Tipps und Tricks
für den leichteren Umgang mit der SAS Software

Almut Hahn
INPUT Clinical Research GmbH
Lütticher Str. 281
52074 Aachen
a.hahn@input-cro.de

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

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

Thomas Rüdiger
AXA Service AG
Colonia-Allee 10-20
51067 Köln
thomas.ruediger@gmx.at

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.
Themen:

1 ODS PDF Bookmarks von SAS Grafiken
2 Komprimierung in SAS
3 Verschiedene SAS-Profile verwalten
4 ODS RTF – Neues aus SAS9
5 Zugriff auf und Schreiben von Excel- und Access-Tabellen
6 Temporäre Buffer
7 Ende gut, alles gut - Grafiken mit True Type Fonts und ODS
7 Java im Data Step
9 Auflösung der Publikumsfrage von der 9. KSFE

Schlüsselworte: ODS PDF, Bookmarks, SAS/GRAPH By-Processing, PROC GREPLAY, description=, des=, #BYLINE, #BYVAL, #BYVAR,
compress, binary, char, uncompressed, strings
Windows, Desktop-Symbol, Verknüpfung, Userprofil, sasuser, SAS-Optionen
Microsoft Excel, Excel, Microsoft Access, Access, Libname-Anweisung, Libname-Engine, Engine Libname,Replace-Option, Datumsangaben, Leerzeichen, ODS RTF, Inhaltsverzeichnis, Contents-Option, Pageof-Option, Publikumsfrage

1 ODS PDF Bookmarks von SAS Grafiken *(Heinrich Stürzl)*

Bei der Erstellung von PDF Dateien mit ODS PDF lassen sich optional Bookmarks (Lesezeichen) automatisch erstellen, die als verlinkte Inhaltsübersicht für das PDF Dokument dienen. Durch Anklicken eines Bookmarks gelangt man direkt zur entsprechenden Seite im Dokument.
Standardmäßig erscheinen die Bookmarks nach Prozeduren geordnet und je nachdem ob der Output mit oder ohne By-Gruppen erzeugt wird innerhalb der Prozedur ein- oder zweistufig aufgebaut und dadurch nicht besonders gut lesbar und übersichtlich.
Für Grafik Output, insbesondere wenn viele Grafiken durch By-Gruppen erstellt werden, lassen sich mittels der Grafik DESCRIPTION= Option und PROC GREPLAY übersichtliche, einstufige Bookmarks erstellen, deren Inhalt man selbst bestimmen kann.

1.1 Standard Bookmarks

Als Beispiel dienen hochaufgelöste Grafiken, die mit PROC GPLOT und PROC UNIVARIATE (hier Histogramme) ohne und mit By-Gruppen erstellt werden.

Beispieldaten:

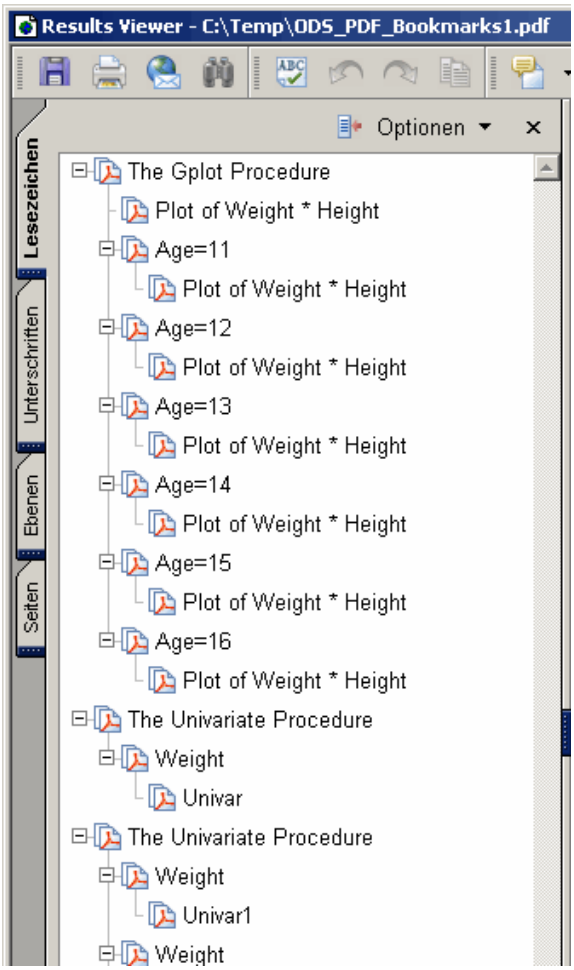
```
PROC SORT DATA=sashelp.class OUT=class;
  BY age ;
RUN;
```

Beispiel 1: Standard

```
ODS PDF FILE="C:\Temp\Bookmarks1.pdf";

PROC GPLOT DATA=class ;
  PLOT weight*height ;
RUN;
  PLOT weight*height ;
  BY age ;
RUN;
QUIT;
PROC UNIVARIATE DATA=class NOPRINT ;
  HISTOGRAM weight ;
RUN;
PROC UNIVARIATE DATA=class NOPRINT ;
  HISTOGRAM weight ;
  BY age ;
RUN;

ODS PDF CLOSE;
```



Die Booksmarks beider Prozeduren sind offensichtlich unterschiedlich. Im Gegensatz zur „echten“ SAS/GRAPH Prozedur GPLOT fehlt bei den Grafiken von PROC UNIVARIATE die By-Gruppen Information in den Bookmarks. Dies gilt auch für Probability Plots (PROBPLOT) und Quantil-Quantil-Plots (QQPLOT).

1.2 DESCRIPTION= Option

Mit Hilfe der DESCRIPTION= Option bzw. ihrer Kurzform DES= der jeweiligen Grafik Anweisung kann man ein Grafik Label mit bis zu 256 beliebigen Zeichen definieren. Dabei können auch die Platzhalter #BYLINE, #BYVAL und #BYVAR im Falle von By-Gruppen verwendet werden.

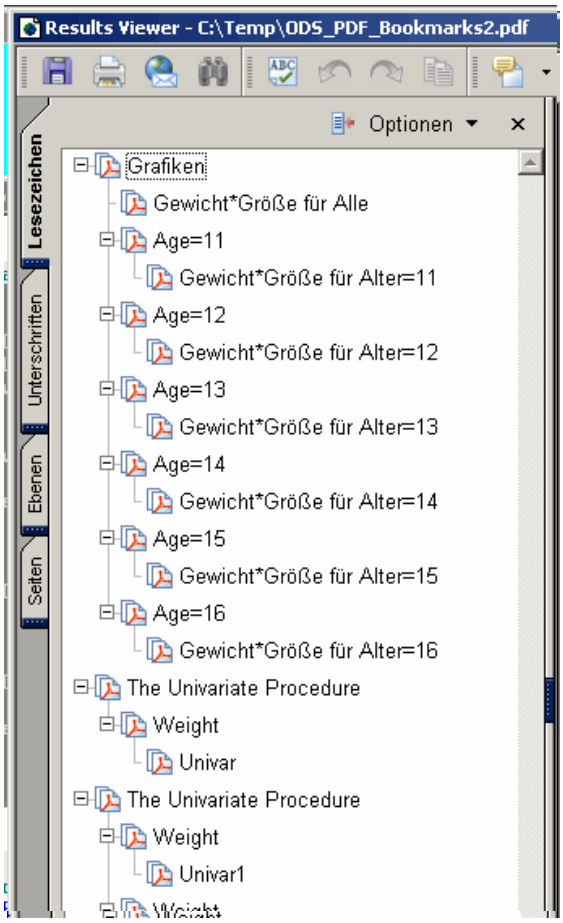
Mit ODS PROCLABEL lässt sich der Haupteintrag der nächsten Prozedur definieren.

Beispiel 2: Mit DES= Option

```
ODS PDF FILE="C:\Temp\Bookmarks2.pdf";

ODS PROCLABEL="Grafiken";
PROC GPLOT DATA=class ;
  PLOT weight*height
  / DES="Gewicht*Größe für Alle";
RUN;
  PLOT weight*height
  / DES="Gewicht*Größe für
Alter=#BYVAL(age)";
  BY age ;
RUN;
QUIT;
PROC UNIVARIATE DATA=class NOPRINT ;
  HISTOGRAM weight
  / DES="Gewicht Histogramm für Alle";
RUN;
PROC UNIVARIATE DATA=class NOPRINT ;
  HISTOGRAM weight
  / DES=" Gewicht Histogramm für
Alter=#BYVAL(age)";
  BY age ;
RUN;

ODS PDF CLOSE;
```



Mit dem Grafiklabel lässt sich im Falle von By-Gruppen nur der zweite Eintrag beeinflussen, während der erste Eintrag standardmäßig als #BYVAR=#BYVAL festgelegt ist.

Das Grafiklabel wird im Unterschied zu SAS/GRAPH Grafiken bei den Grafiken von PROC UNIVARIATE leider nicht für die Bookmarks verwendet. Dies gilt auch für Probability Plots (PROBPLOT) und Quantil-Quantil-Plots (QQPLOT).

1.3 DESCRIPTION= Option und PROC GREPLAY

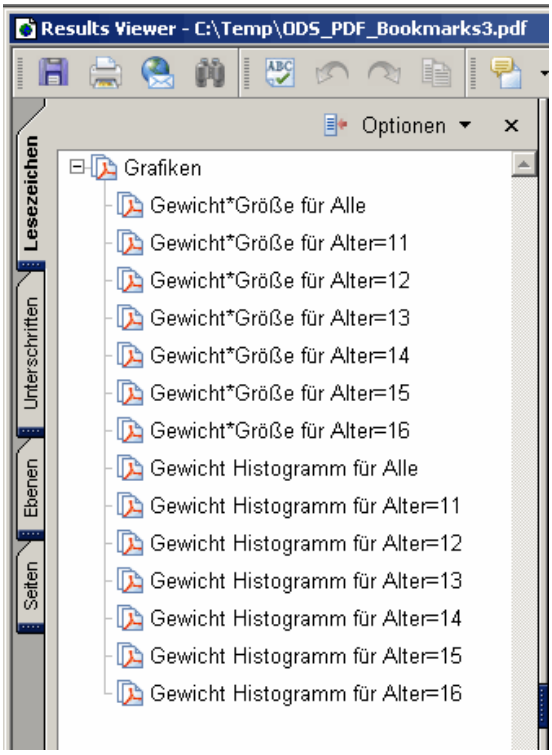
Um einstufige Bookmarks zu erhalten muss man wie folgt vorgehen:
1. Grafiklabel mit DES= Option bei der Erstellung definieren.
2. Grafiken nicht direkt ausgeben lassen, sondern in einem Katalog zwischenspeichern.
3. Grafiken mit PROC GREPLAY aus dem Katalog heraus ausgeben lassen.

