Tipps und Tricks für den leichteren Umgang mit der SAS Software

Martin Kappler
BGFA
Bürkle-de-la-Camp-Platz 1
44789 Bochum
kappler@bgfa.ruhr-uni-bochum.de

Carina Ortseifen
Universitätsrechenzentrum Heidelberg
Im Neuenheimer Feld 293
69120 Heidelberg
carina.ortseifen@urz.uni-heidelberg.de

Grischa Pfister
iCASUS GmbH
Vangerowstraße 2
69115 Heidelberg
g.pfister@icasus.de

Heinrich Stürzl
Dade Behring Marburg GmbH
Emil-von-Behring Str. 76
35041 Marburg
Heinrich_Stuerzl@DadeBehring.com

Zusammenfassung

In Form von kurzen Beiträgen 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.

Schlüsselworte: Animierte GIFs, BINARYw., BINARY64., Binär, Binärformat, Bitmuster, Boolescher Wert, Boolesche Variable, DIR DOS-Kommando, DOS-Kommando DIR, Dualzahl, Funktionen, ODS (Output Delivery System), Perl Reguläre Ausdrücke, Projekt (Enterprise Guide), Reguläre Ausdrücke, SAS Enterprise Guide, SAS-Funktionen, SAS/GRAPH, Systemoption VALIDVARNAME=, Template (ODS), VALIDVARNAME= Systemoption, Verzeichnis einlesen.

Inhalt

1 Englischsprachiges SAS mit deutschem Output

(Grischa Pfister)

Mit Version 9 der SAS Software steht den Anwendern der sog. „national language support“ zur Verfügung, mit anderen Worten, es gibt wieder einmal eine deutschsprachige SAS-Version. Für erfahrene SAS Benutzer muss dies aber nicht unbedingt von Vorteil sein, denn nicht nur die Ausgabe im OUTPUT-Fenster erfolgt auf Deutsch (Beschriftung der Prozedurausgaben), sondern auch die Meldungen im LOG-Fenster werden entsprechend ausgegeben. Das kann die Fehlersuche aber deutlich erschweren, da ich nicht davon ausgehe, dass z.B. bei http://support.sas.com nach deutschprachigen Fehlermeldungen gesucht werden kann. Zudem liegt die aktuelle Dokumentation großteils in Englisch vor und Englisch ist auch die lingua franca der internationalen SAS community.
Vorteil des deutschsprachigen OUTPUT ist dagegen die bessere Lesbarkeit und die direkte Übernahme in andere Programme.
Daraus ergibt sich die Frage, ob es möglich ist, die Vorteile des deutschsprachigen OUTPUTs mit dem englischen LOG zu kombinieren?
Um dies beantworten zu können, müssen zwei beteiligte Elemente genauer betrachtet werden, der National Language Support (NLS) und das Output Delivery System (ODS).

Der National Language Support

Um den NLS im Sinne dieses Tipps zu nutzen, müssen bei der Installation von V9 mindestens zwei Sprachen installiert werden, z.B. Englisch und Deutsch.

SAS legt unter seinem Root-Verzeichnis eine Verzeichnisstruktur an (in diesem Falle „!sasroot\nls\de“ und „!sasroot\nls\en“), die die sprachspezifischen Teile enthält. Beim Aufruf von SAS wird die sasv9.cfg aus dem jeweiligen Unterverzeichnis verwendet, die dann die entsprechenden Ressourcen allokiert.

Das Output Delivery System

Das ODS kombiniert die Ausgabe von Prozeduren oder Data Steps mit einem table template zu einem ODS-Output-Objekt. Dieses Objekt wird dann weiterverarbeitet und mit Hilfe eines style templates für die geöffneten ODS-Kanäle interpretiert. Das table template enthält die wesentlichen Informationen über den tabellarischen Aufbau des Prozedur-Outputs, insbesondere auch die (deutschen bzw. englischen) Beschriftungen. Alle templates werden in itemstores gespeichert, das ist ein neuer SAS-Member Typ seit V8.

Die Lösung

Die Lösung besteht darin, ein englischsprachiges SAS mit den deutschsprachigen ODS templates zu kombinieren. Zu diesem Zweck wird ein Libname-Statement auf das Verzeichnis „!sasroot\nls\de\sashelp“ abgesetzt, hier befindet sich nämlich der itemstore „tmplmst.sas7bitm“, das ist der deutschsprachige „template master“. Mit Hilfe des ODS-Path-Statements wird der Suchpfad für die Templates erweitert, so dass die deutschsprachigen zuerst gefunden werden.

```
Libname german "!sasroot\nls\de\sashelp";
Ods path sasuser.templat (update) german.tmplmst sashelp.tmplmst;
```

Literatur

[1] SAS Online-Hilfe und SAS Online-Doc

2 Reguläre Ausdrücke aus Perl in SAS

(Martin Kappler)

Reguläre Ausdrücke bieten eine leistungsfähige und flexible Methode zur Textbearbeitung und –analyse. Mit ihnen können komplizierte Strukturen in Texten gefunden und ersetzt werden. Die Möglichkeiten von Regulären Ausdrücken gehen deutlich über die von SAS mitgelieferten Funktionen zur Textbearbeitung hinaus. Dennoch

gilt, fast alles was mit Regulären Ausdrücken machbar ist, ließe sich auch (oft allerdings wesentlich komplizierter) mit den bekannten Funktionen wie etwa `SCAN` oder `SUBSTR` realisieren.

Nachdem in SAS mit der Version 6 auf experimenteller Basis SAS-eigene Reguläre Ausdrücke (regular expression, RX) eingeführt wurden, ist diese Funktionalität in Version 9 erweitert um Reguläre Ausdrücke aus der Programmiersprache Perl (perl regular expression, PRX). Mit Einführung der Regulären Ausdrücke aus Perl soll eine

- geringere Fehleranfälligkeit
- leichtere Handhabung
- leichtere Lesbarkeit
- bessere Performance

der Regulären Ausdrücke erreicht werden.

In den Tipps und Tricks auf der KSFE 2003 [1] wurden die SAS-eigenen Regulären Ausdrücke bereits ausführlich vorgestellt. Da sich im Vergleich dazu mit den Regulären Ausdrücken aus Perl im wesentlichen nur die Syntax des Suchmusters verändert hat, wird im folgenden schwerpunktmäßig diese Syntax näher erläutert, während auf die Anwendungsgebiete und Verwendung der Funktionen im Data-Step kürzer eingegangen wird.

Eine genauere Beschreibung der Verwendung von Regulären Ausdrücke aus Perl in SAS findet sich in der OnlineDoc [2]. Ein Schnell-Einstieg und eine Kurz-Übersicht zu Regulären Ausdrücken aus Perl findet sich z.B. in [3]. Ein ausführlicheres Tutorial zum Thema findet sich in [4]. Dabei ist allerdings zu beachten, dass in SAS einiges der Perl-Funktionalität nicht verfügbar ist [2].

2.1 SAS Funktionen und Call-Routinen für Reguläre Ausdrücke aus Perl

Für die SAS-eigenen Regulären Ausdrücke sind die Funktionen und Call-Routinen RXCHANGE, RXMATCH, RXPARSE, CALL RXCHANGE, CALL RXSUBSTR und CALL RXFREE aus SAS 8 bekannt. Diese Funktionen stehen für Reguläre Ausdrücke aus Perl ebenfalls zur Verfügung und sind lediglich mit dem Kürzel PRX statt RX am Anfang versehen. Zusätzlich wurden für Reguläre Ausdrücke aus Perl folgende Funktionen und Call-Routinen eingeführt:

- PRXPOSN, PRXPAREN,
- CALL PRXPOSN, CALL PRXNEXT, CALL PRXDEBUG.

