Tipps und Tricks für einen leichteren Umgang mit der SAS Software

Carina Ortseifen	Grischa Pfister	Olaf Schoffer	Lars Tschiersch
Universität Heidelberg	ICASUS GmbH	Universität Dortmund	Universität Dortmund
Universitätsrechen-zentrum	Vangerowstraße	Lehrstuhl Statistik und Ökonometrie	Institut für Wirtschafts- und Sozialstatistik
INF 293	69117	Vogelpothsweg 87	Vogelpothsweg 78
69120 Heidelberg	Heidelberg	44221 Dortmund	44221 Dortmund
carina.ortseifen@ urz.uni-heidelberg.de	g.pfister@icasus. de	schoffer@statistik. uni-dortmund.de	tschiersch@statistik. uni-dortmund.de

Zusammenfassung

In Form von Kurzvorträgen werden nützliche Lösungen zu Problemen und Fragestellungen vorgestellt, die bei der täglichen Arbeit mit der SAS Software auftreten können. Es werden dabei nicht unbedingt neue Prozeduren, Optionen oder Module vorgestellt. Stattdessen soll die effektive Anwendung vorhandener Anweisungen und Prozeduren an Beispielen aufgezeigt werden.

Keywords: Tabulator-getrennte Werte, Infile, Delimiter=, Hexadezimalkodierung, Rank(), Input, Doppelpunkt, Output, Set, Lag(), Expand, Kummulieren, Standardisierung, Stdize, Missing value, ODS Rtf, Grafiken, Vektorformat, True Type Schriften, Automatisierter Grafikimport, Streudiagramm, Pointlabel, Setinit, Notfall-Setinit.

Inhalt

1. Tabulator getrennte Werte einlesen
2. Doppelpunkt im Input-Statement
3. Datenwerten eine Struktur zuweisen
4. Einzelne Werte allen Beobachtungen zuweisen
5. Duplikate löschen (Lag-Funktion)
6. Werte kumulieren mit und ohne der Prozedur Expand

7. Werte standardisieren mit und ohne der Prozedur Stdize
8. Der Wert von Missing Values (Mathematik außer Kraft gesetzt?)
9. ODS Rtf und die automatisierte Einbettung von SAS/GRAPH Output
10. Einzelne Punkte im Streudiagramm beschriften
11. Einspielen des Setinits

1 Einlesen von Datensätzen mit Tabulator als Trennzeichen

von Olaf Schoffer und Lars Tschiersch

Das Einlesen von tabulator-separierten (*tab-delimited*) Dateien ist in `Proc Import` mit der Option `Dbms=Tab` vorgesehen.

```
Proc Import Datafile=Dateiname Out=SAS-Datensatz
            Dbms=Tab Replace;
   Getnames=yes;
Run;
```

Dies ist jedoch auch sehr einfach mit dem `Infile`-Statement im DATA-Step möglich. Dazu muss aber der Tabulator als Trennzeichen in der `Delimiter`-Option angegeben werden: Zeichen '09' hexadezimal kodiert!

```
Data dummy;
   Infile Dateiname Delimiter='09'x;
   Input variable1 variable2;
Run;
```

Weitere mögliche Trennzeichen können beispielsweise codiert werden mit:

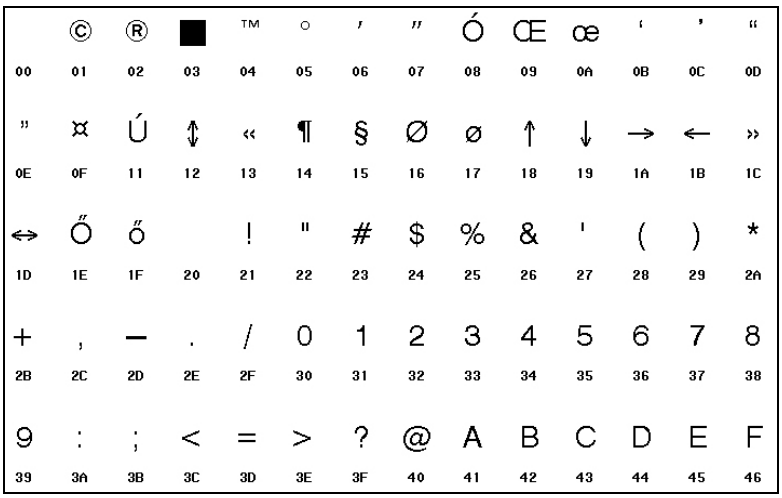
```
       ,      '2C'x
       ;      '3b'x
       :      '3a'x
Leerzeichen   '20'x
```

Natürlich lassen sich diese Zeichen auch ohne Hexadezimalcodierung als `','`, `';'`, `':'` bzw. `' '` (d.h. ohne `X`) angeben.

Zur gemeinsam Darstellung aller Zeichen einer beliebigen SAS-Schriftart mit ihrem Hexadezimalcode kann `Proc Gfont` verwendet werden.

```
Proc Gfont Name=SAS-font Nobuild
   Romhex;
Run;
```

Mit der Angabe von `Romhex` werden die Hexadezimalcodes direkt unter das jeweilige Zeichen gesetzt. Die Zeichen der SAS-Schriftart entsprechen jedoch nicht vollständig der Zeichentabelle des jeweiligen Systems.



Abschließend sei noch erwähnt, dass sich die Zeichencodierung zwischen den Systemen (z. B. Mainframe und Windows-PC) unterscheiden kann. Zuverlässige Informationen liefert die Funktion `Rank` für die vom System verwendete *collating sequence* (genaueres siehe SAS OnlineDoc).

2 Verwendung des Doppelpunkt im INPUT-Statement

von Olaf Schoffer und Lars Tschiersch

Ist ein Datenwert zu lang oder zu kurz für ein vorgegebenes Informat, gibt es beim Einlesen oft Fehler oder der Datenwert wird abgeschnitten. Die Verwendung des Doppelpunkt zwingt SAS dazu, in solchen Fällen den folgenden Datenwert nach dem nächsten Trennzeichen (z. B. Leerzeichen) einzulesen und nicht bereits innerhalb des zu langen Datenwerts.