Beispiel 3: Mit DES= Option und PROC GREPLAY

```
* Anzeige ausschalten, Grafiken in Katalog schreiben;
GOPTIONS NODISPLAY;
PROC GPLOT DATA=class GOUT=work.grafiken;
  PLOT weight*height
  / DES="Gewicht*Größe für Alle";
RUN;
  PLOT weight*height
  / DES="Gewicht*Größe für
Alter=#BYVAL(age)";
  BY age ;
RUN;
QUIT;
PROC UNIVARIATE DATA=class NOPRINT
GOUT=work.grafiken;
  HISTOGRAM weight
  / DES="Gewicht Histogramm für Alle";
RUN;
PROC UNIVARIATE DATA=class NOPRINT
GOUT=work.grafiken;
  HISTOGRAM weight
  / DES="Gewicht Histogramm für
Alter=#BYVAL(age)";
  BY age ;
RUN;

ODS PDF FILE="C:\Temp\Bookmarks3.pdf";

* Grafiken mit PROC GREPLAY aus dem Katalog heraus anzeigen;
GOPTIONS DISPLAY;
ODS PROCLABEL "Grafiken";
PROC GREPLAY IGOUT=work.grafiken NOFS;
```



```
  REPLAY _ALL_;
  DELETE _ALL_; * alle Grafiken löschen für Programmwiederholung;
RUN;
QUIT;

ODS PDF CLOSE;
```

Diese Methode hat außerdem den Vorteil, dass die Grafiken verschiedener Prozeduren unter einem Eintrag zusammengefasst werden können.
Darüber hinaus kann bei Bedarf auch die Reihenfolge geändert werden. Siehe [1].

Literatur

[1] Spruck, D., Kawohl M. (2005): *Sort Your SAS Graphs and Create a Bookmarked PDF Document Using ODS PDF*, Paper CC01 der PhUSE 2005 in Heidelberg. http://www.lexjansen.com/phuse/index.htm

2 Komprimierung in SAS *(Thomas Rüdiger)*

2.1 Zusammenfassung

Speicheroptimierung läßt sich in SAS für eine Ausgabedatei über die Compress-Optionen ‘binary‘, ‘char‘ und ‘no‘ erreichen. Für die bestmögliche Option läßt sich zwar keine Formel angeben, jedoch Spielregeln. ‘Compress=Binary‘ scheint bei großen Datenmengen der geeignete System-Default.

2.2 Einleitung

“Komprimierung in SAS lohnt sich trotz einiger Nachteile.“

File-Komprimierung ist ein Prozess, der die pro Datensatz benötigten Bytes reduziert.
In einem komprimierten File ist für jeden Datensatz die Record-Länge variabel, in einem unkomprimierten File fix.

Vorteile von Komprimierung sind idR Speicherreduzierung, mehr Daten bei weniger Speicherbedarf, weniger I/O-Prozesse zum Lesen oder Schreiben von Daten, geringeres Risiko von PROC SORT-Abbrüchen

Nachteile von Komprimierung sind erhöhter CPU-Bedarf zum Lesen von Daten aufgrund des Dekomprimierungs-Overheads sowie in einigen Fällen eine Erhöhung des Speichervolumens aufgrund Komprimierung-Patterns im Ausgabefile.

Die compress-Option kennt seit der Version SAS/V8 drei Anwendungs-Ebenen (System, Library und Tabelle) und drei Ausprägungen (NO, BINARY und CHAR).

Optimale Speicheroptimierung findet individuell auf Tabellenebene statt.

SAS-Syntax Tabellen-Ebene:

```
DataBinary(compress=binary)
Char(compress=char)
Uncompressed(compress=no);
X=1.2345;run;
```

SAS-Syntax Libname-Ebene:

```
libname MYDATA '.' compress=binary;
```

SAS-Syntax System-Ebene:

```
options compress=binary;
```

Gibt man compress auf Tabellenebene an, wird der Default für Library und System überschrieben, die Ebene Library hat höhere Priorität als die Ebene System.

Die optimale compress-Option hängt von der Datenstruktur und den Dateninhalten ab.

NO | N
läßt die Datensätze in der Ausgabe-Datei unkomprimiert (feste Länge)

YES | CHAR | Y | ON
staucht die Datensätze in der Ausgabe-Datei mit RLE (Run Length Encoding): RLE reduziert gleiche aufeinanderfolgende Zeichen (einschließlich Blanks) auf 2- oder 3-Byte-Darstellungen. RLE berücksichtigt laut “SAS Hilfe“ insbesondere Textfelder.

BINARY
staucht die Datensätze in der Ausgabe-Datei mit RDC (Ross Data Compression):
RDC kombiniert RLE und „Sliding-window compression“. Diese Methode eignet sich vor allem bei großen Blöcken von binären Daten/numerischen Variablen (mehrere hundert Bytes oder größer). Die Recordlänge sollte in diesem Fall mehrere hundert Bytes oder größer sein.

Das “strings“-Kommando auf Linux/UNIX-Systemen erlaubt Einblicke in die physische Struktur von SAS-Dateien. Neben den Daten werden auch System- und Dictionary-Informationen bei der Komprimierung berücksichtigt.

Um die Dateigröße von unkomprimierten Ausgabe-Datei zu schätzen, gibt die "SAS Hilfe" eine Formel mit Angaben aus dem PROC CONTENTS an: Je kleiner Data Set Page Size und Anzahl Datensätze und je größer Max Obs per Page, desto kleiner die Size.

Observations 100
Max Obs per Page 101
Data Set Page Size 4096

number of data pages = 1 + (floor(Observations / Max Obs per Page))

size = 1024 + (Data Set Page Size * number of data pages)

Für die Schätzung der Dateigröße von komprimierten SAS-Tabellen ist derzeit keine Formel in der SAS Hilfe hinterlegt: Der Wert für Max Obs per Page ist im PROC CONTENTS bei komprimierten SAS-Tabellen missing. In Einzelfällen gab es bei unkomprimierten SAS-Tabellen Ungenauigkeiten in der Schätzung. Reihenfolge geändert werden.

2.3 Test zur Ermittlung der optimalen Compress-Option

Die Speicheroptimierung läßt sich testen mit unterschiedlicher Anzahl Datensätzen und unterschiedlichen Datenstrukturen.

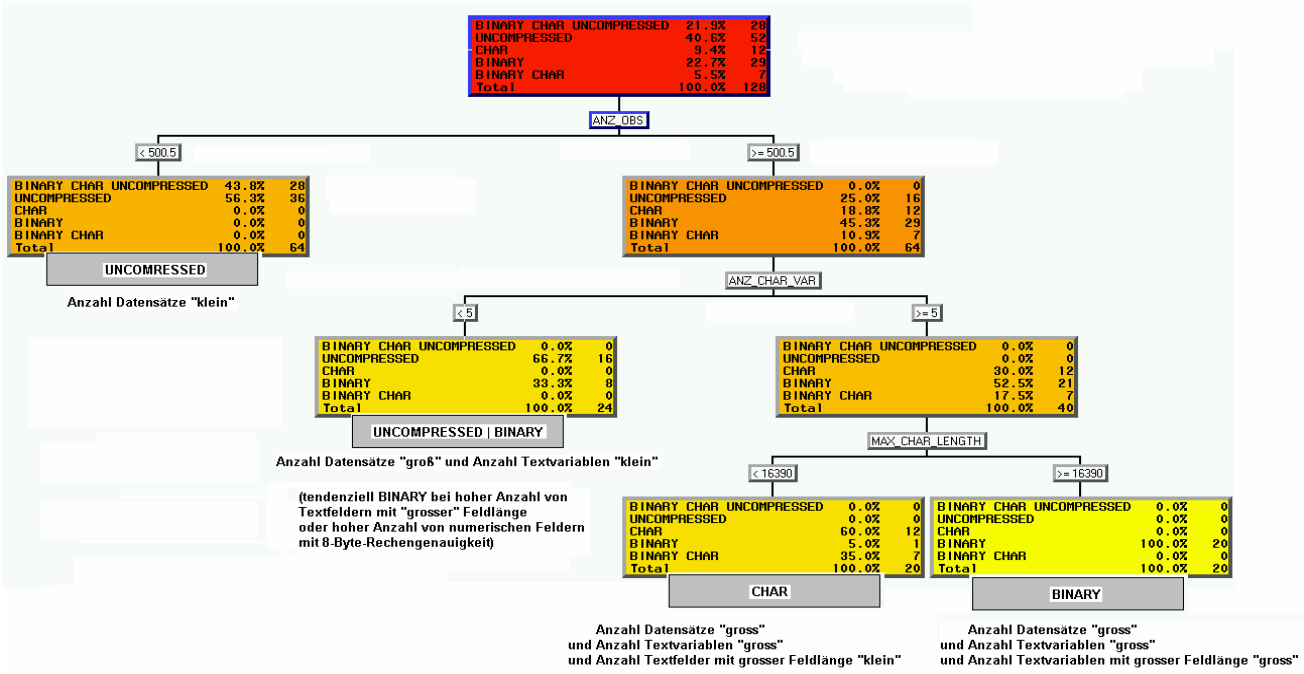
I N P U T -Parameter:

MacroVariable	**Beschreibung (Testwerte)**
VarOrder	array) Variablen-Reihenfolge array-weise (smallnum bigchar bignum)
	var) Variablen-Reihenfolge entsprechend Variablentyp Anz_Big_Num_Var
Anz_Obs	Anzahl Datensätze (1|1000)
Anz_Small_Num_Var	Anzahl numerischer Felder mit 3 Byte-Genauigkeit (0|10)
Anz_Big_Num_Var	Anzahl numerischer Felder mit 8 Byte-Genauigkeit (0|10)
Anz_Char_Var	Anzahl Textfelder (0|10)
Max_Small_Num_Length	Länge (3-8) für numerische Felder mit 3-Byte-Genauigkeit (3|8)
Max_Char_Length	Länge für max. 16-stellige Textfelder (16|32767)

O U T P U T :

Variable	**Beschreibung**
Compress	Berechneter optimaler Kompressions-Typ ("Zielvariable")
Factor	Berechneter Faktor Speicherreduzierung
MinBytes	Benötigte Bytes bei optimaler Komprimierung
DataSetPageSize	DataSet Page Size
MaxObsPerPage	Max Obs Per Page
Estimated_Size	Benötigte Bytes bei optimaler Komprimierung, geschätzt

Hinweise für eine mögliche Vorhersage des optimalen Compress-Typs ergeben sich per Entscheidungsbaum.



Der optimale Compress-Typ läßt sich aufgrund von Datenstruktur
und Anzahl erwartbarer Datensätze in der Ausgabe-Datei nur in etwa vorhersagen.

Die optimale Compress-Option läßt sich nur durch direkten Vergleich
über alle Input-Datensätze hinweg ermitteln.

