Der Datenschritt
Ein mächtiges Werkzeug innerhalb der SAS-Umgebung

Carina Ortseifen
Universitätsrechenzentrum Heidelberg
Im Neuenheimer Feld 293
Heidelberg
carina.ortseifen@urz.uni-heidelberg.de

Zusammenfassung

Der SAS-Datenschritt kann manchmal mysteriös sein, aber auch ungeheuer mächtig. In diesem Beitrag werden auf Anweisungen und Optionen eingegangen, die für die tägliche Arbeit hilfreich sein können: Die DATA-Anweisung mit ihren Datei-Optionen, Möglichkeiten der Variablendeklaration, Einsatzmöglichkeiten der internen Variablen _N_, Verarbeitung von BY-Gruppen, Auslesen bestimmter Beobachtungen und die Entwicklung eigener Funktionen.

Schlüsselwörter: Datenschritt, DATA-Anweisung, LENGTH-Anweisung, _N_, POINT=-Option, BY-Gruppenverarbeitung, FIRST., LAST., FCMP-Prozedur, _NULL_, OUTPUT-Anweisung, RETAIN-Anweisung implizit, SUM-Anweisung, COMPRESS=-Option, REUSE=-Option, REPLACE=-Option, PGM=-Option

1 Einführung

Der SAS-Datenschritt ist manchmal rätselhaft und mysteriös. Oder ist Ihnen völlig klar, was die beiden folgenden Datenschritte genau machen:

```
Data;
   x+y;
Run;
```

```
Data _Null_;
   If 0 Then Set test Nobos=obs;
   Call Symputx("nobs", nobs);
   Stop;
Run;
```

Auf der anderen Seite ist er aber auch sehr mächtig. Während man beispielsweise mit dem SAS/Enterprise Guide 10 Klicks braucht, um eine neue Variable anzulegen, benötigen Sie im Datenschritt eine Anweisung:

```
Data test;
   a = 1;
Run;
```

In den folgenden Abschnitten werden wir uns daher mit dem Datenschritt näher auseinander setzen; aber nicht in Form einer klassischen Einführung von Beginn an und ohne Behandlung des Program Data Vectors (PDV). Den PDV finden Sie ausführlich beschrieben in [2] und [3]. Stattdessen betrachten wir die Anweisung DATA genauer und wie im SAS System Variablen deklariert werden können. Anschließend beschäftigen wir uns mit der internen Variablen _N_, der BY-Gruppenverarbeitung im Datenschritt, und der Auswahl bestimmter Beobachtungen. Ein weiterer Abschnitt wird den Dateioptionen und der Entwicklung eigener Funktionen gewidmet. Alle in den folgenden Abschnitten vorgestellten SAS-Programme finden Sie als gezippte Datei im deutschsprachigen SAS-Wiki unter http://de.saswiki.org bei dieser Langfassung.

2 Die Anweisung DATA

Wir wenden uns wieder dem ersten Beispiel-Datenschritt zu [7]. Was ist das Ergebnis von:

```
Data;
   x+y;
Run;
```

?

Mit der Anweisung

```
Data dateiname;
```

wird die SAS-Tabelle dateiname angelegt. Da alle Programmierer möglichst effektiv sein wollen (manchmal auch „schreibfaul“), genügt es auch nur

```
Data ;
```

zu schreiben. Die erzeugte SAS-Tabelle wird automatisch nummeriert: data1, data2 … Die Standardbibliothek ist dabei WORK. Im Log-Fenster erscheint: work.data1 …

Mit der Systemoption USER= kann eine andere, eigene Standardbibliothek festgelegt werden:

```
Option User=meinbib;
```

Der Vorteil ist, dass jetzt alle mit eingliedrigem Namen erzeugten Tabellen in der Bibliothek „meinbib“ landen und nicht automatisch mit dem Ende der SAS-Sitzung gelöscht werden. (Nachteil: Sie müssen selbst für Ordnung in diesem Ordner sorgen!)

