $ alter;
     LINES;
Fritz   Petra   Heidelberg 31
Fischer Ulrich  Heidelberg 46
Meier   Hans    Walldorf   25
Rost    Werner  Mannheim   34
Schulz  Karin   Mannheim   27
RUN;
/*--- Sortierung der Adressen ---*/
PROC SORT DATA=adressen;
     BY DESCENDING name;
RUN;
/*--- Ende des eingeschobenen Prozedurschritts ---*/
PROC PRINT DATA=adressen;
RUN;
```

Das Programm wird mit F3 ausgeführt. Das Ausgabe- und das Protokollfenster werden über *File* → *Save as* in zwei verschiedene Dateien, etwa `C:\kurs\aufgabe1.out` und `C:\kurs\aufgabe1.log` abgespeichert.

❸ Die Anweisung TITLE kann beispielsweise wie folgt verändert werden:

```
TITLE 'Nach Namen sortierte Adressen';
```

Belassen Sie sie aber an der gleichen Stelle im Programm. Würden Sie die geänderte Anweisung ganz ans Ende schreiben, im Anschluß an den PROC PRINT-Schritt, würde sie zunächst keinerlei Wirkung zeigen. Probieren Sie das ruhig aus!

C.2 Kapitel 4

❶ Sie können das Programm von der Diskette mittels *File* → *Open* öffnen. Wählen Sie das Unterverzeichnis PROGRAMS und die Textdatei `KA03-01.SAS`. Fügen Sie Zeilennummern über das Kommando `NUMS ON` ein und löschen Sie die Mitarbeiterdaten über das Zeilenkommando `D`. Fügen Sie neue Zeilen mit `I` ein und tragen Sie eigene Namen ein. Ein Beispiel ist in Abbildung C.1 dargestellt.

Abb. C.1: Beispiellösung zu Aufgabe 1 von Kapitel 4

❷ Das um den Sortierschritt erweiterte Programm könnte die folgende Form haben:

```
title;
data club1;
   input idno name $ team $ strtwght endwght;
   cards;
1049 Amelia yellow 145 124
1246 Ravi yellow 194 177
1078 Ashley red 127 118
1221 Jim yellow 220 .
4010 Tom green 200 170
3101 Anne blue 130 125
run;
/*--- Neuer Prozedurschritt ---*/
PROC SORT DATA=club1;
     BY idno;
RUN;
proc print data=club1;
   title 'Weight Club Members';
run;
```

C.3 Kapitel 5

❶ Für die Adressen der Freunde werden neben Vorname und Name, die Straße, Postleitzahl und Wohnort eingetragen. Vielleicht hätten Sie auch gerne das Geburtsdatum, aber wie man Datumsangaben behandelt, lernen Sie erst in Kapitel 7.

```
DATA freunde;
     INPUT vorname $ &:30. name $ &:30 plz $
           ort $ &:30 strasse $ &30.;
     LINES;
Anneliese  Schwarz  69123 Heidelberg  Marktstraße 4
Udo  Freund  69117 Heidelberg  Hauptstr. 5
Jan Ullrich  Schmitt  69120 Heidelberg  INF 680
....
```

In dieser ersten Variante werden die Variablen listengesteuert eingelesen. Da außer der fünfstelligen Postleitzahl längere Namen, eventuell auch mit

Leerzeichen, auftreten können, wird das $-Zeichen um &:30 erweitert. Damit muß der Abstand zwischen zwei Ausprägungen aber mindestens zwei Leerzeichen betragen.

Fragen Sie sich, warum die Postleitzahl nicht als numerische, sondern als Textvariable eingelesen wird? Bei numerischen Variablen werden führende Nullen automatisch entfernt. Die Postleitzahl 01234 würde damit zu 1234, was der Deutschen Bundespost und Freunden aus den neuen Bundesländern sicher keine Freude bereiten würde. Bei Textvariablen passiert dies nicht: 0 bleibt 0, auch am Anfang.

Um die gleichen Angaben spaltengesteuert einzulesen, werden sie zunächst in Spalten angeordnet und die Anweisung INPUT entsprechend geändert:

```
DATA freunde;
     INPUT vorname $ 1-11 name $ 13-19 plz $ 21-25
           ort $ 27-36 strasse $ 38-50;
     LINES;
Anneliese   Schwarz 69123 Heidelberg Marktstraße 4
Udo         Freund  69117 Heidelberg Hauptstr. 5
Jan Ullrich Schmitt 69120 Heidelberg INF 680
....
```

Hier ist nun kein zweites Leerzeichen zur Abtrennung erforderlich. Die einzelnen Worte könnten sogar direkt aneinander grenzen. Und schließlich noch die formatgesteuerte Variante:

```
DATA freunde;
     INPUT vorname $11. @13 name $7. @21 plz $5.
           @27 ort $10. @38 strasse $13.;
     LINES;
Anneliese   Schwarz 69123 Heidelberg Marktstraße 4
Udo         Freund  69117 Heidelberg Hauptstr. 5
Jan Ullrich Schmitt 69120 Heidelberg INF 680
....
```

Hier wird der absolute Spaltenzeiger verwendet und die Formate $w..

Die Datenschritte werden mit *File* → *Save As* oder  in den drei Dateien freunde1.sas, freunde2.sas und freunde3.sas abgespeichert.

❷ Um die Datei permanent abzulegen, muß eine Bibliothek referenziert und ein zweiteiliger Dateinamen verwendet werden.

```
LIBNAME meine 'C:\kurs';
DATA meine.freunde;
     INPUT vorname $ &:30. name $ &:30 plz $
           ort $ &:30 strasse $ &30.;
     LINES;
Anneliese  Schwarz  69123 Heidelberg  Marktstraße 4
Udo  Freund  69117 Heidelberg  Hauptstr. 5
Jan Ullrich  Schmitt  69120 Heidelberg  INF 680
....
```

Das Programm wird unter `mfreunde.sas` abgespeichert. Mit dem Dateimanager (oder dem Explorer unter Windows 95) wurde die permanente SAS-Datei gesucht (Stichwort: `freunde.*`) und im Verzeichnis `C:\kurs` unter dem Namen `freunde.sd2` gefunden.

Nach erneutem SAS-Aufruf wird das folgende Programm ausgeführt:

```
LIBNAME meine 'C:\kurs';
TITLE 'Meine Freunde';
PROC PRINT DATA=meine.freunde;
RUN;
```

Im Ausgabefenster erscheint die Liste. Die Anweisung TITLE ist nicht unbedingt erforderlich. Sie wurde bisher auch noch gar nicht richtig erklärt, das folgt erst in Kapitel 10.

❸ Das Programm `KA05-08.SAS` wird über *File* → *Open* → *Read File* oder im Programmfenster geöffnet.

```
DATA adressen;
     INPUT  name $ 1-10 betrag 12-16 @@;
     LINES;
Fritz P.    5.00 Fischer U. 10.00 Meier H.    5.00
Rost W.     5.00 Schulz K.  15.00
;
```

Je nachdem, wie lange Sie gewartet haben, bis Sie das Programm unterbrochen haben, wurden unterschiedlich viele Beobachtungen in die Datei

eingetragen. Aber immer die gleichen: nur Herr Fritz mit seinen 5 DM. Die anderen Angaben wurden ignoriert.

```
DATA adressen;
     INPUT  name $ 1-10 betrag 12-16;
     LINES;
Fritz P.    5.00 Fischer U. 10.00 Meier H.    5.00
Rost W.     5.00 Schulz K.  15.00
;
```

Wird der Zeilenhalter @@ vergessen, dann werden nur die Herren Fritz und Rost erfaßt, da mit einem Abarbeiten der Anweisung INPUT die jeweilige Datenzeile als beendet betrachtet wird.

