SAS-Praktikum

Institut für Stochastik, Universität Ulm
Vorlesungsbegleiter; Stand: 2. Juli 2009

Inhaltsverzeichnis

1 Einführung in SAS

1.1 Starten von SAS

Das Programmpaket SAS befindet sich installiert auf einem Rechner des Rechenzentrums (RZ) der Universität. Somit ist es für alle Kursteilnehmer notwendig, neben einem gültigen Account auf den Rechnern der Fakultät (Thales, Turing,.....) ebenfalls einen gültigen Account für die Rechner des Rechenzentrums zu besitzen (Antrag stellen usw.), um auf das Programmpaket SAS zugreifen zu können. Im Folgenden wird dies vorausgesetzt.

Der Rechner des RZ, auf dem SAS verfügbar ist, hat den Namen andromeda. Somit muss zuerst eine Verbindung vom Rechner der Fakultät zur andromeda hergestellt werden. Im Folgenden sehen Sie den Ablauf für jeden Start einer SAS-Sitzung von einem Rechner (z.B. der Turing) unserer Fakultät:

1. Anmelden auf dem Fakultätsrechner
2. Verbindung zu RZ-Rechner herstellen:

   `turing$ ssh -X -l [username] login.rz.uni-ulm.de`

   wobei der username der beim kiz (Uni-Mail) ist.
3. Eingabe des geforderten Passwortes (normalerweise das Mail-Passwort, es sei denn es wurde geändert)
4. Bereitstellen des SAS-Programmpaketes:

   `andromeda$ option sas92`
5. Aufruf von SAS:

   `andromeda$ sas &`

Danach öffnen sich auf Ihrem Monitor mehrere Fenster, u.a. der SAS-Programm-Editor, das SAS-LOG-Fenster und das SAS-Output-Fenster. Zusätzlich erscheint die SAS-Toolbox.

Beim erstmaligen Aufruf wird von „`option sas`“ ein persönliches SAS-Verzeichnis angelegt, das dann unter der Umgebungsvariable `$SASHOME` zugänglich ist. Ein Wechsel in dieses Verzeichnis kann also z.B. durch „`cd $SASHOME`“ erfolgen.

1.2 Die SAS-Oberfläche

1.2.1 Der Editor

Tastaturkürzel:

Basis	Strg+X	Wechsel Einfüge-Modus / Überschreibe-Modus
Ausführung	F3	Submit
	F4	Recall
Kopieren / Einfügen	Strg + F	Mark (beginnt markierten Bereich)
	Strg + K	Cut (Bereich ausschneiden)
	Strg + R	Store (Kopieren)
	Strg + T	Einfügen
Weitere	Strg + Y	In Kommandomodus wechseln / Kommmandomodus verlassen

Weitere Kürzel gibt es bei Tools->Options->Keys

Zeilen löschen / einfügen Um Zeilen einzufügen, gehen Sie mit dem Cursor in den Bereich mit den Zeilennummern, und geben „i“ ein, bzw. z. B. „i5“, um 5 Zeilen nach der aktuellen einzufügen. Um Zeilen zu löschen, kann man gleich vorgehen, nur statt „i“ ist „d“ einzugeben.

1.2.2 Hilfe zu SAS

Zur SAS-Version 9 gibt es eine Seite, auf der die gesamte Dokumentation verlinkt ist:
http://support.sas.com/onlinedoc/913/docMainpage.jsp

1.3 Grundprinzipien der SAS-Programmierung

- Befehle hören mit einem Semikolon auf.
- Groß- und Kleinschreibung spielen keine Rolle.
- Variablennamen können Buchstaben, Zahlen und Unterstriche (_) enthalten, können aber nicht mit einer Zahl beginnen. Systemvariablen beginnen typischerweise mit einem _.
- Kommentare haben immer folgende Form: `/* Kommentar */`.

1.4 Der DATA-Step

Als grobe Einteilung gilt: Der Data-Step dient dem Auslesen, der Manipulation und dem Schreiben von Daten, wohingegen der Proc-Step der Auswertung und (graphischen) Ausgabe dient.

Wir empfehlen, der Übersicht halber Data- und Proc-Steps mit `RUN;` abzuschließen, es ist aber eigentlich nicht unbedingt nötig. Nur einmal am Ende der Datei sollte es stehen, damit SAS das Programm auch ausführt.