In einem Datenschritt können mehrere Tabellen verarbeitet werden. Dabei kann aus n vorhandenen Tabellen mit den Anweisungen SET bzw. MERGE eine gemacht werden.

```
Data gesamt;
   Set datei1 datei2;
Run;
```

```
Data gesamt;
   Merge datei1 datei2;
   By id;
Run;
```

Es können aber auch mehrere Dateien aus einer Datei erzeugt werden mithilfe der Anweisung OUTPUT. Normalerweise wird am Ende des Datenschritts die Beobachtung in eine Tabelle geschrieben. (Genauer: Die Werte aus dem PDV werden in die Ausgabetabelle eingetragen, vgl. [2].) Es wird ein implizites Output ausgeführt.

```
Data ;
   a = 1;
Run;
```

entspricht dabei

```
Data ;
   a = 1;
   Output;
Run;
```

(Als implizites Output muss man die Anweisung aber nicht wirklich hinschreiben.) Verwendet man die Anweisung OUTPUT dagegen explizit, wird das implizite Output überschrieben. Fortan werden nur die Beobachtungen aus dem PDV in die Tabelle eingetragen, wenn ein OUTPUT erscheint. Anwendung findet diese Anweisung beim Umstrukturieren von Tabellen als Alternative zur Prozedur TRANSPOSE:

```
Data breit;
   Input id $ a b c;
   Datalines;
A1 3 4 5
A2 4 . 2
;
Run;

Data lang (Drop=a b c);
   Set breit;
   zeit=1;
   wert=a;
   If not Missing(wert) Then Output;
   zeit=2;
   wert=b;
   If not Missing(wert) Then Output;
   zeit=3;
   wert=c;
   If not Missing(wert) Then Output;
Run;
```

und beim Aufteilen in mehrere Tabellen:

```
Data dateiM dateiW ;
   Set sashelp.class;
   If (sex="M") Then Output dateiM;
   Else              Output dateiW;
Run;
```

Mit der DATA-Anweisung können also ein oder mehrere Tabellen erzeugt werden. Aber manchmal will man den SAS-Datenschritt als „Taschenrechner“ verwenden, ohne eine Tabelle zu erzeugen. Oder man möchte seine eigenen Reports generieren. Dann schreibt man einfach

```
Data _Null_;
```

Die folgenden Anweisungen werden ausgeführt, aber keine Tabelle erzeugt:

```
Data _Null_;
   a=11; b=12; c=a*b;
   Put "Das Produkt ist " c=;
Run;
```

3 Variablendeklaration

Neue Variablen können im Datenschritt auf (mindestens) drei verschiedene Arten definiert werden. Am einfachsten geht es mit der Zuweisung:

```
variable = wert;
```

Links vom Gleichheitszeichen steht der Name der Variablen. Handelt es sich um einen unbekannten Variablennamen, wird die neue Variable neu angelegt; ist der Name bereits vorhanden, wird die alte Variable überschrieben.
Rechts vom Gleichheitszeichen steht der Wert (oder ein Ausdruck), der der Variablen zugewiesen wird.

```
x = 5;
a = "Ulm";
```

Der numerischen Variablen x wird der Wert 5 zugewiesen und x hat per Standardeinstellung (Default) die Länge 8.
Die Variable a ist vom Typ Character, also eine Textvariable mit der Länge 3, weil der Wert Ulm drei Zeichen lang ist. Der Textwert muss mit Anführungszeichen „maskiert“ werden – denn ohne Anführungszeichen würde SAS prüfen, ob die beiden Variablen a und ulm den gleichen Wert haben.

Die zweite Variante, um Variablen zu deklarieren, erfolgt mit der Anweisung LENGTH:

```
LENGTH x 8 a $10;
```

Der Vorteil gegenüber der direkten Zuweisung besteht darin, dass die LENGTH-Anweisung gleichzeitig die Reihenfolge der Variablen in der SAS-Tabelle festlegt.

Und als drittes gibt es noch die implizite Zuweisung, in der SAS-Literatur auch SUM-Anweisung genannt (obwohl an dieser Stelle kein Schlüsselwort SUM auftaucht!) [4].

```
x + 1;
x + y;
```

Die Variable x ist dabei vom Typ numerisch und hat die Länge 8. Diese implizite Zuweisung ist sehr nützlich bei Additionen und Iterationen:

```
Data;
   Set sashelp.class;
   If (sex = "F") Then index + 1;
   Put index;
Run;
```

Das Auszählen der Mädchen (sex="F") in der Tabelle funktioniert aber nur, weil die implizite Zuweisung ein implizites Retain enthält. Damit wird der aktuelle Wert der Variable index von einer zur nächsten Beobachtung behalten. Ohne dieses Retain würde der Wert immer wieder auf 0 zurückgesetzt werden.

