5 Aufbau und Beschreibung eines SAS-Auswertungsprogrammes

Jedes SAS-Programm beginnt mit einem Dateneingabeschritt (= **data step**) zum Erzeugen und Verändern eines SAS-Datenfiles und dann folgen Arbeitsschritte (= **proc steps**), die die Daten der SAS-Datei verarbeiten, d.h. der **data**-Step stellt die Daten für die Auswertung im **proc**-Step bereit.

Ein **data**-Step wird mit dem Schlüsselwort DATA und ein **proc**-Step mit dem Schlüsselwort PROC begonnen. Danach folgen noch weitere Angaben, die SAS-Statements genannt werden. Näheres dazu, siehe "*2.4 SAS-Statements*".

Der **data**-Step teilt SAS mit wo die Daten zu finden sind, wie viele Variablen eingelesen werden sollen, wie sie heißen und welchem Datentyp sie angehören (numerisch oder alphanumerisch), ob neue Variablen definiert oder Variablenausprägungen zusammengefasst wurden und ob Datensätze (über Bedingungen) von der Auswertung ausgeschlossen werden sollen.

Die Gesamtheit aller im SAS-Programm aufgeführten Anweisungen bezeichnet man als Programmcode. Dieser ist einfacher zu überblicken, wenn man zusammengehörende Programmteile einrückt und die Befehle damit strukturiert.

Es empfiehlt sich auch Kommentarzeilen mit kurzen Erklärungen einzusetzen. Einzeiligen Kommentaren wird ein Stern vorangestellt und ein Semikolon angehängt. Mehrzeilige Kommentare beginnen mit den beiden Zeichen /* und enden mit den beiden Zeichen */. Die Programmzeilen dazwischen heißen Kommentarzeilen.

Beispiele:

einzeiliger Kommentar	**mehrzeiliger Kommentar**
***** Alter in Gruppen fassen **;**	/***** PROC PRINT ; VAR nr alter geschlecht ; RUN; ***** /

Mit Hilfe von Kommentaren kann man auch Programmbefehle aus der Auswertung ausschließen. SAS überliest Text oder Anweisungen, die als Kommentar gekennzeichnet wurden.

5.1 Wie weiß SAS, wo die Daten zu finden sind?

Das Einlesen der Daten erfolgt im **data**-Step. Es hängt davon ab, wie die Daten vorliegen. Diese können direkt in das Auswertungsprogramm eingegeben werden, als externes Datenfile (ASCII-Code) oder bereits als SAS-Datenfile (.sas7bdat) vorliegen.

5.1.1 Daten sind direkt im SAS Programm enthalten

Bei der **direkten Dateneingabe**, gibt man die Daten in das SAS-Auswertungsprogramm ein. Der Beginn der Dateneingabe wird durch das DATALINES-Statement gekennzeichnet und das Ende durch ein Semikolon angezeigt.

Die zum Einlesen notwendige INPUT-Anweisung übergibt SAS die Variablennamen, teilt dem Programm mit welchem Datentyp (numerisch oder alphanumerisch) die Daten angehören und wie sie eingegebenen wurden. Hat man die Variablenausprägungen mit Buchstaben oder Buchstabenzahlenkombinationen beschrieben, muß zwischen Variablennamen und Spaltenangabe ein Dollar-Zeichen ($) eingetippt werden (siehe *3.1 Der INPUT-Befehl*):

Beispiel: `INPUT nr **geschlecht $** groesse gewicht alter ;` .

Die direkte Dateneingabe sollte man nur bei kleinen Datenmengen verwenden, da das Auswertungprogramm sonst sehr unübersichtlich wird.

Beispiel:

```
* Speicherort für die SAS-Daten festlegen ;
LIBNAME name 'festplatte:\ordner' ;
* Daten einlesen ;
DATA name.daten ;
* Daten ;
INPUT  nr  geschlecht  groesse  gewicht  alter ;
* Definition neuer Variablen ;
⋮
DATALINES;
1  m  180  70  25
2  w  175  60  32
3  w  160  50  23
4..m  185  95  42
;
RUN;
* Datenauswertung ;
SAS - Prozeduren zur Datenauswertung
```

Soll das SAS-Datenfile nicht permanent sondern nur als Arbeitsdatei (SAS löscht die Daten beim Beenden) angelegt werden, entfällt das LIBNAME-Statement und die DATA-Anweisung lautet **DATA name ;**.

5.1.2 Daten liegen als externe Daten in einem ".dat-File" vor

Bei dieser Methode erfolgt die **Dateneingabe in einer separaten Datei** (ASCII-Datei mit der Extension .dat) auf die im zugehörigen Auswertungsprogramm mit der SAS-Anweisung **INFILE 'DATEINAME' ;** ein Hinweis gemacht werden muß.

Die INPUT-Anweisung übergibt SAS die Variablennamen, teilt SAS mit welchem Datentyp (numerisch oder alphanumerisch) die Daten angehören und wie sie eingegebenen wurden. Hat man die Variablenausprägungen mit Buchstaben oder Buchstabenzahlenkombinationen beschreiben, muß zwischen Variablennamen und Spaltenangabe ein Dollar-Zeichen ($) eingetippt werden, z.B. `INPUT nr **geschlecht $** groesse gewicht alter ;` (siehe *3.1 Der INPUT-Befehl*) .

Beispiel:

```
* Speicherort für die SAS-Daten festlegen ;
LIBNAME name 'festplatte:\ordner' ;
* Daten einlesen ;
DATA name.daten ;
* Daten aus der externen Datendatei einlesen ;
INFILE 'E:\doktorarbeit\sas\name.DAT' ;
* Datenbeschreibung laut Datenbogen ;
INPUT  nr  geschlecht  groesse  gewicht  alter ;
* Definition neuer Variablen ;
⋮
* Datenauswertung ;
SAS - Prozeduren zur Datenauswertung
```

Möchte man das SAS-Datenfile nicht permanent sondern nur als Arbeitsdatei anlegen, lässt man das LIBNAME-Statement weg und schreibt die DATA-Anweisung als **DATA name ;**. Das Programm löscht die Daten beim Beenden von SAS.

5.1.3 Daten liegen als SAS-Datenfile vor

SAS-Datenfiles erhält man dann, wenn Daten aus anderen Programmen, z.B. Excel, importiert werden oder die Dateneingabe direkt in SAS erfolgt, z.B. mit dem ViewTable-Editor oder SAS/INSIGHT.

Da Variablennamen und -typ (numerisch oder alphanumerisch) im Datenfile vorgegeben wurden, benötigt SAS hier keine INPUT-Anweisung (siehe *3.1 Der INPUT-Befehl*).

Beispiel:

```
* Speicherort für die SAS-Daten festlegen ;
LIBNAME name 'festplatte:\ordner' ;
* Daten einlesen ;
DATA name; SET name.daten ;
* Definition neuer Variablen ;
⋮
* Datenauswertung ;
SAS - Prozeduren zur Datenauswertung
```

Möchte man das SAS-Datenfile nicht permanent sondern nur als Arbeitsdatei anlegen, fällt das LIBNAME-Statement weg und die DATA-Anweisung lautet **DATA name ;**. SAS legt die Daten dann nur temporär als Arbeitsdatei an und löscht sie beim Beenden von SAS.

5.2 Wie ist ein SAS-Auswertungsprogramm aufgebaut ?

Die Daten werden, wie in 4.1.1 - 4.1.3 beschrieben, eingelesen und dann mit SAS-Anweisungen analysiert und ausgewertet.

Die Namen der Variablen und den Variablentyp (numerisch oder alphanumerisch) erhält SAS über die **INPUT**-Anweisung (siehe *3.1 Der INPUT-Befehl*) oder über die Spaltenbeschriftung im SAS-Datenfile.

Mit **Optionen** und zusätzlichen Programmbefehlen kann man das Aussehen des Programms und die Ausgabe der Daten beeinflussen. SAS benötigt diese Angaben nicht zum Auswerten der Daten. Sie helfen dabei die Auswertungsergebnisse zu beschreiben, zu strukturieren und zu ordnen.

Die folgenden Unterpunkte 5.2.1 - 5.2.11 zeigen die Reihenfolge der Statements in einem SAS-Programm auf (siehe nachfolgende Programmstruktur). Ausgegangen wird dabei von Daten, die schon als SAS-Datenfile vorliegen. Die aufgeführten Programm-Komponenten können genauso bei der direkten Dateneingabe als auch bei der Dateneingabe mit externen Datenfiles eingesetzt werden. Wichtig ist, dass die richtige Reihenfolge der Statements eingehalten wird.

Bei den folgenden Unterpunkten ist immer vermerkt, ob SAS diese Programm-Komponente benötigt oder nicht.