Der Doppelpunkt wird dazu im `Input`-Statement zwischen der Definition einer Variablen und dem Informat angegeben. Das Informat `$27.` dient hier dem Einlesen von Zeichenketten der Länge 27, `3.1` dem Einlesen numerischer Werte mit einer Ziffer vor und einer Ziffer nach dem Komma. Durch `:` werden Zeichenketten der Länge 27 trotz voranstehender Leerzeichen korrekt eingelesen. Längere Zeichenketten werden jedoch nach der 27. Stelle abgeschnitten. Für das numerische Informat bewirkt `:`, dass nur die Anzahl der Nachkommastellen entscheidend ist.

```
Data a;
  Input a : $27. b : 3.1;
  Datalines;
   dies_ist_ein_zu_langes_Wort 1234.5
   kurzes_Wort 3.4
Run;

Proc Print;
Run;
```

Output:

```
Obs    a                                b

 1     dies_ist_ein_zu_langes_Wort    1234.5
 2     kurzes_Wort                       3.4
```

Das Ersetzen der zweiten Programmzeile durch

(i) `Input a $27. b : 3.1;`
(ii) `Input a : $27. b 3.1;`

ergibt folgenden Output:

(i)
```
Obs    a                           b
 1     dies_ist_ein_zu_langes_W    .
```

(ii)
```
Obs    a                               b
 1     dies_ist_ein_zu_langes_Wort   12.3
 2     kurzes_Wort                    3.4
```

Das Weglassen des Doppelpunkt führt also für jede der beiden Variablen zu unerwünschten Ergebnissen.

Übrigens sind auch `'&'` bzw. `'~'` an gleicher Stelle wie der Doppelpunkt anwendbar. Bei Verwendung von `&` werden Zeichenketten einschließlich(!)

einzelner Leerzeichen eingelesen. Das Trennen von Folgewerten erfolgt dann mittels zweier hintereinanderstehender Leerzeichen. Wird hingegen ~ (Tilde) benutzt, werden Leerzeichen innerhalb von " ... " nicht als Trennzeichen, sondern als Buchstabe gelesen und die Anführungsstriche (") am Anfang und Ende der Zeichenkette entfernt.

3 Zuordnen einer Struktur zu Datenwerten, die im DATA-Step eingelesen werden

von Olaf Schoffer und Lars Tschiersch

Mitunter gibt es im Bereich der Varianzanalyse bzw. Versuchsplanung vorgegebene Daten, denen eine Struktur zugeordnet ist, welche in den Daten jedoch nicht enthalten ist. Die nachfolgende Strukturierung des DATA-Steps ermöglicht die effiziente Zuordnung dieser Struktur zu den einzulesenden Daten. Strukturierung bedeutet hierbei, dass Klassifikationsvariablen den Datensatz gemäß einer von außen vorgegebenen Struktur in Gruppen bzw. Klassen unterteilen.

Beispiel: Die vorgegebenen Datenwerte 2, 3, 4, 5, 6, 7 sind zunächst in drei gleich große Gruppen und anschließend innerhalb dieser Gruppen nochmals in zwei Untergruppen aufzuteilen.

Die Strukturierung wird erzeugt, indem das `INPUT`-Statement innerhalb von `DO`-Schleifen eingesetzt wird, welche die Struktur vorgeben. In jedem Schleifendurchlauf wird genau eine Beobachtung eingelesen (nicht ein Datenwert) und gemeinsam mit dem aktuellen Wert der Schleifenvariablen `I` und `J` mittels des `OUTPUT`-Statements in den Datensatz geschrieben.

```
Data a;
   Do i=1 To 3;
      Do j=1 To 2;
         Input y @@;
         Output;
      End;
   End;
   Datalines;
2 3 4 5 6 7
;
Run;
```

```
Proc Print;
Run;
```

Output:

```
Obs    i    j    y

 1     1    1    2
 2     1    2    3
 3     2    1    4
 4     2    2    5
 5     3    1    6
 6     3    2    7
```

4 Zuordnen einzelner Werte eines Datensatzes zu allen Werten eines anderen Datensatzes

von Olaf Schoffer und Lars Tschiersch

Mitunter sollen alle Beobachtungen eines Datensatzes mit einem einzelnen Wert oder einer einzelnen Beobachtung aus einer vorherigen Analyse verglichen werden. Dazu ist dieser Wert/diese Beobachtung allen Beobachtungen des Datensatzes zuzuordnen. Diese Zuordnung ist neben dem Vergleich beispielsweise auch für Standardisierungen oder die Selektion von Beobachtungen verwendbar.

Gegeben seien folgende Datensätze:

```
Data Beobachtung;
   x=Normal(1);
Run;

Data Datensatz;
   Do i=1 To 10;
      Output;
   End;
Run;
```

Die Zuordnung der Beobachtung aus Datensatz `BEOBACHTUNG` zu allen Beobachtungen von Datensatz `DATENSATZ` erfolgt in DATA-Step `ZUSAMMEN`. Die Zeile `If _N_=1 Then Set Beobachtung;` bewirkt, dass die Werte aus Datensatz `BEOBACHTUNG` in den Datensatz `ZUSAMMEN` übernommen werden, sobald die erste Zeile mittels `SET DATENSATZ;` eingelesen wird. Für die übrigen Zeilen werden die Werte der Beobachtung aus Datensatz `BEOBACHTUNG` beibehalten (implizites retain). Der Selektion von Beobachtungen mittels `If i>x;` dient zur Veranschaulichung des Nutzens einer solchen Zuordnung.

```
Data Zusammen;
   If _N_=1 Then Set Beobachtung;
   Set Datensatz;
   If i>x;
Run;
Proc Print;
Run;
```

Output:

```
Obs        x         i

 1     1.80482      2
 2     1.80482      3
 3     1.80482      4
 4     1.80482      5
 5     1.80482      6
 6     1.80482      7
 7     1.80482      8
 8     1.80482      9
 9     1.80482     10
```

Vergleiche mit Werten aus vorherigen Analysen sind ebenfalls mittels Makrovariablen realisierbar. Der Programmieraufwand dazu ist nur wenig größer, die Wahl des Verfahrens ist somit „Geschmackssache“.

```
Data temp;
   Set Beobachtung;
   Call Symput('x',x);
Run;
```

```
Data zusammen;
   Set datensatz;
   If i>&x;
Run;
Proc Print;
Run;
```

Output:

```
Obs     i

 1      2
 2      3
 3      4
 4      5
 5      6
 6      7
 7      8
 8      9
 9     10
```

Vergleiche auch: Tipps&Tricks-Beitrag KSFE 2000 (Gießen) von H. Stürzl: „Special Merge (one-to-many)“.