```
DATA adressen;
     INPUT  name $ 1-10 betrag 12-16;
     LINES;
Fritz P.
5.00
Fischer U.
10.00
...
RUN;
```

Wenn der Betrag in einer separaten Zeile steht, wird dieser als eigene Beobachtung eingelesen und die Zahl als Name aufgefaßt. Die Variable `betrag` hat nur fehlende Ausprägungen.

❹ Da in der Aufgabenstellung nicht genauer spezifiziert wurde, wie die Daten in der externen Datei vorliegen, muß zunächst mit Hilfe eines Editors oder dem Type-Kommando unter DOS die Datei betrachtet werden. Mit dem SAS-Editor erscheint folgendes Bild:

```
Kelchblattlänge und -breite, Blütenblattlänge und
-breite, jeweils in cm, Irisart
 5.1 3.5 1.4 0.2 Setosa
 7.0 3.2 4.7 1.4 Versicolor
 6.3 3.3 6.0 2.5 Virginica
 4.9 3.0 1.4 0.2 Setosa
...
```

Der zugehörige Datenschritt, mit dem diese Datei in eine SAS-Datei umgewandelt werden kann, hat die Form:

```
DATA biblio.iris;
     INFILE 'A:\daten\iris.dat' FIRSTOBS=3;
     INPUT kelch_l kelch_b bluete_l bluete_b
           art $ &30.;
RUN;
```

❺ Bevor die Dateien `staedte` und `tour97` genauer untersucht werden können, muß das Verzeichnis, in dem sie sich befinden, referenziert werden. Anschließend hilft die Prozedur CONTENTS bei der Beantwortung der Fragen:

```
LIBNAME disk 'A:\AUFGABEN';
PROC CONTENTS DATA=disk.staedte;
RUN;
PROC CONTENTS DATA=disk.tour97;
```

Die Datei `staedte` enthält 263 Beobachtungen und die Variablen `stadt`, `land` und `zahl`. Die Datei `tour97` besteht aus 139 Beobachtungen und den Variablen `fahrer`, `startnr`, `zeit`, `endzeit`, `berg` und `sprint`.

❻ Sie benötigen die LIBNAME-Anweisung zur Referenzierung der Bibliothek und einen Datenschritt.

```
LIBNAME biblio 'C:\kurs';
DATA biblio.stammdat;
LIBNAME biblio 'C:\';
DATA biblio.stammdat;
     INFILE 'A:\firma\stammdat.dat';
     INPUT nr nname $ 6-12 vname $ 14-22 ort $ 24-34
           alter sex $ 40 famstand $ 43 kinder 44
           eintritt 52-53;
RUN;
```

C.4 Kapitel 6

❶ Die sortierte Datei wurde unter dem Namen `s_ punkte` abgelegt.

```
PROC PRINT DATA=s_punkte;
RUN;
PROC PRINT DATA=biblio.punkte;
RUN;
```

❷ Damit die im letzten Kapitel erzeugte Datei `biblio.iris` nicht überschrieben wird, wird die Option OUT= in der PROC SORT-Anweisung verwendet. Mit der VAR-Anweisung wird die Variablenliste eingeschränkt.

```
PROC SORT DATA=biblio.iris OUT=iris;
     BY kelch_l;
RUN;
PROC PRINT DATA=iris;
     VAR art kelch_l kelch_b;
RUN;
```

❸ Zunächst muß eine Referenz auf die Diskette und das Verzeichnis `aufgaben` gelegt werden, bevor der PROC SORT-Schritt ausgeführt werden kann. Da die Diskette mit einem Schreibschutz versehen wurde und der Originalzustand der Datei erhalten bleiben soll, muß die Option OUT= verwendet werden.

```
LIBNAME disk 'A:\aufgaben';
PROC SORT DATA=disk.staedte OUT=biblio.staedte;
     BY land zahl;
RUN;
```

Gibt man die Städte jetzt noch getrennt nach Bundesland mit der Prozedur PRINT aus, kann die Frage nach den drei größten Städten in jedem Bundesland leicht beantwortet werden.

```
PROC PRINT DATA=biblio.staedte;
     BY land;
RUN;
```

In Berlin-Brandenburg haben die Städte Potsdam, Cottbus und Brandenburg a. d. Havel die meisten Einwohner, in Baden-Württemberg Stuttgart, Mannheim und Karlsruhe usw.

❹ Nachdem die Datei auf der Begleitdiskette referenziert wurde, wird mit der Prozedur SORT die Datei nach der Variablen `berg` absteigend sortiert und die Liste mit einem PROC PRINT-Schritt ausgegeben.

```
PROC SORT DATA=biblio.tour97;
     BY DESCENDING berg;
RUN;
PROC PRINT DATA=biblio.tour97;
RUN;
```

C.5 Kapitel 7

❶ Die Datei `neu` wird angelegt. Allerdings hat sie nur 1 Beobachtung und 0 Variablen, d. h., sie besteht eigentlich aus nichts. Im Protokollfenster erscheint die Meldung:

```
1    data neu;run;
NOTE: The data set WORK.NEU has 1 observations and 0
      variables.
NOTE: The DATA statement used 12.31 seconds.
```

❷ Die Datei `ws` enthält zwei numerische Variablen, `alfa` und `x`, und eine Beobachtung. Die Variablen wurden implizit durch die Zuweisung gebildet. Die beiden Anweisungen werden einmal ausgeführt und die Datenwerte in die Datei eingetragen.

❸ Ohne die Anweisung OUTPUT enthält die Datei nur eine Beobachtung. Die Variable `i` hat den Wert `10`. Das heißt, es wird nur die letzte Zuweisung in der Datei abgelegt. Mit OUTPUT werden für jeden Schleifendurchlauf die Werte in die Datei als eigene Beobachtung eingetragen. Setzt man den Schleifenzähler `i` daher auf 100 hoch, werden 100 Beobachtungen erzeugt.

❹ Damit nur `Hans` in der Liste erscheint, müssen die fehlenden Angaben ausgeschlossen werden.

```
PROC PRINT DATA=gruppe;
     WHERE alter NE . AND alter<20;
RUN;
```

❺ Die Daten können entweder in eine externe Datei eingetragen werden oder wie im folgenden direkt im Programmfenster erfaßt werden:

```
DATA labor;
     INPUT labor $ substanz $ menge @@;
     LINES;
a x 100.00 b x  90.00
a x 110.00 b x 105.00
a x 105.00 b x  98.00
a y 110.00 b y 120.00
a y 100.00 b y 130.00
a y  95.00 b y 110.00
a z 200.00 b z 180.00
a z 170.00 b z 170.00
a z 150.00 b z 140.00
RUN;
```

```
DATA sub_x;
     SET labor;
     WHERE substanz="x";
RUN;

PROC PRINT DATA=sub_x;
     WHERE menge<100;
RUN;

DATA pruefen;
     SET labor;
     WHERE x>100 AND y>100 AND z>160;
RUN;
```

❻ Wenn die Anweisung ELSE unterbleibt, wird der Variablen sex bei allen Beobachtungen, deren Namen nicht in der Liste erscheinen, kein Wert zu-

gewiesen. In einer Liste würde sie für diese Beobachtungen einen Punkt als Zeichen des fehlenden Werts aufzeigen.

❼ Analog zu Programm KA07-19.SAS muß man schreiben:

```
PROC FORMAT;
     VALUE $fsexneu m=männlich
                    w=weiblich;
RUN;
```

❽ Die beiden Dateien werden mittels MERGE in einem Datenschritt verknüpft. Die BY-Variable ist die Personalnummer. Da diese allerdings in der Datei punkte mit pnr bezeichnet wird, in stammdat dagegen mit nr muß mit RENAME eine Anpassung erfolgen:

```
DATA biblio.firma;
     MERGE biblio.stammdat
           biblio.punkte(RENAME=(pnr=nr));
     BY pnr;
RUN;
```

❾ Die permanenten Formate werden über die LIBRARY-Bibliothek und die Option LIB= der PROC FORMAT-Anweisung definiert.