Genau wie für die SAS-eigenen Regulären Ausdrücke muss zur Verwendung der Regulären Ausdrücke aus Perl im Data-Step zunächst ein Suchmuster definiert werden. Dies geschieht mit PRXPARSE, wobei eine neue Variable geschaffen wird, die eine Referenz auf das Suchmuster legt. Um den Verbrauch von Speicherplatz zu minimieren und damit auch die Performance zu steigern, bietet es sich an, das Suchmuster nur ein einziges Mal zu Beginn des Datensatzes zu definieren. Wenn das Suchmuster nicht mehr benötigt wird, sollte der belegte Speicher mit CALL PRXFREE wieder frei gegeben werden.

```
DATA bsp;
  SET bsp END=final;
  RETAIN prx;
  IF _N_=1 THEN prx=PRXPARSE("/<PRX>/");
  ...
  Anwendung von PRX-Funktionen und sonstige DATA-Step Anweisungen
  ...
  IF final THEN CALL PRXFREE(prx);
  DROP prx;
RUN;
```

2.2 Aufbau eines Suchmusters

Die Suchmuster von Regulären Ausdrücken aus Perl sind grundsätzlich in Trennzeichen eingeschlossen. Im Regelfall werden dafür Schrägstriche / verwendet. Vorangestellt wird ein Kürzel, das kennzeichnet, was mit dem Suchmuster getan werden soll. Ein "m" wird für Suchen nach einem Muster verwendet (match) und ein "s" für Suchen und Ersetzen (substitution). Das "m" kann dabei auch weggelassen werden.

- `m/<PRX>/ = /<PRX>/`
- `s/<PRX>/<PRX>/`

Trennzeichen
Als Trennzeichen können neben / auch viele andere Zeichen verwendet werden. In diesem Fall darf "m" jedoch nicht weggelassen werden.

- `m&<PRX>&  = m%<PRX>%  = m^<PRX>^  = …`
- `s&<PRX >&<PRX>& = s%<PRX>%<PRX>% = s^<PRX>^<PRX>^ = …`

Die Verwendung von alternativen Trennzeichen ist insbesondere dann sinnvoll, wenn nach dem Schrägstrich selbst gesucht werden soll, dessen Sonderfunktion als Trennzeichen dann hinderlich ist.

Optionen
Hinter dem letzten Trennzeichen können Optionen gesetzt werden. Verschiedene Optionen können dabei durch Aneinanderreihung der Buchstaben kombiniert werden.

Am häufigsten findet wohl die Option "i" Verwendung, die bewirkt, dass ohne Unterscheidung von Groß- und Kleinschreibung nach dem eingegebenen Suchmuster gesucht wird.

- m/<PRX>/i = /<PRX>/i
- s/<PRX>/<PRX>/i

In den Regulären Ausdrücken aus Perl bedeutet die Eingabe eines Leerzeichens grundsätzlich, dass auch ein Leerzeichen gesucht werden soll, während in den SAS-eigenen Regulären Ausdrücken Leerzeichen zur Strukturierung und besseren Lesbarkeit des Suchmusters verwendet werden können. Durch die Option "x", die Leerzeichen ignoriert, ist dies auch für die Regulären Ausdrücke aus Perl möglich.

Die Option "o" bewirkt, dass ein Suchmuster im Data-Step nur ein einziges Mal definiert wird. Damit wird also der gleiche Effekt erzielt, wie im obigen Umweg über `RETAIN` und die Abfrage nach `_N_=1`.

Sonderzeichen
Wie auch schon für die SAS-eigenen Regulären Ausdrücke sind zur einfacheren Erstellung von Suchmustern eine Vielzahl von Teilmustern vordefiniert.

- \w Sucht ein Wortzeichen (a-z, A-Z, 0-9, _)
- \d Sucht ein Zahlzeichen (0-9)
- \s Sucht ein White-Space-Zeichen (Leerzeichen, Tabulator, …)
- \b Sucht das vorangehende Suchmuster an einer Wortgrenze
- \n Sucht nach einem Zeilenwechsel
- . Sucht ein beliebiges Zeichen (bis auf Zeilenwechsel)

Die Komplementärmenge dieser Sonderzeichen lässt sich i.A. einfach durch die Ersetzung des Klein-Buchstabens durch den entsprechenden Groß-Buchstaben suchen. So sucht z.B. \W nach allen Zeichen, die kein Wortzeichen sind, und \D nach allen Zeichen, die kein Zahlzeichen sind usw.

Operatoren
In den Suchmustern können eine Reihe von Operatoren verwendet werden. Die meisten beziehen sich darauf, wie oft ein bestimmtes Suchmuster gefunden werden soll.

- | Logisches Oder
- ^ Sucht ein Suchmuster am Beginn des Textes
  Ausnahme: Innerhalb von […] ist ^ ein logisches Nicht
- $ Sucht ein Suchmuster am Ende des Textes
- ? Sucht ein Suchmuster 0 oder 1 Mal
- * Sucht ein Suchmuster 0 oder mehrere Male
- + Sucht ein Suchmuster 1 oder mehrere Male
- {n} Sucht ein Suchmuster n Mal
- {n,m} Sucht ein Suchmuster n bis m Male

Gruppierungen
Eigene Suchmuster können durch Gruppierung von Zeichen konstruiert werden.

- […] Gruppiert einzelne Zeichen
  (`/[HLM]aus/` findet "Haus", "Laus" und "Maus")
- (…) Gruppiert verschiedene Suchmuster (und speichert das Gefundene)
  `/(Kl|Schm)aus/` findet 'Klaus' und 'Schmaus'
- [a-z] Sucht nach einem Kleinbuchstaben
- [K-M] Sucht nach 'K', 'L' oder 'M'
- [1-9] Sucht nach den Ziffern 1 bis 9

2.3 Beispiele von Suchmustern

```
/data /
```
Sucht nach dem Wort "data" gefolgt von einem Leerzeichen.

```
/data /i
```
Sucht nach dem Wort "data" ohne Unterscheidung von Groß- und Kleinschreibung gefolgt von einem Leerzeichen.

```
/1[89]\d{2}/
```
Sucht nach "1" gefolgt von "8" oder "9" gefolgt von genau zwei Ziffern, d.h. 1800 bis 1999.

```
s/ +/ /
```

Ersetzt mehrere Leerzeichen durch ein einziges.

```
/(\d{6})[- ](\w{1,3})/
```

Das Suchmuster besteht aus drei Gruppierungen. Die erste Gruppierung wird gebildet durch runde Klammern, in denen 6 Ziffern gesucht und das Gefundene zwischengespeichert wird. Die zweite besteht aus den eckigen Klammern in denen ein Bindestrich oder ein Leerzeichen gesucht wird. Die dritte Gruppierung besteht wieder aus runden Klammern, so dass das Gefundene ebenfalls zwischengespeichert wird. Hier werden ein bis drei Wortzeichen gesucht. Gefunden werden Texte vom Format "140299-BOT" oder "181098 D".

2.4 Anwendungsbeispiele

Basis für die folgenden Anwendungsbeispiele bildet ein Datensatz mit einer Textvariable, die Einträge enthält, die mit dem obigen Suchmuster gefunden werden.

```
DATA bsp;
  INPUT id $20.;
  DATALINES;
140299-BOT
181098 D
28195-DO
;
RUN;
```

2.4.1 Funktion PRXPOSN

Über die Funktion PRXPOSN ist es möglich, auf identifizierte Teilmuster zuzugreifen, die durch runde Klammern in einem Suchmuster zwischengespeichert wurden. Dazu wird im Funktionsaufruf die Nummer des Klammerpaares angegeben, das ausgelesen werden soll.

```
DATA bsp1;
  SET bsp END=final;
  prx=PRXPARSE("/(\d{6})[- ](\w{1,3})/o");
  IF PRXMATCH(prx, id) THEN DO;
    dat=PRXPOSN(prx, 1, id);
    ort=PRXPOSN(prx, 2, id);
  END;
  ELSE PUT "WARNUNG: " id " nicht korrekt!";
  IF final THEN CALL PRXFREE(prx);
RUN;
PROC PRINT; RUN;
```

Log: `WARNUNG: 28195-DO  nicht korrekt!`

Output:

```
Beob.       id       prx  dat     ort
  1    140299-BOT    1   140299  BOT
  2    181098 D      1   181098  D
  3    28195-DO      .
```

Die Funktion PRXMATCH gibt die Stelle zurück, an der das Suchmuster zum ersten Mal gefunden wurde und 0, falls das Suchmuster nicht gefunden werden konnte. Damit kann das Ergebnis von PRXMATCH auch als logische Variable für das Finden eines Suchmusters verwendet werden.