Struktur eines SAS-Programmes:

5.2.1
```
OPTIONS PAGESIZE=65 LINESIZE=75 NOLABEL;
```

5.2.2
```
* Libname-Anweisung ;
LIBNAME libname 'Pfad' ;
```

5.2.3
```
* Daten aus Excel importieren ;
PROC IMPORT OUT=libname.daten
            DATAFILE="Laufwerk:\Ordner\name_exceldatei.xls"
            DBMS=EXCEL2000 REPLACE;
            SHEET="Name des Tabellenblattes" ;
     GETNAMES=YES;
RUN;
```

5.2.4
```
* WERTEETIKETTEN (Klartexte für Codierungen, Formate);
PROC FORMAT;
    VALUE name1x    1='text max.16 Zeichen'
                    2='text max.16 Zeichen'
                    ⋮
                    n='text max.16 Zeichen'   ;
   ⋮
    VALUE name2x    1='text max.16 Zeichen'
                    2='text max.16 Zeichen'
                    ⋮
                    n='text max.16 Zeichen'   ;
```

5.2.5
```
* ------ Datenaufruf:                                  ------ ;
DATA name    ; SET libname.DATEN;
```

5.2.6
```
* -------Externe Daten                                 ------ ;
INFILE 'Laufwerk:\ordner\name.DAT' ;
```

5.2.7
```
* ------ Datenbeschreibung                             ------ ;
INPUT variable1 variable2 variable3 variable4 variable5
      variable6 ...
                        ;
```

5.2.8
```
* Definition neuer Variablen ;
* ;
```

5.2.9
```
* Variablenetiketten (Variablenbeschriftung);
LABEL /* Beispiel: NR = 'Lfd. Nummer' */
      varname1 = 'text max.40 Zeichen'
      varname2 = 'text max.40 Zeichen'
      ⋮
               ;
```

5.2.10
```
* Verknuepfung von Variablen und Werteetiketten ;
FORMAT varname1 name1x.  varname2 name2x.
       varname3 varname4 name3x. varmame5 name4x.
                                 ;
```

5.2.11
```
* direkte Dateneingabe ;
DATALINES ;
⋮
```

```
Daten
.
.
.
;
```

5.2.12

```
* ------------------- AUSWERTUNGSTEIL ------------------- ;
* Haeufigkeiten fuer qualitative (nominale) Variablen ;
  PROC FREQ DATA=name ;
      TABLES variablenliste ;
  RUN;
* Stat. Masszahlen fuer quantitative (stetige) Variablen ;
  PROC MEANS DATA=name MEAN MIN MAX MAXDEC=2 ;
      VAR variablenliste ;
  RUN;
```

5.2.1 Grundlegende Optionen vorgeben

Optionen, die generell gelten (SYSTEM OPTIONS), gibt man als erste Zeile in das Auswertungsprogramm ein. Sie beginnen mit dem Schlüsselwort OPTIONS und enden mit einem Semikolon ; ***SAS benötigt sie nicht zum Auswerten der Daten.*** Läßt man sie weg, arbeitet SAS mit den Standardeinstellungen.

Im Beispiel sind nur die wichtigsten Optionen aufgeführt. Weitere Optionen findet man in der SAS-Hilfe oder im SAS Handbuch "Base SAS® 9.1 Proceedures Guide".

```
OPTIONS LINESIZE=130 PAGESIZE=69 NOLABEL ;
```

Beschreibung der Optionen:

LINESIZE=130 legt die Anzahl der Zeichen pro Zeile fest, im Beispiel sind maximal 130 Zeichen pro Zeile erlaubt, dann beginnt SAS eine neue Zeile.

PAGESIZE=69 gibt die Anzahl der Zeilen pro Seite vor, im Beispiel sind maximal 69 Zeilen pro Seite zulässig, dann beginnt SAS eine neue Seite.

NOLABEL verhindert, dass SAS die Variablennamen sowohl als Variablenname als auch als Variablenbeschreibung ausgibt.

5.2.2 LIBNAME-Anweisung definieren

Mit einer LIBNAME-Anweisung wird zu Beginn einer SAS-Sitzung ein Ordner auf einem Laufwerk festgelegt, in welchem die zu erstellenden SAS-Datenfiles gespeichert bzw. aus welchem bestehende SAS-Datenfiles gelesen werden sollen (siehe *1.4 SAS-Libraries und Libnames*).

Beispiel:

```
LIBNAME meyer 'H:\hugo' ;
```

meyer ist der LIBNAME. SAS stellt die Verbindung zum Ordner **hugo** auf dem Datenträger **Laufwerk H:** her.

5.2.3 Daten aus Excel importieren

Diese Programmschritte sind nur dann nötig, wenn die Daten in einer Excel-Arbeitsmappe vorliegen. Genauere Angaben siehe *4.5.2 EXCEL-Daten über die Prozedur IMPORT in SAS einlesen und als SAS-Datenfile abspeichern***.**

SAS liest die Exceldaten ein, wandelt sie in SAS-Daten um und speichert sie als SAS-Datenfile ab.

Allgemeine Form der PROC IMPORT:

```
PROC IMPORT OUT=libname.name_sasdatei
            DATAFILE="Festplatte:\Ordner\name_exceldatei.xls"
            DBMS=EXCEL2000 REPLACE;
            SHEET="Name des Tabellenblattes" ;
     GETNAMES=YES;
RUN;
```

Erläuterung der SAS-Statements:

PROC IMPORT ist der SAS-Befehl zum Einlesen von Daten aus anderen Programmen.

OUT=name.sasdatei); **name** ist dabei der Name des Libnames und **name_sasdatei** der Name des zu erstellenden SAS-Datenfiles

DATAFILE="Festplatte:\Ordner\dateiname.xls" gibt an, wo die Excel-Arbeitsmappe mit den einzulesenden Daten gespeichert ist; **name_dateiname.xls** steht für den Namen der Excel-Arbeitsmappe.

DBMS=EXCEL2000 REPLACE legt fest welches Programm für die Dateneingabe verwendet wurde. Die Angabe **DBMS=EXCEL2000** gibt an, dass für die Dateneingabe mindestens mit Excel 2000 gearbeitet wurde und die **Option REPLACE** bewirkt, dass bei mehrfachem Einlesen das SAS-Datenfile immer wieder überschrieben wird.

SHEET="Tabellenblattname" teilt SAS den Namen des einzulesenden Tabellenblattes mit.

Statement **GETNAME=YES ;** übergibt SAS die Spaltenbeschriftung der Exceltabelle als Variablennamen.

Statement **RUN ;** beendet die Prozedur.

Beispiel:

```
PROC IMPORT OUT=meyer.daten
            DATAFILE="H:\doktorarbeit\exceldaten.xls"
            DBMS=EXCEL2000 REPLACE;
            SHEET="Tabelle1" ;
     GETNAMES=YES;
RUN;
```

Im Beispiel werden die Daten aus dem Tabellenblatt **Tabelle1** der Excelarbeitsmappe **exceldaten.xls**, die im Verzeichnis **H:\doktorarabeit** zu finden ist, in das SAS-Datenfile **daten** eingelesen und dieses in dem durch den Libname **meyer** festgelegten Ordner gespeichert.

5.2.4 VALUE-Anweisung definieren

VALUE-Anweisungen sind für die Programmausführung nicht notwendig. Man benötigt sie dann, wenn den verschlüsselten Ausprägungen einer Variablen, ein Klartext zugeordnet werden soll. VALUES werden auch als Werteetiketten oder Formate bezeichnet.

Den Ausprägungen stetiger Variablen weist man keinen Klartext zu, z.B. Größe, Gewicht, Laborwerten.

Allgemeine Form der FORMAT-Anweisung:

```
PROC FORMAT ;
    VALUE name  zahl1 = 'Klartext'
                zahl2 = 'Klartext'
                ⋮
                      ;
```

PROC FORMAT = SAS-Prozedur, die benutzerdefinierte Formate für die Ausgabe der Variablen festlegt. Welches Format zu welcher Variablen gehört wird an anderer Stelle vorgegeben.

VALUE = SAS-Statement für die Definition von Werteetiketten.

name = ein vom Benutzer festgelegter Name für die Werteetikette, über den sie angesprochen werden kann. Er darf nur aus Buchstaben (ä, ö, ü und ß nicht verwenden!!!) und Zahlen bestehen, wobei das erste Zeichen immer ein Buchstabe sein muß und nicht doppelt verwendet werden. Um die Namen der Werteetiketten von den Variablennamen zu unterscheiden, sollte man ein zusätzliches Zeichen an den Namen anhängen (im Beispiel ein x) oder einen völlig neuen Namen wählen. Es gelten die Vorgaben für SAS-Namen (siehe *1.3 Regeln für die Definition von Dateinamen und Libnames, Variablennamen und -typen*).

zahl = als Ausprägung verwendete Zahl.

'Klartext' = Text, der anstelle der Zahl ausgegeben werden soll. Er muß in Hochkomma eingeschlossen werden und darf max. 16 Zeichen lang sein (zusätzliche Zeichen werden beim Ausdrucken ignoriert).

; = beendet die Definition der Werteetikette.