Aber aufgepasst! Es gibt damit drei Arten zum Addieren:

1. Die einfache Variante:
   `Ergebnis = variable1 + variable2;`
2. Die Summenfunktion:
   `Ergebnis = Sum(variable1, variable2);`
3. Die implizite Zuweisung:
   `Ergebnis + (variable1 + variable2);`

Und alle können zu unterschiedlichen Ergebnissen führen, wie folgendes Beispiel zeigt:

```
Data werte;
  Input vorher nachher @@;
  Datalines;
10  9
11  14
```

```
15  .
21 11
;

Data test;
   Set werte;
   diff1 = vorher - nachher;
   diff2 = Sum(vorher,-nachher);
   diff3 = 0;
   diff3 + (vorher - nachher);
   label
      diff1="Einfache Subtraktion"
      diff2="Mit Summenfunktion"
      diff3="Mit impliziter Zuweisung"
   ;
Run;
Proc Print Label;
Run;
```

Mit folgendem Ergebnis:

Obs	vorher	nachher	Einfache Subtraktion	Mit Summen-funktion	Mit impliziter Zuweisung
1	10	9	1	1	1
2	11	14	-3	-3	-3
3	15	.	.	15	0
4	21	11	10	10	10

Die Unterschiede treten natürlich nur dort auf, wo es fehlende Werte gibt. Während die einfache Subtraktion dann ebenfalls einen fehlenden Wert produziert, liefert die SUM-Funktion die Summe der nichtfehlenden Werte, also 15. Und die implizite Zuweisung liefert 0. Probieren Sie selbst aus, was passiert, wenn die Zuweisung der 0 zur Variable diff3 auskommentiert ist.

So, und jetzt dürfte Ihnen auch klargeworden sein, welches Ergebnis der erste Datenschritt im Abschnitt 1 produziert. Falls doch nicht, dann lesen Sie einfach nochmal nach.

4 Orientierung im Datenschritt mit der internen Variable _N_

In jedem Datenschritt wird automatisch die interne Variable _N_ angelegt. Dabei wird bei jeder Iteration des Datenschritts, also bei jeder neuen Beobachtung der Wert von _N_ um 1 erhöht.

Obwohl die Variable _N_ selbst nicht in die neue SAS-Tabelle aufgenommen wird, gibt es zahlreiche Einsatzmöglichkeiten:

- Erzeugung einer eindeutigen Variablen: Wenn die Tabelle nach einem bestimmten Kriterium sortiert werden soll, später aber wieder der Ausgangszustand hergestellt werden muss, kann man vor dem Sortieren durch Zuweisung der Variable _N_ eine neue Variable schaffen, die eindeutige Werte enthält.
  ```
  id = _N_;
  ```

- Bedingtes Ausführen einer Anweisung nach jeder i-ten Beobachtung:
  ```
  If (mod(n, 5) = 0) Then …;
  ```

Eine weitere Anwendung kann die Bestimmung der Anzahl der Beobachtungen einer SAS-Tabelle sein:

```
%Let nobs=;

Data _null_;
   Set sashelp.class;
   Call Symputx("nobs",_n_);   * _N_ dient als Zähler;
Run;

%Put nobs = &nobs;
```

Der Wert der Variablen _N_ wird mit der CALL SYMPUTX-Routine der Makrovariable nobs zugewiesen, eigentlich für jede Beobachtung. Doch da die Routine erst am Ende des Datenschritts ausgeführt wird, enthält nobs die gewünschte Anzahl der Beobachtungen.
Hinweis: Die CALL Routine SYMPUTX ist ab SAS Version 9 verfügbar und erspart die explizite Umwandlung eines Textes in eine numerische Variable.