2.4.2 Funktion PRXPAREN

Die Funktion PRXPAREN gibt als Ergebnis die Nummer des Klammernpaares zurück, das als letztes identifiziert werden konnte. Dies kann sinnvoll sein, wenn aus Fließtext Schlüsselwörter gesucht und diese in eine kategorielle Variable übertragen werden sollen.

```
DATA bsp2;
  SET bsp END=final;
  prx=PRXPARSE("/(BOT\b)|(D\b)|(DO\b)/o"); /* Muster 1, ok */
  * prx=PRXPARSE("/(BOT)|(D)|(DO)/o");     /* Muster 2, ?? */
  * prx=PRXPARSE("/(BOT)|(DO)|(D)/o");     /* Muster 3, ok */
  IF PRXMATCH(prx,id) THEN ort=PRXPAREN(prx);
  IF final THEN CALL PRXFREE(prx);
RUN;
PROC PRINT; RUN;
```

Output

```
Beob.      id      prx  ort
  1    140299-BOT   1    1
  2    181098 D     1    2
  3    28195-DO     .    3
```

Vorsicht ist geboten, wenn die runden Klammern in einer Oder-Verknüpfung stehen wie im Beispiel. Dann bricht das Suchmuster ab, sobald eine Übereinstimmung gefunden wird. Wird die Angabe "\b" für die Suche an einer Wortgrenze weggelassen (Muster 2), so wird für die Beobachtung "28195-DO" das "D" gefunden und

PRXPAREN gibt als Ergebnis 2 statt der gewünschten 3 für "DO" zurück. Umgangen werden kann das Problem mit Zusatzinformationen in den Teilsuchmustern wie im Beispiel mit "\b" oder durch Umsortierung, so dass Teilsuchmuster, die Teilmengen von anderen sind, zuerst gesucht werden (Muster 3).

2.4.3 Funktion PRXCHANGE

Die Funktion PRXCHANGE ersetzt im durchsuchten Text das gefundene Suchmuster durch das Ersetzungsmuster. Hierfür wird in jedem Fall ein Regulärer Ausdruck für Suchen und Ersetzen "s" benötigt.

```
DATA bsp3;
  SET bsp END=final;
  prx=PRXPARSE("s/(\d{4})9(\d)/${1}199$2/o");
  id=PRXCHANGE(prx,1,id);
  IF final THEN CALL PRXFREE(prx);
RUN;
PROC PRINT; RUN;
```

Output

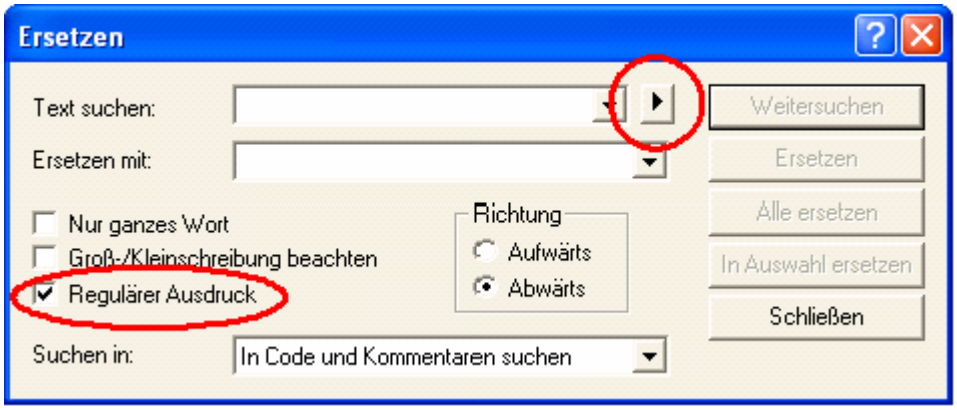
```
Beob.      id        prx
  1   14021999-BOT   1
  2   18101998 D     1
  3   28195-DO       .
```

Auf im Suchmuster gefundene Teilsuchmuster, die durch runde Klammern zwischengespeichert wurden, kann auch schon innerhalb des Suchmusters zugegriffen werden. Dies geschieht durch $x oder auch \x, wobei x für das x-te Klammerpaar im Suchmuster steht. Da im Beispiel eine Zahl auf das erste Einfügen der Zwischenspeicherung folgt, wird die Identifizierung des Klammerpaars von geschweiften Klammern umschlossen: ${1}.

2.5 Reguläre Ausdrücke im Enhanced Editor

Reguläre Ausdrücke lassen sich auch im "Suchen und Ersetzen"-Dialog des Enhanced Editor einsetzen, indem im entsprechenden Fenster die Checkbox "Regulärer Ausdruck" angewählt wird (Abb. 1).

Abbildung 1: "Suchen und Ersetzen"-Dialog des Enhanced-Editors



Vorsicht ist allerdings geboten, da sich die Syntax der hier verwendeten Regulären Ausdrücke sowohl von der SAS-eigenen Syntax als auch leicht von der Perl-Syntax unterscheidet.
Die wichtigsten Unterschiede zur Perl-Syntax sind hier aufgeführt:

- Eingabe des Suchmusters ohne Anführungs- und Trennzeichen
- Verwendung von (…), {n,m} und ? nicht möglich
- Sonderzeichen \b für White-Space-Zeichen (statt \s)
- Sonderzeichen \< und \> für Wortanfang und –ende (statt \b)

Um ohne genaue Kenntnis der Syntax schnelle Abfragen durchführen zu können, sind die wichtigsten Sonderzeichen direkt anwählbar über eine Schaltfläche rechts neben dem "Text suchen"-Feld (Abb. 1).

Anwendungsmöglichkeiten
Eine einfache Anwendung ist gegeben durch die Suche nach Wörtern, die immer gleich anfangen. Der Code dazu ist z.B.
\<file
um nach Wörtern zu suchen, die mit file beginnen.

Für die Suche nach der Verwendung von SAS-erzeugten Variablen oder Namen, die von Unterstrichen umschlossen sind, kann folgendes Suchmuster verwendet werden:

```
\<_\w+_\>
```

Dabei sucht \<_ einen Unterstrich am Wortanfang, \w+ ein oder mehrere Wortzeichen und _\> einen Unterstrich am Wortende.

Um potentielle Programmierfehler abzufangen, kann z.B. nach Zeilen gesucht werden, die nicht mit einem Semikolon abgeschlossen sind. Der Reguläre Ausdruck hierfür (in der Syntax des Enhanced Editors) wäre:

```
[^;\b] *\n
```

Am besten liest sich dieser Reguläre Ausdruck von hinten. Es wird nach einem Zeilenumbruch gesucht, vor dem 0 oder mehrere Leerzeichen stehen, vor dem weder ein Semikolon noch ein White-Space-Zeichen stehen.

Eine weitere sinnvolle Anwendungsmöglichkeit ist für das Problem gegeben, dass in SAS-Makros dem %else-Statement kein Kommentar der Form *...; vorangehen darf, da dann das zugehörige %if-Statement nicht mehr erkannt wird. Vorkommen von solchen Kommentaren vor einem %else können mit folgendem Regulären Ausdruck gefunden werden:

```
\*.*;\b*%else\b
```

Gesucht wird dabei zunächst ein * (als Sonderzeichen muss er mit \ demaskiert werden) gefolgt von 0 oder mehreren beliebigen Zeichen (.*), gefolgt von einem ; , womit der Kommentar identifiziert ist. Dem Kommentar folgen 0 oder mehrere White-Space-Zeichen, gefolgt von %else, gefolgt von einem White-Space-Zeichen.