Beispiel:

```
PROC FORMAT;
    VALUE gruppex 1 = 'Medizin'
                  2 = 'Placebo'
              ;

    VALUE medx 0 = 'kein Medikament'
               1 = 'Schmerzmittel'
               2 = 'Beruhigungsmittel'
           ;

    VALUE janeinx 0 = 'nein'
                  1 = 'ja'
              ;

    ⋮
```

Im Beispiel werden drei Werteetiketten definiert. Die erste erhält den Namen ***gruppex*** und teilt SAS mit, dass die **Zahl *1* den Klartext *Kontrolle*** und die **Zahl *2* den Klartext *Patient***

erhalten soll, die zweite heißt ***medx*** und weist den Zahlen ***0, 1 und 2*** die Klartexte ***kein Medikament***, ***Schmerzmittel*** und ***Beruhigungsmittel*** zu und die dritte lautet j***aneinx*** und veranlasst SAS anstelle der Zahlen ***0*** und ***1*** den Text ***nein*** bzw. ***ja*** auszugeben.

Eine Werteetikette wird mit dem SAS-Statement VALUE eingeleitet und nach der Aufzählung sämtlicher Codierungen mit einem Semikolon beendet!!!

An dieser Stelle im Programm findet nur die Definition der Werteetiketten statt. Zu welcher Variablen welche Werteetikette gehört, steht weiter hinten im Programm.

5.2.5 Festlegen, welche Daten verwendet werden sollen

An dieser Stelle gibt man SAS vor, mit welchem Datenfile die Auswertung stattfinden soll.

Gibt man die Anweisung **DATA name;** ein, sucht SAS im Arbeitsverzeichnis WORK nach dem Datenfile mit dem Namen **name** und wertet die darin gespeicherten Daten mit vorgegebenen Auswertungsbefehlen aus. Verwendet man stattdessen die Statements **DATA name ; SET libname.name ;**, schreibt SAS die Daten aus dem permanenten Datenfile libname.**name** (libname=Ordner, in dem die Daten zu finden sind) in die Arbeitsdatei **name** im Arbeitsverzeichnis WORK und arbeitet mit diesen Daten.

Beispiele:

```
OPTIONS PAGESIZE=65 LINESIZE=75 NOLABEL;
⋮
DATA paula;
⋮
```

⇨SAS sucht im Arbeitsverzeichnis WORK nach dem Datenfile **paula** und verwendet die darin abgelegten Daten für die Auswertung.

```
OPTIONS PAGESIZE=65 LINESIZE=75 NOLABEL;
LIBNAME meyer 'H:\meyer' ;
⋮
DATA willi; SET meyer.daten ;
⋮
```

⇨SAS schreibt die Daten aus dem Datenfile **daten** des Ordners **H:\meyer** in die Arbeitsdatei **willi** und arbeitet mit diesem Datenfile weiter.

5.2.6 Externe Daten

Mit dem SAS-Statement INFILE teilt man SAS mit, dass Daten aus einer separaten Datei eingelesen werden sollen, wie die Datei mit den Daten heißt und wo sie zu finden ist (siehe *4.3 Dateneingabe über externe Datenfiles*). **Liegt ein SAS-Datenfile vor oder werden die Daten direkt ins Programm eingegeben, entfällt dieser Programmschritt.**

Den Aufbau der Daten erfährt SAS aus dem nachfolgenden INPUT-Statement (siehe *4.1 Der INPUT-Befehl)*. **Liegt ein SAS-Datenfile vor oder werden die Daten direkt ins Programm eingegeben, entfällt dieser Programmschritt.**

Externe Datenfiles liegen in der Regel als Dateien mit der Extension **.dat** vor und enthalten Daten im ASCII-Format (Textformat).
Allgemeine Form:

```
INFILE 'laufwerk:\ordner\dateiname.dat' ;
INPUT variablenliste ;
⋮
```

Beispiel:

```
INFILE  'D:\sas\daten.dat' ;
INPUT   nr    sex    alter    groesse    gew  ;
⋮
```

⇨Im Beispiel liest SAS die Daten aus der Datei **daten.dat** ein. Die Datei wurde auf Laufwerk **D:** im Ordner **sas** abgelegt und enthält die Variablen **nr**, **sex alter**, **groesse** und **gew**, deren Ausprägungen durch mindestens ein Leerzeichen getrennt eingegeben wurden.

5.2.7 Beschreibung der einzulesenden Daten

Hier erfolgt die Beschreibung der zu verwendenden Daten und die Festlegung des Datentyps bei der direkten Dateneingabe bzw. der Verwendung von Daten aus externen Datenfiles mit der INPUT-Anweisung (siehe *4.1 Der Input-Befehl*). **Liegt ein SAS-Datenfile vor oder werden die Daten aus einem anderen Programm exportiert, entfällt dieser Programmschritt.**

5.2.8 Neue Variablen definieren

Neue Variablen werden aus bereits vorhandenen Variablen gebildet. Das kann durch die Angabe eines arithmetischen Ausdrucks, durch das Zusammenfassen von Variablenausprägungen oder Variablen, durch das Festlegen von Bedingungen oder das Bilden von Gruppen geschehen.

Um zu gewährleisten, dass die ursprünglichen Variablen weiter genutzt werden können, sollte man immer neue Namen verwenden.

name = Name des Datenfiles.

Bei der Eingabe von Formeln gilt Punkt vor Strich, Klammern werden zuerst aufgelöst.

1. durch die Angabe eines arithmetischen oder logischen Ausdruckes:

Links vom Gleichheitszeichen steht der Name der neuen Variablen. Im arithmetischen Ausdruck auf der rechten Seite dürfen alle üblichen Operatoren (+ = Addition, - = Subtraktion, * = Multiplikation, / = Division, ** = Potenzieren) verwendet werden:

Beispiele mit Erklärungen:

```
fieber_diff = fieber_nach - fieber_vor ;
```

⇨Die neue Variable **fieber_diff** wird aus der Differenz der Variablen **fieber_nach** und **fieber_vor** gebildet.

```
groesse = groesse+100 ;
```

⇨Zur Variablen **groesse** wird bei allen Datensätzen die Zahl 100 addiert.

```
groesse = groesse*100 ;
```

⇨Variable **groesse** wird bei allen Datensätzen mit der Zahl 100 multipliziert.

als SAS-Statements:

```
* Fieberdifferenz berechnen ;
fieber_diff = fieber_nach - fieber_vor ;
* Variable groesse um 100 erhöhen ;
groesse = groesse+100 ;
```

2. durch IF-Anweisungen

Verwendet man Text für die Ausprägungen neuer Variablen, muss man darauf achten, dass die erste Ausprägung so viele Zeichen aufweist, wie die längste. Gegebenenfalls muß mit Leerzeichen aufgefüllt werden. Ist die erste Ausprägung kürzer als die übrigen, schneidet SAS die Zeichen ab, die die Anzahl der Zeichen der ersten Ausprägung übersteigen.

Definiert man die Ausprägungen der neuen Variablen mit Zahlen, kann diesen über die Value-Anweisung und die Format-Anweisung ein Klartext zugeordnet werden (siehe *5.2.4 VALUE-Anweisung definieren* und *5.2.10 Werteetiketten mit den zugehörigen Variablen verknüpfen*).

In IF-Anweisungen können logische Operatoren eingesetzt werden:

Zeichen	Bedeutung
>	größer als
>=	größer gleich
<	kleiner als
<=	kleiner gleich
=	gleich
AND	logisches UND (=beide Bedingungen müssen zutreffen)
OR	logisches ODER (eine der beiden Bedingungen muß zutreffen)
NOT	logisches NICHT

Beispiele mit Erklärung:

Variable **rauchen** verwendet die Ausprägungen **0="nicht rauchen", 1="mäßig rauchen", 2="stark rauchen"**. Die Ausprägungen sollen nun zu "**Nichtraucher**" und "**Raucher**" zusammengefasst werden:

```
IF rauchen = 0  THEN raucheng = 'Nichtraucher' ;
                ELSE raucheng = 'Raucher' ;
```

⇨Die neue Variable **raucheng** bekommt die Ausprägung **Nichtraucher**, wenn Variable **rauchen** den Wert **0** enthält und die Ausprägung **Raucher** wenn Variable **rauchen** einen **von 0 verschiedenen Wert** aufweist.

Können fehlende Angaben vorkommen, würde SAS diese Beobachtungen den Rauchern zuordnen, weil diese bei numerischen Variablen intern als Punkt gesetzt werden und dieser von der Null verschieden ist. Um das zu vermeiden, ändert man die Bedingung wie folgt:

```
IF rauchen = 0  THEN raucheng = 'Nichtraucher' ;
IF rauchen > 0  THEN raucheng = 'Raucher' ;
```

Variable **ktemp** enthält gemessene Fieberwerte, die in die Ausprägungen erniedrigt, normal und erhöht eingeteilt werden sollen. Die Fieberwerte liegen bei allen Beobachtungen vor.

```
IF ktemp <=36.5                THEN ktempg = 'erniedrigt' ;
IF ktemp >36.5 AND ktemp <=37  THEN ktempg = 'normal' ;
IF ktemp >37                   THEN ktempg = 'erhöht' ;
```

⇨Die neue Variable **ktempg** bekommt die Ausprägung ***erniedrigt***, wenn Variable **ktemp** einen Wert enthält, der **kleiner oder gleich 36.5** ist, die Ausprägung ***normal***, wenn Variable **ktemp** einen Wert aufweist, der **zwischen 36.6 und 37** liegt (d.h. größer als 36.5 und kleiner oder gleich 37) und die Ausprägung ***erhöht***, wenn Variable **ktemp** eine Zahl **größer als 37** beinhaltet.