5 Löschen von Duplikaten (Anwendung der LAG-Funktion)

von Olaf Schoffer und Lars Tschiersch

Die `LAG`-Funktion „verzögert“ zeitlich angeordnete Daten um einen Zeitpunkt. Sie verschiebt also die Merkmalswerte einer Variablen nach unten. Die `LAG`-Funktion benötigt jedoch nicht unbedingt zeitlich angeordnete Daten. Sie kann auch für andere Daten sinnvoll angewendet werden. Beispielsweise zur Löschung von Duplikaten oder zum Vergleich eines Merkmalswerts mit seinem Vorgänger. Nachfolgend wird die Löschung von Duplikaten mittels `LAG`-Funktion vorgestellt (erhalten bleiben nur die Beobachtungen, in denen der jeweilige Wert zum ersten Mal auftritt).

Gegeben sei der Datensatz:

```
Data a;
   Do i=1 To 100;
      X=Ranbin(1,80,0.6);
      Output;
   End;
Run;
```

Dazu wird der Datensatz zunächst gemäß der Variable `X` sortiert, die keine Duplikate aufweisen soll. Im DATA-Step `C` werden dann durch die Zeile `If x Ne Lag(x);` nur Beobachtungen übernommen, deren Merkmalswert für die Variable `X` sich von ihrem Vorgänger unterscheidet. Abschließend wird die ursprüngliche Reihenfolge wieder hergestellt (durch Sortieren nach `I`).

```
Proc Sort Data=a Out=b;
   By x; Run;
Data c;
   Set b;
   If x Ne Lag(x); Run;
Proc Sort;
   By i; Run;
Proc Print; Run;
```

Output:

Obs	i	x	Obs	i	x
1	1	52	14	31	45
2	2	40	15	33	41
3	3	49	16	48	53
4	4	51	17	63	57
5	5	42	18	69	54
6	7	48	19	98	56
7	9	55			
8	11	44			
9	13	43			
10	16	50			
11	18	46			
12	19	39			
13	23	47			

Anmerkung: Dieses spezielle Problem der Entfernung von Duplikaten wäre effektiver mit der Option `NODUPKEYS` in der ersten `SORT`-Prozedur umzusetzen (dann kann DATA-Step `C` entfallen). Da die `LAG`-Funktion auch für andere Anwendungen flexibel einsetzbar ist, wurde jedoch bewusst auf diese „Abkürzung" verzichtet.

Alternativ kann die `FIRST.<Variablenname>`-Funktion verwendet werden. Dazu wird DATA-Step `C` durch folgenden ersetzt:

```
Data c;
   Set b;
   By x;
   If First.x=1;
Run;
```

Vergleiche auch: Tipps&Tricks-Beitrag KSFE 2000 (Gießen) von H. Stürzl: „Dubletten Check".

6 Kumulieren von Werten einer Spalte im DATA-Step und mit PROC EXPAND

von Olaf Schoffer und Lars Tschiersch

Das fortlaufende Kumulieren von Werten einer Variable findet nicht nur in der Zeitreihenanalyse Anwendung. Auch für die Berechnung der Lorenzkurve, für die empirische Verteilungsfunktion und darauf basierende Tests, in der Qualitätskontrolle (CUSUM-Karte) und kaufmännische Anwendungen wird es benötigt.

Im DATA-Step werden jedoch neue Variablen aus den vorhandenen üblicherweise „horizontal" berechnet, d.h. der *i*-te Merkmalswert der neuen Variable ergibt sich aus den *i*-ten Merkmalswerten der anderen Variablen. Flexibler ist da bereits die `LAG`-Funktion. Zur Berechnung der fortlaufenden kumulierten Summe eignet sie sich aber auch nicht.

Gegeben sei der Datensatz:

```
Data a;
   Do i=1 To 10;
      Output;
   End;
Run;
```

Nachfolgend werden verschiedene Methoden im DATA- und PROC-Step vorgestellt, die Merkmalswerte eines Datensatzes fortlaufend zu kumulieren. Zunächst seien zwei Möglichkeiten im DATA-Step `B` präsentiert (das `RETAIN`-Statement ist nur für die zweite der beiden Möglichkeiten notwendig). In der ersten Möglichkeit wird eine implizite Variablendeklaration mit implizitem retain verwendet, was bei der zweiten Möglichkeit explizit durchgeführt wird.

```
Data b;
   Set a;
   i_sum_1+i;                  /* Möglichkeit 1 */
   Retain i_sum_2 0;
   i_sum_2=Sum(i_sum_2,i);     /* Möglichkeit 2 */
Run;
Proc Print;
Run;
```

Output:

```
Obs     i    i_sum_1    i_sum_2

  1     1        1          1
  2     2        3          3
  3     3        6          6
  4     4       10         10
  5     5       15         15
  6     6       21         21
  7     7       28         28
  8     8       36         36
  9     9       45         45
 10    10       55         55
```

Eine weitere Möglichkeit, Merkmalswerte zu kumulieren, bietet der PROC-Step mit der Prozedur `EXPAND` aus SAS/ETS. Das `CONVERT`-Statement wandelt numerische Variablen gemäß der gewählten Optionen um. Die Option `TRANSFORM=(SUM)` summiert die Merkmalswerte der betrachteten Variablen fortlaufend auf.

```
Proc Expand Data=a Out=c;
   Convert i=i_sum_3 /Transform=(Sum);
Run;
```

```
Proc Print;
Run;
```

Output:

```
Obs    TIME    i_sum_3     i

  1      0         1       1
  2      1         3       2
  3      2         6       3
  4      3        10       4
  5      4        15       5
  6      5        21       6
  7      6        28       7
  8      7        36       8
  9      8        45       9
 10      9        55      10
```

7 Standardisierung mit und ohne PROC STDIZE

von Olaf Schoffer und Lars Tschiersch

Zur Standardisierung von Variablen wählt man intuitiv die STANDARD-Prozedur. Sie bietet jedoch nur die Möglichkeiten der Standardisierung gemäß Mittelwert und Varianz. Daher wurde zunächst das weit flexiblere Makro STDIZE[59] entwickelt. In der SAS-Version 7 wurde daraus die Prozedur STDIZE.