Literatur

[1] W. Herff, C. Ortseifen, G. Pfister, H. Stürzl. *Tipps & Tricks für einen leichteren Umgang mit der Base SAS Software*. C. Becker, H. Redlich (Hrsg.), Proceedings der 7. Konferenz der SAS®-Anwender in Forschung und Entwicklung (KSFE), Shaker-Verlag, Aachen, 2003

[2] SAS Institute Inc. (2003): *Functions and Call Routines : Pattern matching using regular expressions (RX) and perl regular expressions (PRX)*. http://support.sas.com/91doc/getDoc/lrdict.hlp/a002288677.htm, SAS Institute Inc., Cary, NC, USA [22.02.2005]

[3] M. Kvale (2000). *perlrequick – Perl regular expressions quick start*. http://search.cpan.org/~nwclark/perl-5.8.6/pod/perlrequick.pod [22.02.2005]

[4] M. Kvale (2000). *perlretut – Perl regular expressions tutorial*. http://search.cpan.org/~nwclark/perl-5.8.6/pod/perlretut.pod [22.02.2005]

3 Animierte GIFs mit SAS/GRAPH

(Grischa Pfister)

Animierte GIFs stellen eine Aneinanderreihung verschiedener im GIF-Format gespeicherter Grafiken dar, die wie eine Dia-Show automatisch hintereinander abgespielt werden. Diese Darstellungsweise kann sinnvoll sein, um zeitliche Verläufe, gruppierte Auswertungen oder generell Prozesse darzustellen. Oft finden sich animierte GIFs im Internet, aber auch die Office-Programme von Microsoft können sie anzeigen. SAS/GRAPH stellt seit der Version 6.12 (da noch experimentell) einen speziellen Device zur Verfügung, der die Erstellung von animierten GIFs direkt aus SAS ermöglicht.

Der Device GIFANIM

Um ein animiertes GIF zu erstellen, müssen bestimmte Grafik-Optionen (GOPTIONS-Statement) in einer bestimmten Reihenfolge gesetzt werden. Die Zieldatei wird mit Hilfe eines Filename-Statements allokiert, über die GOPTION gsfname=*fileref* wird die Grafikausgabe entsprechend umgeleitet. Vor Erstellen der ersten Grafik wird zudem die GOPTION gsfmode=replace gesetzt, um eine eventuell schon bestehende Grafik zu überschreiben. Danach wird gsfmode=append gesetzt, denn der Trick besteht darin, dass die verschiedenen Grafiken in ein und derselben Datei gespeichert werden. **Vor** Erzeugen der letzten Grafik muss dann die GOPTION gepilog="3B"x gesetzt werden, damit das GIF als animiert erkannt wird. Im Programm sieht der Ablauf dann standardmäßig so aus:

```
Filename out 'pfad/name.gif';

Goptions
  reset     = all
  device    = gifanim
  gsfname   = out
  gsfmode   = replace;
* GP Erzeugen der ersten Grafik *;
...
Goptions gsfmode = append;
* GP Erzeugen der weiteren Grafiken *;
...
* GP Setzen von gepilog VOR der letzen Grafik *;
Goptions gepilog = "3B"X;
* GP erzeugen der letzten Grafik *;
...
```

Über eine Reihe von weiteren Grafik-Optionen werden die Animations-Parameter gesetzt:

Iteration = n	Anzahl der Durchläufe (0 bedeutet Endlosschleife)
Delay = n	Pause zwischen Grafiken in 1/100 sec
[No]Transparency	Bildhintergrund transparent schalten

Hier ein vollständiges Beispiel:

Mit Hilfe des FILENAME-Statements wird die Zieldatei allokiert.

```
Filename out 'd:/Temp/gpTest001.gif';
```

Die oben besprochenen Grafik-Optionen werden gesetzt, zusätzlich werden noch die Schriftarten und –größen für die Beschriftung vorgegeben, sowie die Ausgabe von deutschen Umlauten ermöglicht (keymap=winansi).

```
Goptions
  reset     = all
  device    = gifanim
  gsfname   = out
  gsfmode   = replace
  iteration = 0
  delay     = 150
  keymap    = winansi
  ftitle    = swiss
  htitle    = 14 pt
  ftext     = swiss
  htext     = 12 pt
;
```

Als erstes wird eine Titelgrafik erzeugt.

```
Title j=c "MDDWNB GmbH"
      j=c h=12 pt "Möbel, die die Welt nicht braucht -
Gesellschaft mit beschränkter Hoffnung"
      j=c " ";

Proc Gslide;
```

```
    Note h=40 pt
    j=c     ' '
    j=c     ' '
    j=c     'Umsatz 1993'
  ;
  Run;
Quit;
```

Jetzt wird gsfmode=replace gesetzt, damit die weiteren Grafiken an die Zieldatei angehängt werden.

```
Goptions gsfmode = append;
```

Über das BY-Statement werden sechs Grafiken erzeugt (drei Länder à zwei Regionen).

```
Proc Gchart data=Sashelp.Prdsale;
    Vbar product / sumvar = actual width=15;
    By country region;
  Run;
Quit;
```

Jetzt wird der Marker für die Animation gesetzt…

```
Goptions gepilog = '3B'x;
```

… und die letzte Grafik wird erzeugt.

```
Proc Gslide;
    Note h=40 pt
    j=c     ' '
    j=c     ' '
    j=c     'Ende'
  ;
  Run;
Quit;
```

Um das ganze anzeigen zu können, wird noch schnell eine HTML-Seite geschrieben, die das animierte GIF anzeigt.

```
Data _Null_;
  File "D:/temp/gpTest.html";
  Put "<html><body>" /
      "<center>" /
      "<H1> Animiertes GIF mit SAS/GRAPH </H1>" /
      "<HR>" /
      "<IMG src=""gpTest001.gif""">" /
      "<HR>" /
      "</center>" /
      "</body></html>";
Run;
```

Literatur

[1] SAS Online-Hilfe und SAS Online-Doc

[2] SAS Anwenderhandbuch im Netz:
http://web.urz.uni-heidelberg.de/statistik/sas-ah/
(Achtung, hier noch mit der Syntax für SAS 6.12!)

4 Probleme mit dem BINARY64. Format

(Heinrich Stürzl)

Das Format BINARY*w*. konvertiert numerische Werte in die Binärdarstellung. Beispielsweise entspricht der Zahl 7 die Dualzahl 0111, wobei die Ziffern von rechts nach links die aufsteigenden Potenzen 0, 1, 2, … zur Basis 2 bedeuten. Also 1*2^0 + 1*2^1 + 1*2^2 + 0*2^3 = 1 + 2 + 4 = 7.

Die Feldbreite w (width) steuert die Länge der Dualzahl (1-64). Die Zahl 7 mit BINARY64. besteht aus 61 Nullen und ‚111'. Mit SAS Version 6.12 funktioniert dies auch korrekt. **Ab Version 8 und auch in Version 9 liefert BINARY*w*. mit 59≤*w*≤64 zumindest für Integer Zahlen falsche Dualzahlen!** Im Log Fenster bekommt man leider KEINEN HINWEIS auf den Fehler.

Beispiel:

```
data _null_;
  zahl=7;
  bits=put(zahl, BINARY64.);
  put _all_;
run;
```

V6.12 (korrekt):

```
ZAHL=7
BITS=0000000000000000000000000000000000000000000000000000000000000111
_ERROR_=0 _N_=1
```

V8.2, V9.1.2, V9.1.3 (FALSCH):

```
zahl=7
bits=0100000000011100000000000000000000000000000000000000000000000000
_ERROR_=0 _N_=1
```

In der SAS Note SN-004572 wird diese Änderung als gewollt dargestellt, um Fließkommazahlen konsistent zum HEX*w*. Format darzustellen. Für Integer Zahlen führt diese Änderung definitiv zu falschen Bitmustern.
Der Fehler tritt erstmals für BINARY59. auf, wie die nachfolgende Übersicht für die Feldbreiten 56≤*w*≤64 zeigt.

```
zahl=7
bits56=0000000000000000000000000000000000000000000000000000000000000111
bits57=0000000000000000000000000000000000000000000000000000000000000111
bits58=0000000000000000000000000000000000000000000000000000000000000111
bits59=0000001110000000000000000000000000000000000000000000000000000000
bits60=0000001110000000000000000000000000000000000000000000000000000000
bits61=0000000111000000000000000000000000000000000000000000000000000000
bits62=0000000011100000000000000000000000000000000000000000000000000000
bits63=1000000000111000000000000000000000000000000000000000000000000000
bits64=0100000000011100000000000000000000000000000000000000000000000000
```

Lösung:
Für **BINARY64.** wird für Integer Zahlen folgender Workaround empfohlen:

```
  Bits64=PUT(PUT(zahl,S370FIB8.),$BINARY64.);
```

Für **BINARY59. – BINARY63.** wird für Integer Zahlen folgender Workaround empfohlen, wobei w der gewünschten Breite entspricht:

```
  Bits_59_63=SUBSTR(Bits64,64-w+1,w);
```

Bemerkenswert an dieser Tatsache ist, dass dies ein Beispiel dafür ist, dass die SAS Versionen 8 und 9 nicht vollständig abwärtskompatibel zur Version 6.12 sind. Das bedeutet, dass SAS Programme unter Version 8 und Version 9 völlig andere Ergebnisse liefern können als unter 6.12, wenn das BINARY*w*. Format mit $59 \leq w \leq 64$ verwendet wird. Dies ist umso problematischer, als es außer der genannten SAS Note wohl keinen weiteren Hinweis auf diese Änderung gibt.