Für diese Aufgabe, Bestimmung der Anzahl der Beobachtungen, gibt es zahlreiche weitere Varianten [1]:

a. _N_ kombiniert mit der Option END= der SET-Anweisung

```
%Let nobs=;
Data _null_;
   Set sashelp.class End=eof;
   If eof Then Call Symputx("nobs",_n_);
Run;
%Put nobs = &nobs;
```

Hierbei wird die CALL Routine SYMPUTX nur noch am Ende des Datenschritts ausgeführt, wenn die Variable eof den Wert TRUE enthält. (Die Option END= definiert eine binäre Variable eof, welche die Werte 0 oder 1 für FALSE bzw. TRUE enthält, je nachdem, welche Beobachtung gerade verarbeitet wird.)

Die Option END= kann auch eingesetzt werden, wenn nach dem Einlesen der letzten Beobachtung weitere Anweisungen ausgeführt werden sollen, bevor der Datenschritt beendet wird, zum Beispiel um zusätzliche Informationen ins Log-Fenster zu schreiben.

b. Option NOBS= der SET-Anweisung und STOP-Anweisung

```
Data _null_;
   Set sashelp.class Nobs=obs;
   Call Symputx("nobs",obs);
   Stop;
Run;
```

Während man in den obigen Beispielen jeweils die gesamte Datei einlesen musste, genügt es mit der Option NOBS= der SET-Anweisung nur die Metainformationen zu lesen, um die Anzahl der Beobachtungen zu erfahren. Die STOP-Anweisung unterbricht das implizite RETURN und beendet den Datenschritt.

c. If 0 THEN SET und STOP-Anweisung

```
Data _null_;
   If 0 Then Set sashelp.class Nobs=obs;
   Call Symputx("nobs",obs);
   Stop;
Run;
```

Dieses Konstrukt – vergleichen Sie es mit dem ersten Beispiel – liest ebenfalls nur die Metainformationen der Tabelle aus und nicht die gesamten Beobachtungen ein.

d. SCL-Funktionen

Mit der SCL-Funktion OPEN() wird auf die Tabelle zugriffen und mit der SCL-Funktion ATTRN() und dem Argument NOBS die Eigenschaft „Anzahl der Beobachtungen“ ausgelesen:

```
Data _Null_;
   dsid=Open("sashelp.class");
   obs=Attrn(dsid,"nobs");
   Call Symputx("nobs",obs);
Run;
```

e. Prozedur SQL

Natürlich gibt es auch Lösungen, die mit der Prozedur SQL funktionieren:

```
Proc Sql Noprint;
   Select Count(*) into :nobs From sashelp.class;
Quit;
```

Die Funktion COUNT(*) ermittelt die Anzahl der Beobachtungen und mit INTO :nobs wird dieser Wert in eine Makrovariable übertragen.

Ab Version 9.3 steht mit der Makrovariable &SYSNOBS dieser Wert direkt zur Verfügung, allerdings darf man dann die Option NOPRINT nicht setzen:

```
Proc Sql;
   Select * From sashelp.class;
Quit;

%Put N = &Sysnobs;
```

Außerdem gibt es immer noch die Dictionary Tables:

```
%Let nobs=;

Proc Sql Noprint;
   Select nobs Into:nobs From dictionary.tables
   Where libname = "SASHELP" and memname="CLASS";
Quit;
%Put nobs = &nobs;
```

5 BY-Gruppen im Datenschritt

Die Anweisung BY kann in jedem Prozedurschritt verwendet werden, um die Prozedur getrennt für Gruppen innerhalb der Tabelle auszuführen. Die Gruppen werden dabei durch die verschiedenen Werte der BY-Variablen definiert.

```
Proc Print Data=sashelp.class;
   Var name;
   By sex;
Run;
```

erzeugt beispielsweise eine Liste der Jungen- (sex="M") und eine Liste der Mädchennamen (sex="F").

Aber auch im Datenschritt kann die BY-Anweisung verwendet werden.

```
Data class;
   Set sashelp.class;
   By sex;
   …
Run;
```

Dabei ist – wie im Prozedurschritt auch – wichtig, dass die Tabelle nach der BY-Variablen sortiert ist. Sobald man jetzt die BY-Anweisung verwendet, werden zwei automatische Variablen erzeugt:

```
   First.sex
   Last.sex
```

Diese sind binär, haben also nur Werte 0 oder 1. First.sex trägt den Wert 1, wenn die 1. Beobachtung der BY-Gruppe auftritt, also beim 1. Mädchen oder beim 1. Jungen; ansonsten hat sie den Wert 0. Last.sex hat den Wert 1 bei der jeweils letzten Beobachtung, ansonsten 0.