Gegeben sei der Datensatz:

```
Data a (Drop=i);
   Do i=1 To 10;
      x=Ranbin(1,100,0.5);
      y=x;
      Output;
   End;
Run;
```

Die (nichtnegativen) Merkmalswerte der Variable Y werden nun als Prozentzahlen standardisiert, sodass der größte Wert gerade die 100% repräsentiert. Die Standardisierung erfolgt mit den Optionen METHOD=MAXABS (alle

[59] http://ftp.sas.com/techsup/download/stat/stdize.sas

Merkmalswerte werden durch den größten Wert geteilt) und `MULT=100` (die standardisierten Merkmalswerte werden mit 100 multipliziert) der Prozedur `STDIZE`. Eine solche Standardisierung ist beispielsweise für die Konstruktion von Lorenzkurven notwendig. Die Lorenzkurve ist ein Diagramm zur Darstellung der Konzentration, beispielsweise von Einkommen. Abgetragen wird hier der kumulierte Anteil der Bevölkerung gegen den kumulierten Anteil des Einkommens.

Anmerkung: Da `METHOD=MAXABS` den betragsmäßig größten Wert zu 1 bzw. -1 standardisiert, lässt sich das Verfahren bei Vorliegen negativer Werte nicht sinnvoll auf das gestellte Problem anwenden.

```
Proc Stdize Method=Maxabs Mult=100 Out=b;
   Var y;
Run;
Proc Print;
Run;
```

Output:

```
Obs     X       y

  1    46     77.966
  2    59    100.000
  3    49     83.051
  4    47     79.661
  5    57     96.610
  6    59    100.000
  7    51     86.441
  8    50     84.746
  9    42     71.186
 10    42     71.186
```

Für die `STDIZE`-Prozedur gibt es verschiedenste Methoden: `MEAN, STD, RANGE, MAD, MEDIAN, MAXABS` usw.

Die gleiche Standardisierung ohne die `STDIZE`-Prozedur erfordert mehr Aufwand:

```
Proc Means Noprint;
   Var y;
   Output Out=c Max=max;
Run;
```

```
Data d;
   If _N_=1 Then Set c;
   Set a;
   Y=x/max*100;
Run;

Proc Print;
Run;
```

Output:

Obs	_TYPE_	_FREQ_	max	x	y
1	0	15	59	46	77.966
2	0	15	59	59	100.000
3	0	15	59	49	83.051
4	0	15	59	47	79.661
5	0	15	59	57	96.610
6	0	15	59	59	100.000
7	0	15	59	51	86.441
8	0	15	59	50	84.746
9	0	15	59	42	71.186
10	0	15	59	42	71.186

8 Der Wert von missing values (Mathematik außer Kraft gesetzt?)

von Olaf Schoffer und Lars Tschiersch

Weshalb sollte man die folgende Umformung anzweifeln?

$$a \leq b \Leftrightarrow a - b \leq 0$$

Wenn missing values (`.`) ins Spiel kommen, gilt diese Umformung nicht mehr!

```
Data a (Keep=x);
   Do i=1 To 5;
      x=1;
      Output;
   End;
Run;
```

```
Data b;
   Set a;
   If x Le Lag(x) Then z1=1;
   If x-Lag(x) Le 0 Then z2=1;
Run;

Proc Print;
Run;
```

Output:

```
Obs    x    z1    z2

  1    1     .     1
  2    1     1     1
  3    1     1     1
  4    1     1     1
  5    1     1     1
```

Offensichtlich ist `Z1` in der ersten Beobachtung nicht identisch zu `Z2`.

Ursache dieses (scheinbaren) Paradoxons ist der durch die `LAG`-Funktion erzeugte missing value in der ersten Beobachtung. Für die übrigen Beobachtungen ist sowohl $1 \leq 1$ als auch $1 - 1 \leq 0$ wahr. In der ersten Beobachtung ergibt `LAG(X)` aber einen missing value. Der Vergleich lautet hier $1 \leq .$ bzw. $1 - . \leq 0$. Da $1 - .$ aber einen missing value ergibt, lautet der zweite Vergleich $. \leq 0$. Laut SAS Online Help ist ein missing value aber kleiner als jede Zahl. Somit ist für die erste Beobachtung nur der zweite Vergleich wahr, nicht jedoch der erste.

9 ODS Rtf und die automatisierte Einbettung von SAS/GRAPH Output

von Grischa Pfister

Problemstellung

Mit Hilfe des Output Delivery Systems (ODS) soll (unter Windows) ein Word-Dokument erstellt werden, das verschiedene SAS-Ausgaben enthält, insbesondere Tabellen und Grafiken. Diese Grafiken sollen einerseits nachbearbeitbar sein – also in einem Vektor-Format vorliegen – andererseits aber

von der Qualität so überzeugen, dass ein Nachbearbeiten gar nicht notwendig ist.

Problem 1

SAS/GRAPH unterstützt das ODS, allerdings gibt es eine Schwierigkeit: Die im SAS verwendbaren True Type Schriftarten wie z.B. „Times New Roman“ oder „Arial“ gehen bei der Umwandlung in die RTF-Datei verloren. Statt eines Text-Objektes enthält die im RTF enthaltene Grafik eine Punktwolke, die nicht mehr editierbar ist (aber ein Editieren notwendig macht...).

Folgendes Programm zeigt das Problem:

```
Goptions
      ftitle="comic sans ms/bo"
      htitle=18pt
      ftext="arial/bo"
      htext=14pt
      dev=emf
;

Ods Rtf File = "c:/ods/page1.rtf";

Proc Print Data = Sashelp.Class;
Run;

Title1 "Anzahl der Obs/Geschlecht";

Proc Gchart Data = Sashelp.Class;
  Vbar sex / width=20;
  Run;
Quit;
Ods Rtf Close;
```

Zunächst werden einige Grafik-Optionen gesetzt. Die Schriftart für Titel und allen Text (ftitle, ftext) wird auf „Comic Sans MS“ eingestellt, das Suffix „/bo“ ist eine Option und steht für „bold“, also Fett-Druck (weitere Möglichkeit ist „it“,“italic“ für kursiv; Schriftart-Optionen werden durch „/“ getrennt, also z.B. „arial /bo/it“). Als Schriftgrößen werden 18 respektive 14 Punkte angegeben. Der Parameter „dev“ (Kurzform von „device“) sagt aus, in welchem Format die Grafik gespeichert werden soll. In diesem Falle ist das „EMF“ (enhanced metafile), eine Weiterentwicklung des bereits in Ver-

sion 6 verfügbaren WMF (windows metafile) Formates. Dabei handelt es sich um eine Vektorformat, das für Microsoft-Produkte optimiert ist. Das Vektorformat hat gegenüber den Pixel-basierten Formaten wie JPEG, GIF oder BMP den Vorteil, dass die Grafik weniger Speicherplatz benötigt und in den Office-Produkten oder den gängigen Grafikprogrammen nachbearbeitbar ist.