```
LIBNAME LIBRARY 'C:\kurs';
PROC FORMAT LIB=LIBRARY;
     VALUE $fstand  l=ledig
                    h=verheiratet
                    w=verwitwet
                    g=geschieden;
RUN;
```

C.6 Kapitel 8

❶ Mit Hilfe eines Editors überzeugt man sich zunächst von der Reihenfolge der Variablen: Nummer, Gruppe, Gewicht vor und nach der Kur.

```
DATA kur;
     INFILE 'A:\daten\kur.dat';
     INPUT nr gruppe gew_vor gew_nach;
RUN;
```

❷ Die Auszählung der beiden Gruppen erfolgt mit der Prozedur FREQ. Man hätte aber genauso gut MEANS einsetzen können. FREQ liefert mit der Option CHISQ zusätzlich einen Signifikanztest.

```
PROC FREQ DATA=kur;
     TABLES gruppe / CHISQ TESTP=(0.5,0.5);
RUN;
```

Die Aufteilung von 14 zu 16 ist nahezu ideal und nicht signifikant von 15:15 verschieden.

❸ Die wichtigsten Kennwerte erhält man mit der Prozedur MEANS. Die Anweisung CLASS trennt diese nach den beiden Gruppen auf.

```
PROC MEANS DATA=kur;
     CLASS gruppe;
     VAR gew_vor gew_nach;
RUN;
```

Den Median und den Quartilsabstand berechnet die Prozedur UNIVARIATE. Da diese Prozedur aber keine CLASS-Anweisung kennt, sondern nur die BY-Anweisung, muß die Datei zuvor sortiert werden.

```
PROC SORT DATA=kur;
     BY gruppe;
RUN;
PROC UNIVARIATE DATA=kur;
     VAR gew_vor gew_nach;
     BY gruppe;
RUN;
```

❹ Die Gewichtsdifferenz wird in einem Datenschritt berechnet. Der Test dieser einer Variablen wird dann mit der Prozedur UNIVARIATE durchgeführt.

```
DATA kur;
     SET kur;
     diff=gew_vor-gew_nach;
RUN;
PROC UNIVARIATE DATA=kur PLOT;
     VAR diff;
RUN;
```

Erstaunlicherweise nehmen nicht alle Teilnehmerinnen im Verlauf der Kur ab, ein Teil nimmt auch zu. Die Gewichtsdifferenzen schwanken zwischen -5 und +10 Kilogramm. Der Wilcoxon-Rangsummentest zeigt eine signifikante Gewichtsdifferenz (p=0.0278) auf.

❺ Die geforderten Konfidenzintervalle um die mittlere Gewichtsdifferenz ermittelt man mit der Prozedur MEANS. Da die Intervalle zunächst für alle, dann aber getrennt für die beiden Gruppen berechnet werden sollen, müssen zwei Prozedurschritte ausgeführt werden.

```
PROC MEANS DATA=kur clm;
     VAR diff;
RUN;
PROC MEANS DATA=kur clm;
     CLASS gruppe;
     VAR diff;
RUN;
```

Das Konfidenzintervall für die gesamte Gruppe beträgt (0.3, 3.8). Die Null ist nicht darin enthalten, weshalb der Hypothesentest in der vorherigen Teilaufgabe einen signifikanten Unterschied aufzeigte. Beschränkt man sich auf die beiden Gruppen, dann relativiert sich die Aussage: Beide Konfidenzintervalle schließen die Null ein. Keine der beiden Behandlungsarten garantiert für sich betrachtet eine Gewichtsreduktion.

❻ Der Zusammenhang wird mit dem Korrelationskoeffizienten nach Pearson und der Prozedur CORR gemessen.

```
PROC CORR DATA=kur;
     VAR gew_vor gew_nach;
RUN;
```

Mit 0.85 ist der Zusammenhang recht hoch.

❼ Ob sich die beiden Kurmaßnahmen unterscheiden, kann man mit dem t-Test und der Prozedur TTEST nachprüfen.

```
PROC TTEST DATA=kur;
     CLASS gruppe;
     VAR diff;
RUN;
```

Mit einem p-Wert von 0.9 wird bestätigt, was bereits vorher die beiden Konfidenzintervalle gezeigt hatten: Die beiden Maßnahmen unterscheiden sich nicht.

C.7 Kapitel 9

❶ Balkendiagramme werden mit der Anweisung VBAR der Prozedur GCHART erzeugt.

```
LIBNAME biblio 'C:\kurs';
PROC GCHART DATA=biblio.firma;
     VBAR wohnort;
RUN;QUIT;
```

❷ Kreisdiagramme erzeugt man mit der Prozedur GCHART und der Anweisung PIE. Falls Sie das Programm zur Erzeugung der Kur-Datei nicht gespeichert haben, müssen Sie dieses zunächst nochmal neu schreiben.

```
PATTERN1 V=SOLID C=BLUE;
PATTERN2 V=SOLID C=RED;
PROC GCHART DATA=kur;
     PIE gruppe;
RUN;
```

Die beiden PATTERN-Anweisungen sind nicht unbedingt erforderlich, aber sie geben der Grafik das „gewisse Etwas“.

❸ Auch hierzu wird die Prozedur GCHART eingesetzt, diesmal allerdings mit der Anweisung HBAR.

```
PROC GCHART DATA=kur;
     HBAR gew_vor / SUBGROUP=gruppe;
RUN;
```

Wenn Sie beim Kreisdiagramm die Farbmuster definiert hatten, werden diese auch im Balkendiagramm eingesetzt, da es sich um globale Anweisungen handelt, deren Wirkung erhalten bleibt (siehe Kapitel 10).

❹ Für das vertikale Balkendiagramm muß die Anweisung VBAR verwendet werden.

```
PROC GCHART DATA=kur;
     VBAR diff / GROUP=gruppe;
RUN;
```

❺ Das Streudiagramm wird mit der Prozedur GPLOT gebildet. Die Punkte werden mit zwei SYMBOL-Anweisungen festgelegt.

```
SYMBOL1 V=DOT C=BLUE;
SYMBOL2 V=DOT C=RED;
PROC GPLOT DATA=kur;
     PLOT gew_nach*gew_vor=gruppe;
RUN;
```

❻ Die zusätzliche Regressionsgerade kann nur dann in das Diagramm eingetragen werden, wenn auf die Unterscheidung der Punkte zwischen den beiden Gruppen verzichtet wird.

```
SYMBOL1 I=RL C=BLACK;
SYMBOl2 V=DOT C=RED;
PROC GPLOT DATA=kur;
    PLOT gew_nach*gew_vor gew_nach*gew_vor / OVERLAY;
RUN;
```

C.8 Kapitel 10

❶ Eine mögliche Beschriftung des Balkendiagramms könnte sein:

```
TITLE C=RED 'Frauen und Männer in der Firma';
FOOTNOTE C=BLUE 'So gut wie ausgeglichen';
```

❷ Der Titel könnte lauten:

```
TITLE C=GREEN 'Wohnorte unserer Mitarbeiter';
```

❸ Das SAS-Programm zum Anzeigen der Tour de France-Fahrer:

```
TITLE 'Tour de France 1997';
TITLE2 'Endplazierung';
OPTIONS LS=64 DATE NOCENTER;
LIBNAME biblio 'A:\aufgaben';
PROC PRINT DATA=biblio.tour97;
RUN;
```

❹ Das Kommando zur Eintragung der Zeilennummern heißt `Nums on`. Analog zu Kapitel 10 wird über *Options* → *Edit Tools* eine neue Schaltfläche definiert (*Add*, *Browse*). (*Save* am Ende nicht vergessen!)

Command	:	`Nums on`
Help Text	:	Zeilenkommandos einschalten
Tip Text	:	Nums on

D

Syntaxelemente

In diesem Abschnitt werden die im Laufe des Kurses behandelten globalen Anweisungen sowie die Anweisungen der Daten- und Prozedurschritte ohne Erläuterungen zusammengestellt.

D.1 Kommentare