Auch hier gilt, wenn fehlende Werte vorkommen können, müssen diese von der Zuordnung in die neue Variable ausgeschlossen werden. Die Bedingungen lauten dann:

```
IF ktemp NE . ktemp <=36.5     THEN ktempg = 'erniedrigt' ;
IF ktemp >36.5 AND ktemp <=37  THEN ktempg = 'normal' ;
IF ktemp >37                   THEN ktempg = 'erhöht' ;
```

⇨**NE .** in der ersten Bedingung steht für NOT EQUAL ., d.h. fehlende Werte werden ausgeschlossen.

Die stetige Variable ***groesse*** soll in 10er Gruppen eingeteilt werden:

```
IF groesse>150 AND groesse<=160 THEN groesseg='151 - 160 cm' ;
IF groesse>160 AND groesse<=170 THEN groesseg='161 - 170 cm';
IF groesse>170 AND groesse<=180 THEN groesseg='171 - 180 cm';
IF groesse>180 AND groesse<=190 THEN groesseg='181 - 190 cm' ;
IF groesse>190 AND groesse<=200 THEN groesseg='191 - 200 cm' ;
```

⇨Variable **groesse** wird in 10er Gruppen eingeteilt und das Ergebnis in Variable **groesseg** geschrieben. Der untere Wert gehört nicht mehr zur Gruppe (**>** - Zeichen), der obere schon (**<=** - Zeichen). Nimmt die Zahl in Variable **groesse** einen Wert **größer als 150 und kleiner oder gleich 160** an, erhält die neue Variable **groesseg** die Ausprägung **151 - 160 cm**. Liegt der Wert in Variable **groesse zwischen 161 und 170**, bekommt Variable **groesseg** die Ausprägung **161 - 170 cm** usw.

```
IF groesse=1.65 THEN groesse=165;
```

⇨Diese Bedingung veranlasst SAS dazu, Variable ***groesse*** mit 165 zu überschreiben, wenn sie den Wert 1.65 enthält.

```
IF nr=75 THEN groesse=165 ;
```

⇨Diese Bedingung bewirkt, dass SAS den Inhalt der Variablen ***groesse*** mit 165 überschreibt, wenn Variable **nr** den Wert 75 enthält.

```
IF nr=75 THEN groesse*100 ;
IF nr=75 THEN groesse=groesse*100 ;
```

⇨SAS gibt für die obere Bedingung eine Fehlermeldung aus und ignoriert die Anweisung, weil der Variablenname vor der Anweisung ***groesse*100** fehlt. Richtig ist daher die zweite Bedingung. Hier multipliziert SAS Variable ***groesse*** mit 100 und schreibt das Ergebnis in Variable ***groesse***, wenn Variable **nr** den Wert 75 annimmt.

als SAS-Statements:

```
* Variable rauchen in zwei Gruppen einteilen ;
IF rauchen = 0  THEN raucheng = 'Nichtraucher' ;
                ELSE raucheng = 'Raucher' ;
* Variable groesse in gleichmaessige Gruppen einteilen ;
IF groesse>150 AND groesse<=160 THEN groesseg='151 - 160 cm' ;
IF groesse>160 AND groesse<=170 THEN groesseg='161 - 170 cm';
IF groesse>170 AND groesse<=180 THEN groesseg='171 - 180 cm';
IF groesse>180 AND groesse<=190 THEN groesseg='181 - 190 cm' ;
IF groesse>190 AND groesse<=200 THEN groesseg='191 - 200 cm' ;
* Inhalt der Variablen groesse mit 165 ersetzen, wenn er 1.65 ist ;
IF groesse=1.65 THEN groesse=165;
* Inhalt der Variablen groesse mit 165 überschreiben, wenn die nr=75 ist ;
IF nr=75 THEN groesse=165 ;
* Inhalt der Variablen groesse mit 100 multiplizieren, wenn die nr=75 ist ;
IF nr=75 THEN groesse=groesse*100 ;
```

3. durch (gleichmäßige) Gruppenbildung

Die Formeln ermitteln für jeden Datensatz die obere Klassengrenze als glatten Wert und sie ordnen die Datenwerte den Klassen zu. Der Wert Null wird immer in die unterste Klasse aufgenommen. Den kleinsten und größten Wert einer Klasse legt man mit der Berechnungsformel fest. Will man anstelle der oberen Klassengrenzen Texte ausgegeben haben, muß man einen VALUE definieren und diesen zuweisen (siehe *5.2.4 VALUE-Anweisung definieren* und *5.2.9 Werteetiketten mit den zugehörigen Variablen verknüpfen*).

Im Beispiel werden die beiden Funktionen **INT(…)** und **ROUND (…)** eingesetzt. Erstere schneidet die Stellen nach dem Komma ab, letztere rundet auf die nächste ganze Zahl, wobei SAS bis zur 4 nach dem Komma abrundet und ab einer 5 nach dem Komma aufrundet.

Beispiele mit Erklärung:

```
alterg = INT(ROUND(alter-1)/10) * 10 + 10 ;
```

⇨aus der stetigen Variablen **alter** wird die in 10-er Klassen eingeteilte neue Variable **alterg** kreiert. Die unterste Klasse enthält Werte zwischen 0 und 10, d.h. die kleinste Zahl der Klasse ist die Null und die größte 10.

SAS berechnet mit der Formel die Ausprägungen 10, 20, 30, 40, …, die die folgende Gruppeneinteilung aufweisen: **10=0-10, 20=11-20, 30=21-30, 40=31-40, 50=41-50**, ...

```
alterg = INT(ROUND(alter)/10) * 10 + 10 ;
```

⇨ aus der stetigen Variablen **alterg** wird die in 10-er Klassen eingeteilte neue Variable **alterg** definiert. Die unterste Klasse beinhaltet Werte zwischen 0 und 9, d.h. die kleinste Zahl der Klasse ist die Null und die größte 9.

Die Formel liefert die Ausprägungen 10, 20, 30, 40, …, die die folgende Gruppeneinteilung aufweisen: **10=0-9, 20=10-19, 30=20-29, 40=30-39, 50=40-49**, ...

```
alterg = INT(ROUND(alter-1)/5) * 5 + 5 ;
```

⇨ aus der stetigen Variablen **alter** wird die in 5-er Klassen eingeteilte neue Variable **alterg** erstellt. Zur untersten Klasse gehören Werte zwischen 0 bis 5, d.h. die kleinste Zahl der Klasse ist die Null und die größte 5.

SAS berechnet mit der Formel die Ausprägungen 5, 10, 15, 20, ..., womit die Gruppen **5=0-5, 10=6-10, 15=11-15, 20=16-20, 25=21-25** ... gebildet werden.

```
alterg = INT(ROUND(alter)/5) * 5 + 5 ;
```

⇨ aus der stetigen Variablen **alter** wird die in 5-er Klassen eingeteilte neue Variable **alterg** erstellt. Die unterste Klasse schließt Werte zwischen 0 und 4, d.h. die kleinste Zahl der Klasse ist die Null und die größte 4.

SAS berechnet mit der Formel die Ausprägungen 5, 10, 15, 20, ..., womit die Gruppen **5=0-4, 10=5-9, 15=10-14, 20=15-19, 25=20-24** ... gebildet werden.

als SAS-Statements:

```
* Alter in 10-er Gruppen einteilen: 10=0-10, 20=11-20, 30=21-30 ...;
alterg = INT(ROUND(alter-1)/10) * 10 + 10 ;
* Alter in 10-er Gruppen einteilen: 10=0-9, 20=10-19, 30=20-29 ...;
alterg = INT(ROUND(alter)/10) * 10 + 10 ;
*Alter in Jahren ermitteln: 5=0-5, 10=6-10, 15=11-15, 20=16-20 ...;
alterg = INT(ROUND(alter-1)/5) * 5 + 5 ;
*Alter in Jahren ermitteln: 5=0-4, 10=4-9, 15=10-14, 20=15-19 ...;
alterg = INT(ROUND(alter)/5) * 5 + 5 ;
```

4. Datumsdifferenzen berechnen

SAS verwaltet Datumsangaben intern mit seriellen Zahlen. Für ein **Datum** wird als **Ausgangsbasis der Wert 1** verwendet und in ganzzahligen Schritten weiter gezählt wird. Das Datum 01.01.1960 entspricht der Ausgangsbasis 1, der 02.01.1960 der Zahl 2, der 03.01.1960 der Zahl 3 usw. .