Leider sieht das Ergebnis dieses Programmes aber nicht wie erwartet aus. SAS kann die True Type Schriftart nicht in das ODS übernehmen. Ergebnis ist die bereits oben erwähnte „Punktwolke".
Ein Blick in das Results-Fenster zeigt jedoch, dass die Grafik korrekt erzeugt wurde. Das Problem liegt also tatsächlich an der Weiterverarbeitung durch das ODS.

Lösung 1 – Schriftarten erhalten

Da das ODS momentan nicht in der Lage ist, die Schriftart korrekt zu übernehmen, muss ein anderer Weg gefunden werden, wie eine Grafik korrekt in Word übernommen werden kann. Eine Lösung bietet SAS/GRAPH schon lange an, das ist der Export der Grafik in eine Datei. Diese Grafik-Datei kann dann wiederum in Word importiert werden.

Das folgende Programm demonstriert, dass SAS selbstverständlich in der Lage ist, eine korrekte EMF-Grafik zu erzeugen:

```
Ods Listing;

Filename gout "c:/ods/graph01.emf";
Goptions
      ftitle="comic sans ms/bo"
      htitle=18 pt
      ftext="comic sans ms/bo"
      htext=14 pt
      dev=emf
      gsfmode=replace
      gsfname=gout
      hsize = 10 cm
      vsize = 15 cm
;
Title1 "Anzahl der Obs/Geschlecht";
```

```
Proc Gchart Data = Sashelp.Class;
  Vbar sex / width=20;
  Run;
Quit;

Filename gout Clear;
Title;
Goptions reset = global;
```

Hier wird zunächst eine Zieldatei allokiert, in der die Grafik gespeichert wird (Anweisung Filename). Es folgen die oben besprochenen Grafik-Optionen, die noch um zwei Positionen ergänzt wurden: „gsfmode=replace“ sagt aus, dass die Zieldatei bei jedem Prozedur-Durchlauf neu geschrieben wird, „gsfname=out“ ist die Referenz auf die allokierte Datei.

Wenn das Programm abgearbeitet ist, kann die Grafik in Word über das Menü „Einfügen → Grafik → Aus Datei...“ in ein Dokument importiert werden. Jetzt sind die Schriftarten korrekt übertragen worden, die Grafik kann über den eingebauten Editor bearbeitet werden.

Problem 2

Diese Methode eignet sich für einzelne Grafiken, macht aber wenig Sinn, wenn eine Vielzahl von Grafiken in einem Dokument enthalten sein soll. Hier kann es auch noch zu einem weiteren Problem kommen: Word-Dokumente, die viele Grafiken enthalten, können unnötig viel Speicher beanspruchen (insb. wenn es sich um Pixel-Grafiken handelt). In Word gibt es deshalb die Möglichkeit, Grafiken nicht in das Dokument direkt zu speichern, sondern nur einen Link auf eine Grafik anzulegen, die dann getrennt von dem Dokument gespeichert ist.

Wie aber kann aus SAS heraus das Word-Dokument so erstellt werden, dass es zuvor erstellte Grafiken automatisch importiert?

Lösung 2 - Automatisierter Grafikimport

Die Lösung liegt in den Besonderheiten von ODS und der RTF-Spezifikation. Das ODS ermöglicht, dass aus SAS heraus generischer RTF-Code geschrieben wird. Der Programmierer hat also die Möglichkeit, direkt in die entstehende RTF-Datei einzugreifen und so zusätzliche Elemente hineinzuschreiben, die über das normale ODS nicht möglich sind. Allerdings besteht hier auch die Gelegenheit, die RTF-Datei durch fehlerhafte Eingaben so zu

beschädigen, dass sie nicht mehr geöffnet werden kann. Vorsicht ist also geboten.

Die RTF-Spezifikation (zu finden z.B. unter msdn.microsoft.com; nach rtf specification suchen) ihrerseits bietet einige wenige Felder an (die den Feldfunktionen von Word entsprechen), und eines dieser Felder dient dem Import von Grafiken bzw. der Einbettung von Grafiken mit Hilfe eines Links.

Die Syntax des RTF-Feldes ist relativ simpel:
{\field {*\fldinst INCLUDEPICTURE "name"\d}}

Die Klammerung "{\field {*\fldinst … }}" kennzeichnet den Ausdruck als Word-Feldfunktion, "INCLUDEPICTURE" ist der Name der Feldfunktion in Word, "name" ist der Name der einzubettenden Grafik (eventuell inklusive (relativer) Pfadangabe) und der Schalter „\d" besagt, das die Grafik nicht importiert, sondern stattdessen ein Link erzeugt werden soll.

Es sollten auf alle Fälle relative Pfade verwendet werden, da das Dokument sonst kaum auf andere Rechner übertragbar wäre.

Das ODS erkennt RTF-Sequenzen daran, dass der ausgegebene Text mit „{" beginnt. Alles was jetzt noch nötig ist, ist ein Data-Step, der die Feldfunktion in die geöffnete RTF-Datei schreibt.

Hier ist das vollständige Beispiel, das mit Hilfe der Makro-Sprache noch weiter automatisiert werden könnte:

```
Ods Listing;

Filename gout "c:/ods/graph01.emf";
*Filename gout "c:/ods/images/graph01.emf";
Goptions
      ftitle="comic sans ms/bo"
      htitle=18 pt
      ftext="comic sans ms/bo"
      htext=14 pt
      dev=emf
      gsfmode=replace
      gsfname=gout
      hsize = 10 cm
      vsize = 15 cm
;

Title1 "Anzahl der Obs/Geschlecht";
```

```
Proc Gchart Data = Sashelp.Class;
  Vbar sex / width=20;
  Run;
Quit;

Ods Listing Close;

Ods Noresults;

Filename gout Clear;
Title;
Goptions reset = global;

Ods Rtf file = 'c:/ods/page2.rtf';
Proc Print Data = Sashelp.Class;
Run;

Data _Null_;
  File Print;
  Put "{\field {\*\fldinst INCLUDEPICTURE
""graph01.emf""\d}}";
*  Put '{\field {\*\fldinst INCLUDEPICTURE
"images/graph01.emf" \* MERGEFORMAT \d }}';
Run;

Ods Rtf Close;
Ods Listing;
```

Neu ist hier der Data-Step, der mit „File Print“ in das Output-Fenster – und damit in das geöffnete ODS – schreibt. Zu beachten ist, dass sich die RTF-Datei und die Grafik im gleichen Verzeichnis befinden. Die beiden auskommentierten Statements zeigen, wie die Grafik in ein anderes Verzeichnis gespeichert und dann in der RTF-Datei ein relativer Link auf dieses Verzeichnis angegeben wird.