*SAS-Syntax für parallelen Compress-Output;

```
data BINARY(compress=binary)
     CHAR(compress=char)
     UNCOMPRESSED (compress=no);
     set INPUT_DATEI;
```

```
run;
```

Es gibt zwar keine Formel für den optimalen Compress-Typ, jedoch gewisse von der "SAS Hilfe" abweichende Spielregeln.

Compress=BINARY:
BINARY hat sich bei großen Datenmengen und bei entweder einem hohen Anteil von numerischen Variablen mit hoher Rechengenauigkeit (8-Byte-Felder) oder vielen Textfeldern mit großer Feldlänge bewährt. Bei großen Datenmengen spricht einiges dafür, BINARY als System-Default zu verwenden ('options compress=binary;'). Lediglich bei wenigen Textfeldern konkurriert COMPRESS=NO. Selbst wenn nur Textvariablen in der SAS-Tabelle vorhanden sind, kann u.U. BINARY besser als CHAR komprimieren.

Compress=CHAR:
Wenn alle Textfelder pro Datensatz konstante Länge haben (z.B. bei ausschließlich kategoriellen Feldern), schnitt die Option CHAR im Vorab-Test immer schlechter ab als die beiden anderen Optionen. Ansonsten schneidet bei großer Anzahl Datensätzen und vielen Textfeldern mit kleiner Länge CHAR am besten ab.

Compress=NO:
Je weniger Datensätze, desto klarer geht die Tendenz in Richtung COMPRESS=NO, allerdings gibt es auch weniger Speicherplatz-Unterschiede pro Compress-Typ.

Die positionelle Anordnung von Variablen nach Variablentyp (z.B. zunächst numerische, dann Textvariablen) bringt keinen zusätzlichen Effekt, jedoch kann es in Einzelfällen bei gleicher Datenstruktur und gleichen Daten bei unterschiedlichen Betriebssystemen zu Unterschieden bezüglich des optimalen Compress-Typs kommen.

Beispiel:
VarOrder=array Anz_Obs=1000 Anz_Small_Num_Var=10
Max_Small_Num_Length=8 Anz_Big_Num_Var=0
Anz_Char_Var=10 Max_Char_Length=16

LINUX Compress=BINARY oder CHAR WIN_PRO Compress=BINARY

Die Reduzierung auf maximal notwendige Feldlängen reduziert zusätzlich den Speicherbedarf.

Maximal notwendige Feldlängen lassen sich mit max(length(Text_Variable)) und "max(trunc(Numerische_Variable))" ermitteln.

Beispiel:
Anz_Obs=1000,Anz_Small_Num_Var=10,Anz_Big_Num_Var=10,Anz_Char_Var=10:

reduzierte Feldlänge: Max_Small_Num_Length=3,Max_Char_Length=16:
1.) CHAR 173056
2.) BINARY 197632
3.) UNCOMPRESSED 279552
1.6-fache Reduktion durch optimalen Compress-Typ

Default-Feldlänge (nicht-reduziert):
Max_Small_Num_Length=3,Max_Char_Length=32767:
1.) BINARY 985600
2.) CHAR 1313792
3.) UNCOMPRESSED 328521216
333.3-fache Reduktion durch optimalen Compress-Typ

985600/173056=5.7-fache Reduktion bei zusätzlich reduzierter Feldlängen

Die Reduzierung auf maximal notwendige Feldlängen bedeutet leider
auch zusätzlich notwendige attrib-Statements beim set|merge|update von Dateien
mit gleichen Variablen.

Beispiel:
Datei A, Feldlänge Variable1 $35 Datei B, Feldlänge Variable1 $70

falsch (Truncation-Fehler): data C;set A B;run;
richtig: data C;attrib Variable1 length=$70;set A B;run;

3 Verschiedene SAS Profile verwalten *(Almut Hahn)*

Man arbeitet an Projekt A, ein Kollege betritt das Büro und fragt „Kannst du mal eben (!) für Projekt B eine Kleinigkeit erledigen". Man möchte die SAS-Session von Projekt A nicht stören. Am liebsten möchte man aber mit den bewährten SAS-Einstellungen auch in Projekt B arbeiten. Mehrere Desktop-Symbole mit unabhängigen Userprofilen erleichtern auf Windows in diesem und anderen Fällen die Arbeit.

3.1 Userprofil bei SAS-Standardinstallation

Man kann sich in SAS allerlei Präferenzen einstellen (z.B. Output-Fenster -> comand line: color background cyan) und dafür sorgen, daß diese auch für zukünftige SAS-

Sessions gespeichert werden (Tools -> Options -> Preferences -> General -> Save settings on exit).

Viele der persönlichen Einstellungen werden in einem Profil gespeichert, das bei Standardinstallation von SAS unter Windwos auf einem festen Pfad - üblicherweise auf C: - gespeichert wird.
Bei Windows NT: C:\programme\sas_institute\sas\,
bei Windows 2000: C:\Dokumente und Einstellungen\<useraccount>\Eigene Dateien\My SAS Files\V8\
(den genauen Pfad findet man in der Konfigurationsdatei sasv8.cfg unter dem Punkt *sasuser*).

Standardmäßig stehen die Profile-Einstellungen nur für eine SAS-Session zur Verfügung, weil der Profile-Katalog nur einmal geöffnet werden kann. Startet man zusätzlich zur laufenden Session eine zweite Session, so findet man darin wieder nur die voreingestellten Optionen vor (im Beispiel: Output-Fenster ist weiß), da das Standardprofil bereits von der ersten SAS-Session in Benutzung ist. Schöner wäre es, auch in einer zweiten Session die persönlichen Einstellungen zur Verfügung zu haben oder sogar in verschiedenen Sessions verschiedene Einstellungen.

Einstellungen, die den Enhanced Editor betreffen (Tools -> Options -> Enhanced Editor bzw. Enhanced Editor Keys), z.B. Short-Cuts, Schriftarten, -größen und -farben (appereance scheme), gehören übrigens nicht zum Profil und stehen bei allen SAS-Sessions gleichermaßen zur Verfügung.

3.2 Unabhängige Userprofile mit Desktop-Symbolen

Es gibt die Möglichkeit, mehrere unabhängige Profile anzulegen, die verschiedenen Desktop-Symbolen zugeordnet werden.

Dazu legt man erst einmal einen Speicherort für ein neues Profil an, z.B. den Pfad C:\MeineSASProfile\Profil2\.

Dann legt man eine neue SAS-Verknüpfung auf den Desktop und begibt sich in deren Eigenschaften (rechte Maus-Taste -> Eigenschaften). Auf dem Karteireiter „Verknüpfung“ wird in der Zeile „Ziel“ der Programmaufruf angezeigt ("<Pfad>\sas.exe"). Man braucht jetzt nur noch dahinter die Option *-sasuser* (wichtig ist das Minuszeichen davor) und den Pfad des neuen Profils anzugeben, so dass dort insgesamt steht (Beispiel Windows NT):
"C:\programme\sas_institute\sas\V8\sas.exe" -sasuser "C:\MeineSASProfile\Profil2\"

Wenn man SAS über dieses neue Desktop-Symbol öffnet, findet man zunächst wieder die Einstellungen des Standardprofils vor. Man kann aber jetzt beliebige Präferenzen einstellen („Save settings on exit“ nicht vergessen!), die unabhängig von den

Einstellungen anderer Desktop-Symbole erhalten bleiben. Wenn man SAS schließt und im Explorer mal unter dem angegebenen Pfad nachschaut (im Beispiel: C:\MeineSASProfile\Profil2\), sieht man, dass SAS das neue Profil hier abgespeichert hat.

Alternativ kann man sich auch das bereits auf den persönlichen Bedarf angepasste Standardprofil von C:\... (s. o.) in das neu angelegte Verzeichnis kopieren, dann muss man mit den Einstellungen nicht bei Null anfangen. Änderungen ab da sind aber natürlich unabhängig vom Standardprofil.

3.3 Userprofile mit anderen Verknüpfungen

Wo immer in Windows Verknüpfungen stehen können, kann man nach demselben Prinzip SAS-Verknüpfungen mit speziellen Einstellungen ablegen.
Man kann also auch im Start-Menü mehrere verschiedene Programmaufrufe für SAS ablegen, die jeder ein eigenes Profil aufrufen. Auch wäre es möglich, in den Unterverzeichnissen verschiedener Projekte oder Arbeitsbereiche SAS-Verknüpfungen mit speziellen Einstellungen zu hinterlegen.

3.4 Weitere Optionen

Die Eigenschaften einer SAS-Verknüpfung kann man nicht nur über die Einstellungen, die in Profilen abgelegt werden, beeinflussen. Es gibt außer der *sasuser* Option auch noch viele weitere Optionen, die man dem Programmaufruf der Verknüpfung mitgeben kann (jeweils mit Minuszeichen davor). Hier einige Beispiele (weitere Optionen findet man in der SAS-Hilfe):

-sasinitialfolder "<Pfad>"
Damit kann man das Verzeichnis festlegen, mit dem beispielsweise bei „open“ oder „save as“ angefangen wird, zu browsen. Der Standard ist hier nämlich der Speicherort von sas.exe, also beispielsweise bei Windows NT
„C:\Programme\SAS_Institute\SAS\V8“ und wer will dort schon seine Programme ablegen?
Man könnte sich durch diese Option verschiedene Desktop-Symbole (oder andere Verknüpfungen) anlegen, die verschiedene Pfade als Arbeitverzeichnis führen: ein SAS-Symbol für Projekt A, ein anderes für Projekt B.

-mergenoby=ERROR
Diese Option sorgt dafür, dass man Fehlermeldungen erhält, wenn man versucht, ohne by-statement zu mergen.

-sasautos "<Pfad>"
Diese Option kann sinnvoll sein, wenn man mit festen Makro-Bibliotheken arbeitet.

-DMS
Läßt beim Öffnen von SAS die Explorer/Results-Fenster am linken Rand weg, so dass mehr Platz für Editor-, Log- und Output-Fenster bleiben. Über View -> Results bzw. View -> Explorer sind aber beide Fenster weiterhin erreichbar.

-MSGLEVEL=i
Zeigt zusätzlich zu Errors, Warnings und Notes außerdem noch Infos an, die oft sehr hilfreiche Hinweise liefern, z.B. beim Überschreiben von Variablen beim Merge.