Literatur

[1] SAS Note SN-004572

5 Neue Funktionen in SAS 9

(Grischa Pfister)

Mit der Version 9 hat SAS wieder einmal eine Vielzahl neuer Funktionen zur Verfügung gestellt. Leider fehlt oft die Zeit, die Online-Hilfe systematisch durchzuschauen. Zu Motivationszwecken folgt hier deshalb eine sehr subjektive Auswahl einiger neuer Funktionen.

5.1 countc()

Diese Funktion zählt das Vorkommen eines oder mehrerer Zeichen in einer gegebenen Zeichenkette. Die Anzahl der Treffer wird in eine numerische Variable gespeichert.

```
rc = countc(string,zeichen<,option(en)>);
```

Als Optionen stehen

i ignore case
t trim trailing blanks (in string und zeichen)
v vice versa

zur Verfügung.

Beispiel

```
Data _Null_;
  string = "  Ein männlicher Briefmark erlebte  ";
  n = countc(string," ","");
  Put n=;
Run;
```

Ergebnis ist 7. Wenn als Option allerdings „t“ gesetzt wird, ist das Ergebnis 0. Das ist auf den ersten Blick etwas verwirrend, liegt aber daran, dass sich die Option sowohl auf die zu durchsuchende als auch auf die gesuchte Zeichenkette bezieht.

5.2 count()

Hier wird geprüft, wie oft eine Zeichenkette in einer anderen enthalten ist.

```
rc = countc(string,zeichen<,option(en)>);
```

Optionen sind in diesem Falle

i ignore case und

t trim trailing blanks (in string und zeichen)

Beispiel

```
Data _Null_;
  string = "Kartoffelbrei mit Ei ist mir einerlei";
  n1 = count(string,"ei ");
  n2 = count(string,"ei ","i"); * ignore Case;
  n3 = count(string,"ei ","t"); * trim Blanks;
  n4 = count(string,"ei ","it"); * ignore Case and trim blanks;
  Put n1= n2= n3= n4=;
Run;
```

Ergebnis: n1=1 n2=2 n3=3 n4=4

5.3 choose[c,n]()

Oft wird eine neue Variable aufgrund der Werte einer oder mehrerer anderer Variablen befüllt. Um dies zu bewältigen, werden oft lange If-Then-Else Konstrukte verwendet. Eine neue Möglichkeit bietet diese Funktion. Sie enthält einfach eine Liste verschiedener Rückgabewerte (entweder numerisch choose**n**() oder alphanumerisch choose**c**()) und bekommt als ersten Parameter eine positive ganze Zahl übergeben. Der Rückgabewert ist dann derjenige, der sich in der Liste an der Position der übergebenen Zahl befindet.

```
rc = choosec(nvar,"wert1","wert2",…);
rc = choosen(nvar,wert1,wert2,…);
```

Besonders interessant wird diese Funktion natürlich im Zusammenspiel mit der von H. Stürzl beschriebenen Booleschen Logik, d.h. die Position, deren Wert zurückgegeben werden soll, wird nicht vorher in einer Variable gespeichert, sondern durch das Erfüllt-/Nichterfülltsein mehrerer Bedingungen berechnet…

Beispiele

```
Data _null_;
  Length farbe $8 wert 3;
  Var = 2;
  farbe = choosec(var,"rot","grün","blau");
  wert = choosen(var,22,44,66,88);
  Put farbe= wert=;
Run;
```

Ergebnis: `farbe = grün wert = 44.`

```
Data _Null_;
  v1 = 100; * v1 = 99;
  v2 = 50;  * v2 = 49;
  v3 = choosec(1 +
               (v1>=100 And v2>=50) +
               (v1>150 And v2>100),
               "Klasse 1","Klasse 2","Klasse 3");
  Put v1= v2= v3=;
Run;
```

Die Variable v3 hat den Wert „Klasse 2“. Wenn einer der auskommentierten Werte genommen wird, enthält sie „Klasse 1“.

5.4 if[cn] ()

Mit dieser Funktion führt SAS endlich den ternären Vergleichsoperator ein, der in Zukunft die meisten einfachen If-Then-Else Konstrukte ablösen wird. Der erste Parameter der Funktion ist eine Bedingung, ist sie erfüllt, wird ein anderer Wert zurückgegeben als wenn sie nicht erfüllt ist. Zusätzlich kann noch ein dritter Rückgabewert definiert werden, wenn die Bedingung in einem missing value resultiert!

`rc = if[c,n](bedingung,wert`$_{true}$`,wert`$_{false}$`<,wert`$_{missing}$`>);`

Beispiele

```
Data _Null_;
  Length value 8;
  Input value @@;
  valueNew = ifn( (value < 3 ), value*2, value*1.5);
  Put value= valueNew=;
  Cards;
  1  2  3  4  5
Run;
```

Ergebnis ist

```
value=1 valueNew=2
value=2 valueNew=4
value=3 valueNew=4.5
value=4 valueNew=6
value=5 valueNew=7.5
```

```
Data a;
  Input a @@;
  Cards;
  1 2 . 4 5 . 7
run;

Data b;
  Set a;
  b=ifn(a, a, 0, -999);
  Put b=;
run;
```

Ergebnis ist

```
b=1
b=2
b=-999
b=4
b=5
b=-999
b=7
```

```
Data b;
  Set a;
  b=ifn(a + ifn( a > 4 ,1,0), a*2, 0, -999);
  Put b=;
run;
```

Ergebnis ist

```
b=2
b=4
b=-999
b=8
b=10
b=-999
b=14
```

5.5 strip()

Hin und wieder enthalten Zeichenketten an Anfang und Ende Leerzeichen. Bisher war die Kombination von trim() und left() das Mittel der Wahl, jetzt gibt es die neue Funktion strip(), die beides in einem Aufruf kombiniert. Außerdem unterscheidet sie sich von trim(), indem Sie einen String mit der Länge null zurückgibt, wenn der ganze String leer ist.

```
rc = strip(string);
```

Beispiele

```
Data _Null_;
  Length string string2 $256;
  string = "  Ein männlicher Briefmark erlebte  ";
  string2 = strip(string);
  len1 = length(string);
  len2 = length(string2);
  Put len1= len2=;
Run;
```

Ergebnis ist

```
len1=34 len2=32.
```

Dieses Beispiel zeigt, dass strip() tatsächlich einen String der Länge null zurückgibt, wenn die untersuchte Zeichenkette leer ist, während die Kombination aus trim() und left() immer einen String der Länge eins liefert.

```
Data _Null_;
  string = "    ";
  If ( trim(left(string = "" )) ) Then
   Put "trim(left(string)): leer";
  Else Put "trim(left(string)): nicht leer";
  test = "->" !! trim(left(string)) !! "<--";
  Put test=;
  If ( strip(string) = "" ) Then
   Put "strip(string): leer";
  Else Put "strip(string): nicht leer";
  test = "->" !! strip(string) !! "<--";
  Put test=;
Run;
```

5.6 compare()

Diese lange vermisste Funktion vergleicht zwei Zeichenketten und gibt den Index zurück, ab dem sie sich nicht mehr gleichen.

```
rc = compare(string1,string2<,option(en)>);
```

Mögliche Optionen sind

i ignore case
l remove trailing blanks
: truncate to same length

Beispiel

```
Data _Null_;
  Length string1 string2 $256;
  string1 = "Ein männlicher Briefmark erlebte";
  string2 = "Ein männlicher Briefmark erlbte";
  index = compare(string1,string2);
  Put index=;
Run;
```

Ergebnis ist

```
index=29
```

5.7 Neue Statistik-Funktionen

SAS9 enthält auch eine Reihe neuer Funktionen für deskriptive Statistik, darunter auch die lange gewünschte Funktion für den Median und andere Perzentile über Variablen, aber auch Funktionen für den k-größten und k-kleinsten Wert und v.a.m.