Das Ergebnis des Data-Steps ist immer eine .sas-Datei, aber meistens eine temporäre.

1.4.1 Einlesen von Text-Daten-Dateien

Die im RZ installierte SAS-Version (sonst wissen wir es nicht so genau) hat leider ein paar „Besonderheiten“beim Einlesen von Textdateien:

- DOS-Zeilenumbrüche machen Probleme (also auch Windows-Dateien).
- Als Dezimaltrenner ist nur der Punkt zulässig, Kommas gehen nicht.

Die von uns gelieferten Daten entsprechen aber alle diese Vorgaben.

Ein einfaches Beispiel: Datei hat 3 Spalten, die erste ist ein String, dann noch 2 Zahlen. Spaltentrenner sind Tabs. Dabei soll die ersten Zeile ignoriert werden, sie enthält Spaltennamen:

```
Name    Size    Price
Dark.Bounty     50.0    0.88
Bounty  50.0    0.88
...
```

Hier der SAS-Code zum Einlesen:

```
DATA choc;
    INFILE "chocolates.dat" DLM="09"x FIRSTOBS=2 /*OBS= */;
    INPUT name$ size price;
RUN;
```

Syntax:

- `DATA`-Zeile Name der (temporäreren) SAS-Datei
- `INFILE`-Zeile:
  - Dateiname
  - `DLM`: Datentrenner, per Default Leerzeichen, Tab kriegt man mit Ö9ẍ
  - `FIRSTOBS`: erste relevante Zeile
  - `OBS`: letzte relevante Zeile (wird selten gebraucht)
- `INPUT`-Zeile: Spaltennamen, für String-Spalten mit $ am Schluss; Spaltennummern können auch angegeben werden, z.B. wäre oben `size 5-7` richtig, wenn „size“ immer in den Spalten 5 bis 7 stünde.
- `RUN;` dient (fast) nur zur Übersicht.

1.4.2 Speichern von SAS-Daten

```
LIBNAME lib "$SASHOME/praktikum";
DATA lib.choc;
    ...
RUN;
```

Speichert die Daten in `~/praktikum/choc.sas7bdat`.

1.4.3 Einlesen von SAS-Daten-Dateien

```
LIBNAME lib "$SASHOME/praktikum";
DATA choc2;
    SET lib.choc;
RUN;
```

liest den Datensatz `lib.choc` (aus der Datei `~/praktikum/choc.sas7bdat`) in `choc2` ein.

1.4.4 einfache Funktionen

Enthält ein Datensatz die Spalten A und B, so kann man die Summe wie folgt berechnen:

```
Summe = A + B;
```

Weitere Operationen sind:

**	Potenzierung
*	Multiplikation
/	Division
+	Addition
-	Subtraktion

Hier noch ein Auszug aus den vielen SAS-Funktionen:

```
arithmetische (SUM MAX MIN MOD SQRT)
Rundungsfunktionen (INT CEIL FLOOR ROUND)
mathematische (EXP GAMMA LOG LOG2 LOG10)
trigonometrische (SIN COS TAN SINH COSH TANH ARSIN ARCOS ARTAN)
Wahrscheinlichkeitsfunktionen (POISSON PROBETA PROBF PROBNORM)
statistische (MEAN N NMISS STD VAR RANGE CV SKEWNESS)
Zufallszahlen (RANNOR NORMAL RANBIN RANPOI RANUNI UNIFORM)
String (LEFT RIGHT SUBSTR LENGTH)
Datum (DATE DAY HOUR TIME)
Kommunikation mit dem Betriebssystem (CMS X)
```

Bei logischen Operationen ist das Ergebnis „1“, wenn die Operation richtig ist, ansonsten „0“.

Beispiel:

```
DATA TEST;
   A='M';
   B='W';
   U=A NE B;
   V=A EQ B;
   W=A LE B;
```

das ERGEBNIS ist: U=1, V=0, W=1 (zur Berechnung von W werden die Bitmuster numerisch verglichen).