Obs	Name	Sex	First.sex	Last.sex
1	Alice	F	1	0
2	Barbara	F	0	0
3	Carol	F	0	0
…	…	…	…	…
9	Mary	F	0	1
10	Alfred	M	1	0
11	Henry	M	0	0
…	…	…	…	…
18	Thomas	M	0	0
19	William	M	0	1

Anwendungsgebiete für die BY-Gruppenverarbeitung im Datenschritt sind:

- Reporting
- Ausgabe in externe Dateien, z.B. HTML
- Berechnung von Zwischensummen in der Tabelle

([4], [7]) Passendes Programmbeispiel: Siehe ByGruppen.sas.

Die BY-Gruppen können auch mit der internen Variable _N_ und der Option END= kombiniert werden, um z.B. eine Dokumentation von Variablen in eine externe HTML-Datei zu erstellen. Das Beispielprogramm finden Sie unter dem Namen By_N_End.sas im deutschsprachigen SAS-Wiki.

6 Nur bestimmte Beobachtungen auslesen

Normalerweise werden alle Beobachtungen einer mit SET gelesenen SAS-Tabelle verarbeitet:

```
Data class;
   Set sashelp.class;
   bmi = weight**2/heigth;
Run;
```

Manchmal möchte man aber nur eine Auswahl und nicht alle Beobachtungen verwenden. Dazu bietet SAS drei Varianten:

- Die Anweisungen IF und WHERE
  Einschränkung mit Hilfe einer logischen Bedingung
- Die Dateioptionen FIRSTOBS= und OBS=
  Verarbeitung eines Datenbereichs von bis
- Option POINT=
  Auslesen bestimmter Beobachtungen

Mit der POINT=-Option können beispielsweise auch Teile einer Tabelle dupliziert und an anderer Stelle eingefügt werden.

Passende Programmbeispiele für die Anweisungen und die Dateioptionen: Siehe BestimmteBeobachtungen.sas. Im Folgenden wird nur die Anwendung der Option POINT= vorgestellt.

```
Data beob;
   Input n @@;
   Datalines;
1 2 5 6 9
;
```

Die Tabelle beob besitzt eine Variable n, 5 Beobachtungen und die Werte 1, 2, 5, 6 u. 9.

```
Data test;
   Set beob;
   Set sashelp.class Point=n;
Run;
```

Im diesem Datenschritt werden nun zunächst aus der Tabelle beob die Werte für n gelesen. Die zweite SET-Anweisung liest sashelp.class ein. Die Option POINT=n beschränkt dabei die Beobachtungen auf die Werte der Variable n aus der ersten Tabelle beob. Ein PROC PRINT-Aufruf liefert:

Obs	Name	Sex	Age	Height	Weight
1	Alfred	M	14	69.0	112.5
2	Alice	F	13	56.5	84.0
3	Henry	M	14	63.5	102.5
4	James	M	12	57.3	83.0
5	Jeffrey	M	13	62.5	84.0

Durch Veränderung der Werte von n können einzelne Beobachtungen dupliziert oder deren Reihenfolge verändert werden. Hat die Variable n beispielsweise die Werte

```
2 2 6 5
```

erhält man folgende Tabelle:

Obs	Name	Sex	Age	Height	Weight
1	Alice	F	13	56.5	84.0
2	Alice	F	13	56.5	84.0
3	James	M	12	57.3	83.0
4	Henry	M	14	63.5	102.5

Die Beobachtung 2 wurde verdoppelt, Henry und James in Beobachtung 5 und 6 haben die Plätze „getauscht“.