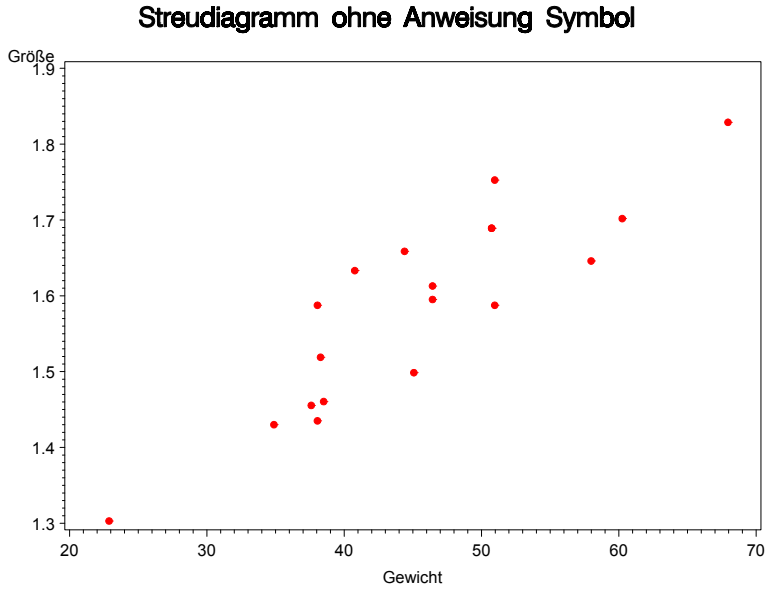
Das Statement „Ods Noresults“ verhindert, dass nach Schließen des RTF-Kanals das Ergebnis von SAS direkt angezeigt wird.

10 Einzelne Punkte im Streudiagramm beschriften

von Carina Ortseifen

Mit der Prozedur `GPLOT` können die Werte zweier Variablen in Form eines Streudiagrammes in einem Koordinatensystem grafisch veranschaulicht werden.
Im folgenden Beispielprogramm werden Größe und Gewicht von Schülern aus der Tabelle sashelp.class als rote Punkte dargestellt, nachdem eine Umrechnung in deutsche Maße erfolgte:

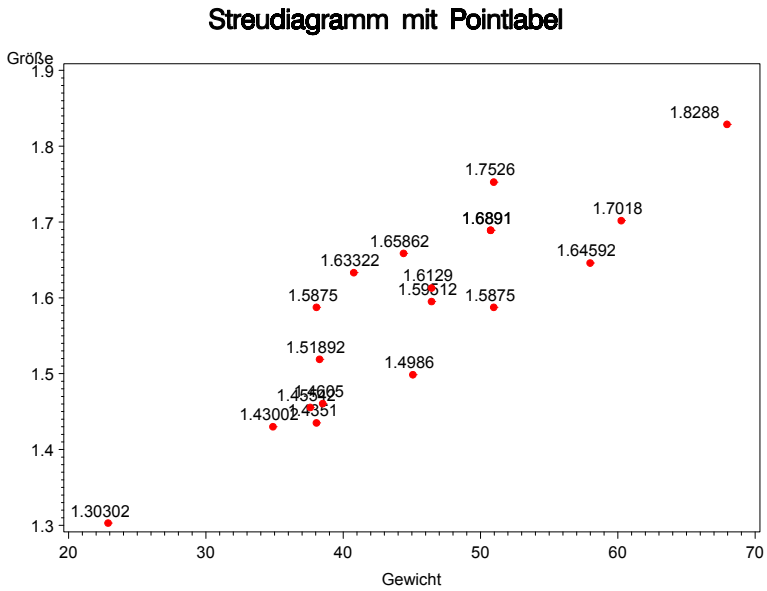
```
Data class;
   Set sashelp.class;
   height=height*2.54/100;
   weight=weight*0.453;
Run;
Title 'Streudiagramm ohne Anweisung Symbol';
Symbol V=Dot C=Red;
Proc Gplot Data=class;
   Plot height*weight;
   Label height='Größe' weight='Gewicht';
Run; Quit;
```



Möchte man die Punkte nun nicht einfach so stehen lassen, sondern in irgendeiner Form beschriften, musste man bislang auf Makros, Annotate-Dateien o.ä. zugreifen. Mit der Version 8.2 der SAS Software steht dafür nun die Option `POINTLABEL` der `SYMBOL`-Anweisung zur Verfügung.

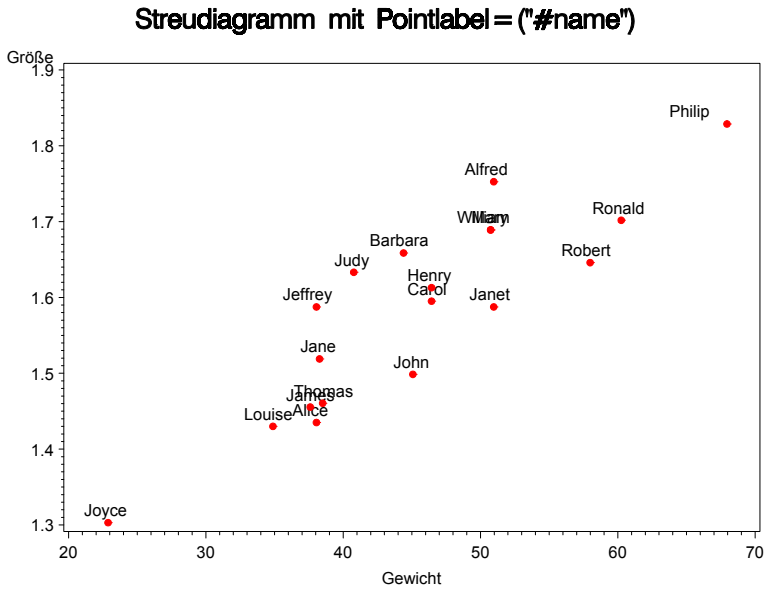
Mit der Option `POINTLABEL` können die Punkte beschriftet werden. Verwendet man die Option `POINTLABEL` ohne weitere Angaben, werden die Punkte mit den Werten der y-Variablen, hier Körpergröße, beschriftet.

```
Symbol V=Dot C=Red Pointlabel;
```