Beispiel

```
Data _Null_;
  array x [5] (. 1 2 3 4);
  v1 = median(of x[*]); * Median;
  v2 = pctl(25, of x[*]); * 25%-Perzentil;
  v3 = largest(2, of x[*]); * zweit-größten Wert;
  v4 = smallest(2, of x[*]); * zweit-kleinster Wert;
Put v1= v2= v3= v4=;
Run;
```

Ergebnis ist

```
v1=2.5 v2=1.5 v3=3 v4=2
```

5.8 Weitere neue Funktionen

Dem geneigten Leser wird außerdem angeraten, folgende Funktionen im Index der Online-Hilfe nachzuschlagen:
round[e,z]()
[ceil,floor,int]z()
vvalue[,x]()
cat[,s,t,x]()
any[alnum,alpha,cntrl,…]()
und viele, viele mehr…

Literatur

[1] SAS Online Hilfe

[2] SAS Language Dictionary V9, download unter http://support.sas.com/documentation/onlinedoc/91pdf/sasdoc_913/base_lrdictionary_8404.pdf

6 Boolesche Werte in SAS

(Heinrich Stürzl)

Boolesche Werte sind die Zahlen 0 für „Falsch“ (false) und 1 für „Wahr“ (true). Im SAS System kommen sie vor als Ergebnis eines Vergleichs, als Inhalt einer Booleschen Variable, als numerischer Wert und als Ergebnis einer logischen Operation.

6.1 Ergebnis eines Vergleichs

Der Vergleich 1 < 5 ist wahr und liefert deshalb den Wert 1. Umgekehrt erhält man den Wert 0, wenn der Vergleich unwahr ist. Diese Eigenschaft läßt sich zum Zählen von Ereignissen mit Hilfe Boolescher Werte nutzen.

Beispiel 1: Wie viele der Variablen x1-x3 haben den Wert 0?

```
Anzahl_Nullen=(x1=0) + (x2=0) + (x3=0);
```

Die zugewiesene Variable enthält die Werte 0, 1, 2 oder 3, je nachdem wie viele der genannten Variablen den Wert 0 haben.

Beispiel 2: Nur positive Werte der Variablen x1-x3 addieren?

```
Summe_der_Positiven=(x1>0)*x1 + (x2>0)*x2 + (x3>0)*x3;
```

Der Wert der Variablen x1-x3 geht nur dann in die Summenbildung ein, wenn die jeweilige Bedingung (*variable*>0) wahr ist.

Beispiel 3: Bildung von Score-Punkten durch Gewichtung bestimmter Ereignisse.

```
score=  (alter > 45) +
      2*(raucher="ja") +
      3*(bmi > 25);
```

Die zugewiesene Variable enthält umso höhere Werte, je mehr der genannten Bedingungen erfüllt sind, wobei die einzelnen Bedingungen unterschiedlich stark gewichtet sein können. Im Beispiel wird Rauchen doppelt so stark gewertet wie das Alter.

6.2 Inhalt einer Booleschen Variable

Auch wenn SAS keinen eigenen Variablentyp für Boolesche Werte kennt, kann man numerische Variablen mit dem Inhalt 0 oder 1 als Boolesche Variablen bezeichnen.

1. Automatisch erzeugte Boolesche Variablen im Data Step
Beispiele: _ERROR_, FIRST. und LAST.Variablen, IN= und END= Tabellenoptionen der SET und MERGE Anweisung

2. Selbstdefinierte Boolesche Variablen[1]

Beispiel:

```
kind=alter < 14;
```

entspricht

```
IF alter < 14 THEN kind=1;
ELSE              kind=0;
```

[1] Falls Boolesche Variablen gespeichert werden, sollten sie die kleinstmögliche Länge numerischer Variablen haben (unter Windows die Länge 3 Byte), um Speicherplatz zu sparen.

6.3 Numerischer Wert

Jeder numerische Wert kann auch als Boolescher Wert interpretiert werden, wobei die Werte 0 und missing value als „Falsch“, alle übrigen numerischen Werte – also alle positiven und negativen Zahlen – als „Wahr“ gelten.

Beispiel: Division nur durchführen, wenn der Nenner ungleich Null ist.

```
IF nenner THEN bruch=zaehler/nenner;
ELSE           bruch=.;
```

Die Abfrage **`IF variable`** entspricht **`IF variable NOT IN (. 0)`**.

6.4 Ergebnis einer logischen Operation mit NOT, AND, OR

Beispiel 1: Wahrheitstabelle eines beliebigen Booleschen Ausdrucks erstellen. z. B. NOT a OR NOT b

```
DATA _NULL_;
  DO a=0,1;
    DO b=0,1;
      x=NOT a OR NOT b;
      PUT a= b= x=;
    END;
  END;
RUN;
```

⇨

```
a=0 b=0 x=1
a=0 b=1 x=1
a=1 b=0 x=1
a=1 b=1 x=0
```

Beispiel 2: Numerischen Wert in einen Booleschen Wert umwandeln.

```
x=NOT(NOT(x));
```

Der Wert 0 und Missing value (false) wird so zum Wert 0. Alle übrigen numerischen Werte (true), werden zum Wert 1.

Literatur

[1] Rick Aster: Professional SAS Programming Shortcuts, SAS Institute Inc. (BBU), 2002.

7 Anwendungen von Tastaturabkürzungen und –makros

(Martin Kappler)

7.1 Vermeidung von Programmierfehlern bei Missings

In den Tipps und Tricks auf der KSFE 2004 [1] wurde das Problem vorgestellt, dass Missings dem Zahlwert –∞ entsprechen, wodurch logische Abfragen durch < bzw. <= eventuell zu unerwünschten Ergebnissen führen.

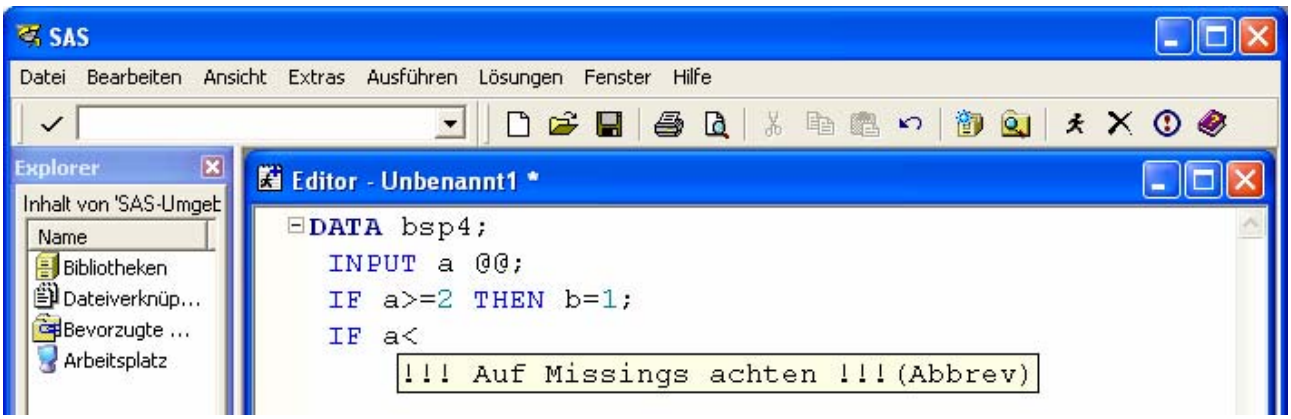
```
DATA bsp4;
  INPUT a @@;
  IF a>=2 THEN b=1;
  IF a< 2 THEN b=0; /* Vorsicht!!! */
  DATALINES;
1 . 2
;
RUN;
```

Output

```
Beob.    a    b
  1      1    0
  2      .    0
  3      2    1
```

Eine Möglichkeit daran erinnert zu werden, besteht im Erzeugen einer neuen Tastaturabkürzung. Dazu wird unter "Extras > Abkürzung hinzufügen" im Feld "Abkürzung" das Kleiner-Zeichen "<" eingetragen und ins Feld "Volltext für Abkürzung" ein Warnhinweis geschrieben, dass an Missings gedacht werden soll. Jedes Mal, wenn nun das Kleiner-Zeichen eingegeben wird, öffnet sich ein kleines Fenster mit dem eingegebenen Warnhinweis (Abb. 2).