Liest SAS ein Exceldatum in der Form 02JUN1931:00:00:00 muss mit der Funktion DATEPART die Uhrzeit abgeschnitten werden.

Beispiel: `gebdatum_neu = DATEPART(gebdatum) ;`

⇨SAS schneidet mit der Funktion die Uhrzeit ab und schreibt das übriggebliebene Datum in die Variable gebdatum_neu. **Es muß ein anderer Variablenname angegebenen werden, weil SAS die Funktion sonst nicht ausführen kann!**

Beispiele mit Erklärung:

Beide Datumsangaben sind als Variablen vorhanden. Berechnet werden soll die Verweildauer. Variable **aufdatum** enthält das Aufnahme- und Variable **entdatum** das Entlassungsdatum.

```
aufenthalt = (entdatum - aufdatum) + 1 ;
```

⇨Die Differenz von Entlassungs- und Aufnahmedatum ergibt die Verweildauer in Tagen. Sie wird in Variable **aufenthalt** geschrieben. Zu den Verweiltagen wird 1 Tag addiert, weil SAS den Aufnahmetag nicht mitzählt und somit beispielsweise bei einer Aufnahme und Entlassung am selben Tag eine Differenz von 0 ermittelt. Dies ist zwar rechnerisch gesehen richtig, aber krankenhaustechnisch nicht korrekt, denn der Patient war ja einen Tag aufgenommen worden.

Beide Datumsangaben sind als Variablen vorhanden. Berechnet werden soll das Alter in Jahren. Variable **aufdat** enthält das Aufnahme- und Variable **gebdat** das Geburtsdatum.


```
alter=INT(aufdat - gebdat) / 365) ;
```

⇨Die angegebene Formel berechnet das ganzzahlige **Alter** in Jahren zum Zeitpunkt der Aufnahme. Da SAS die Differenz in Tagen berechnet, wird durch 365 dividiert um in Jahre umzurechnen. Die Funktion **INT( )** schneidet die Nachkommastellen ab.

Nur eine der Datumsangaben liegt als Variable vor. Berechnet werden soll das Alter zu einem bestimmten Stichtag. Bekannt ist das Geburtsdatum in Variable **gebdat**, der Stichtag soll der **26.07.2008 sein**.

```
*  Aus den Komponenten des Stichtages ein Datum definieren ;
sticht=26;  stichm=07;  stichj=2008;
stich_dat = MDY(stichm,sticht,stichj) ;
* Alter in Jahren berechnen ;
alter=INT(stich_dat - gebdat) / 365) ;
```

⇨MDY ist eine SAS-Datumsfunktion und steht für MONTH DAY YEAR. Sie berechnet aus den Variablen **sticht**, **stichm** und **stichj** das Datum des Stichtages. Die Differenz von Geburts- und Stichtagdatum ergibt das Alter in Tagen. Sie wird in Variable **alter** geschrieben. Mit der Division durch 365 wird das Alter in Jahre umgerechnet und mit der Funktion **INT(...)** die Dezimalstellen abgeschnitten.

Die Datumskomponenten liegen nicht in Form eines Datums vor. Variable **entmonat** enthält den Entlassungsmonat, Variable **entjahr** das Entlassungsjahr, Variable **gebmonat** den Geburtsmonat und Variable **gebjahr** das Geburtsjahr. Berechnet werden soll das Alter bei der Entlassung.

```
* Alter berechnen ;
alter=INT((entjahr-gebjahr) + (entmonat-gebmonat)/12) ;
```

⇨Berechnet wird die Differenz zwischen dem Entlassungsjahr und dem Geburtsjahr zuzüglich der Differenz aus dem aktuellen Entlassungsmonat und dem Geburtsmonat dividiert durch 12 (Umrechnung in Jahre). Mit der Funktion **INT(...)** werden die Dezimalstellen abgeschnitten und Variable alter als ganze Zahl ausgegeben.

5. Ausprägungen von Variablen kombinieren

Manchmal möchte man wissen in welcher Kombination die Ausprägungen zweier oder auch mehrerer Variablen auftreten. Im Grunde könnte man dieses Problem über Kontingenztafeln lösen, was aber bei vielen zu verknüpfendem Variablen sehr aufwendig und unübersichtlich werden kann.

Da SAS zwischen numerischen und alphanumerischen Variablen unterscheidet, gibt es auch drei verschiedene Verfahren.

Numerische Variablen verknüpfen. Numerische Variablen enthalten nur Zahlen; für fehlende Werte setzt SAS einen Punkt. Am einfachsten ist es, wenn die Ausprägungen der Variablen einstellig sind.

Beispiel:

Die Variablen **kompl1** und **kompl2** enthalten jeweils die beiden Zahlen 0 und 1, die **Zahl 0 steht für nein** und die **Zahl 1 für ja**. Jetzt sollen die beiden Variablen zu einer Variablen mit dem Namen **kompl_ges** zusammengefasst werden, damit das Vorliegen beider Komplikationen ausgezählt werden kann.

```
* Komplikationen zusammenfassen ;
kompl_ges = 10*kompl1 + kompl2 ;
```

⇨SAS kombiniert nun pro Beobachtung die beiden Variablen **kompl1** und **kompl2** und bildet die Variable **kompl_ges** mit den möglichen Ausprägungen **00**, **01**, **10** und **00**, wobei Variable **kompl1** mit 10 multipliziert und Variable **kompl2** dazuaddiert wird. .In der Auswertung tauchen nur die Ausprägungen auf, die in der Datenmenge vorkommen.

Die 10er Stelle zeigt dann den Inhalt der Variablen **kompl1** und die 1er Stelle den Inhalt der Variablen **kompl2**. Würde man einfach nur die Summe bilden, würde SAS die Ausprägung 0, 1 und 2 errechnen, die Information, ob hinter der Zahl 1 die Komplikation1 oder die Komplikation2 steckt, geht dann verloren.

Ausprägung 00 bedeutet, dass keine der beiden Komplikationen auftrat, Ausprägung 01, dass nur Komplikation2 vorkam, Ausprägung 10, dass nur Komplikation1 vorlag und Ausprägung 11, dass beide Komplikationen vorhanden waren.

Textvariablen kombinieren. Textvariablen oder alphanumerische Variablen setzen sich aus Buchstaben oder Buchstabenzahlenkombinationen zusammen; für fehlende Werte setzt SAS ein Leerzeichen.

Beispiel:

Variable **blutgruppe** enthält den Blutgruppennamen **A**, **B**, **AB** oder **0** und Variable **rhesus** die Ausprägungen **+** oder **-**. In der neuen Variablen **blut_rh** sollen die Ausprägungen der beiden Variablen zusammengestellt werden.

```
* Blutgruppe und Rhesusfaktor in eine Variable ;
blut_rh = TRIM(blutgruppe) || TRIM(rhesus) ;
```

⇨SAS kombiniert nun pro Beobachtung die beiden Variablen **blutgruppe** und **rhesus** und bildet die Variable **blut_rh** mit den möglichen Ausprägungen **A+**, **B+**, **AB+**, **0+**, **A-**, **B-**, **AB-** und **0-**. In der Auswertung tauchen nur die Ausprägungen auf, die in der Datenmenge vorkommen. Die Funktion **TRIM** entfernt überflüssige Leerzeichen aus den Variablen.

Numerische Variablen und Textvariablen zusammensetzen. Numerische Variablen enthalten nur Zahlen; für fehlende Werte setzt SAS einen Punkt, alphanumerische Variablen dagegen setzen sich aus Buchstaben oder Buchstabenzahlenkombinationen zusammen; für fehlende Werte setzt SAS ein Leerzeichen.

Beispiel:

Zusammengefaßt werden sollen die Angaben des Hauptgenotyps und des Subgenotyps von an Hepatitis C erkrankten Personen. Der Hauptgenotyp wurde als Variable **hauptgtyp** mit den Zahlen 1, 2 und 3 und der Subgenotyp als Variable **subgtyp** mit den Buchstaben a, b und c eingegeben. Diese beiden Variablen sollen in Variable **genotyp** kombiniert werden.

```
* Hauptgenotyp und Subgenotyp in eine Variable zusammenfassen ;
genotyp = '␣␣' ;
   SUBSTR(genotyp,1,1) = hauptgtyp ;
   SUBSTR(genotyp,2,1) = subgtyp ;
```

␣ = Leerzeichen

⇨SAS kreiert Variable **genotyp** mit zwei Leerzeichen und schreibt mit der Funktion SUBSTR als 1. Zeichen den Inhalt der Variablen **hauptgtyp** und als 2. Zeichen den Inhalt der Variablen **subgtyp** in die Variable **genotyp**. Die Zahl 1 hinter dem Variablennamen im Funktionsaufruf besagt, dass die Zahl aus Variable hauptgtyp an erster Stelle in die

Variable eingefügt werden soll, die 1 hinter dem Variablennamen sagt aus, dass der Buchstabe aus Variable subgtyp an zweiter Stelle eingefügt werden soll.