```
* Dieser Kommentar gilt bis zum Semikolon;

/* In diesem Kommentar können Semikolons enthalten sein ;
   PROC HELP; z. B. gibt es nicht;
*/
```

D.2 Der Datenschritt

```
/* Rohwerte im Programmfenster */
DATA dateiname|_NULL_|datei1 datei2 ...;
     INPUT ...;
     LINES;
rohwerte
RUN;

/* Rohwerte in einer externen Textdatei */
DATA dateiname|_NULL_|datei1 datei2 ...;
```

```
INFILE 'Pfad+Dateiname'|referenzname|CARDS
      <MISSOVER> <PAD> <FIRSTOBS=n> <OBS=n>
      <LRECL=n>;
INPUT ...;
```

Die Anweisung INPUT kann folgende Variationen aufweisen:

```
/* Listengesteuertes Einlesen */
INPUT variable1 <$> <&> <:> variable2 ...;

/* Spaltengesteuertes Einlesen */
INPUT variable1 <$> n1-n2 variable2 ...;

/* Formatgesteuertes Einlesen */
INPUT @1 variable1 <$> einleseformat @i|+j variable2 ... ;

/* Mehrere Eingabezeilen pro Beobachtung */
INPUT #1 variable1 ...       #2 variable... ...;

/* Mehrere Beobachtungen pro Eingabezeile */
INPUT variable1 ... @@;
```

Zugriff auf bestehende SAS-Dateien

```
SET datei <datei2> ...;
MERGE datei1 datei2;
```

Weitere Anweisungen (in alphabetischer Reihenfolge):

```
DROP variablen-liste;
IF logische bedingung;
IF logische bedingung THEN anweisung;
   <ELSE anweisung;>
KEEP variablen-liste;
RENAME altvar1=neuvar1 altvar2=neuvar2 ...;
RETAIN variablen-liste;
```

D.3 Prozeduren

Prozeduren, die wie INSIGHT oder GOPTIONS ohne zusätzliche Optionen und Anweisungen aufgerufen werden, werden in dieser alphabetischen Zusammenstellung nicht berücksichtigt.

```
PROC CONTENTS DATA=Bibliotheksname.Datei|_ALL_
              <POSITION>
              <DIRECTORY>;

PROC CORR DATA=Datei <SPEARMAN> <KENDALL> <ALPHA>
           <NOPRINT> <OUTP=Datei> <OUTS=Datei>;
     VAR Variablen-Liste;
     <WITH Variablen-Liste;>

PROC FREQ DATA=Datei <ORDER=DATA|FORMATTED|FREQ>;
     TABLES Variablen-Liste </ <NOCOL> <NOROW> <NOPERCENT>
                    <EXPECTED> <DEVIATION> <CELLCHI2>
                    <CHISQ> <EXACT>;
     * TABLES (Variablen-Liste)*(Variablen-Liste);

PROC GCHART DATA=SAS-Datei;
     VBAR Variablen-Liste / <DISCRETE>
                   <MIDPOINTS=n1 n2 ...> <LEVELS=n>
                   <TYPE=FREQ|PERCENT|MEAN|SUM>
                   <SUMVAR=Variable> <ERRORBAR=TOP>
                   <GROUP=Variable> <SUBGROUP=Variable>;
     HBAR ...;
     PIE ...;

PROC GPLOT DATA=SAS-Datei;
     PLOT y-Variablen-Liste*x-Variablen-Liste
              </ <OVERLAY> <LEGEND> <REQEQN>>;

PROC MEANS DATA=Datei <FW=n> <MAXDEC=n> <NOPRINT>
           <N> <MEAN> <STD> <MIN> <MAX> <VAR> ...
           <CLM> <LCLM> <UCLM> <ALPHA=>
           <T> <PRT>;
     <CLASS Variablen-Liste;>
     <VAR Variablen-Liste;>
     <OUTPUT OUT=Datei;>

PROC NPAR1WAY DATA=datei <WILCOXON>;
     CLASS Variable;
     <VAR Variablen-Liste;>
     <EXACT WILCOXON;>
```

```
PROC PRINT <DATA=Datei> <NOOBS>;
     <VAR Variablen-Liste;>
     <WHERE where-Bedingung>;
     <SUM Variablen-Liste;>
     <BY Variablen-Liste;>

PROC SORT <DATA=Datei> <OUT=Datei>;
     BY <DESCENDING> Variablen-Liste;

PROC TTEST DATA=Datei;
     CLASS Variable;
     <VAR Variablen-Liste;>

PROC UNIVARIATE DATA=Datei <NOPRINT> <PLOT> <NORMAL>;
     <VAR Variablen-Liste;>
     <OUTPUT OUT=Datei Kennwert=Variablen-Liste <...>;>
```

D.4 Lokale Anweisungen

Die beiden folgenden Anweisungen können in jedem Datenschritt und in den meisten Prozedurschritten eingesetzt werden.

```
FORMAT variable formatname. variable2 formatname2. ...;
LABEL  variable='etikett' variable2='etikett2' ...;
```

D.5 Globale Anweisungen

```
OPTIONS  Option|NO-Option Option=Wert ...;
GOPTIONS Goption=Wert ...;

TITLE<1|2|...10> <<C=Farbe> <H=n <Einheiten>>
                <J=L|C|R> 'Text'> ... ;
FOOTNOTE<1|2|...10> <<C=Farbe> <H=n <Einheiten>>
                <J=L|C|R> 'Text'> ...;