Alternativ kann man auch den Namen einer Variablen angeben (beispielsweise der Name der Schüler),

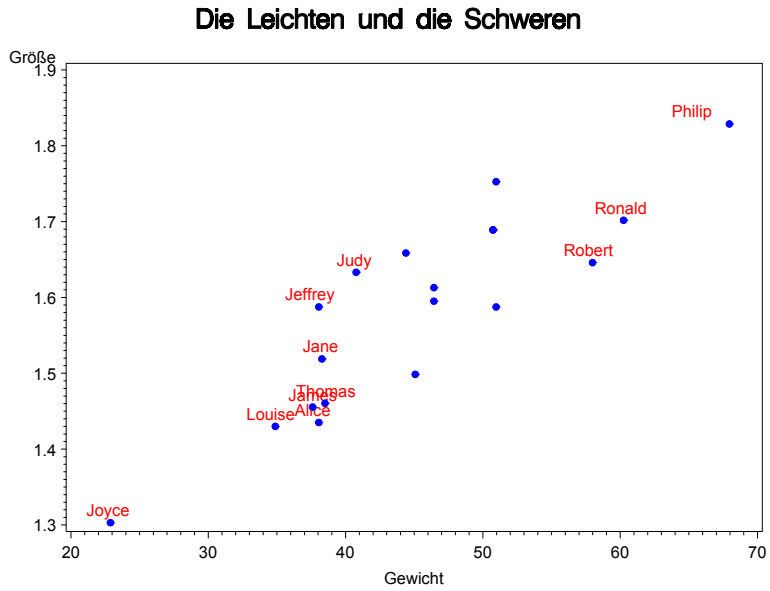
```
Symbol V=Dot C=Red Pointlabel=('#name');
```



so dass die Punkte im Streudiagramm mit den Schülernamen beschriftet werden.

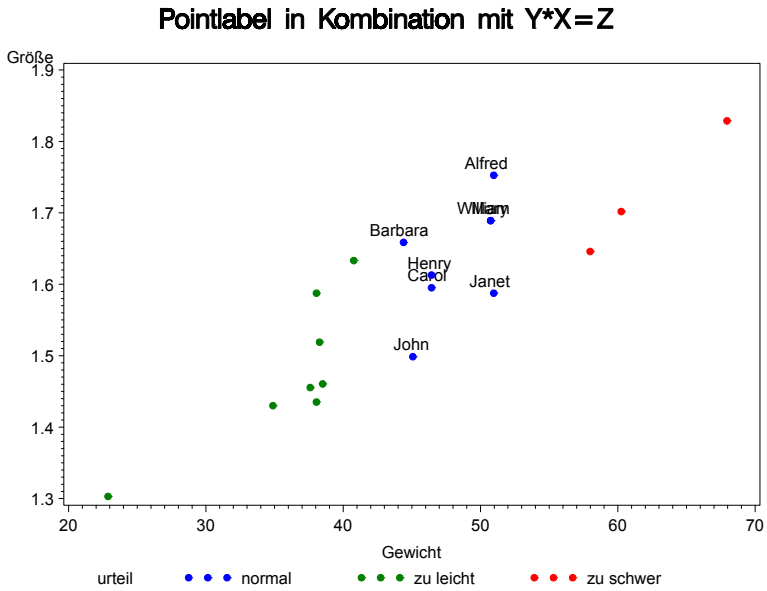
Sollen nur einzelne Punkte beschriftet werden, wird zunächst in einem Datenschritt eine neue Variable angelegt, die die entsprechenden Texte enthält, hier die Variable wert mit den Namen der Schüler, die einen Body Mass Index (bmi) kleiner 20 bzw. größer 25 haben::

```
Data class2;
   Set class;
   bmi=(weight/2)/(((height+100)/100)**2);
   If bmi<20 Or bmi>25 Then wert=name;
Run;

Symbol V=Dot C=Blue Pointlabel=('#wert' C=Red);
Proc Gplot Data=class2;
   Plot heigth*weight;
Run;Quit;
```



Im nächsten Beispiel werden die drei BMI-Gruppen (zu leicht, normal und zu schwer) farblich unterschiedlich dargestellt. Dazu werden drei Symbolanweisungen definiert und die Plot-Anweisung erweitert. Zusätzlich werden mittels der `POINTLABEL`-Option die normalgewichtigen Personen mit Namen gekennzeichnet.

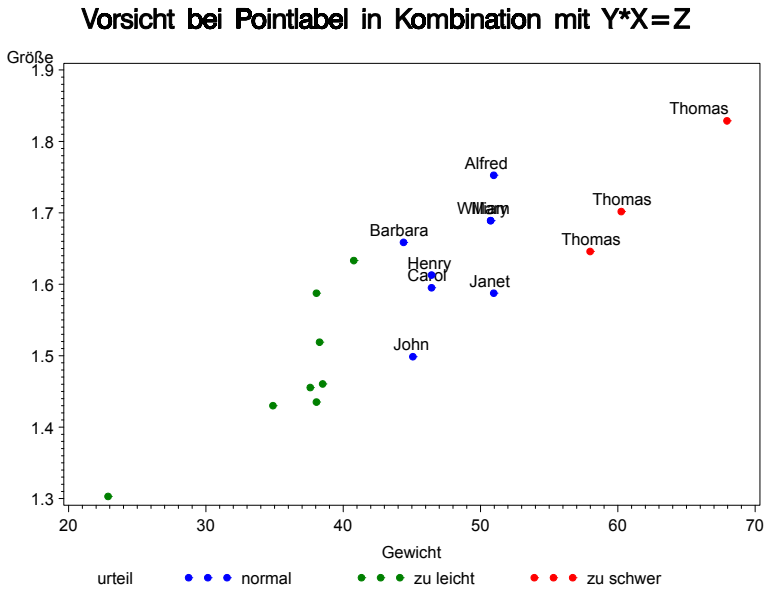
```
Data class3;
   Set class2;
   If bmi<20 Then urteil='zu leicht';
   Else If bmi <25 Then urteil='normal';
   Else urteil='zu schwer';
Run;

Title 'Pointlabel in Kombination mit Y*X=Z';
Symbol1 V=Dot C=Blue Pointlabel=('#name');
Symbol2 V=Dot C=Green
Symbol3 V=Dot C=Red;
Proc Gplot Data=class3;
   Plot height*weight=urteil;
   Label height='Größe' weight='Gewicht';
Run;Quit;
```



Aber genau in diesem Fall, Plot Y*X=Z, können Probleme auftreten, die tückisch sind, weil keine Fehlermeldung im Log-Fenster erscheint, wie das abschließende Beispiel zeigen soll.