Abbildung 2: Anzeige eines Warnhinweises über eine Tastaturabkürzung



7.2 Auskommentieren eines Teils einer Zeile

Über die Tastenkombinationen Ctrl+7 bzw. Ctrl+Shift+7 ist im Enhanced Editor eine schnelle Auskommentierung bzw. Entkommentierung von ganzen Zeilen möglich. Sollen jedoch einzelne Teile einer Zeile auskommentiert werden, so funktioniert dies nicht.
Abhilfe schafft hier ein Tastaturmakro, das sich entweder aufzeichnen oder im Menü unter "Extras > Tastaturmakros > Makros > Erstellen" erzeugen lässt. Dazu werden nacheinander die folgenden Kommandos ausgewählt und in den Inhalt des Tastaturmakros geschoben:

- Auswahl ausschneiden
- Zeichenkette "/*" einfügen
- Aus Zwischenablage einfügen
- Zeichenkette "*/" einfügen

Nachdem dem Makro noch eine freie Tastenkombination als Shortcut zugewiesen wurde, ist darüber eine Auskommentierung von Zeilenteilen möglich.

Literatur

[1] C. Ortseifen, G. Pfister, O. Schoffer, L. Tschiersch. *Tipps und Tricks für einen leichteren Umgang mit der SAS Software.* D. Beyer, Ortseifen C. (Hrsg.), Proceedings der 8. Konferenz für SAS®-Anwender in Forschung und Entwicklung (KSFE), Shaker Verlag, Aachen, 2004

8 Fünf Stolperfallen beim Arbeiten mit der SAS Enterprise Guide Software

(Carina Ortseifen)

Die SAS Enterprise Guide Software (SAS/EG) ist eine Thin Client Software, die seit dem Erscheinen von SAS 9 jedem SAS Anwender unter Windows zur Verfügung steht.
Für Anfänger stellt dieses Modul eine bequeme Möglichkeit dar, ohne Programmierung statistische Analysen mit SAS durchzuführen. Aber auch langjährige Anwender der SAS Software profitieren davon, etwa indem sie sich für komplexere Prozeduren wie SQL oder MIXED Musterprogramme erstellen lassen, die sich dann problemlos weiterbearbeiten lassen.

Am Anfang kann man allerdings über eine der folgenden fünf Hürden stolpern:

8.1 Installation

Die SAS Enterprise Guide Software ist ein eigenständiges Modul, auf einer eigenen CD (eventuell auch zwei CDs).
Ab Version SAS 9 ist der SAS/EG lizenzrechtlich Bestandteil von Base SAS. Aber bei der Installation des ‚normalen' SAS Systems wird die CD nicht angefordert. D.h. man muss als Anwender daran denken, dass diese CD nachträglich installiert werden muss.

8.2 Nicht immer Windows-gemäß

Gerade dem Windows-Anwender bereitet es große Schwierigkeiten, dass Änderungen in SAS-Tabellen sofort physikalisch wirksam sind und kein ‚Rückgängig' diese wieder umkehrt. Während in Programmen wie Word oder Excel einzelne Änderungen rückgängig gemacht werden können oder das stundenlang geänderte Dokument einfach am Ende des Tages verworfen werden kann, sind Änderungen an SAS-Tabellen sofort und unwiderruflich aktiv.
Der Anfänger muss lernen, am Ende des Tages Sicherungen für den nächsten Tag oder zu Beginn der Arbeit sofort Sicherungskopien anzufertigen, um mögliche Fehlschläge zu vermeiden.

8.3 Projekte …

An einem Rechner arbeiten ist wunderschön. Doch manchmal will man seine Projekte doch zu einem Kollegen übertragen, um mit ihm Probleme zu besprechen oder in die SAS-Beratung am URZ Heidelberg gehen.
Da ist es dann schon häufig passiert, dass die Projekte zwar übertragen wurden, aber die SAS-Tabellen, die nicht im Projekt abgespeichert wurden, zu Hause vergessen wurden und die ganze Beratung bzw. das Kollegengespräch vertagt werden musste.
Der SAS/EG Nutzer muss daran denken, dass die Tabellen nur als Link im Projekt enthalten sind und die Quellen daher nicht automatisch mitkopiert werden.

Denkt man daran, die Tabellen mit zu übertragen, liegt aber auf dem Zielrechner eine andere Verzeichnis-Struktur vor, dann muss man im Projektfenster die Eigenschaften der Tabelle ändern (Tabelle im Projektfenster anklicken, rechte Maustaste, Eigenschaften und Ändern).

8.4 Unterschiedliche Einstellungen

Für viele SAS Anwender stellt die SAS/EG Software eine gute Einstiegsmöglichkeit in SAS dar. Intensivere Arbeit erfordert dann aber doch einen Übergang in das normale SAS System.
Eine große Hürde stellt dabei – gerade für unkundige Anfänger – die unterschiedliche Einstellung der SAS Systemoption Validvarname dar.

Während im SAS/EG Validvarname=Any gilt und damit auch Variablennamen wie Alter?, Wütend und Binde-strich gültig sind, gilt für SAS 9 wie auch für SAS 8.2: Validvarname=V7. Und oben genannte Beispielnamen sind ungültig.

Ein Ausweg wird in der SAS Note SN-012607 beschrieben: Man solle in SAS 8 bzw. SAS 9 die Einstellung der Systemoption Validvarname ändern.

Eine Alternative, allerdings nicht unbedingt für Anfänger geeignet, besteht darin, über *Extras > Optionen > Neues SAS-Programm > Zusätzlicher SAS-Code: Eigenen SAS-Code vor ausgeführtem Code einfügen* zu aktivieren und unter *Bearbeiten* die Systemoption mit Hilfe der Anweisung Options im SAS/EG zu setzen.

8.5 Nicht alles ist im SAS/EG machbar

Nicht alle aus der ‚normalen' SAS Umgebung bekannten Prozeduren lassen sich im Code-Fenster des SAS/EG aufrufen. Ein Beispiel hierfür ist die Prozedur FSEDIT. Versucht man diese in einem Code-Fenster des SAS/EG 2.1 aufzurufen, erhält man die Fehlermeldung:

```
ERROR: Terminal does not have full-screen capabilities.
```

Im SAS/EG 3.0 wird stattdessen die Prozedur FSBROWSE ausgeführt.

9 Verzeichnisinhalte einlesen

(Grischa Pfister)

In der Praxis stellt sich immer wieder die Aufgabe, den Inhalt eines bestimmten Verzeichnisses einzulesen, z.B. um alle Dateien automatisiert einzulesen, die Existenz einer bestimmten Datei zu überprüfen oder um in einer programmierten Anwendung eine Auswahlliste anzuzeigen.
Es gibt verschiedene Wege, zu einem Ergebnis zu kommen, unter anderem die Verwendung von SCL (Sas Component Language) Funktionen, die seit V8 auch im Data Step erlaubt sind, oder die Allokation von Windows-spezifischen DLLs.
Hier soll jedoch eine andere Methode Verwendung finden, das altbekannte FILENAME-Statement.

9.1 Das FILENAME-Statement

In der Regel wird das FILENAME-Statement verwendet, um eine externe Datei zu allokieren, in die dann Daten geschrieben bzw. aus der Daten gelesen werden. Allerdings bietet das Statement wesentlich mehr Möglichkeiten, es können mit Hilfe des FILENAME-Statements z.B. emails verschickt werden, Dateien von FTP-Servern gelesen oder dorthin geschickt werden etc. Dafür muss lediglich die entsprechende Device-Angabe für die Zugriffsmethode ergänzt werden, z.B. **CATALOG, FTP, EMAIL**, …
Ein weiterer Device ist **PIPE**, mit dessen Hilfe Befehle oder Programmaufrufe an das Betriebssystem übergeben werden können. Das macht dann Sinn, wenn das Ergebnis des Befehls/aufgerufenen Programms eine Ausgabe ist, die normalerweise in der Konsole landen würde (also vom aufgerufenen Programm an STDOUT geschrieben wird), denn mit Hilfe eines Data Steps und der INFILE-Anweisung wird die Ausgabe des Kommandos direkt in SAS eingelesen.