AXIS<1|2|...> <LABEL=NONE>;
PATTERN<1|2|...> <V=EMPTY|SOLID|...> <C=Farbe>;
SYMBOL<1|2|...> <I=NONE|JOIN|RL> <V=DOT|STAR|...>
                <H=n> <C=Farbe>;
```

D.6 Kommandos

Kommando	Funktionalität
INClude	Programm im Programmfenster öffnen
FILe	aktuelles Fenster abspeichern
SUBmit	Programm ausführen
RECall	Programm zurückholen
CLEAR	aktuellen Fensterinhalt löschen
NUMS ON	Zeilenkommandos im Programmfenster einschalten
	In den Fenstern bewegen
BACKward	zurückblättern
FORWARD	vorblättern
RIght	nach rechts blättern
LEft	nach links blättern
TOP	an den Anfang springen
BOTTOM	an das Ende springen
	Zwischen den Fenstern wechseln
PGM	Programmfenster
LOG	Protokollfenster
OUTput	Ausgabefenster
MANAGER	Ausgabe-Manager
HELP	Online-Hilfe
KEYS	Funktionstastenbelegung
OPTIONS	Systemoptionen
DLGLIB	Libraries-Fenster
LIBname	Libname-Fenster
DIRectory	Directory-Fenster
Variables	Variablenfenster
END	Rückkehr zum Programmfenster
	Zur Betriebssystemebene wechseln
X	Wechsel, ohne die SAS-Umgebung zu verlassen
BYE	SAS-Sitzung beenden
	Modul-spezifische Kommandos
ASSist	SAS/ASSIST
ADD	neue Einträge in FSEDIT-Masken aufnehmen
INSight	SAS/INSIGHT
BUIld	Prozedur BUILD
AUTOSPLIT	Trennen mit ENTER in SCL-Einträgen
AF	Ausführung einer Anwendung

Kommando	Funktionalität
	Kommandos in Selektionsspalten von Fenstern
R	Umbenennen des Eintrags
D	Löschen des Eintrags
V	Bestätigen der Löschaktion
C	Abbrechen der Löschaktion
	Zeilenkommandos
COLS	Zeilenlineal einschalten
I	Leerzeile einfügen
C, M	Zeile kopieren bzw. verschieben
A, B, O	nach/vor der aktuellen Zeile bzw. diese überlagern
D	Zeile löschen
JC, JL, JR	Zeile ausrichten
TC, TS	Zeilen verbinden bzw. teilen
$>, <$	Zeilen nach rechts bzw. links verschieben

E

Fehlermeldungen

Sobald man mit dem SAS-System arbeitet, gleichgültig ob als Anfänger oder als „alter Hase“, produziert man Fehler, die Hinweise, Warnungen und Fehlermeldungen im Protokollfenster zur Folge haben können. Manchmal wird dem Anwender in der Fehlermeldung sofort ein Lösungsweg aufgezeigt. In anderen Fällen, besonders dann, wenn Folgefehler auftauchen, ist die Ursache nicht sofort einsichtig. Anfänger und fortgeschrittene Anwender unterscheiden sich nicht selten vor allem darin, daß der Fortgeschrittene bereits Strategien entwickelt hat, mit den Fehlermeldungen des Systems umzugehen. Dieser Abschnitt will die wichtigsten Fehlermeldungen und deren Ursachen sowie einige Lösungsansätze aufzeigen.

Die Daten- und Prozedurschritte sowie die globalen Anweisungen, aus denen sich ein SAS-Programm zusammensetzt, werden sukzessive von oben nach unten abgearbeitet. Erst wenn ein Daten- oder Prozedurschritt abgearbeitet ist, kommt der nächste an die Reihe. Das SAS-System kann dabei zwei Arten von Fehlern erkennen: Syntaxfehler und Datenfehler. Fehler in der Logik des Programmaufbaus dagegen kann das System nicht erkennen, es sei denn, diese führen wiederum zu Syntax- oder Datenfehlern.

E.1 Syntaxfehler

Bevor ein Programm ausgeführt wird, gleichgültig, ob es ein Daten- oder ein Prozedurschritt ist, wird dessen Syntax überprüft. Es wird ermittelt, ob die verwendeten Anweisungen und Optionen korrekt geschrieben und ob deren Positionen zulässig sind.

Bei einigen Schlüsselwörtern kann das SAS-System Schreibfehler erkennen. Es führt die Prozedur oder den Datenschritt mit der vermeintlich richtigen Anweisung oder Option aus. Als Hinweis für den Anwender wird im Protokollfenster eine Warnung ausgesprochen:

```
2    PORC PRINT DATA=biblio.firma;
     ----
     14
3    RUN;

WARNING 14-169: Assuming the symbol PROC was misspelled as
                PORC.
```

Der Fehler wird unterstrichen und mit einer Nummer versehen: 14. Auf diese Nummer bezieht sich dann die unten angefügte Warnung 14-169. Der Fehler wird behoben und der korrigierte Prozedurschritt ausgeführt.

Anders sieht es aus, wenn das System kein passendes Schlüsselwort finden kann, wie im nächsten Beispiel:

```
4   PROC PRINT DATA=biblio.firma;
5        BAR nr;
         ---
         180
6   RUN;
ERROR 180-322: Statement is not valid or it is used out of
               proper order.
NOTE: The SAS System stopped processing this step because of
      errors.
```

Die Anweisung BAR ist im Zusammenhang mit der Prozedur PRINT unbekannt, weshalb das System folgende (hier übersetzte) Fehlermeldung ausgibt: Die Anweisung ist ungültig (im Zusammenhang mit dieser Prozedur) oder sie steht an der falschen Stelle. Wieder wird die betreffende Anweisung unterstrichen, mit einer Nummer versehen und die Fehlermeldung ausgegeben. Der Prozedurschritt wird abgebrochen.

Eine andere Fehlermeldung tritt auf, wenn man alle Anweisungen und Optionen korrekt eingibt, aber ein Semikolon vergißt. (Dies ist einer der häufigsten Fehler.)

```
7   PROC PRINT DATA=biblio.firma
8         VAR nr;
          --- --
          202 202
9   RUN;

ERROR 202-322: The option or parameter is not recognized.

NOTE: The SAS System stopped processing this step because of
      errors.
NOTE: The PROCEDURE PRINT used 0.55 seconds.
```

Das Semikolon fehlt in der ersten Programmzeile, am Ende der PROC PRINT-Anweisung. Als falsch wird jedoch die VAR-Anweisung markiert. Die Meldung lautet, daß die (unterstrichenen) Optionen oder Paramter unbekannt sind.

Nun, was ist passiert? Das SAS-System hält sich ja bekanntlich nicht an die Zeilen, sondern liest solange weiter, bis ein Semikolon auftaucht. Die Anweisung PROC PRINT endet daher erst mit der 2. Zeile. Die Begriffe VAR und nr werden als Optionen der Anweisung interpretiert. „Völliger Blödsinn“, denken Sie jetzt sicherlich. Aber dem SAS-System fehlt es hier an der notwendigen Intelligenz zur Behebung des Fehlers.

Jetzt gilt es für den Anwender, selbstbewußt zu sein und sich nicht verwirren zu lassen. Wenn Sie davon überzeugt sind, daß die Anweisung VAR nr; korrekt ist, dann richten Sie Ihren Blick nach oben, in die Zeile vorher, und prüfen, ob das letzte Zeichen vor VAR ein Semikolon ist. Falls Sie keins finden, dann fügen Sie es ein, und das Problem sollte behoben sein.

```
10    PROC PNT DATA=biblio.firma;
ERROR: Procedure PNT not found.
11    RUN;

NOTE: The SAS System stopped processing this step because of
      errors.
```

Auch in diesem Fall kann der Prozedurschritt nicht ausgeführt werden: Der Prozedurname PNT ist unbekannt. Diese Meldung tritt auch dann auf, wenn es die Prozedur im Prinzip gibt, aber auf Ihrem Rechner nicht installiert wurde.

Werden in einem Prozedurschritt falsche Variablennamen angegeben oder die Namen nur fehlerhaft geschrieben, dann erscheint eine Fehlermeldung der folgenden Gestalt:

```
12   PROC PRINT DATA=biblio.firma;
13        VAR pnr;
ERROR: Variable PNR not found.
14   RUN;

NOTE: The SAS System stopped processing this step because of
      errors.
```

Die Variable `pnr` konnte nicht in der Datei `biblio.firma` gefunden werden, weshalb der komplette Prozedurschritt abgebrochen wird.

Wird ein falscher Dateiname angegeben, reagiert das System mit folgender Meldung:

```
15   PROC PRINT DATA=biblio.firms;
ERROR: File BIBLIO.FIRMS.DATA does not exist.
16   RUN;

NOTE: The SAS System stopped processing this step because of
      errors.
```

In diesem Beispiel wurde der Dateiname falsch geschrieben. Die Meldung tritt aber auch auf, wenn die Datei in einer anderen Bibliothek abgelegt wurde. Wird dazu im Unterschied der Bibliotheksname falsch geschrieben, dann meldet das System:

```
17   PROC PRINT DATA=bilbio.firms;
ERROR: Libname BILBIO is not assigned.
18   RUN;

NOTE: The SAS System stopped processing this step because of
      errors.
```

Werden der Dateiname und die Option DATA= ganz vergessen, müssen zwei Fälle unterschieden werden. Entweder es tritt folgende Fehlermeldung auf:

```
12   PROC PRINT;
ERROR: There is not a default input data set
       (_LAST_ is _NULL_).
13        VAR nr;
14   RUN;

NOTE: The SAS System stopped processing this step because of
      errors.
```

Diese Meldung besagt, daß im Laufe dieser Sitzung keine Datei erzeugt wurde, weshalb die Systemvariable _LAST_ den Wert _NULL_ hat. Oder, im zweiten Fall, wenn zuvor eine Datei erzeugt wurde, wird die Prozedur mit dieser Datei ausgeführt, ohne Fehlermeldung zwar, aber vielleicht nicht mit dem Resultat, das Sie erwartet haben.

Viele der bisher angeführten Fehlermeldungen treten auch im Datenschritt auf: unbekannte Bibliothek oder Datei, falsche Anweisung oder eine Anweisung an der falschen Stelle. Wird jedoch eine falsche Variable benannt, dann erscheint im Protokollfenster folgender Hinweis:

```
15   DATA NEU;
16        SET biblio.firma;
17        gesamt=teil1+teil2;
18   RUN;

NOTE: Variable TEIL2 is uninitialized.
NOTE: Missing values were generated as a result of
      performing an operation on missing values. Each place
      is given by: (Number of times) at (Line):(Column).
      31 at 17:18
NOTE: The data set WORK.NEU has 31 observations and 16
      variables.
```

Die Variable `teil2` ist unbekannt, ihr wurde bisher noch kein Wert zugewiesen (`uninitialized`). Der Datenschritt wird nun aber nicht abgebrochen, sondern die Variable `teil2` wird implizit neu angelegt. Sie wird mit fehlenden Werten besetzt (=initialisiert). Die Summe von `teil1` und `teil2` kann gebildet werden. Allerdings, und das besagt die nächste Meldung, kann die Gesamtpunktzahl bei allen `31` Beobachtungen an der Stelle `17:18` nicht gebildet werden. `17` bezieht sich dabei auf die Zeilennummer des Programms, `18` auf die Spalte. Die betreffende Operation ist die Addition (+).

Ein vergessenes Semikolon im Datenschritt kann erstaunliche, aber auch katastrophale Folgen haben, wie das nächste Beispiel zeigt.

```
19   DATA NEU
20        SET biblio.firma;
21        gesamt=teil1+teil2a;
22   RUN;

NOTE: Variable TEIL2A is uninitialized.
NOTE: Variable TEIL1 is uninitialized.
NOTE: Missing values were generated as a result of
      performing an operation on missing values. Each place
      is given by: (Number of times) at (Line):(Column).
      1 at 21:18
NOTE: The data set WORK.NEU has 1 observations and 3
      variables.
NOTE: The data set WORK.SET has 1 observations and 3
      variables.
NOTE: The data set BIBLIO.FIRMA has 1 observations and 3
      variables.
```

Die Anweisung DATA wurde nicht mit einem Semikolon abgeschlossen, weshalb das System die SET-Anweisung nicht erkennt und somit die beiden Variablen `teil1` und `teil2a` nicht finden kann. Sie müssen neu initialisiert werden (mit den entsprechenden Meldungen).

Anschließend kommen die zunächst kurios erscheinenden, dann aber erschreckenden Meldungen. Es wurden drei Dateien angelegt: `neu`, `set` und, das ist das eigentlich tragische, `biblio.firma` mit jeweils einer Beobachtung und 3 Variablen. Die schöne Firmendatei wurde ohne Warnung überschrieben und damit zerstört. Nicht passiert wäre dies, wenn die Systemoption NOREPLACE aktiviert worden wäre:

```
OPTIONS NOREPLACE;
```

Im Protokollfenster erschiene daraufhin die zusätzliche Zeile

```
WARNING: Data set BIBLIO.FIRMA not replaced because of
         NOREPLACE option.
```

Die permanente SAS-Datei `biblio.firma` behält mit der Option NOREPLACE ihre alte Form. Aber diese Option hilft nur, wenn man nicht im nächsten Schritt

doch die Datei ändern möchte, denn sonst muß man ständig die Option REPLACE ein- bzw. ausschalten.

Wenn das Vergessen eines Semikolons der häufigste Fehler beim Arbeiten mit dem SAS-System ist, dann ist der zweithäufigste Fehler das Vergessen eines Anführungszeichens. Verzeichnisse und komplette Dateinamen in den Anweisungen FILENAME, LIBNAME und INFILE müssen genauso wie Ausprägungen von Textvariablen in Anführungszeichen eingeschlossen werden. Ob einfache oder doppelte Anführungszeichen verwendet werden, spielt keine Rolle. Wichtig ist nur, daß man jeweils die gleichen am Anfang und Ende der Zeichenkette verwendet.

```
27   LIBNAME biblio 'C:\kurs;
28   PROC PRINT DATA=biblio.firma;
29   RUN;
```

Unterläuft einem dieser Fehler bei einer Anweisung wie LIBNAME, dann merkt man an der unterbliebenen Rückmeldung über den erfolgten Zugriff auf die Bibliothek, bzw. die Meldung, daß das Verzeichnis nicht existiert, daß hier etwas schiefgelaufen sein muß. Auch nach dem Prozedurschritt wird keine Datei im Ausgabefenster dargestellt, und auch die Meldung über die beanspruchte Zeit erscheint nicht.

In dieser Situation jetzt einfach das Programm zurückzuholen und erneut auszuführen, bringt nicht die Lösung, sondern nur weitere Fehlermeldungen:

```
30   LIBNAME biblio 'C:\kurs;
                    -
                    22
ERROR: Libname BIBLIO is not assigned.
ERROR: Error in the LIBNAME or FILENAME statement.
ERROR 22-7: Invalid option name C.
31   PROC PRINT DATA=biblio.firma;
ERROR: Libname BIBLIO is not assigned.
32   RUN;
NOTE: The SAS System stopped processing this step because of
      errors.
NOTE: The PROCEDURE PRINT used 0.44 seconds.
```

Das neue Anführungszeichen in Zeile 30 schließt das erste aus Zeile 27 ab. Da die Zeichenkette innerhalb der beiden Anführungszeichen aber nicht als gültiges Verzeichnis ausgemacht werden konnte, kann die Bibliothek biblio nicht zugeordnet werden. Der in dieser LIBNAME-Anweisung auftretende Fehler wird

allerdings erst beim nächsten Zeichen angezeigt, dem eigentlichen Verzeichnisnamen. `C:\kurs` wird als Option betrachtet und das System versucht, es entsprechend zu interpretieren.

Was jetzt hilft, ist die LIBNAME-Anweisung mit dem vergessenen Anführungszeichen zu komplettieren und auszuführen. Die gewünschte Rückmeldung über die Zuordnung wird dabei erscheinen.

War Ihnen das Fehlen des Anführungszeichen dagegen sofort aufgefallen, hilft es nicht, es in der Anweisung LIBNAME zu korrigieren und das korrigierte Programm auszuführen: Sie werden keinerlei Rückmeldungen vom SAS-System erhalten.

```
27   LIBNAME biblio 'C:\kurs;
28   PROC PRINT DATA=biblio.firma;
29   RUN;
30   LIBNAME biblio 'C:\kurs';
31   PROC PRINT DATA=biblio.firma;
32   RUN;
33   LIBNAME biblio 'C:\kurs';
34   PROC PRINT DATA=biblio.firma;
35   RUN;
```

Warum? Die erste Zeichenkette enthält den Text
„`C:\kurs; PROC PRINT DATA=biblio.firma; RUN; LIBNAME biblio`“.
Dann folgt das Verzeichnis „`C:\kurs`“, danach die 2. Zeichenkette „`PROC PRINT ...`“. Die Anweisung LIBNAME wird nicht mit einem Semikolon beendet, weshalb das System diese Anweisung nicht überprüfen kann.

Um nun aber doch zum Ziel zu kommen, muß die Zeichenkette mit `';` (Zeile 30 im folgenden Protokollfenster) geschlossen werden. Danach kann das korrigierte Programm ausgeführt werden.

```
27  LIBNAME biblio 'C:\kurs;
28  PROC PRINT DATA=biblio.firma;
29  RUN;
30  ';
NOTE: Library BIBLIO does not exist.
31  LIBNAME biblio 'C:\kurs';
NOTE: Libref BIBLIO was successfully assigned as follows:
      Engine:        V612
      Physical Name: C:\kurs
```

Analog dazu gehen Sie vor, wenn Sie doppelte und einfache Anführungszeichen verwechselt haben. Zuerst schließen Sie die zuletzt geöffnete Zeichenkette, dann die verbleibende (Zeile 37 im folgenden Protokollfenster).