Das obige Programm wird nur um eine `POINTLABEL`-Option in der dritten `SYMBOL`-Anweisung erweitert:

```
Title
   'Vorsicht bei Pointlabel in Kombination mit Y*X=Z';
Symbol1 V=Dot C=Blue Pointlabel=('#name');
Symbol2 v=Dot C=Green;
Symbol3 V=Dot C=Red Pointlabel=('#name');
Proc Gplot Data=class;
   Plot height*weight=urteil;
Run;Quit;
```



Und plötzlich ist nicht mehr Philip der große schwere, sondern dieser heißt laut Grafik Thomas. Und die beiden anderen rot gefärbten auch.

Zu diesem Fehlverhalten gibt es zahlreiche SAS Notes von SAS Institute (http://support.sas.com), u.a. SN-009736.

11 Einspielen des SAS Setinits

von Carina Ortseifen

Das Setinit ist für den autorisierten Betrieb der SAS Software erforderlich. Jeder Lizenznehmer erhält dieses mit dem Installationsmedium oder zur Verlängerung einer vorhandenen Installation (nach Begleichung der Rechnung über die Lizenzgebühren).

Das Setinit enthält neben den Angaben zum Lizenznehmer und dem Ablauf der Lizenzperiode die jeweilige SAS-Version, das Betriebssystem sowie die Namen der lizensierten SAS-Module.

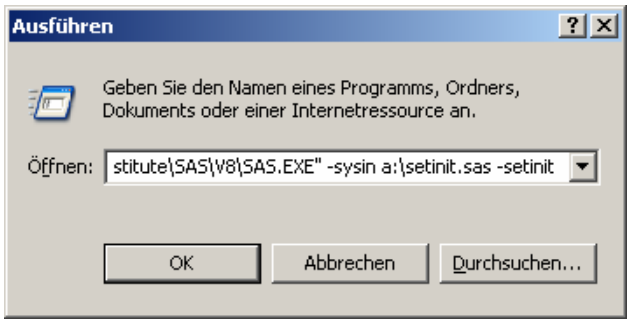
Auf manchen Installations-CD ist ein gültiges Setinit bereits vorinstalliert, z.B. bei der Learning Edition oder der Mining Challenge CD. In allen anderen Fällen jedoch enthält die Installations-CD ein veraltetes Setinit, das im

Laufe der Installation bzw. im Rahmen der Verlängerung durch ein aktuelles ersetzt werden muss.

Das Setinit besteht aus einem einzigen Prozedurschritt, `PROC SETINIT` mit mindestens den drei folgenden Anweisungen: `SITEINFO`, `CPU` und `EXPIRE`. Der Setinit-Prozedurschritt kann etwa folgende Form haben:

```
PROC SETINIT RELEASE='8.2';
   SITEINFO NAME='LANDESLIZENZ BADEN WUERTTEMBERG,
      SITE=87654321 OSNAME='WIN' RECREATE
      BIRTHDAY='23NOV1992'D EXPIRE='30APR2004'D
      PASSWORD=1234567890;
   CPU  MODEL=' ' MODNUM=' ' SERIAL=' '
      NAME=CPU000;
   EXPIRE 'BASE' 'GRAPH' 'ETS' 'FSP' 'AF' 'OR'
      'IML'  'QC' ' … ' '30APR2004'D / CPU=CPU000;
SAVE; RUN;
```

Wird die SAS Software neu installiert, überträgt man dieses Setinit in ein SAS-Programm, nennt dieses setinit.sss (unter Windows, bzw. setinit.sas) und gibt im Laufe der Installation den vollständigen Pfadnamen dieser Datei an.



Um eine bestehende SAS-Installation zu verlängern, kann dieser Prozedurschritt interaktiv im SAS-Editorfenster oder im Batchmodus ausgeführt werden. Unter Windows öffnet man für den Batchmodus beispielsweise über *Start* > *Ausführen* ein Kommandofenster und trägt dort das Kommando

```
"C:\Programme\SAS Institute\SAS\V8\SAS.EXE"
-sysin A:\setinit.sas -setinit
```

ein. Vorausgesetzt, SAS wurde standardmäßig installiert und die Setinit-Datei liegt im Hauptordner von Laufwerk A:.
Eine dritte Möglichkeit besteht darin, im Windows-Explorer mit dem Cursor

auf die Setinit.sss-Datei zu zeigen und über das rechte Maustastenmenü SAS im Batchmodus anzustoßen.

Die interaktive Variante, d.h. den Proc Setinit-Schritt ins SAS-Editorfenster übertragen und ausführen, funktioniert allerdings nur solange die alte Lizenz noch nicht ausgelaufen ist. Sind die drei Monate Kulanzzeit verstrichen, muss das Setinit im Batchmodus s.o. eingespielt werden.

Was kann man tun, wenn man die Lizenz bewusst nicht verlängert hat, etwa weil ein Systemwechsel stattfand, von z.B. Unix nach Windows und einem Anwender ein älteres Sicherungsband mit Unix-SAS-Tabellen in die Hände fällt, die dringend benötigt werden, ein SAS-Katalog mit wichtigen Eingabemasken und/oder Formaten vergessen wurde zu konvertieren oder ein Anwender mit ‚alten' Unix-Daten kommt?

Für diesen Fall kann man bei SAS Institute ein Notfall-Setinit erhalten, das für 6 Tage gilt. Entweder wendet sich der Systembetreuer direkt an seinen Betreuer bei SAS oder online über http://www.sas.com/getsetnow. Mit diesem Notfallsetinit kann die SAS für Unix-Software nochmals aktiviert und die Konvertierung vorgenommen werden.

Literatur

[1] Diskussionsbeitrag per Email von Adolf Quast über die SAS-Infos Diskussionsliste am 8.9.2003