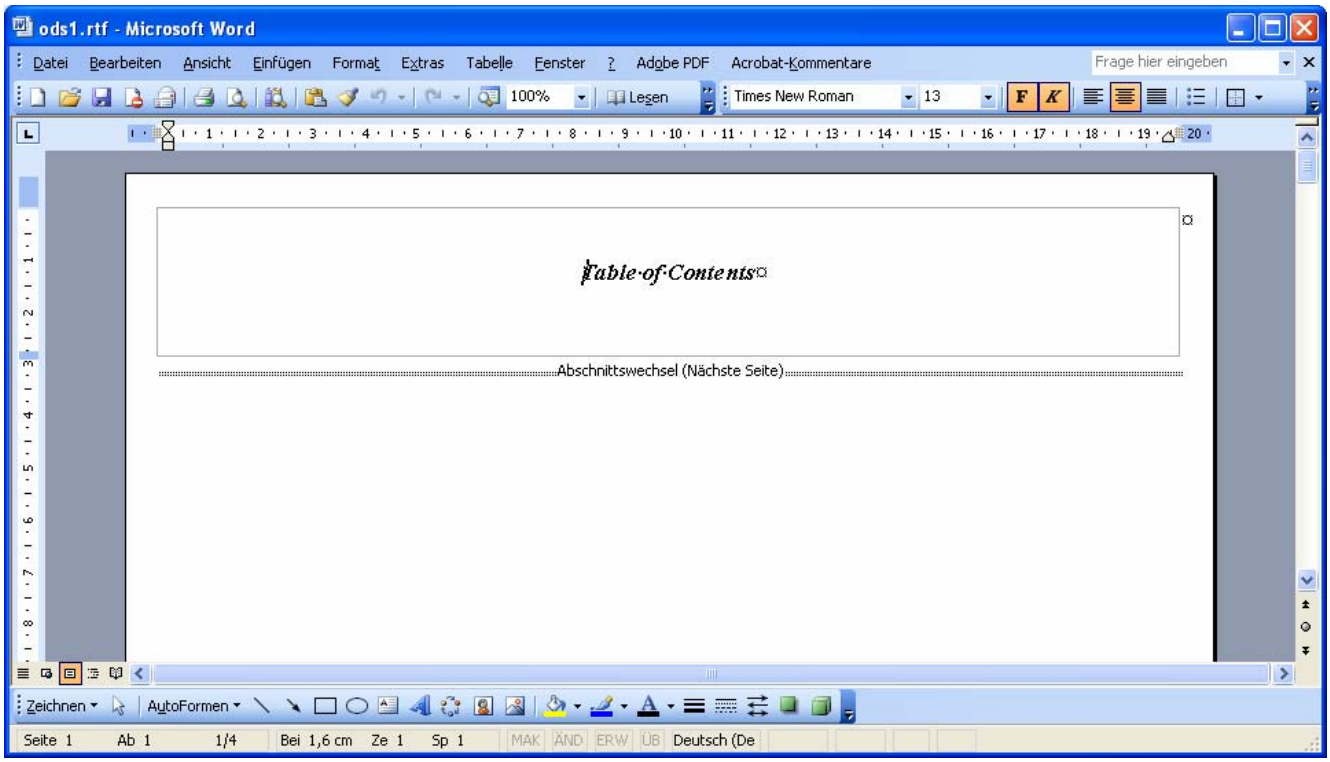
4 ODS RTF –Neues aus SAS9 *(Carina Ortseifen)*

Werden in einem Verarbeitungsschritt zahlreiche Tabellen, Grafiken und Listen erzeugt und mittels ODS RTF in ein Word-Dokument überführt, dann kann es hilfreich sein, wenn man am Anfang des Word-Dokuments ein Inhaltsverzeichnis erhält, das einen eleganten Zugriff auf einzelne Ergebnisse erlaubt.
Die Anweisung ODS RTF stellt dazu die Option Contents bereit.

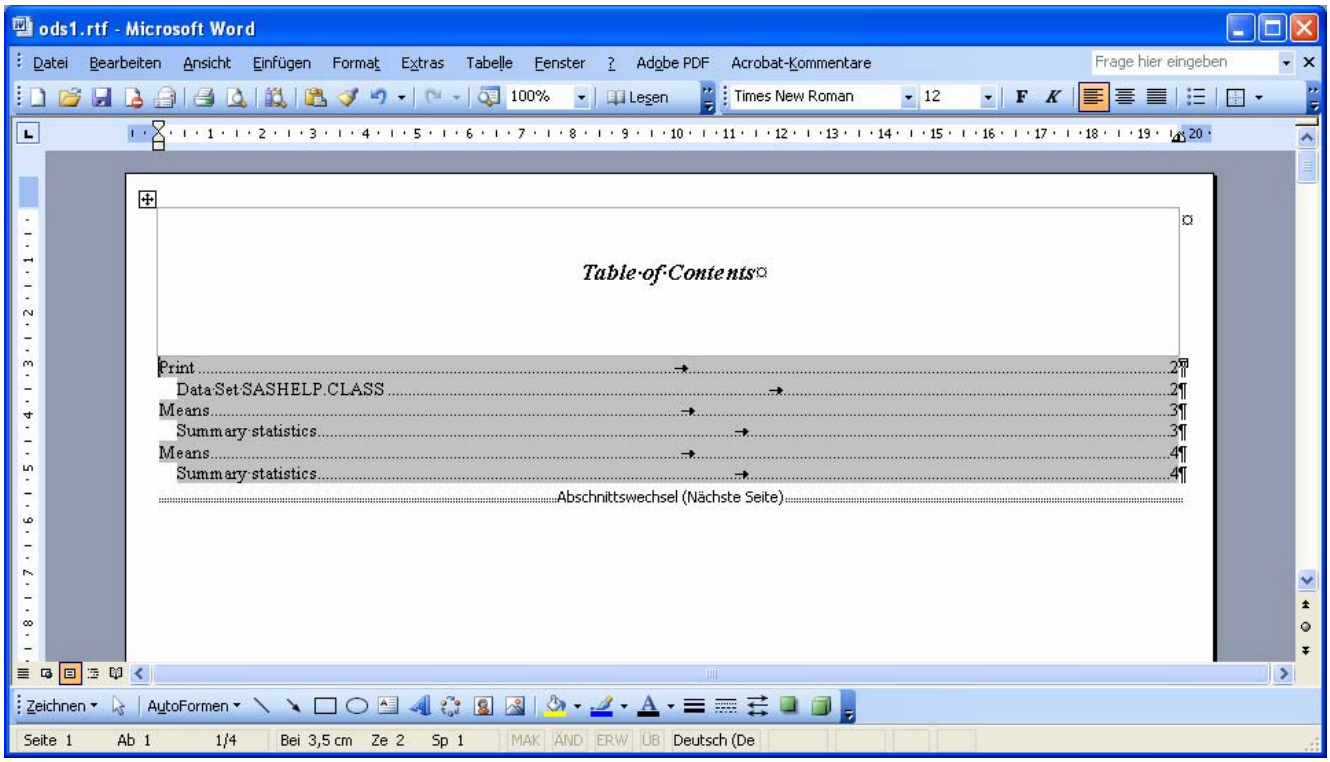
```
Ods RTF FILE='c:\kurs\ods1.rtf' CONTENTS;
PROC PRINT DATA=sashelp.class;
RUN;
PROC MEANS DATA=sashelp.class;
   VAR age;
RUN;
PROC MEANS DATA=sashelp.class;
   VAR weight height;
RUN;
ODS RTF CLOSE;
```

Wird das Programm im SAS-System ausgeführt und anschließend das Word-Dokument geöffnet, sehen Sie folgende erste Seite

(Wenn Sie sich über Extras > Optionen nicht alle Formatierungszeichen anzeigen lassen, dann fehlt bei Ihnen der Absatzwechsel.)



Zwar gibt es eine Überschrift für das Inhaltsverzeichnis, aber das eigentliche Verzeichnis fehlt. Erst wenn Sie aus dem Rahmen vor den Abschnittswechsel mit dem Cursor wechseln und F9 drücken, wird das Inhaltsverzeichnis eingefügt.



Und nun kann mittels der Steuerungstaste und der rechten Maustaste zu den einzelnen Abschnitten gesprungen werden.

Eine weitere nette Sache im Zusammenhang mit ODS RTF ist die Eintragung der Seitennummern und der Gesamtzahlen des Dokuments: Pageof.
Innerhalb von Titeln und Fußzeilen (leider nicht mit Bodytitle, vgl. SN-015727) kann folgender Ausdruck angezeigt werden: Page x of y.

Dazu notwendig ist allerdings, dass mit der Option Escapechar= per ODS-Anweisung ein Sonderzeichen definiert wird, das vor der Option pageof dieses als Option kenntlich macht. Die Option pageof selbst wird in geschweifte Klammern eingefügt, denen das Sonderzeichen vorausgeht.

```
ODS ESCAPECHAR = '*';
ODS RTF FILE='c:\kurs\test.rtf';
TITLE J=R 'Page *{pageof}';
PROC PRINT DATA=sashelp.class;
RUN;
ODS RTF CLOSE;
```

Ein entsprechendes „Seite von“ für die deutschsprachigen Nutzer ist der Autorin leider nicht bekannt.

Literatur

[2] SAS Note SN-015727, Writing Page X Of Y in RTF does not work with BODYTITLE.
(http://support.sas.com/techsup/unotes/SN/015/015727.htm, 25.01.2006)

5 Zugriff auf und Schreiben von Excel- und Access-Tabellen *(Carina Ortseifen)*

Das SAS System bietet zahlreiche Möglichkeiten, auf Microsoft Excel- und Access-Tabellen zuzugreifen und Excel- bzw. Access-Tabellen zu erzeugen. Auf früheren Konferenzen wurden einige bereits vorgestellt:

- **DDE**
  Die einfachste Möglichkeit hinsichtlich erforderlicher SAS-Module stellt der Dynamische Datenaustausch (DDE) dar, der bereits mit Base SAS funktioniert. Mit einer Filename-Anweisung wird die Verbindung zwischen SAS und Excel aufbaut und anschließend der Transfer in einem Datenschritt bewerkstelligt. Die einzelnen Schritte wurden ausführlich im SAS-Anwenderhandbuch im Netz beschrieben.

Der Zugriff ist allerdings nur für Excel- und Word-Tabellen möglich. Auf Access-Tabellen kann damit nicht zugegriffen werden.

- **ODBC**
  Eine weitere Möglichkeit stellt die Open Database Connectivity (ODBC) dar, die allerdings das SAS-Modul SAS/ACCESS TO ODBC voraussetzt und vergangenes Jahr auf der 9. KSFE in Berlin von Rudolph u.a. vorgestellt wurde.

- **Libname-Engine**
  Als dritte Variante bietet sich, sofern eine Lizenz dafür vorhanden ist, das SAS-Modul SAS/ACCESS TO PC File Formats an, das zum einen die Prozeduren Import und Export anbietet, die interaktive Variante mit dem Import- und Export-Wizard und seit der Version 9 von SAS die entsprechende Libname-Engine, womit echtes MS Excel-Format geschrieben wird, keine CSV-Datei.

Da DDE und ODBC bereits mehrfach auf den KSFE Gegenstand von Vorträgen waren und die Prozeduren Import und Export sowie der Import- und Export-Wizard schon lange verfügbar sind, möchte ich mich in meinem Tipp auf die vielleicht noch nicht allen bekannte Libname-Engine beschränken.