7 Datei-Optionen

Datei-Optionen werden in Klammern nach SAS-Tabellennamen geschrieben. Z. B.

```
Data class;
   Set sashelp.class (FIRSTOBS=5 OBS=10);
   bmi = weight**2/heigth;
Run;
```

Mit FIRSTOBS= und OBS= haben wir zwei Datei-Optionen im vorherigen Abschnitt kennengelernt. Weitere Datei-Optionen sind DROP=, KEEP= und RENAME=. Eine

Erläuterung dazu finden Sie in [3]. Diese Datei-Optionen können dabei nicht nur im Datenschritt sondern auch in Prozedurschritten eingesetzt werden.

Datei-Optionen, die dagegen nur im Zusammenhang mit DATA-Anweisung verwendet werden können, sind [8]:

- COMPRESS=
- REUSE=
- $POINTOBS=
- REPLACE=
- PGM=

Diese Dateioptionen können auch als Systemoptionen verwendet werden. Dann gelten Sie nicht für die betroffene Tabelle, sondern generell für alle Tabellen.

a. Bessere Speicherplatznutzung: COMPRESS= YES und REUSE=YES

SAS-Tabellen werden standardmäßig mit fester Breite gespeichert. D.h. alle Beobachtungen nehmen gleich viel Platz ein, gleichgültig, ob sie Daten enthalten oder nicht. (Das steuert die Systemoption COMPRESS=NO.) Mit der Dateioption COMPRESS=YES kann dieses Verhalten für einzelne Tabellen geändert werden. Die Tabellen nehmen dann – in der Regel – weniger Speicherplatz in Anspruch.
Zusätzlich erlaubt REUSE=YES, dass der durch Löschen von Beobachtungen freiwerdende Speicherplatz an neue Beobachtungen vergeben wird. (Ansonsten werden neue Beobachtungen unten an die Tabelle angehängt.)

b. Bestimmte Beobachtungen – auch bei Views: $POINTOBS=YES

Die Dateioption $POINTOBS=YES garantiert, dass die POINT=-Option auch bei Views funktioniert.

c. Ersetzen von Tabellen: REPLACE=YES

Standardmäßig werden im Datenschritt Tabellen neu angelegt oder vorhandene Tabellen überschrieben, ohne dass der Anwender explizit um Erlaubnis gefragt wird. Das garantiert die Systemoption REPLACE=YES. Es mag aber durchaus Szenarien geben, wo dieses Standardverhalten unerwünscht ist; etwa wenn man mit wichtigen Daten arbeitet, die nicht aus Versehen überschrieben werden dürfen. Dann kann die Systemoption REPLACE=NO gesetzt werden und

```
Data sasuser.test;
Run;
```

ist nicht möglich bzw. es erscheint eine entsprechende Meldung.

```
NOTE: The data set SASUSER.TEST has 1 observations and 0 variables.
WARNING: Data set SASUSER.TEST was not replaced because of NOREPLACE
option.
NOTE: DATA statement used (Total process time):
```

Soll eine Tabelle jedoch bewusst überschrieben werden, setzt man die Dateioption REPLACE=YES:

```
Data sasuser.test (Replace=Yes);
   Set sashelp.class;
Run;
```

Wichtig ist: Diese Option gilt nur für permanente Tabellen. Tabellen in der Bibliothek Work sind davon ausgeschlossen.

Beispielprogramme zum Einsatz dieser Optionen: Siehe Compress.sas.

Im folgenden Abschnitt wird noch eine Option vorgestellt, keine Datei- oder Systemoption, aber eine nützliche Option der DATA-Anweisung:

d. Kompilierter Datenschritt: PGM=

SAS-Datenschritte werden in zwei Schritten abgearbeitet: Zunächst werden sie kompiliert und anschließend ausgeführt. Standardmäßig passiert dies in einem „Rutsch“. Der Anwender führt das Programm aus („submit“) und sieht das Ergebnis. Die Schritte können aber auch getrennt werden: Das kompilierte Programm kann mit der Option PGM= in einem SAS-Katalog abgespeichert werden und zu einem späteren Zeitpunkt, in einem weiteren Datenschritt ausgeführt werden. Hierzu ein Beispiel aus dem Programm pgm.sas, das Sie auch im deutschsprachigen SAS-Wiki (de.saswiki.org) finden:

```
Libname pgms "c:\test";
Data out / Pgm=pgms.prog01;
   Set sashelp.class;
   x=weight**2/height;
Run;
```