Weitere logische Operationen sind:

```
& = AND
| = OR
```

Hier noch ein Beispiel für das Aneinanderhängen von Strings, das manchmal gebraucht wird:

```
   A='HAUS';
   B='GAST';
   C=B||A;
```

`C` hat den Inhalt 'GASTHAUS'

1.4.5 Arrays

Normales Einlesen von Daten mit inhaltlich gleichen Spalten (und bilden des Minimums):

```
   INPUT F1 F2 F3 F4 F5 F6 F7 F8 F9 F10;
   B = MIN(F1, F2, F3, F4, F5, F6, F7, F8, F9, F10);
```

Gleiches leistet folgende Anweisungen im DATA-Step:

```
   INPUT F1-F10;
   B=MIN(OF F1-F10);
```

Verwendung des `ARRAY`-Stichwortes für Schleifen (erfordert INPUT-Anweisung wie oben):

```
ARRAY X (*) F1-F10;  /* definiert den indizierten Bereich X */
A=X(5);   /* A enthält das fünfte Elemente des Bereichs X */

ARRAY TEST (*) F1-F10 N1-N100 X1-X10;
DO I = 1 TO DIM(TEST);
  TEST(I) = I * 2;
END;
```

1.4.6 If und Schleifen

Einfaches Beispiel:

```
IF B_LAND='V' OR B_LAND='T' THEN
    REGION='WESTEN';
ELSE
    REGION = 'SONST ';
```

Datenauswahl mit `IF`:

```
IF A='weiblich';
```

Es werden nur jene Beobachtungen in der Datenmatrix behalten, bei denen gilt: A='weiblich'. Das 'Subsetting IF' eliminiert jene Beobachtungen aus der Datenmatrix, für welche die Bedingung nicht erfüllt ist.

SELECT-Anweisung (Fallunterscheidung):

```
SELECT (A);
     WHEN (1) B = 3;
     WHEN (2) ;
     WHEN (3) B = 4;
   OTHERWISE B = 0;
  END;
```

While-Schleife:

```
DO WHILE (K_ALTER(I) NE .);
     KINDER_Z = KINDER_Z + 1;
     I = I + 1;
END;
```

DO UNTIL-Schleife:

```
DO UNTIL (bedingung); anweisungen; END;
```

Wiederholungsschleifen:

```
DO I=1 TO 10;               /* I nimmt die Werte 1, 2, ..., 10 an */
DO K=1 TO 10 BY 2;          /* 1, 3, 5, 7, 9 */
DO ZAEHLER=1 TO 2 BY 0.2 WHILE (J NE .); /* 1, 1.2, 1.4, 1.6,
                               1.8, 2, solange J nicht "missing value" */
DO I=1, 10, 100;            /* 1, 10 , 100 */
DO ALPHA='JAN', 'FEB', 'MAR'; /* JAN, FEB, MAR */
```

1.4.7 DROP, KEEP, OUTPUT und co

Behalten / Löschen bestimmter Spalten:

```
KEEP GES_UMS;
DROP UMS1 UMS2;
```

Bei mehereren Datensätzen gleichzeitig:

```
DATA EINS (DROP=I J K) ZWEI (KEEP=A B C D) DREI (DROP=A C D);
```

Zeilen behalten / löschen:

```
IF BUNDLAND = 'TIR' THEN OUTPUT;
DELETE;
```

Zeilen zu Datensätzen zuordnen:

```
DATA INVENT BESTELL FEHL;
INPUT ART_NR IST SOLL MINDEST;
OUTPUT INVENT;
IF MINDEST < IST THEN DO;
  B_MEN = MINDEST - IST;
  OUTPUT BESTELL;
END;
IF IST NE SOLL THEN OUTPUT FEHL;
```

Aneinanderhängen von Datensätzen (die beide die Spalte `NAME` enthalten):

```
SET ALLE;
SET ABTEIL1 ABTEIL2; BY NAME;
```

Verbinden von Datensätzen anhand einer Spalte:

```
DATA KUNDEN;
 MERGE NAMEN WOHNORT;
 BY KDN_NR;
```

1.5 Simulation von Zufallsvariablen

Bezeichnungen in SAS:

- *Verteilungsfunktion:* $F(x) = P(X \leq x)$: CDF(...)
- *(Zähl-)Dichte*: PDF(...)
- *Simulation*: RAN...

Die Verwendung wird bei der Poissonverteilung genauer erläutert, bei anderen Verteilungen erfolgt sie analog.