5.1 Beispiele

Im Unterschied zur normalen Libname-Anweisung für den Zugriff auf SAS-Bibliotheken, wo dem Schlüsselwort Libname der Name der Bibliothek und der Windows-Ordner folgt, in dem die SAS-Tabellen enthalten sind,

```
LIBNAME bib 'C:\kurs\';
```

benennt man im Zusammenhang mit der Libname-Engine für Excel-Tabellen direkt den Namen der Excel-Arbeitsmappe:

```
LIBNAME bibexcel  'C:\kurs\iris.xls';
LIBNAME bibaccess 'C:\kurs\kdat.mdb';
```

Die Engine Excel bzw. Access wird von SAS automatisch gewählt, was die entsprechende Rückmeldung im Protokollfenster widerspiegelt:

```
1    LIBNAME bibexcel "C:\kurs\iris.xls";
NOTE: Libref BIBEXCEL was successfully assigned as follows:
      Engine:        EXCEL
      Physical Name: C:\kurs\iris.xls
```

Auf die einzelnen Arbeitsblätter bzw. Tabellen innerhalb der Excel-Arbeitsmappe bzw. Access-Tabelle kann nun mit jeder Prozedur direkt darauf zugegriffen werden:

```
PROC UNIVARIATE DATA=bibexcel.iris;
RUN;
```

Mit Hilfe eines Datenschritts kann das Arbeitsblatt bzw. die Tabelle in eine SAS-Tabelle überführt werden.

```
DATA sasdaten.iris;
   SET bibexcel.iris;
RUN;
```

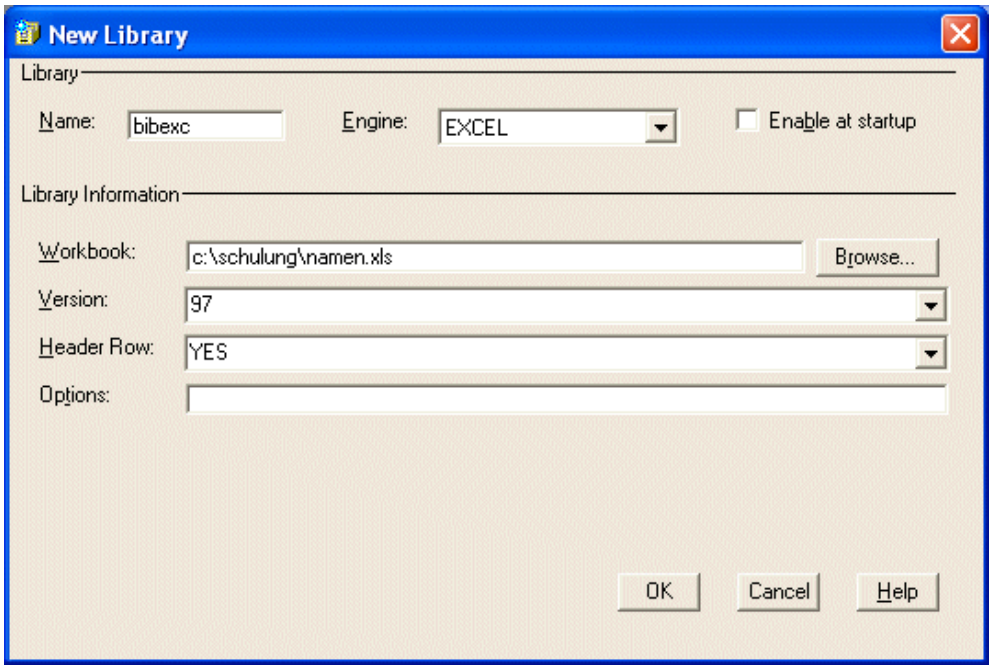
Probleme bereiten dem Anwender häufig die Namen der einzelnen Excel-Arbeitsblätter, wenn sie Sonderzeichen wie $ oder ähnliches enthalten. (Tückisch kann dabei sein, dass das Dollarzeichen innerhalb von Excel nicht erkennbar ist, sondern erst beim Eintrag in der SAS-Bibliothek sichtbar wird.) Im SAS-Programm wird dann in aller Regel eine Maskierung der Sonderzeichen fällig:

```
DATA bibexcel."tabelle1$"N;
```

Der mit Sonderzeichen versehene Name wird dazu in Anführungszeichen gefolgt von dem Buchstaben N (natürlich ist auch Klein-n möglich). Im Abschnitt ?.3 wird im Zusammenhang mit Sonderzeichen in den Namen von Excel-Spalten nochmals auf diese in der SAS Dokumentation als ‚SAS name literals' bezeichneten Namen hingewiesen.

5.2 Optionen der Anweisung Libname

Von der Prozedur Import sind Ihnen vielleicht Optionen bekannt, mit deren Hilfe die Übertragung der Werte spezifiziert werden kann: Getnames= beispielsweise für das Ablesen der Variablennamen aus der ersten Beobachtungszeile der Excel-Tabelle.



Diese Optionen können auch in der Anweisung Libname angeführt werden, wenn beispielsweise der Standard verändert werden soll:

```
LIBNAME bibexcel  'C:\kurs\iris.xls' GETNAMES=No SCANTEXT=No;
```

Weitere Optionen sind Mixed=, Usedate=, Scantime= und für Access Scanmemo=. Die komplette Liste kann in der Online-Hilfe zur Prozedur Import eingesehen werden.

Zur Anweisung Libname gibt es auch eine interaktive Variante: Über das Fenster „New Library“ wird dabei die Engine Excel (oder Access) ausgewählt werden. Daraufhin werden die Excel- bzw. Access-spezifischen Werte abgefragt. Die Optionen können ebenfalls in ein Feld eingetragen werden.

5.3 Datumsangaben, Sonderzeichen und Sonderfälle

Datumsangaben, Zeilenwechsel innerhalb einer Zelle, Sonderzeichen in Variablennamen, Labels und ausgeblendete Spalten bzw. Zeilen bereiten häufig Probleme und sollen daher im folgenden kurz behandelt werden.

5.3.1 Datumsangaben

Überträgt man die folgende Tabelle als Excel-Arbeitsblatt mithilfe einer wie oben beschriebenen Libname-Anweisung gefolgt von einem Datenschritt

Name	Zahl	Datum
Hans	12	01.12.2000
Jutta	14	29.02.2004
Anneliese	112,34	01.01.2006

werden die Datumsangaben mit dem SAS-Format Date9. belegt und innerhalb der SAS-Welt sehen die Werte wie folgt aus:

VIEWTABLE: Work.Sasdat

	Name	Zahl	Datum
1	Hans	12	01DEC2000
2	Jutta	14	29FEB2004
3	Anneliese	112.34	01JAN2006

Auch Datumsangaben mit zwei Nachkommastellen oder solche, die nur Tag und Monat enthalten, werden korrekt, d.h. mit dem Format Date9. eingelesen. Denn intern überführt Excel alle Datumsangaben in komplette Angaben, bestehend aus Tag, Monat und Jahr. (Intern wird das ganze in die Anzahl der Tage seit dem 1.1.1900 umgerechnet.) Berücksichtigen sollte man dabei, dass Excel die zweistelligen

Jahresangaben in den Bereich 1930-2029 einordnet und keine Datumsangaben vor 1900 akzeptiert.

Erweitert man den Datenschritt um eine Format-Anweisung für die Variable Datum,

```
LIBNAME excel "C:\kurs\excel2sas.xls";
DATA sasdat;
   SET excel."tabelle1$"n;
   FORMAT datum DDMMYY10.;
RUN;
```

werden die Werte in der ursprünglichen Form wiedergegeben.

	Name	Zahl	Datum
1	Hans	12	01/12/2000
2	Jutta	14	29/02/2004
3	Anneliese	112.34	01/01/2006

Wurde das Datum in Excel als Text eingegeben, muss mit Hilfe der Input- bzw. Put-Funktion der Text in einen numerischen Wert verwandelt werden.

5.3.2 Zeilenumbrüche

Enthalten Excel-Zellen Zeilenumbrüche (Linefeed mit Alt + Return, also Zeichenkette '0A'x), dann werden diese Sonderzeichen als leeres Rechteck sichtbar.
(Wurden diese Zeichen mit SAS 6.12 problemlos übertragen, führten sie mit SAS 8.2 für jeden Umbruch zu einer neuen Beobachtung und damit zu einer völlig kaputten Tabelle. In SAS 9 ist dieser Fehler wieder behoben.)

Excel

	D
	Text
.2006	Dies ist ein langer Text mit Zeilenumbruch
?. Jan	Mal schauen, wie der rüberkommt::::)
. Jan	Und jetzt?

SAS

```
The SAS System        09:26 Tuesday, January 31

Text

Dies ist ein langer Text mit □Zeilenumbruch
Mal schauen, wie der □rüberkommt::::)
Und jetzt?
```

5.3.3 Abfrage für fehlende Werte

Sollen die Beobachtungen herausgefiltert werden (mit einer If- oder Where-Anweisung), deren Textwerte einer fehlenden Angabe entsprechen (also durch ein Leerzeichen wiedergegeben werden), dann kann man im Programmschritt normalerweise die Formulierung

```
Wert = ' '
```

verwenden. Bei Abfragen auf Excel- oder Access-Tabellen funktioniert dies jedoch nicht. Ein Versuch

```
DATA sasdat;
   SET excel."tabelle1$"n;
   WHERE name = ' ';
RUN;
```

endet mit dem folgendem Ergebnis:

```
NOTE: There were 0 observations read from the data set
EXCEL.tabelle1$.
      WHERE name=' ';.
```

Die SAS-Note SN016741 bietet einen Ausweg, um diese Abfrage korrekt zu formulieren:

```
DATA sasdat;
   SET excel."tabelle1$"n;
   WHERE name IS NULL;
RUN;
```

Bei numerischen Werten tritt dieses Problem nicht auf. Die Abfrage `WHERE zahl=.;` ist eine gültige Abfrage.

5.3.4 Formeln und ausgeblendete Spalten/Zeilen in der Excel-Tabelle

Beim Import nach SAS werden alle Spalten und Zeilen übernommen, auch die ausgeblendeten.
Und sollten in Ihren Excel-Zellen Formeln enthalten sein, dann wird in die SAS-Tabelle nur das Ergebnis übernommen, nicht jedoch der Rechenausdruck. (Hier passen einfach die Philosophien von SAS als Statistikprogramm und Excel als Tabellenkalkulation nicht zueinander.)