Das Programm wird in der Bibliothek pgms unter dem Namen prog01 abgespeichert. Beachten Sie hier, dass PGM= nach einem Schrägstrich eingefügt wurde (und nicht in Klammern steht, also streng genommen keine Datei-Option ist, sondern eine Option der DATA-Anweisung).

```
Data Pgm=pgms.prog01;
Run;
```

Im diesem Datenschritt wird das kompilierte Programm mit PGM=pgms.prog01 aufgerufen und ausgeführt und das erwartete Ergebnis, nämlich eine neue Datei, erzeugt:

```
NOTE: DATA STEP program loaded from file PGMS.PROG01.
NOTE: There were 19 observations read from the data set
SASHELP.CLASS.
NOTE: The data set WORK.OUT has 19 observations and 6 variables.
```

Mit der Anweisung REDIRECT und den Optionen INPUT und OUTPUT können die zu lesende Tabelle und die zu erzeugende Tabelle angepasst werden, analog zu Argumenten in Funktionen oder den Parametern in Makros:

```
Data Pgm=pgms.prog01;
   Redirect Output out=sasuser.classneu;
Run;

NOTE: DATA STEP program loaded from file PGMS.PROG01.
NOTE: There were 19 observations read from the data set
SASHELP.CLASS.
NOTE: The data set SASUSER.CLASSNEU has 19 observations and 6 varia-
bles.
```

8 Eigene Funktionen im Datenschritt nutzen

Das SAS-System kennt zwei Methoden, um eigene Funktionen definieren zu können:

1. Die Makrosprache
2. Die Prozedur FCMP

SAS-Makros können so entwickelt werden, dass sie einen Rückgabewert ausgeben, der dann im Datenschritt weiterverarbeitet wird:

```
%Macro Addiere(a,b);
   %Let return=%Sysevalf(&a+&b);
   &return
%Mend;

Data _Null_;
   s=%addiere(1,2);
   Put s=;
Run;
```

Mit der Prozedur FCMP können Funktionen entwickelt werden, die nicht nur SAS-Syntax nutzen, sondern auch auf DLLs, C- und Java-Programme zugreifen. Wie Sie beispielsweise das Alter berechnen können, wenn das Geburtsdatum von Patienten und

ein Stichtag gegeben sind, können Sie unter [6] nachlesen. Weitergehende Anwendungen der Prozedur FCMP siehe [5].

Literatur

[1] S. Annapareddy: Finding the number of observations in a SAS dataset
http://studysas.blogspot.de/2008/08/finding-number-of-observations-in-sas.html

[2] B. Gigic, A. Deckert: SAS Backstage.
In: Proceedings zur 16.Konferenz der SAS-Anwender in Forschung und Entwicklung, S. 103-111. Shaker-Verlag, 2012.

[3] B. Gigic: SAS DATA Step – Optionen vs. Anweisungen.
In: Proceedings zur 17. Konferenz der SAS-Anwender in Forschung und Entwicklung, to appear. Shaker-Verlag, 2013.

[4] A.X. Li: The Essence of DATA Step Programming
SAS Global Forum 2011, Paper 269-2011
http://support.sas.com/resources/papers/proceedings11/269-2011.pdf

[5] A. Menrath: Schöne neue Welt – So können Sie fehlende SAS Funktionalitäten mit PROC FCMP nachrüsten.
In: Proceedings zur 17. Konferenz der SAS-Anwender in Forschung und Entwicklung, to appear. Shaker-Verlag, 2013.

[6] C. Ortseifen et.al.: Tipps und Tricks für den leichteren Umgang mit SAS
13. KSFE 2009 Halle
http://de.saswiki.org/wiki/PROC_FCMP_%E2%80%93_Eigene_Funktionen_im_Datenschritt_erstellen

[7] G. Pfister: Der Data Step – das unbekannte Wesen?
Vortrag im Rahmen des SAS-Treff am URZ am 01.02.2002
http://www.urz.uni-heidelberg.de/statistik/kurse_sastreff.html

[8] S.D. Riba: The DATA Statement: Efficiency Techniques
SUGI 24 (1999), Paper 71
http://www2.sas.com/proceedings/sugi24/Begtutor/p71-24.pdf