```
34  LIBNAME biblio 'C:\kurs";
35  PROC PRINT DATA=biblio.firma;
36  RUN;
37  ";';
NOTE: Library BIBLIO does not exist.
38  LIBNAME biblio 'C:\kurs';
NOTE: Libref BIBLIO was successfully assigned as follows:
      Engine:        V612
      Physical Name: C:\kurs
39  PROC PRINT DATA=biblio.firma;
40  RUN;

NOTE: The PROCEDURE PRINT used 0.44 seconds.
```

Während man beim Ausführen einer LIBNAME-Anweisung sofort eine Rückmeldung vom System erhält und das Vergessen eines Anführungszeichens somit leicht entdeckt werden kann, ist dies bei FILENAME-Anweisungen oder bei Zuweisungen im Datenschritt nicht so einfach ersichtlich. Ein wichtiges Indiz dafür ist die die folgende Fehlermeldung:

```
41  DATA neu;
41        INPUT nr @5 text $40.;
42        gruppe='SAS;
43        LINES;
44  0001 Frau Gabriele Mayer, Psychologie, 1. Semester
45  0002 Herr Ludger Scholz, VWL, 6. Semester
46  0003 Frau Sabine Annabauer, Psychologie, 3. Semester
47  0004 Frau Gabriele Mayer, Psychologie, 1. Semester
WARNING: The current word or quoted string has become more
         than 200 characters long.  You may have unbalanced
         quotation marks.
NOTE: A single quote (') will terminate this quoted string.
48  0005 Herr Ludger Scholz, VWL, 6. Semester
49  0006 Frau Sabine Annabauer, Psychologie, 3. Semester
50  ;
51  PROC PRINT DATA=neu;
52  RUN;
```

In der 42. Programmzeile wurde das schließende Anführungszeichen vergessen. Das SAS-System erkennt nun nicht die nachfolgende LINES-Anweisung und die Datenwerte, sondern interpretiert alle Zeichen als zur Variablen gruppe gehörig. Da Textvariablen maximal 200 Zeichen lang sein können, liest das System die nächsten 200 Zeichen ein und bricht dann ab mit obiger Warnung. Gleichzeitig wird die Vermutung geäußert, daß der Fehler mit einem Ungleichgewicht von Anführungszeichen zusammenhängen könnte. Der auf die Warnung folgende Hinweis gibt zusätzlich noch an, welche Art von Anführungszeichen unausgewogen ist: in obigem Fall das einfache. Wenn man das Programmfenster genau betrachtet, fällt auch auf, daß der Datenschritt noch nicht beendet wurde: *DATA STEP running.*

Genau wie oben bei der LIBNAME-Anweisung hilft es auch hier nicht, das Programm zu korrigieren und neu auszuführen. Zunächst muß der offene Textstring geschlossen und der Datenschritt beendet werden.

```
53  ';
54  RUN;

ERROR: No CARDS or INFILE statement.
NOTE: The SAS System stopped processing this step because of
      errors.
WARNING: The data set WORK.NEU may be incomplete.  When this
         step was stopped there were 0 observations and 3
         variables.
WARNING: Data set WORK.NEU was not replaced because this
         step was stopped.
NOTE: The DATA statement used 12.41 seconds.
```

Das Semikolon schließt die Zeichenkette, RUN den Datenschritt ab. Da das SAS-System weder eine INFILE- noch eine CARDS- bzw. LINES-Anweisung finden konnte, wurde der Datenschritt abgebrochen. Jetzt kann das Programm korrigiert und erneut ausgeführt werden:

```
55  DATA neu;
56       INPUT nr @5 text $40.;
57       gruppe='SAS';
58       LINES;

NOTE: The data set WORK.NEU has 3 observations and 3
      variables.
NOTE: The DATA statement used 1.42 seconds.
```

E.2 Datenfehler

Datenfehler können, von wenigen Ausnahmen abgesehen, nur in Datenschritten auftreten, da nur an dieser Stelle externe Daten in das SAS-System überführt werden und Diskrepanzen zwischen Dateibeschreibung und Datenwerten auftreten können.

Da die wichtigsten in Frage kommenden Fehler bereits im Kapitel 5 ausführlich besprochen wurden, werden hier nur die Fehlermeldungen und ihre Ursachen zusammengestellt. Die Lösungsmöglichkeiten werden angesprochen, für eine ausführlichere Erläuterung wird allerdings auf Kapitel 5 verwiesen.

Man kann zwei Arten von Datenfehlern unterscheiden: falsche Deklaration des Variablentyps und fehlerhafte Wahl des Einlesevorgangs. Wird eine Textvariable nicht als solche deklariert, erscheint die Meldung:

```
NOTE: Invalid data for VARIABLE in line x xx-xx.
```

Der Datenschritt wird ausgeführt, allerdings wird die betreffende Variable mit einem fehlenden Wert belegt. Ein $-Zeichen in der INPUT-Anweisung nach dem Variablennamen beseitigt das Problem.

Der folgende Hinweis deutet daraufhin, daß das SAS-System beim Übertragen der Datenwerte über das Ende einer Datenzeile hinausgeraten ist und in der nächsten Zeile mit der Aktion fortfährt.