Es können die folgenden Kombinationen vorkommen: 1a, 1b, 1c, 2a, 2b, 2c, 3a, 3b und 3c. Kombinationen, die nicht vorkommen, werden von SAS ignoriert.

Will man **einer Zahl immer den gleichen Buchstaben** zuordnen, geht man wie folgt vor.

Beispiel:

Von der Tumorklassifikation TNM wurde das Stadium des Primärtumors als Zahl (0, 1, 2, 3 und 4) in Variable **t** eingegeben und jetzt soll dieser Zahl der Buchstabe T vorangestellt werden. Das Ergebnis wird in Variable **t_stadium** gespeichert.

```
* Stadium des Primaertumors den Buchstaben T voranstellen ;
t_stadium = 'T␣' ;
    SUBSTR(t_stadium,2,1)=t
```

␣ = Leerzeichen

⇨SAS legt Variable **t_stadium** an und schreibt den Buchstaben T in diese Variable. Mit der Funktion **SUBSTR** wird der Inhalt der Variablen **t** für jede Beobachtung an den Buchstaben T in Variable **t_stadium** angehängt. Die Zahl 2 im Funktionsaufruf besagt, dass die Zahl an zweiter Stelle in die Variable eingefügt werden soll, die 1 sagt aus, dass der Inhalt von Variable t einstellig ist.

5.2.9 Variablenetiketten definieren

Variablenetiketten sind für die Programmausführung nicht notwendig. Sie beschreiben einen verkürzten Variablennamen näher und werden bei den durchgeführten Berechnungen zusätzlich zum Variablennamen ausgegeben.
Variablen, deren Namen eindeutig ist, benötigen keine Variablenetikette. So sind beispielsweise die Angaben **geschlecht = 'Geschlecht'** oder **alter = 'Alter' überflüssig**.

Auch im Programm neu definierte Variablen kann man eine Variablenetikette zuweisen.

Allgemeine Form der LABEL-Anweisung:

```
LABEL variablenname = 'Beschreibung der Variablen'
      variablenname = 'Beschreibung der Variablen'
      variablenname = 'Beschreibung der Variablen'
      ⋮
                    ;
```

Links vom Gleichheitszeichen steht der verwendete Variablenname und rechts davon der zugehörige Klartext (= Variablenetikette) in einfache Hochkomma eingeschlossen. Die Beschreibung darf **maximal 40 Zeichen lang** sein. **Nach der letzten** Variablenetikette schließt ein Strichpunkt die Beschreibung der Variablen ab.

LABEL = SAS-Statement für die Definition von Variablenetiketten.

variablenname = der vom Benutzer festgelegte Namen der Variablen.

'Beschreibung der Variablen' = Text, der zusätzlich zum Variablennamen ausgegeben werden soll.

; = beendet die Definition aller Variablenetiketten.

Als SAS-Statements:

```
* ------ Variablenetiketten : ----------------- ;
LABEL th     = 'Behandlungsart'
      sex    = 'Geschlecht'
      groesse= 'Koerpergroesse in cm'
      gew    = 'Koerpergewicht in kg'
      med    = 'Medikation'
      alterg = 'Alter gruppiert'
      ⋮
                ;
```

Beispiel:

ohne Variablenetikette

TH

TH	Frequency	Percent	Cumulative Frequency	Cumulative Percent
Medizin	47	47.48	47	47.48
Placebo	52	52.52	99	100.00

mit Variablenetikette

Behandlungsart

TH	Frequency	Percent	Cumulative Frequency	Cumulative Percent
Medizin	47	47.48	47	47.48
Placebo	52	52.52	99	100.00

5.2.10 Werteetiketten mit den zugehörigen Variablen verknüpfen

Für die Programmausführung ist die Verknüpfung der Werteetiketten (Formate) mit den Variablen nicht notwendig. Hat man jedoch Werteetiketten festgelegt, sollte man sie auch verwenden.

Die Definition von Formaten erfolgt am Anfang eines Auswertungsprogramms unter der PROC FORMAT und ist unter *5.2.3 VALUE-Anweisung definieren* beschrieben.

Man kann sich die Definition und die Zuweisung der Formate sparen, wenn man bei der Definition einer neuen Variablen den zu verwendenden Klartext gleich mit angibt:

Beispiel:

```
IF rauchen=0 THEN raucheng='Nichtraucher' ;
             ELSE raucheng='Raucher' ;
```

Der erste Klartext sollte so viele Zeichen aufweisen, wie der längste. Benötigt er weniger Zeichen, muss mit Leerzeichen aufgefüllt werden.

Trifft ein Format für mehrere Merkmale zu, so muss man dieses nicht für jede dieser Variablen festlegen, sondern man schreibt die Werteetikette ein einziges Mal und gibt in der Formatanweisung an, welche Variablen diese Werteetikette benutzen sollen.

Allgemeine Form der FORMAT-Anweisung:

```
FORMAT varname1 formatname1. varname2 formatname2.
       varname3 varname4 formatname3.
              ⋮
                    ;
```


FORMAT = SAS-Anweisung, mit der man festlegt welche Werteetikette für welche Variable verwendet werden soll.

VARNAME = Name der Variable(n), der/denen ein Format zugewiesen werden soll.

formatname = ein vom Benutzer festgelegter Name für die Werteetikette, über den die Werteetikette angesprochen werden kann. Definiert werden die Formate als VALUE unter der Prozedur PROC FORMAT.

; = beendet die Verknüpfung der Variablennamen und Werteetiketten.

Wichtig!!! Nach jedem Format muß ein Punkt (im Beispiel: .) gesetzt werden, um die Verknüpfung abzuschließen.

Beispiel:

```
PROC FORMAT ;
    VALUE janeinx 0='nein' ;
                  1='ja' ;
    VALUE gruppex 1 = 'Medizin'
                  2 = 'Placebo' ;
    VALUE medx 0 = 'kein Medikament'
               1 = 'Medikament'
               2 = 'Beruhigungsmittel' ;

⋮
FORMAT behgruppe gruppex. medikament medx.
       kompl_vor kompl_nach janeinx.
                 ⋮
                       ;
```

Der Variablen **behgruppe** wird das Format **gruppex**, der Variablen **medikament** die Werteetikette **medx** und den beiden Variablen **kompl_vor** und **kompl_nach** das Format **janeinx**. zugewiesen. Erst dann weiß SAS, welchen Klartext die als Zahlen verschlüsselten Ausprägungen der einzelnen Variablen erhalten.

ohne Format

med

med	Frequency	Percent	Cumulative Frequency	Cumulative Percent
0	33	33.33	47	33.33
1	31	31.31	64	64.64
2	35	35.35	99	100.00

mit Format

Medikation

med	Frequency	Percent	Cumulative Frequency	Cumulative Percent
kein Medikament	33	33.33	47	33.33
Schmerzmittel	31	31.31	64	64.64
Beruhigungsmittel	35	35.35	99	100.00

5.2.11 Direkte Dateneingabe

Dieser Programmschritt ist nur notwendig, wenn die Daten direkt ins Programm eingegeben werden sollen. Man tippt die die Datenzeilen unter dem Statement DATALINES ein und SAS liest sie nach den Vorgaben im INPUT-Statement ein.

Nach der letzten Datenzeile muss ein Semikolon eingegeben werden und anschließend der Befehl RUN ; folgen (siehe *4.2 Direkte Dateneingabe*).

Beispiel:

```
⋮
INPUT nr   geschlecht groesse gewicht alter med rauchen datum
                          ;
⋮
DATALINES ;
001 2 180 80.0 29 1 0 121093
002 1 160 45.9 30 3 0 101092
003 1 170 80.0  . 1 0 160993
          ;
RUN;
⋮
```

Diese Art der Dateneingabe ist nur dann sinnvoll, wenn wenig Datensätze vorliegen, weil das Auswertungsprogramm sonst sehr unübersichtlich wird. Bei vielen Datensätzen empfiehlt sich die Dateneingabe in ein externes Datenfile (siehe *4.3 Dateneingabe über externe Datenfiles*)

5.2.12 Auswertungsteil

Hier gibt man die Befehle zum Analysieren der Daten ein. Die Auswertung der Daten geschieht in SAS durch die Angabe von Prozeduren (siehe *3.4 SAS-Statements (SAS-Anweisungen, SAS-Auswertungsbefehle)*).

Beispiel:

```
* ============================================================ ;
* --------------------   Auswertungsteil  ---------------------- ;
*============================================================== ;
* Haeufigkeiten fuer qualitative (nominale) Variablen ;
  PROC FREQ DATA=name ;
      TABLES variablenliste ;
  RUN;
* Stat. Masszahlen fuer quantitative (stetige) Variablen ;
  PROC MEANS DATA=name MIN MAX MEAN MEDIAN MAXDEC=2 ;
      VAR variablenliste ;
  RUN;
```

Die Prozedur **PROC FREQ** im Beispiel errechnet für nominale Variablen Häufigkeitslisten und prozentuale Anteile.