9.2 Der DIR-Befehl

Dieser DOS-Befehl gibt eine Liste des aktuellen Verzeichnisses aus. Über Optionen kann das Aussehen der Liste angepasst werden. Die wichtigsten Optionen sind:

/? zeigt Informationen über den Befehl an
/B gibt nur eine Liste der Dateinamen aus
/A:D listet nur Verzeichnisse auf
/A:-D listet nur Dateien auf
/-C kein Tausendertrennpunkt in der Größenangabe

Beispiele

Dieses einfache Beispiel setzt einen DIR-Befehl ab und gibt das eingelesene Ergebnis in das LOG aus.

```
Filename in PIPE "Dir D:\";

Data _Null_;
  Infile in;
  Input;
  Put _infile_;
Run;
```

Sollen nur die enthaltenen Dateien gelesen werden, bietet sich folgende Syntax an.

```
Filename in PIPE "Dir D:\ /B/A:-D";

Data Work.Contents;
  Length file $256;
  Infile in;
  Input file;
Run;

Proc Print;
Run;
```

Dieses dritte Beispiel liest auch die Dateieigenschaften mit ein.

```
Filename in PIPE "Dir D:\ /A:-D/-C";

Data Work.Contents (drop=start);
  Length file $256 cDate $10 cTime $5 cSize $32
         nDate nTime nSize 8;
  Format nDate eurdfdd10. nTime time5.;
  Infile in;
  Input;
  * GP Leerzeilen ignorieren *;
  If ( _infile_ Ne "");
  * GP Kopfdaten und Zusammenfassung ignorieren *;
  If ( substr(_infile_,1,1) Ne " ");
  Put _infile_;
  cDate = scan(_infile_,1," ");
  cTime = scan(_infile_,2," ");
  cSize = scan(_infile_,3," ");
  * GP Dateinamen koennen Leerzeichen enthalten... *;
```

```
  start = indexw(_infile_,cSize) + length(compress(cSize)) + 1;
  file  = substr(_infile_,start);
  * GP Werte in Zahlen konvertieren *;
  nDate = input(cDate,ddmmyy10.);
  nTime = input(cTime,time5.);
  nSize = input(cSize,best.);
Run;

Proc Print;
Run;
```

Literatur

[1] SAS Online-Hilfe und SAS Online-Doc

10 Publikumsfrage

Auf der Vorjahreskonferenz in Schmalkalden wurde am Ende der Tipps und Tricks Sitzung eine Frage ans Publikum gestellt, deren Auflösung hier vorgestellt wird. Am Ende finden Sie dann noch die neue Publikumsfrage.

10.1 Publikumsfrage der 8. KSFE 2004 Schmalkalden

Aufgabenstellung

Ausgangspunkt war folgende SAS-Tabelle A:

```
Obs     a     b     c     d
1       371   .     .     .
2       .     1203  .     .
3       .     .     5     .
4       .     .     .     23
```

Diese Tabelle A sollte mit Hilfe eines SAS-Programms überführt werden in die SAS-Tabelle B:

```
Obs      a        b      c      d

  1      371     1203     5     23
```

Lösungen

Die Tabelle B besteht aus den Diagonalelementen der Matrix A und es waren Mittel und Wege gefragt, dies mit der SAS Software zu realisieren.
Insgesamt wurden 64 Lösungen von 21 Teilnehmern bzw. Teilnehmergruppen eingereicht. Die Anzahl der Lösungen pro Teilnehmer bzw. Gruppe bewegte sich zwischen einer und 16 (!) verschiedenen Lösungen und zeigten die vielfältigen Möglichkeiten auf, die mit der SAS Software zur Verfügung stehen.
Ungefähr die Hälfte der Lösungen (n=35) beruht auf einem Datenschritt entweder kombiniert mit weiteren Datenschritten oder einem Prozedurschritt. Ein weiteres Drittel (n=20) wählte eine Prozedur aus, um zur Lösung zu kommen, etwa Means oder Summary. Und neun weitere Lösungen beruhen auf anderen Wegen (Proc Iml, Sql …).
Im folgenden sollen stellvertretend für die eingereichten 64 Möglichkeiten je eine Prozedur- und eine Datenschrittlösung vorgestellt werden. Die übrigen Varianten sind im SAS-Anwenderhandbuch im Netz unter
http://www.urz.uni-heidelberg.de/statistik/sas-ah/ im Kapitel 5 nachzulesen.

Lösung mit einer Prozedur

Am häufigsten wurde die Prozedur Means eingesetzt, 10 mal, aber es gab auch Lösungen mit Summary (5 mal), Univariate (2), Report, Tabulate und Standard (je 1).
Wegen der Einfachheit und weil die Prozedur Means eine der Statistikprozeduren aus SAS/Base ist, die der Anfänger als erstes kennenlernt, hier mein „Prozedurfavorit“:

```
Proc Means Data=a Noprint;
   Output
      Out=b(Drop=_Type_ _Freq_)
      Sum=;
Run;
```

Die Output-Anweisung wird verwendet, um das Ergebnis in die neue Tabelle B raus zuschreiben. Die Wahl der Statistik (hier: Sum) spielt eigentlich keine Rolle, man könnte genau so gut N=, Mean= o. ä. wählen, da die restlichen Werte in der Spalte fehlten und daher bei der Berechnung der Kenngröße ignoriert werden.

Ein Nachteil dieser Methode ist, die Nicht-Verallgemeinerbarkeit, was aber in der Aufgabenstellung so nicht genannt wurde, aber sich nachträglich vielleicht doch als wichtig erweisen könnte. Dieses Verfahren ist nur bei numerischen Variablen einsetzbar und auch nur dann, wenn die Werte außerhalb der Diagonalen allesamt fehlend sind.

Betrachten wir also die

Lösung mit einem Datenschritt

Die 35 verschiedenen Lösungen mit Hilfe eines Datenschritts zeigen sehr anschaulich die Vielfalt der Programmierungsmöglichkeiten im SAS System. Die Aufgabe wurde gelöst mit den Anweisungen Retain (7 mal), Array (6), Summenbildung (4), Mergen von vier Dateien (4), Bildung einer neuen Variablen und Proc Transpose (3) sowie der Lag-Funktion, Call Vname, Update (je 1 mal) und anderen Varianten.

Beispielhaft soll hier die Lösung über Arrays aufgezeigt werden, da sie wegen der Unabhängigkeit von den übrigen Werten leicht zu verallgemeinern ist.

```
Data ab;
   Set a;
   Keep diag;
   Array x a--d;
   Do i=1 To 4;
      If i=_N_ Then diag=x(i);
   End;
Run;
Proc Transpose data=ab out=b(Drop=_Name_Rename=(Col1=a Col2=b
Col3=c Col4=d));
Run;
```

Allen Einsendern von Lösungen sei an dieser Stelle recht herzlich gedankt. Einige haben es bereits getan, aber vielleicht haben jetzt auch Sie Lust bekommen für die neue Frage ans Publikum?

10.2 Publikumsfrage der 9. KSFE 2005 Berlin

Gegeben ist folgende SAS-Tabelle:

```
data ksfe9;
   input string $50.;
   datalines;
Der Mond, unser Nachbar
Alter: ca. 4 Mrd. Jahre
Schwerkraft: 1,62 m/s²
Die minimale Oberflächentemperatur beträgt -160°C,
während maximal bis zu +130°C erreicht werden.
Die Entfernung zur Erde beträgt ca. 3,84e+05 km.
Ein Sonnentag dauert 2.551.443,0 Sekunden.
Der Atmosphärendruck beträgt ,1E-11 bar.
;
```

Finden Sie alle Zahlen in den Zeilen des Textes und bestimmen Sie den Beginn der Zahl im String und die Länge der Zahl.

Ein **Tipp**, wenn Sie die Aufgabe mit Perl Regular Expressions lösen möchten:
Über `(?(condition)yes-regexp|no-regexp)` ist eine If-Then-Else Abfrage innerhalb des Suchmusters möglich
(siehe http://search.cpan.org/~nwclark/perl-5.8.6/pod/perlretut.pod).

Bitte schicken Sie Ihre Lösung bis zum 30. Januar 2006 an kappler@bgfa.ruhr-uni-bochum.de oder carina.ortseifen@urz.uni-heidelberg.de.

Die besten Lösungen werden auf der 10. KSFE in Hamburg vorgestellt.