1.5.1 Diskrete Verteilungen

- Poissonverteilung

$$P(X = k) = \frac{\lambda^k}{k!} \cdot e^{-\lambda} \qquad \forall k \geq 0$$

in SAS: $P(X \leq k)$ ist implementiert unter CDF('POISSON', k, λ), die Berechnung ist wie folgt möglich:

```
DATA p(KEEP=w);
    lambda=2;
    k=3;
    w = CDF('POISSON', lambda, k);
RUN;

PROC PRINT;
RUN;
```

Analog: $P(X = x)$ ist PDF('POISSON', x, λ)

Simulation einer Zufallsvariablen:

```
DATA sim(KEEP=x);
    seed=0;
    lambda=2;
    x = RANPOI(seed, lambda);
RUN;
```

Simulation von 10 Realisierungen:

```
DATA sim(KEEP=x);
    seed=0;
    lambda=2;
    DO i=1 TO 10;
        x = RANPOI(seed, lambda);
        OUTPUT;
    END;
RUN;
```

- Binomialverteilung

$$P(X = k) = \binom{n}{k} p^k (1-p)^{n-k} \qquad \forall k = 0, \ldots, n$$

$P(X = k)$ = PDF('BINOM', k, p, n)
Simulation: RANBIN(seed, k, p)

1.5.2 Stetige Verteilungen

- Gleichverteilung in [a, b]
$f(x) = \frac{1}{b-a} \cdot 1_{[a,b]}(x), \qquad a < b$
Simulation: RANUNI(seed)
- Gamma-Verteilung
$f(x) = \frac{\lambda^\alpha}{\Gamma(\alpha)} \cdot x^{\alpha-1} \cdot e^{-\lambda x} \cdot 1_{[0,\infty)}(x), \qquad \alpha, \lambda > 0$
$f(x)$ = PDF('GAMMA', x, α, $1/\lambda$)
(Simulation: RANGAM)
- Normalverteilung
$X \sim \mathrm{N}(\mu, \sigma^2), \mu \in \mathbb{R}, \sigma^2 \in \mathbb{R}_+$
$f(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\{-\frac{(x-\mu)^2}{2\sigma^2}\}$
Dichte: PDF('NORMAL', x)
Simulation: RANNOR(seed)
Quantile: PROBIT(alpha)
- Chi-Quadrat-Verteilung
$X_1, \ldots, X_n$ unabhängig, N(0,1)-verteilt
$\Rightarrow U_n = \sum_{i=1}^n X_i^2$ Chi-Quadrat-verteilt
f(x) = PDF('CHISQ', x, n)
$\chi^2_{\alpha,n}$=CINV(α, n)
keine direkte Simulation in SAS
- t-Verteilung
$X \sim \mathrm{N}(0,1), U_n \sim \chi^2_n$ unabhängig
$\Rightarrow T_n = \frac{X}{\sqrt{\frac{U_n}{n}}} \sim t_n$
f(x) = PDF('T', x, n)
$t_{\alpha,n}$ = TINV(α, n)
keine direkte Simulation in SAS
- F-Verteilung
$U_m \sim \chi^2_m, U_n \sim \chi^2_n$ unabhängig
$\Rightarrow W_{m,n} = \frac{U_m/m}{U_n/n}$
f(x) = PDF('F', x, m, n)
$F_{\alpha,m,n}$ = FINV(α, m, n)
keine direkte Simulation in SAS

1.6 Der PROC-Step

1.6.1 Grundlagen

SAS unterscheidet beim PROC-Step zwischen zwei Arten von Optionen, im Handbuch genannt „Options“ und „Statements“, hier werden auch immer diese Begriffe verwendet.

Bsp.:

```
PROC <Name der Prozedur> Option1 Option2 ...;
    Statement1;
    Statement2;
    ...
RUN;
```

Statements sind im Prinzip das gleiche wie eine Option, nur dass diese meist so lang sind (bzw. sein können), dass sie nicht mehr sinnvoll in eine Zeile passen würden. Von der Syntax her können Optionen und Statements allerdings nicht ausgetauscht werden!