Der **PROC MEANS** - Befehl dient der Ermittlung statistischer Maßzahlen wie z.B. Mittelwert, Standardabweichung, Median, Minimum und Maximum. Man verwendet ihn nur für stetige Variablen.

Nach jeder Befehlsfolge muss der SAS-Befehl `RUN ;` erscheinen, da der davor stehende Programmteil sonst nicht ausgeführt wird.

Die am häufigsten verwendeten Prozeduren sind in Kapitel 5 anhand von Beispielen beschrieben.

5.3 Beispielprogramme

Im Folgenden werden Programmbeispiele zur direkten Dateneingabe, zu Daten aus externen Datenfiles und zu als SAS-Datenfile vorliegende Daten beschrieben.

5.3.1 Programm zur direkten Dateneingabe (siehe *4.2 Direkte Dateneingabe*)

5.2.1

```
OPTIONS PAGESIZE=65 LINESIZE=75 NOLABEL;
```

5.2.2

```
* Libname-Anweisung ;
LIBNAME libname 'Pfad' ;
```

5.2.4

```
* WERTEETIKETTEN ;
PROC FORMAT;
    VALUE therapix 1='OP'
                   2='Medikamente'
                   3='Kur' ;
    VALUE gruppex 1='Patient'
                  2='Kontrolle' ;
    VALUE agx 1='<= 20 Jahre'
              2=' > 20 Jahre' ;
```

5.2.5

```
* ------ Datenaufruf:                                ------ ;
DATA name    ; SET libname.DATEN;
```

5.2.7

```
* Datenbeschreibung laut Datenbogen ;
INPUT    NR  1- 3  SEX 4  GROESSE 5-7  GEW 8-10
      ALTER 11-12  THERAPIE 13  GRUPPE 14  AT 15-16
         AM 17-18  AJ 19-20  ET 21-22  EM 23-24  EJ 25-26 ;
```

5.2.8

```
* Eingabe neuer Variablen ;
IF alter <= 20 THEN ag=1 ; ELSE ag=2;
gew = gew / 10;
```

5.2.9

```
* Variablenetiketten ;
LABEL SEX = 'Geschlecht'
      GROESSE = 'Koerpergroesse in cm'
      GEW = 'Koerpergewicht in kg'
                ;
```

5.2.10

```
* Verknuepfung von Variablen und Werteetiketten ;
FORMAT therapie therapix.  gruppe gruppex.  ag agx.
                   ;
```

5.2.11

```
DATALINES;
/* Daten */
;
RUN;
```

5.2.12

```
* Auswertungsteil ;
SAS - Befehle zum Auswerten der Daten
```

5.3.2 Programm zur Dateneingabe über externe Datenfiles (siehe 4.3 Dateneingabe *über externe Datenfiles*)

5.2.1
```
OPTIONS PAGESIZE=65 LINESIZE=75 NOLABEL;
```

5.2.2
```
* Libname-Anweisung ;
LIBNAME libname 'Pfad' ;
```

5.2.4
```
* WERTEETIKETTEN ;
PROC FORMAT;
    VALUE therapix 1='OP'
                   2='Medikamente'
                   3='Kur' ;
    VALUE gruppex 1='Patient'
                  2='Kontrolle' ;
    VALUE agx 1='<= 20 Jahre'
              2=' > 20 Jahre' ;
```

5.2.5
```
* ------ Datenaufruf:                              ------ ;
DATA name    ; SET libname.DATEN;
```

5.2.6
```
* Daten aus der externen Datendatei einlesen
INFILE 'E:\doktor\SASDAT\name\name.DAT' ;
```

5.2.7
```
* Datenbeschreibung laut Datenbogen ;
INPUT    NR 1 - 3  SEX 4  GROESSE 5-7  GEW 8-10
      ALTER 11-12  THERAPIE 13  GRUPPE 14  AT 15-16
         AM 17-18  AJ 19-20  ET 21-22  EM 23-24  EJ 25-26 ;
```

5.2.8
```
* Eingabe neuer Variablen ;
IF alter <= 20 THEN ag=1 ; ELSE ag=2;
gew = gew / 10;
```

5.2.9
```
* Variablenetiketten ;
LABEL SEX = 'Geschlecht'
      GROESSE = 'Koerpergroesse in cm'
      GEW = 'Koerpergewicht in kg'
                ;
```

5.2.10
```
* Verknuepfung von Variablen und Werteetiketten ;
FORMAT therapie therapix.  gruppe gruppex.  ag agx.
                    ;
```

5.2.12
```
* Auswertungsteil ;
SAS - Befehle zum Auswerten der Daten
```


5.3.3 Programm mit Daten aus einem SAS-Datenfile (siehe *4.5 EXCEL-Daten in SAS einlesen und als SAS-Datenfile abspeichern*)

5.2.1
```
OPTIONS PAGESIZE=65 LINESIZE=75 NOLABEL;
```

5.2.2
```
* Libname-Anweisung ;
LIBNAME libname 'Pfad' ;
```

5.2.3
```
* Daten aus Excel importieren ;
PROC IMPORT OUT=libname.daten
            DATAFILE="Laufwerk:\Ordner\name_exceldatei.xls"
            DBMS=EXCEL2000 REPLACE;
            SHEET="Name des Tabellenblattes" ;
     GETNAMES=YES;
RUN;
```

5.2.4
```
* WERTEETIKETTEN ;
PROC FORMAT;
    VALUE therapix 1='OP'
                   2='Medikamente'
                   3='Kur' ;
    VALUE gruppex 1='Patient'
                  2='Kontrolle' ;
    VALUE agx 1='<= 20 Jahre'
              2=' > 20 Jahre' ;
```

5.2.8
```
* Eingabe neuer Variablen ;
IF alter <= 20 THEN ag=1 ; ELSE ag=2;
gew = gew / 10;
```

5.2.9
```
* Variablenetiketten ;
LABEL SEX = 'Geschlecht'
      GROESSE = 'Koerpergroesse in cm'
      GEW = 'Koerpergewicht in kg'
                ;
```

5.2.10
```
* Verknuepfung von Variablen und Werteetiketten ;
FORMAT therapie therapix.  gruppe gruppex.  ag agx.
                    ;
```

5.2.12
```
* Auswertungsteil ;
SAS - Befehle zum Auswerten der Daten
```

5.4 Erstellen eines SAS-Auswertungsprogramms

Im Folgenden wird davon ausgegangen, dass die Daten in einer Excel-Arbeitsmappe vorliegen.

Die Beschreibung zu den am häufigsten verwendeten Auswertungsbefehlen finden Sie in Kapitel 6 der Anleitung.

1) Falls noch nicht geschehen: SAS starten und die F5-Taste zum Aktivieren des Programmfensters drücken oder den Befehl WINDOW ✧ PROGRAM EDITOR wählen.

2) Wenn zuvor schon eine Datei geladen wurde diese, falls notwendig, speichern und dann den -Schalter (= Programmeditor leeren) drücken.

3) Die folgenden Programmzeilen in den Program Editor eingeben (siehe *4.5.2 EXCEL-Daten über die Prozedur IMPORT in SAS einlesen und als SAS-Datenfile abspeichern*)

```
00001 OPTIONS LINESIZE=130 PAGESIZE=69 NOLABEL ;
00002 LIBNAME name 'Laufwerk:\Ordner' ;
00003
00004 PROC IMPORT OUT=name.daten
00005            DATAFILE="Laufwerk:\Ordner\dateiname.xls"
00006            DBMS=EXCEL2000 REPLACE ;
00007            SHEET="Name des Tabellenblattes" ;
00008     GETNAMES=YES;
00009 RUN;
00010
00011
00012 DATA name ; SET name.daten;
00013
00014
00015 * ============================================ ;
00016 * ------------ AUSWERTUNGSTEIL --------------- ;
00017 * ============================================ ;
00018 * Beschreibung der eingelesenen Daten ;
00019 PROC CONTENTS DATA=name;
00020 RUN;
```

Erläuterung der SAS-Statements in der `PROC IMPORT`:

OUT=name.sasdatei); **name** ist dabei der Name des Libnames und **name_sasdatei** der Name des zu erstellenden SAS-Datenfiles

DATAFILE="Festplatte:\Ordner\dateiname.xls" gibt an, wo die Excel-Arbeitsmappe mit den einzulesenden Daten gespeichert ist; **name_dateiname.xls** steht für den Namen der Excel-Arbeitsmappe.

DBMS=EXCEL2000 REPLACE legt fest welches Programm für die Dateneingabe verwendet wurde. Die Angabe **DBMS=EXCEL2000** gibt an, dass für die Dateneingabe mindestens mit Excel 2000 gearbeitet wurde und die **Option REPLACE** bewirkt, dass bei mehrfachem Einlesen das SAS-Datenfile immer wieder überschrieben wird.

SHEET="Tabellenblattname" teilt SAS den Namen des einzulesenden Tabellenblattes mit.

Statement **GETNAME=YES ;** übergibt SAS die Spaltenbeschriftung der Exceltabelle als Variablennamen.