```
NOTE: SAS went to a new line when INPUT statement reached
      past the end of a line.
```

Wurde der doppelte Zeilenhalter verwendet, hat diese Meldung nichts weiter zu bedeuten. Bei unvollständigen ASCII-Dateien, die mittels INFILE eingelesen werden, kann diese Meldung allerdings anzeigen, daß ein Fehler aufgetreten ist. Dieser läßt sich mit den Optionen MISSOVER und PAD umgehen.

F

Glossar

In diesem Glossar werden wichtige Begriffe, die in den Handbüchern und der Online Dokumentation von SAS Institute sowie in diesem Buch vorkommen, zusammengefaßt.

alphanumerische Variable siehe Textvariable

Anweisung Bestandteil eines SAS-Programms. Eine Anweisung beginnt mit einem Schlüsselwort und endet mit einem Semikolon. Anweisungen, die nur innerhalb von Daten- und Prozedurschritten eingesetzt werden können, werden als lokale Anweisungen bezeichnet. Unter globalen Anweisungen versteht man Anweisungen, die an jeder beliebigen Stelle im Programm erscheinen können.

Ausgabefenster Standardfenster des SAS-Systems, in dem Prozedurresultate angezeigt werden.

Beobachtung Unter einer Beobachtung versteht man Personen, Probanden oder allgemein Versuchsobjekte, für die in der Datei Datenwerte eingetragen werden. Wenn man sich eine Datei als Tabelle vorstellt, dann bilden die Beobachtungen die Zeilen dieser Tabelle.

Bibliothek Unter einer Bibliothek versteht man alle SAS-Dateien und -Kataloge, die sich innerhalb einer physikalischen Rechnereinheit befinden. Auf der PC-Ebene entspricht diese Rechnereinheit einem Verzeichnis.

data set siehe SAS-Datei

data value siehe Datenwert

Datenschritt Ein Datenschritt beginnt mit dem Schlüsselwort DATA. Diesem folgen Dateinamen sowie Anweisungen und Optionen entsprechend der jeweiligen Syntax. Mit einem Datenschritt werden SAS-Dateien angelegt, bestehende Dateien verändert oder Datenschrittanweisungen ausgeführt.

Datenwert Ausprägung einer Variablen einer bestimmten Beobachtung. Stellt man sich eine Datei als Tabelle vorstellt, sind die Datenwerte die einzelnen Zellen dieser Tabelle.

Dateioption Option, die in Klammern eingeschlossen im Anschluß an den Namen einer Datei erscheint, z. B. um eine Aktion auf bestimmte Beobachtungen oder Variablen einzuschränken.

Etikett Bezeichnung, die den (auf 8 Zeichen beschränkten) Variablennamen ersetzt. Das bis zu 40 Zeichen lange Etikett (oder Label) kann in einem Datenschritt permanent oder in einem Prozedurschritt nur für diesen Programmschritt mit der Variablen verknüpft werden.

Format Bezeichnung für die Datenwerte einer Variablen. Das SAS-System unterscheidet zwischen Einleseformaten für das Erstellen von Dateien und Ausgabeformaten für die Darstellung sowie zwischen numerischen und alphanumerischen Formaten für numerische bzw. Textvariablen.

Kommando Befehl, um eine Aktion auszulösen. Ein Kommando wird über die Kommandozeile, das Menü, eine Funktionstaste oder die Tools-Leiste an das SAS-System übergeben.

Label siehe Etikett

library siehe Bibliothek

numerische Variable Variable, deren Werte nur Zahlenwerte sind.

observation siehe Beobachtung

permanent SAS-Dateien und -Kataloge heißen permanent, wenn sie nicht in der SASWORK-Bibliothek abgelegt sind. Sie werden nach Beendigung der SAS-Sitzung nicht automatisch gelöscht.

Programmfenster Standardfenster des SAS-Systems, in dem Programme geöffnet oder in das Programme geschrieben werden.

Protokollfenster Standardfenster des SAS-Systems, in dem Hinweise und Fehlermeldungen zu ausgeführten Programmen angezeigt werden.

Prozedur Vorgefertigte Routine zur Ausführung spezieller Aufgaben. Der Aufruf einer Prozedur erfolgt entsprechend ihrer festgelegten Syntax.

Prozedurschritt Bestandteil eines SAS-Programms. Ein Prozedurschritt beginnt mit dem Schlüsselwort PROC gefolgt vom Prozedurnamen sowie Anweisungen und Optionen entsprechend der Syntax.

SAS-Datei Eine SAS-Datei ist eine in einer bestimmten Form abgelegte Datei (auf dem PC mit der Endung *.SD2), die nur vom SAS-System selbst gelesen werden kann. Sie enthält die Dateibeschreibung (Name und Typ der Variablen, evtl. auch Etiketten und Formate) und die Datenwerte. Für jede Beobachtung liegt genau ein Wert pro Variable vor. Es werden temporäre und permanente SAS-Dateien unterschieden.

SAS-Katalog Ein SAS-Katalog ist eine in einer bestimmten Form abgelegte Datei (auf dem PC mit der Endung *.SC2), die Anwendungsentwicklungen, Eingabemasken, Grafiken, Formate, Tastenbelegungen, Tools und anderes mehr enthält.

SAS-Programm Aus Daten- und Prozedurschritten sowie globalen Anweisungen bestehender Text (oder Textdatei), der zur Ausführung an das System übergeben wird.

SASWORK Bibliothek der temporären SAS-Dateien und SAS-Kataloge.

Setinit Lizenzcode, der jährlich eingespielt werden muß, bevor das System gestartet werden kann.

Schlüsselwort Ein Schlüsselwort leitet eine Anweisung ein. Die Syntax eines Daten- oder Prozedurschritts bestimmt, welche Schlüsselwörter jeweils zulässig sind.

Screen Control Language (SCL) SCL ist eine SAS-spezifische Programmiersprache zur Steuerung von FSEDIT-Eingabemasken und SAS/AF-Anwendungen.

statement siehe Anweisung

string Zeichenkette, die neben Zahlen auch Buchstaben und Sonderzeichen enthalten kann.

Structured Query Language (SQL) SQL ist eine generelle Abfragesprache für Datenbanksysteme, also nicht SAS-spezifisch.

Syntax Die korrekte Schreibweise, Zulässigkeit und Abfolge von Anweisungen, Schlüsselwörtern und Optionen in Daten- und Prozedurschritten und globalen Anweisungen.

temporär SAS-Dateien und -Kataloge heißen temporär, wenn sie in der SASWORK-Bibliothek abgelegt sind und nach Beendigung der Sitzung gelöscht werden.

Textvariable Variable, deren Werte Buchstaben, Ziffern und Sonderzeichen bis zu einer Länge von 200 Zeichen enthalten können. Sie wird auch als alphanumerische oder Stringvariable bezeichnet.

Transportdatei Datei, die dem Austausch systemspezifischer Dateien zwischen verschiedenen Anwendungsprogrammen oder unterschiedlichen Versionen eines Programms dient.

Variable Merkmal, das für alle Beobachtungen in der Datei eingetragen wird. Wenn man sich eine Datei als Tabelle vorstellt, dann bilden die Variablen die Spalten dieser Tabelle. Das SAS-System unterscheidet zwischen numerischen und alphanumerischen Variablen. Letztere, auch Textvariablen genannt, können im Unterschied zu den numerischen Variablen auch Buchstaben und Sonderzeichen, wie ? und $, enthalten.

Literaturverzeichnis

[1] Bortz, J. (1993): *Statistik für Sozialwissenschaftler*, Springer-Lehrbuch, 4. Auflage. Springer-Verlag, Berlin Heidelberg New York.

[2] Geißler, H., Ortseifen, C. (1996): *Datenaustausch mit SAS*, Skriptum des Universitätsrechenzentrums Heidelberg. (`http://web.urz.uni-heidelberg.de/Dokumentation/SAS/sas-datenaustausch.ps`)

[3] Hartung, J. (1991): *Statistik*. Oldenbourg Verlag, München.

[4] Hoaglin, D.C., Mosteller, F., Tukey, J.W. (1983): *Understanding robust and exploratory data analysis*. John Wiley & Sons, New York.

[5] SAS Institute Inc. (1996): *SAS Companion for the Microsoft Windows Environment, Version 6, Second Edition*. Cary, NC: SAS Institute Inc.

[6] SAS Institute Inc. (1993): *SAS Companion for the UNIX Environments: Language, Version 6, First Edition*. Cary, NC: SAS Institute Inc.

[7] SAS Institute Inc. (1995): *SAS Guide to the REPORT Procedure: Reference, Release 6.11*. Cary, NC: SAS Institute Inc.

[8] SAS Institute Inc. (1989): *SAS Guide to the SQL Procedure: Usage and Reference, Release 6, First Edition*. Cary, NC: SAS Institute Inc.

[9] SAS Institute Inc. (1990): *SAS Language: Reference, Version 6, First Edition*. Cary, NC: SAS Institute Inc.

[10] SAS Institute Inc. (1997): *SAS Macro Language: Reference, First Edition*. Cary, NC: SAS Institute Inc.

[11] SAS Institute Inc. (1990): *SAS Procedures Guide, Version 6, Third Edition*. Cary, NC: SAS Institute Inc.

[12] SAS Institute Inc. (1990): *SAS/GRAPH Software: Reference, Version 6, First Edition, Volume 1 and 2*, Cary, NC: SAS Institute Inc.

[13] SAS Institute Inc. (1991): *SAS/GRAPH Software: Usage, Version 6, First Edition, Volume 1 and 2*, Cary, NC: SAS Institute Inc.

[14] SAS Institute Inc. (1989): *SAS/IML Software: Usage and Reference, Version 6, First Edition*. Cary, NC: SAS Institute Inc.

[15] SAS Institute Inc. (1993): *SAS/INSIGHT User's Guide, Version 6, Second Edition*. Cary, NC: SAS Institute Inc.

[16] SAS Institute Inc. (1989): *SAS/STAT User's Guide, Version 6, Fourth Edition, Volume 1 and 2*. Cary, NC: SAS Institute Inc.

[17] SAS Institute Inc. (1996): *SAS/STAT Software, Changes and Enhancements through Release 6.11*. Cary, NC: SAS Institute Inc.

Abbildungsverzeichnis

Tabellenverzeichnis

Stichwortverzeichnis