Als weitere Möglichkeit, das Verhalten einer Prozedur zu beeinflussen gibt es noch das Setzen von globalen Variablen, wie folgende Ausgabeoptionen:

1. `TITLE 'mein Titel';`
2. `FOOTNOTE 'meine Fußnote';`
3. auch: `TITLE1`, `TITLE2`, ..., analog bei `FOOTNOTE`

Eine häufige Option ist `DATA`. Das dient der Auswahl des zu bearbeitenden Datensatzes. Das Einlesen der Daten mit Selektion nach Wert in einer Spalte ist auch möglich:

```
PROC ... DATA = daten(WHERE = (x > 5));
```

Ein häufiges Statement ist `VAR`. Das dient der Auswahl der zu bearbeitenden Spalten.

```
PROC ...;
    VAR Spalte1 Spalte7;
```

Ein weiteres häufiges Statement ist `CLASS`. Das dient der Auswahl der Spalte, nach der die Einteilung in Klassen / Kategorien stattfindet. Bei einem Datensatz mit Preisen von Eiern in einem Supermarkt könnten z.B. in einer Spalte „Gueteklasse“ Werte A und B stehen. Dann könnte man die Einteilung in Kategorien wie folgt machen:

```
PROC ...;
    CLASS Gueteklasse;
```

Das kommt z.B. bei Tests vor, in denen man zwei Stichproben vergleicht, die anhand der ausgewählten Spalte unterteilt werden.

Als letztes häufiges Statement sei hier `BY` genannt. Es dient zur Sortierung oder Gruppierung / Aufteilung der Stichprobe.

```
PROC ...;
    BY Merkmal;
```

Im Gegensatz zu `CLASS`, bei dem die Einteilung z.B. in 2 zu vergleichende Gruppen geschieht, ist hier die Einteilung in 2 Gruppen, die getrennt analysiert werden sollten.

Hat man z.B. Äpfel und Birnen würde die Einteilung mit `CLASS` dafür sorgen, dass Äpfel mit Birnen verglichen werden, während die Einteilung mit `BY` dafür sorgt, dass die Äpfel und die Birnen jeweils untereinander verglichen werden, z.B. Apfelsorte 1 mit Apfelsorte 2 und getrennt davon Birnensorte 1 mit Birnensorte 2.

1.6.2 PRINT

Optionen:

1. `DATA`
2. `NOOBS` - lässt Beobachtungsnummer weg
3. `LABEL` - Labels verwenden (zusammen mit Statement)

Statements:

1. `BY` - Gruppieren nach einer Spalte
2. `LABEL` - Labels definieren - aktivieren mit der Option

Bsp.:

```
DATA test;
    DO i=1 TO 30;
        x = RANNOR(1);
        y = RANPOI(1, 1);
        z = x + 2 * y;
        OUTPUT;
    END;
RUN;

TITLE 'mein Titel';
FOOTNOTE 'meine Fussnote';

PROC SORT DATA = test;
    BY y;
RUN;

PROC PRINT DATA = test NOOBS LABEL;
    BY y;
    VAR x z;
    LABEL z='Funktion';
RUN;
```

1.6.3 SORT

Zum Sortieren eines Datensatzes

Hier ein Beispiel: Sortiert Datensatz `test` nach der Spalte `y`.

```
PROC SORT DATA = test;
    BY y;
RUN;
```

1.7 Graphische Ausgaben

Empfehlung: Am Anfang Optionen zurücksetzen:

```
GOPTIONS RESET=GLOBAL;
```

1.7.1 GPLOT

Erzeugt Streu-Diagramme.

Globale Option:

1. `SYMBOL INTERPOL=JOIN` für durchgezogene Linien, also eine Kurve

Bsp.:

```
GOPTIONS RESET=GLOBAL;

DATA test;
    DO i=1 TO 30;
        x = RANNOR(1);
        y = RANPOI(1, 1);
        z = x + 2 * y;
        IF (z > 1) THEN
            mark = 1;
        ELSE
            mark = 2;
        OUTPUT;
    END;
    KEEP x y mark;
RUN;

SYMBOL1 COLOR = RED VALUE = STAR;
SYMBOL2 COLOR = BLUE VALUE = PLUS;

PROC GPLOT DATA = test;
    PLOT y*x=mark / HMINOR=0 VMINOR=0;
RUN; QUIT;
```

Achtung! Das `QUIT;` am Ende bewirkt das wirkliche Abschließen des Plots.

1.7.2 GCHART

Histogramme, Block-, Balken-, Kuchen- und Sterngraphiken

1.7.3 G3D

3D-Plots