5.4 Erzeugen von Excel- und Access-Tabellen

Beim Exportieren von SAS-Tabellen in Excel- bzw. Access-Tabellen können jeweils vier Ausgangssituationen unterschieden werden, die dem Anwender das weitere Vorgehen erleichtern[1]:

5.4.1 Excel

Vorausgesetzt wird für alle folgenden Beispiele, dass folgende Libname-Anweisung ausgeführt wurde:

```
LIBNAME e 'C:\kurs\test.xls';
```

5.4.1.1 Excelmappe test.xls ist noch nicht vorhanden

Um eine neue Excelmappe zu erstellen, benötigt man einen Datenschritt oder einen Proc Copy-Schritt (bzw. Proc Datasets mit Copy). Probleme treten dabei keine auf.
Beispiel:

```
DATA e.class ;
  SET sashelp.class ;
RUN ;
```

5.4.1.2 Excelmappe test.xls ist bereits vorhanden

(a) Ist die Excelmappe dagegen bereits vorhanden und soll diese komplett ersetzt werden, tritt ein Problem auf, da die Replace-Option nicht unterstützt wird.
Daher müssen zunächst alle Arbeitsblätter in der Excelmappe gelöscht werden, z. B. mit der Prozedur Delete.

```
PROC DELETE DATA = e._all_ ;
RUN ;
PROC COPY IN=sashelp OUT=e ;
  SELECT class.workers ;
RUN ;
```

(b) Soll nicht die gesamte Mappe, sondern nur ein gleichnamiges Excel-Arbeitsblatt durch einen Datenschritt ersetzt werden, kann dies – wegen der fehlenden Replace-Option – auch nicht direkt umgesetzt werden.
Wieder muss mittels Proc Delete das Arbeitsblatt gelöscht und anschließend mit einem Datenschritt erzeugt werden.

[1] Für die Zusammenstellung dieser vier Konstellationen sowie die anregende Diskussion bei den diversen Problemen mit dem Import von Excel-Tabellen danke ich Herrn Adolf Quast von der Deutschen Bank AG ganz herzlich.

```
PROC DELETE DATA = e.class ;
RUN ;
DATA e.class ;
  SET sashelp.class ;
RUN ;
```

(c) Will man ein zusätzliches Arbeitsblatt anlegen, kann man dies wiederum ohne Probleme mittels eines Datenschritts oder Proc Copy ausführen.

```
DATA e.workers ;
  SET sashelp.workers ;
RUN ;
```

Anmerkungen:
(1) Die Prozedur DELETE kann problemlos auch auf nicht vorhandene Dateien angewendet werden. Es erscheint dann lediglich eine Warnung im Log-Fenster, dass die zu löschende Tabelle nicht existiert.

(2) Das Fehlen der Option Replace bedeutet, es können neue Arbeitsmappen und Arbeitsblätter angelegt werden, aber Änderungen an vorhandenen Arbeitsblättern können ebenso wenig vorgenommen werden wie die Ersetzung kompletter Arbeitsmappen.

(3) Wurde eine neue Excel-Arbeitsmappe angelegt, ist es wichtig, die SAS-Bibliothek zu löschen (entweder durch Beendung der SAS-Sitzung oder mittels

```
LIBNAME e;
```

Ansonsten kann die Tabelle in Excel nicht geöffnet werden.

5.4.2 Access

Vorausgesetzt wird für alle folgenden Beispiele, dass folgende Libname-Anweisung ausgeführt wurde:

```
LIBNAME a 'C:\kurs\test.mdb';
```

5.4.2.1 Access-Datenbank test.mdb ist noch nicht vorhanden

Um eine neue Access-Datenbank zu erstellen, muss diese zunächst manuell angelegt werden. Anschließend können mit einem Daten- oder Prozedurschritt die Daten eingefügt werden.
Beispiel:

```
DATA a.class ;
  SET sashelp.class ;
RUN ;
```

5.4.2.2 Access-Datenbank test.mdb ist bereits vorhanden

(a) Ist die Access-Datenbank dagegen bereits vorhanden und soll diese komplett ersetzt werden, tritt ein Problem auf, da die Replace-Option nicht unterstützt wird.
Daher müssen zunächst alle Tabellen in der Datenbank gelöscht werden, z. B. mit der Prozedur Delete.

```
PROC DELETE DATA = a._ALL_ ;
RUN ;
PROC COPY IN=sashelp OUT=e ;
  SELECT class workers ;
RUN ;
```

(b) Soll nicht die gesamte Datenbank, sondern nur eine Tabelle ersetzt werden, kann dies – wegen der fehlenden Replace-Option – nicht direkt umgesetzt werden.
Wieder muss mittels Proc Delete das Arbeitsblatt gelöscht und anschließend mit einem Datenschritt erzeugt werden.

```
PROC DELETE DATA = a.class ;
RUN ;
DATA a.class ;
  SET sashelp.class ;
RUN ;
```

(c) Will man eine zusätzliches Tabelle anlegen, kann man dies wiederum ohne Probleme mittels eines Datenschritts oder Proc Copy ausführen.

```
DATA a..workers ;
  SET sashelp.workers ;
RUN ;
```

Literatur

[3] Rudolph, P.E., Tuchscherer, A., Ortseifen, C. (2005): *Datenaustausch zwischen SAS 9.1.3 SP1 und MS Excel / Access 2003 mittels ODBC.*
In: Rödel, E., Bödeker, R.-H. (Hrsg.): SAS: Verbindung von Theorie und Praxis. Proceedings der 9. Konferenz für SAS-Anwender in Forschung und Entwicklung (KSFE), Shaker-Verlag, Aachen, 351-367.

[4] SAS-Anwenderhandbuch im Netz.
(http://www.urz.uni-heidelberg.de/statistik/sas-ah/index.html, 25.01.2006)

[5] SAS-Note SN-016741: Subsetting on a missing value from Excel data. (http://support.sas.com/techsup/unotes/SN/016/016741.html, 25.01.2006)

6 Temporäre Buffer

Temporäre Buffer können für eine Vielzahl verschiedener Aufgaben genutzt werden, z.B. um Ergebnisse zwischenzuspeichern oder um Programmcode zusammenzusetzen. Dabei sollte der Entwickler sich einerseits keine Gedanken über den eigentlichen Speicherort machen müssen und andererseits sollte das Programm keine absoluten Pfade enthalten (wie z.B. „C:\Temp\GPfister"), damit es auf andere Rechner übertragbar ist.
Hier werden vier verschiedene Varianten gezeigt, um auf solche temporären Buffer zuzugreifen, alle verwenden das FILENAME-Statement.

6.1 Das SAS Arbeitsverzeichnis verwenden

Wie weiter oben bereits erwähnt, hat eine SAS-Sitzung immer auch ein Arbeitsverzeichnis. In diesem Verzeichnis werden alle Dateien angelegt, bei deren Allokation kein Pfad mitgegeben, sondern nur ein Dateibname angegeben wurde. Das Beispiel zeigt die Verwendung eines einfachen Filename-Statements ohne Pfadangabe.

```
Filename buffer "tmp.txt";
Data _null_;
  File buffer dlm=";";
  Set Sashelp.Class;
  Put name sex age weight height;
Run;
```

Gespeichert wird die Ausgabe unter dem Pfad, der als SAS-Arbeitsverzeichnis eingestellt ist. Er wird i.d.R. in der Statuszeile angezeigt. Über einen Doppelklick auf diese Statuszeile kann ein Dialogfenster geöffnet werden, in dem das Arbeitsverzeichnis umgestellt werden kann. Diese Umstellung kann aber auch im Programm erfolgen, wie das folgende Beispiel zeigt.

```
X "cd d:\temp";
Filename buffer "tmp.txt";
Data _null_;
  File buffer dlm=";";
  Set Sashelp.Class;
  Put name sex age weight height;
Run;
```

In diesem Falle wird die Datei unter „D:\temp\tmp.txt“ gespeichert.

6.2 Einen bereits allokierten Pfad verwenden

Diese Methode kann für temporäre Buffer verwendet werden, aber auch, um Dateien gezielt in einem bestimmten Verzeichnis anzulegen, ohne dass das Verzeichnis hartkodiert im Programm stehen muss. Dabei wird die Funktion pathname() genutzt, sie

gibt den physischen Pfad eines Librefs oder Filerefs zurück. Daraus kann dann ein Filename-Statement konstruiert werden.

```
Filename buffer "%Sysfunc(pathname(WORK))/tmp.txt";
Data _null_;
  File buffer dlm=";";
  Set Sashelp.Class;
  Put name sex age weight height;
Run;
```

Im Beispiel wird die Datei unter dem Pfad der WORK-Library angelegt, deren Verzeichnis mit Beenden der SAS-Sitzung wieder gelöscht wird.

6.3 Von SAS einen Buffer allokieren lassen

Der unter 6.2 vorgestellte Ansatz lässt sich noch kürzer schreiben. Das Filename-Statement kann selbständig einen temporären Buffer allokieren.

```
Filename buffer TEMP;
```

Dieser Buffer wird ebenfalls unter dem Pfad der WORK-Library angelegt, die Benennung erfolgt aber automatisch. Hier ein Beispiel in dem ein Programm aus mehreren Makroaufrufen zusammengesetzt wird: Es werden zunächst zwei Makros definiert, die eine Ausgabe in das LOG schreiben. Dann wird eine Tabelle erzeugt, die mehrere Aufrufe dieser Makros in einer bestimmten Reihenfolge enthält (sozusagen eine Mini-Variante einer Metadaten-Steuerung). Anschließend wird aus der Tabelle ein temporäres Programm mit den Aufrufen geschrieben und dann mit Hilfe von %INCLUDE abgeschickt.

```
%Macro One;  %Put --> Macro One; %Mend;
%Macro Two;  %Put --> Macro Two; %Mend;

Data Work.Meta;
  callMacro = "One";  Output;
  callMacro = "Two";  Output;
  callMacro = "Two";  Output;
  callMacro = "One";  Output;
Run;

Filename buffer TEMP;

*GP Makroaufrufe zusammensetzen
    und zwischenspeichern       *;
Data _null_;
  Set Work.Meta;
  File buffer;
  Put '%' callMacro;
Run;
```

```
%Include buffer;
```

6.4 Die Windows Zwischenablage benutzen

Die Zwischenablage ist unter Windows ein Speicherbereich in den mit Copy & Paste Inhalte eingefügt bzw. daraus entnommen werden können. Das Filename-Statement unterstützt diesen Buffer ebenfalls. So können z.B. ganz einfach Daten interaktiv nach Excel übernommen werden.

```
Filename buffer CLIPBRD;

Data _null_;
  File buffer dlm="09"x;
  Set Sashelp.Class;
  Format weight height commax10.2;
  Put name sex age weight height;
Run;
```

Das Beispiel kopiert die Tabelle Sashelp.Class in die Zwischenablage. Die Option für den Delimiter »dlm=“09“x« bedeutet, dass als Trennzeichen Tabulatoren verwendet werden. In Excel können die Daten dann einfach mit STRG+V bzw. über das Menü Bearbeiten → Einfügen in ein Arbeitsblatt kopiert werden.

Literatur

[1] SAS-Online Hilfe zu Filename-Statement (hier auch unter dem jeweiligen Betriebssystem schauen!)