4) Sie ersetzen mit Hilfe des Befehls EDIT ✧ REPLACE den Text *name* an jeder Stelle durch eine beliebigen maximal 8-stelligen Namen. **Aktivieren Sie dabei die Kontrollfelder MATCH WHOLE WORD ONLY (= Nur ganzes Wort suchen) und MATCH CASE (= Groß-/Kleinschreibung),** (siehe *3.2.7 Ersetzen von beliebigen Zeichenketten*).

5) Speichern Sie das Programm unter einem beliebigen Name (beispielsweise auswertung.sas) im gewünschten Ordner (siehe *3.2.2 Speichern einer Programmdatei oder einer Datendatei mit der Extention .dat während einer SAS-Sitzung*).

6) Danach beginnen Sie mit der Beschreibung/Definition Ihrer Variablen (siehe *5.2.8 Neue Variablen bilden*) und der Eingabe der gewünschten Auswertungsbefehle (siehe ??).

7) Führen Sie Ihr Programm aus (siehe *5.7 Starten des Programms*).

8) Speichern Sie Ihre Programmdatei bevor Sie SAS beenden (siehe *5.6 Programm speichern*).

5.5 Laden des SAS-Auswertungsprogrammes

1) Falls noch nicht geschehen: SAS starten und die F5-Taste zum Aktivieren des Programmfensters drücken oder den Befehl WINDOW ✧ PROGRAM EDITOR wählen.

2) Wenn zuvor schon eine Datei geladen wurde diese, falls notwendig, speichern und dann den -Schalter (= Programmeditor leeren) drücken.

3) Aktivieren Sie den Programmeditor und öffnen Ihr Auswertungsprogramm im zutreffenden Ordner.

⇨ auf dem Bildschirm erscheinen die Statements Ihres Programms.

5.6 Programmdatei speichern

1) Die Programmdatei (Endung .SAS) in den Programmeditor laden.

2) Führen Sie den Befehl FILE ✧ SAVE AS (beim ersten Speichern oder wenn das Programm unter einem anderen Namen abgelegt werden soll) oder den Befehl FILE ✧ SAVE (beim erneuten Speichern) aus und stellen Sie im Feld SPEICHERN IN das gewünschte Laufwerk ein.

Die beiden folgenden Schritte entfallen, wenn der Befehl FILE ✧ SAVE verwendet wurde:

3) Öffnen Sie den zutreffenden Ordner und wählen Sie im Listenfeld DATEITYP den Dateityp SAS FILES (*.SAS).

4) Geben Sie den Namen des zu speichernden Programms in das Feld DATEINAME ein und betätigen Sie den Schalter SPEICHERN.

Selbstverständlich können Sie Ihre SAS-Programme auch über den Windows Explorer (**nicht** den Internet-Explorer!) auf den betreffenden Datenträger speichern.

5.7 Starten des Programmes

Wenn Sie Ihr Programm gestartet (= F8-Taste oder den Schalter in der Symbolleiste drücken) haben, ist für die Dauer der Programmausführung kein Zugriff auf das Programm möglich. Sie sehen an den Meldungen, die im LOG-Fenster angezeigt werden, daß die SAS-Anweisungen in Ihrem Programm abgearbeitet werden. SAS entfernt die Programmzeilen

aus dem Programm-Editor-Fenster, speichert sie in den Arbeitsspeicher, liest Zeile für Zeile und führt die Befehle aus.

Zeigen sich rote Zeilen im LOG-Fenster, so handelt es sich um Fehlermeldungen. SAS bricht die Fortführung des Programms je nach Programmstelle nach dem Erscheinen der ersten Fehlermeldung ab oder überspringt die fehlerhaften Programmstellen.

Zum Ermitteln des Fehlers aktiviert man das LOG-Fenster (F6-Taste drücken oder Befehl WINDOW ✧ LOG) und blättert es nach dem Fehler durch. Hat man ihn gefunden, springt man zum Programmfenster (F5-Taste betätigen oder Befehl WINDOW ✧ PROGRAM EDITOR), ruft das Programm durch Drücken der F4-Taste oder Befehl RUN ✧ RECALL LAST SUBMIT aus dem Arbeitsspeicher in den Programmeditor zurück, verbessert den Fehler und speichert das Programm.

Nach dem Korrigieren aller Fehler löscht man den Inhalt des LOG-Fensters durch Drücken des Schalters  in der Symbolleiste oder durch das Ausführen des Befehls EDIT ✧ CLEAR ALL, ruft den Programm-Editor auf und startet das Programm erneut. Nach einem fehlerfreien Programmlauf erscheint das Ergebnis im OUTPUT-Fenster.

5.8 Ergebnisse im OUTPUT-Fenster betrachten und in eine Datei speichern

SAS führt nach dem Starten des Auswertungsprogramms die eingegebenen Befehle aus. Das Ergebnis des Programmlaufes erscheint, falls keine Fehler auftreten, im OUTPUT-Fenster, einer temporären Datei.

Man aktiviert dieses durch Drücken der F7-Taste oder wählt den Befehl WINDOW ✧ OUTPUT. Das Editieren der angezeigten Daten im OUTPUT-Fenster ist nicht möglich, deshalb speichert man diese in eine permanente Datei ab, die dann als ASCII-Datei mit der Extension **.lst** in jedes Textverarbeitungsprogramm oder jeden Editor geladen werden kann.

1) Aktivieren Sie das OUTPUT-Fenster durch Drücken der F7-Taste oder Ausführen des Befehls Befehl WINDOW ✧ OUTPUT.

2) Speichern Sie den Inhalt des OUTPUT-Fenster über den Befehl FILE ✧ SAVE AS unter einem beliebigen Namen im gewünschten Ordner.

   SAS muß im Listenfeld DATENTYP den Listeneintrag LIST FILES anzeigen.

5.9 OUTPUT-Datei in WinWord einlesen

1) Starten Sie WORD (z.B. über den Befehl START ✧ PROGRAMME ✧ WORD 2003) und führen Sie den Befehl DATEI ✧ ÖFFNEN aus.

2) Wählen Sie im Listenfeld DATEITYP den Listeneintrag ALLE DATEIEN (*.*), stellen Sie im Listenfeld SUCHEN IN den Pfad zur *OUTPUT*-Datei ein.

3) Markieren Sie den Namen der *OUTPUT*-Datei und betätigen Sie den Schalter ÖFFNEN.

4) Führen Sie den Befehl BEARBEITEN ✧ ALLES MARKIEREN aus und wählen Sie anschließend den Befehl FORMAT ✧ ZEICHEN.
5) Öffnen Sie die Karteikarte SCHRIFT und aktivieren Sie im Listenfeld SCHRIFTART die Schriftart COURIER NEW.

6) Rufen Sie den Befehl BEARBEITEN ✧ ERSETZEN auf.

7) a) Positionieren Sie den Cursor im Feld SUCHEN NACH, halten Sie die ALT-Taste gedrückt halten und tippen Sie **gleichzeitig im aktivierten Nummernfeld** auf der Tastatur den **Zahlencode 0131** ein. Im Textfeld erscheint das Zeichen **ƒ**. Setzen Sie die Einfügemarke in das Feld ERSETZEN DURCH und geben Sie einen Bindestrich ein. Nach dem Betätigen des Schalters ALLE ERSETZEN tauscht Word alle ƒ gegen Bindestriche aus.

   b) Manchmal wurde auch das Zeichen **Š** verwendet. Dieses kann man wie beschrieben durch die Eingabe des **Zahlencodes 0138** in einen Bindestrich umwandeln.

8) Dann geben Sie im Feld SUCHEN NACH auf die oben genannte Weise den Zahlencode 0130 ein. Das Zeichen , zeigt sich. Im Feld ERSETZEN DURCH die Taste ALTGR und gleichzeitig die  -Taste drücken. Nach dem Betätigen des Schalters ALLE ERSETZEN tauscht Word alle , gegen senkrechte Striche | aus.

9) Danach tippen Sie im Feld SUCHEN NACH auf die oben genannte Weise den **Zahlencode 0136** eing. Im Textfeld steht das Zeichen ^. Den Cursor in Feld ERSETZEN DURCH positionieren und ein **+** eintippen. Nach dem Betätigen des Schalters ALLE ERSETZEN tauscht Word alle ^ gegen + aus.

   Manchmal wurde auch das Zeichen **„** oder **‡** oder **‰** oder **…** oder **‹** oder **Œ** verwendet. Dieses kann man wie beschrieben durch die Eingabe des **Zahlencodes 0132** bzw. **0135** bzw. **0137** bzw. **0133** bzw. **0139** bzw. **0140** in ein **+** umwandeln.

10) Wählen Sie den Befehl DATEI ✧ SPEICHERN UNTER…, stellen Sie im Listenfeld DATENTYP den Typ WORD-DOKUMENT (*.DOC) ein und speichern Sie das Ergebnis unter dem vorgeschlagenen Namen in den gewünschten Ordner.

11) Führen Sie den Befehl DATEI ✧ BEENDEN aus, um WORD zu schließen und zu SAS zurückzukehren.