7 Ende gut, alles gut – Grafiken mit True Type Fonts und ODS

SAS/GRAPH unterstützt bereits seit Version 6 neben den eigenen Schriftarten (Swiss, Xswiss,…) auch normale True Type Fonts wie z.B. Arial oder Times New Roman. Allerdings gibt es in der Version 8 Probleme in der Zusammenarbeit mit dem ODS, wie im Rahmen der Tipps und Tricks auch mehrfach angesprochen. Mit Version 9 gibt es jetzt eine Neuerung, die SAS/GRAPH und das ODS voll kompatibel macht.

7.1 Die Problemstellung

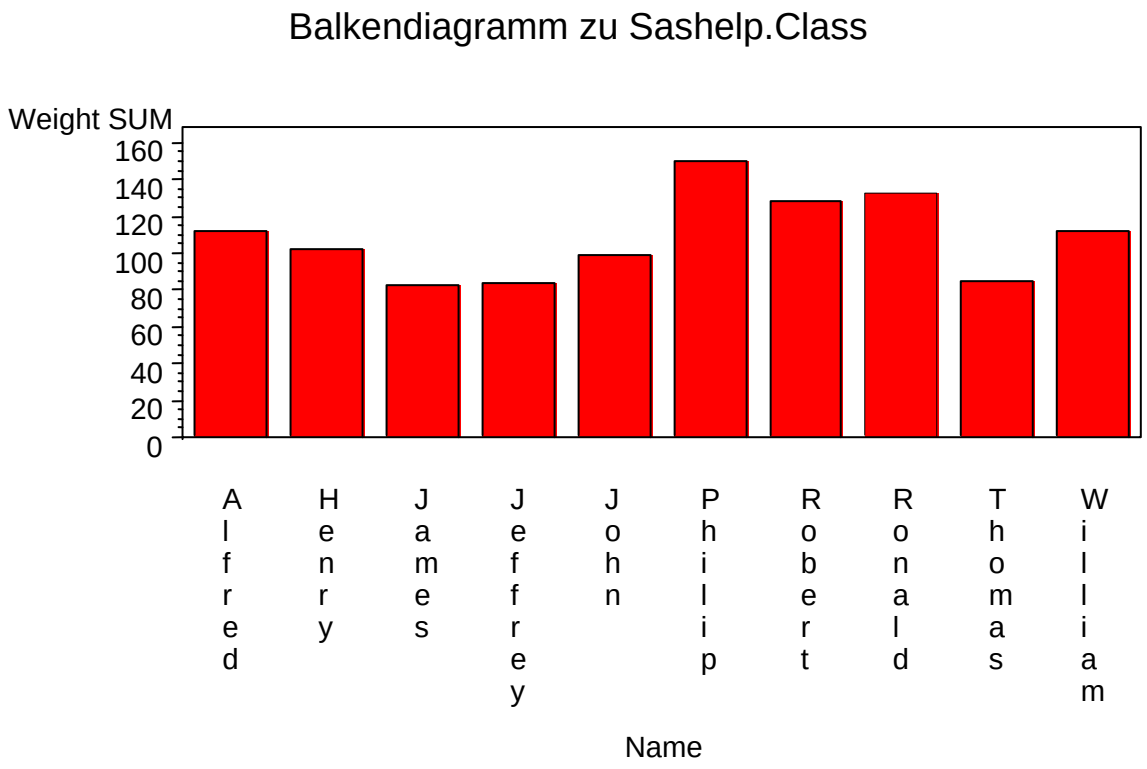
Folgendes Beispiel illustriert die Problematik: Über Graphik-Optionen wird Arial als Schriftart angewählt und eine Grafik erzeugt.

```
Goptions
  ftitle = "Arial" htitle = 16 pt
  ftext  = "Arial" htext  = 12 pt
;

Title "Balkendiagramm zu Sashelp.Class";
Proc Gchart data=Sashelp.Class(where=(sex="M"));
```

```
  Vbar name / sumvar = weight;
  Run;
Quit;
```

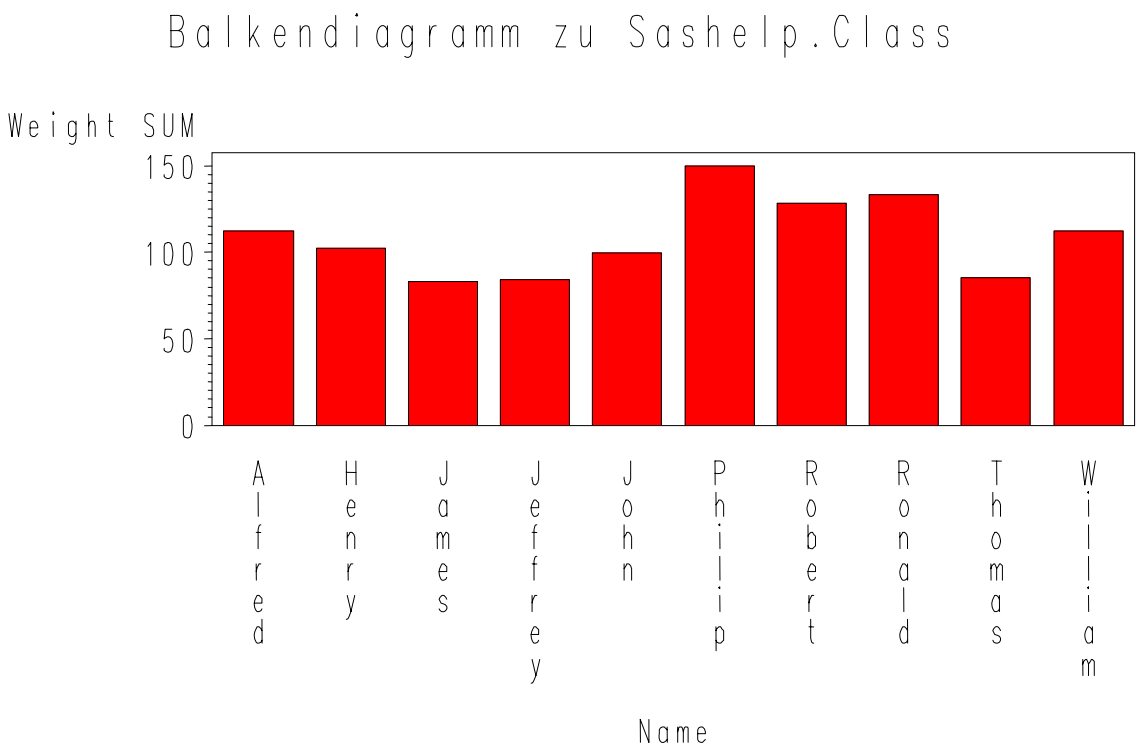
Das Ergebnis zeigt, dass die Schriftart Arial problemlos angezeigt werden kann.



Jetzt wird das Programm um ODS-Klammern ergänzt und nochmals abgeschickt.

```
Ods Rtf file="d:/tmp.rtf";
Proc Gchart data=Sashelp.Class(where=(sex="M"));
…
Ods Rtf Close;
```

Das wenig erfreuliche Ergebnis sieht so aus:



Das Log enthält folgenden Warnhinweis:

```
WARNING: Font Arial could not be used.
         Font SIMULATE substituted for font Arial.
```

Offensichtlich kann die Schriftart Arial nicht gefunden werden, stattdessen wird eine SAS-eigene Schriftart verwendet. Um zu verstehen, warum die Schrifart zwar am Bildschirm angezeigt werden kann (und auch in exportierten Grafiken enthalten ist), bei Einbettung in das ODS aber nicht gefunden wird, ist ein genauerer Blick auf SAS/GRAPH notwendig.

7.2 Das Device-Konzept von SAS/GRAPH

SAS/GRAPH verwendet ein sogenanntes Device-Konzept. Dabei wird die Graphik zunächst unabhängig vom eigentlichen Ausgabeziel erzeugt und in einem Katalog-Eintrag gespeichert. Diese Device-unabhängige Graphik wird dann von einem bestimmten Device interpretiert und dargestellt. Typische Devices sind „Win“ – der Bildschirm oder „gif“, hier wird die Grafik mit dem „graphics interchange format“ in eine Datei gespeichert.

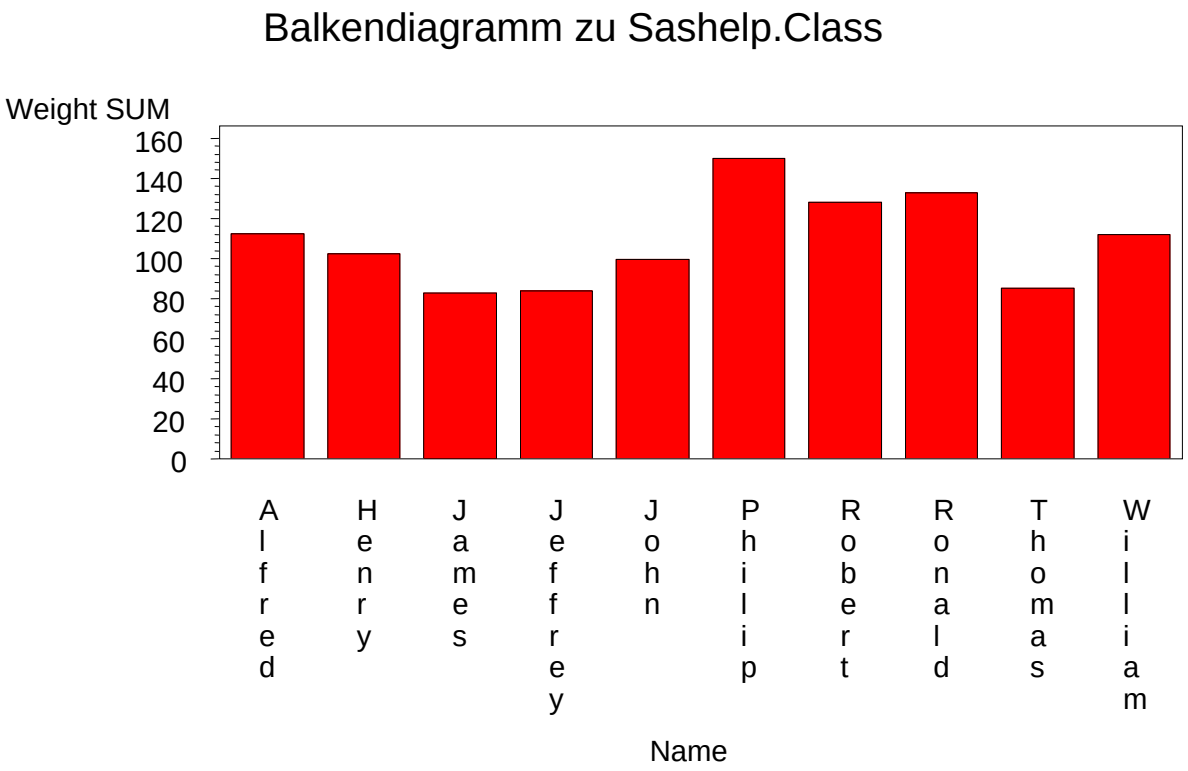
In Version 6 mussten Schriftarten bei jedem Device einzeln bekannt gemacht werden, der Device enthielt eine Übersetzungliste, die den Namen des True Type Fonts und einen internen Namen enthielt. Über diesen internen Namen konnte die Schriftart dann verwendet werden. Mit Version 8 wurde dieser Weg vereinfacht, die meisten Devices konnten (unter Windows) die eigentlichen Schriftartnamen verwenden. Nur mit dem ODS funktioniert das nicht, weil die dort automatisch verwendeten Devices die Schriftarten nicht finden.

7.3 Die Free Type Library unter V9

Deshalb wurde für V9 eine Erweiterung hinzugefügt, die Free Type Library. Diese Library wird von den ODS-Devices verwendet und enthält registrierte True Type Fonts. Der Unterschied ist, dass hier die Schriftarten im SAS registriert werden müssen. Zu diesem Zweck gibt es im SAS Registrier-Editor unter *Core→ Printing→ Freetype→ Fonts* einen Bereich, in dem die registrierten Schriftarten angezeigt werden, sowie die neue Prozedur FONTREG, mit deren Hilfe die Schriftarten installiert werden.

```
Proc Fontreg;
  Fontfile "C:/Windows/Fonts/arial.ttf";
Run;
```

Dieses Beispiel registriert die Schriftart Arial, wenn das Test-Programm nochmals aufgerufen wird, enthält die via ODS erzeugte Graphik die gewünschte Schriftart.



Noch ein Hinweis, es müssen alle Varianten einer Schriftart (fett, kursiv, fett und kursiv) einzeln installiert werden, allerdings zeigt der Registrier-Editor nur einen Eintrag an. Deshalb kann Proc FONTREG auch so benutzt werden, dass alle in einem Verzeichnis enthaltenen True Type Fonts auf einen Schlag registriert werden.

Literatur

[1] TS 674: An Introduction to Exporting SAS/GRAPH Output to Microsoft Office SAS Release 8.2 and higher.

[2] Base SAS 9.1.3 Procedures Guide, Chapter 22 The FONTREG Procedure.

8 Java im Data Step

Eine neue – in SAS Version 9.1.3 noch experimentelle – Erweiterung ermöglicht die Verwendung von Java Klassen im Data Step. Das ermöglicht zum einen die Wiederverwendung bereits in Java implementierter Geschäftslogiken und zeigt, dass SAS ein in hohem Maße integrierbares und offenes System ist. Für Anwender mit weniger komplexen Anforderungen bietet sich die Möglichkeit, eigene Funktionen zu entwickeln und so das SAS-Leistungsspektrum nach ihren Wünschen zu erweitern.
Zu diesem Thema gibt es eine sehr gute Aufbereitung, die von Richard A. DeVenezia auf der 30. SUGI vorgestellt wurde und die als Grundlage dieses Tipps dient. An dieser Stelle erfolgt deshalb nur eine relativ kurz gehaltene Dokumentation der Syntax, ausführlich kann sie in den unter Literatur angegebenen Quellen nachgelesen werden.

8.1 Die Java Runtime Environment (JRE)

Bei der Installation von SAS 9 wird auch eine JRE installiert, die benötigt wird, um Java-Programme auszuführen. Das ist zunächst einmal für die Clients der BI-Plattform gedacht wie z.B. die Management-Konsole. Aber auch der Data Step verwendet diese JRE, wenn Java-Klassen instantiiert werden. Allerdings muss die JRE die verwendeten Klassen auch finden können. Laut offizieller Dokumentation kann der von der JRE verwendete Klassenpfad auf verschiedenen Wegen erweitert werden, leider funktionierte bei mir keine der Methoden. Allerdings hat die JRE auch einen Standardpfad, unter dem sie nach Klassen sucht. Die neue Prozdedur JAVAINFO zeigt diesen Pfad unter „java.class.path=“ an, allerdings existiert das hier aufgelistete Verzeichnis „classes“ unter dem Pfad „…\SAS Institute\Shared Files\JRE\1.4.1“ nicht. Wird dieses Verzeichnis angelegt, können eigene Klassen dort hineinkopiert werden und stehen ohne Anpassung des Klassenpfades zur Verfügung.

8.2 Verwenden von Java Klassen

Die Verwendung von Java Klassen besteht aus folgenden Schritten:

1.) Erzeugen eines Objektes
2.) Lesen und Setzen von Attributen
3.) Aufruf von Methoden
4.) Beenden des Objektes

Diese Schritte sollen anhand eines Beispieles kurz vorgestellt werden. Die zugrundeliegende Java-Klasse hat folgende Definition:

```
public class Test001 {
  public String lastName = "no lastname given";
  public String firstName = "no firstname given";
  public int    age = 0;

  public String toString() {
    return ("this is " + firstName + " " + lastName +
```

```
            ", age of " + age);
  }
}
```

Diese sehr einfache Klasse besitzt drei Felder, die von außen gelesen und gesetzt werden dürfen, die Zeichenketten lastName und firstName, sowie den Integer-Wert age. Die einzige definierte Methode heißt toString und gibt eine Zeichenkette zurück.

8.2.1 Erzeugen eines Objekts

Es gibt zwei Varianten eine Instanz einer Klasse zu erzeugen, bei beiden muss die Declare-Anweisung verwendet werden. Die erste Variante weist zunächst einer Variablen den Typ „javaobj“ zu und erzeugt dann im zweiten Schritt das eigentliche Objekt. Dabei wird als Parameter der Name der eigentlichen Klasse übergeben.

```
Data _Null_;
  Dcl javaobj obj1;
  Obj1 = _new_ javaobj(„Test001“);
  ...
```

Die zweite Variante ist eine Kurzform die Zuweisung und Instantiierung in einen Schritt zusammenfasst.

```
Dcl javaobj obj1("Test001a");
```

In beiden Fällen steht jetzt mit „obj1“ ein Objekt zur Verfügung, dessen Attribute gelesen oder gesetzt und dessen Methoden aufgerufen werden können.

8.2.2 Lesen und Setzen von Attributen

Das Lesen und Setzen von Attributen geschieht indirekt über verschiedene Zugriffsmethoden. Hintergrund ist, dass Java als typenbasierte Sprache wesentlich mehr Datentypen kennt als SAS. Während SAS nur einen numerischen Typen anbietet, unterscheidet Java byte, short, int, long, float und double (sog. primitive Datentypen). Der Java Datentyp String ist eigentlich eine Klasse. Die Zugriffsmethoden übersetzen zwischen den SAS Datentypen und den Java Typen, dabei liest ein Teil den Wert aus dem Attribut – sog. „getter“-Methoden, der andere schreibt in das Attribut hinein – die „setter“-Methoden.
Außerdem bietet Java noch die Möglichkeit Attribute auf Klassen- oder auf Instanzebene anzulegen, sodass eine Vielzahl unterschiedlicher Methodenaufrufe zusammenkommt. In Kurzdarstellung lässt sich das so formulieren:

`obj1.`	`get`
	`set`
		`Static`	`Byte`
			`Short`
			`Int`
			`Long`
			`Float`
			`Double`
			`String`
				`Field(„name“,var)`

Typische Methodenaufrufe für das Lesen oder Setzen von Attributen sind:

```
obj1.getStringField(„firstName“,string);
obj1.setIntField(„age“,33);
obj1.getStaticIntField(„counter“,n);
```

Die Datentypen boolean und char werden in 9.1 noch nicht unterstützt.

8.2.3 Aufruf von Methoden

Für den Aufruf von Methoden gilt ähnliches wie für die Attribute. Erstens gibt es Statische Methoden auf Klassenebene und zweitens werden die Methoden nach dem Type ihres Rückgabewertes unterschieden. Hier ist neben String und den numerischen Typen auch „Void“ gültig - für Methoden ohne Rückgabewert.

`obj1.`	`call`	`Static`	`Void`
			`Byte`
			`Short`
			`Int`
			`Long`
			`Float`
			`Double`
			`String`
				`Method(„name“<,para`$_1$`,…,para`$_n$`<,rcvar>>)`

Die an den Methodenaufruf übergebenen Parameter müssen exakt der Signatur der Java-Methode entsprechen. Hier gibt es ein Problem: SAS Zeichenketten werden in den Java-Typ String übersetzt, numerische Werte in den Typ double. Wenn die Definition der Methode in Java aber einen anderen numerischen Datentypen erwartet, kann die Methode nicht aus dem Data Step heraus aufgerufen werden. In diesem Falle muss eine sog. Wrapper-Klasse geschrieben werden, die die Datentypen von dem Format das SAS liefert in den korrekten Typen für die Java-Klasse übersetzt.
Der dritte Datentyp, der aus SAS heraus an eine Java-Methode übergeben werden kann ist ein Java-Objekt, wobei auch hier die Signatur der Java-Methode den gleichen Objekt-Typen enthalten muss.
Typische Aufrufe von Methoden eines Java-Objektes sind:

```
obj1.callStringMethod(„toString“,string);
obj1.callIntMethod(„add“,1,5,result);
```

Es können übrigens auch SAS Arrays übergeben werden, die dann in entsprechenden Java-Arrays gespeichert werden!

8.2.4 Beenden des Objektes

Am Ende eines Data Steps sollte das Java-Objekt mit der delete()-Methode beendet werden.

```
obj1.delete();
```

8.3 Beispiel für die Verwendung der Klasse Test001

Hier ein paar Beispiele zur Verwendung der unter 8.2 vorgestellten Klasse Test001. Zunächst wird eine Instanz erzeugt und die toString-Methode aufgerufen.

```
Data _Null_;
  Length string $256;
  Declare javaObj obj1("Test001");
  rc = obj1.callStringMethod("toString",string);
  Put rc= string=;
  obj1.delete();
Run;
```

Im Log steht

```
WARNING: The JAVAOBJ is an experimental feature in this release and
is not intended for use in the development of production
applications.
rc=0 string=this is no firstname given no lastname given, age of 0
```

Da keines der drei Attribute gesetzt wurde, werden die Default-Werte eingesetzt. Das zweite Beispiel füllt zunächst die Attribute und ruft dann die Methode auf.

```
Data _Null_;
  Length string $256;
  Declare javaObj obj1("Test001");
  obj1.setStringField("lastName","Pfister");
  obj1.setStringField("firstName","Grischa");
  obj1.setIntField("age",36);
  obj1.callStringMethod("toString",string);
  Put string=;
  obj1.delete();
Run;
```

Jetzt steht im Log

```
string=this is Grischa Pfister, age of 36
```

Weitere Beispiele für die Verwendung finden sich in der Literatur.

Literatur

[1] Richard A. DeVenezia: Java in SAS. JavaObj, a DATA Step Component Object. SUGI 30, Paper 241.

[2] http://support.sas.com/rnd/base/topics/datastep/dot/javaobj.html