Grundlegende Einführung in SAS

Institut für Stochastik, Universität Ulm

21. Mai 2015

Starten von SAS

Das Programmpaket SAS befindet sich installiert auf einem Rechner des Rechenzentrums (KIZ) der Universität. Somit ist es für alle Kursteilnehmer notwendig, neben einem gültigen Account auf den Rechnern der Fakultät 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.

Auf den Computerpools des KIZ ist SAS installiert. Will man von einem anderen Rechner aus SAS starten kann man z.B. eine Verbindung zum Rechner „zeus“ herstellen.

Von Linux-Rechnern aus:

1. Verbindung zu RZ-Rechner herstellen:
   `turing$ ssh -X [username]@zeus.rz.uni-ulm.de`
   wobei der username der beim kiz (Uni-Mail) ist.

2. Eingabe des Linux-Passwortes (Mail-Passwort)

3. `zeus$ export DISPLAY`

4. Bereitstellen des SAS-Programmpaketes:
   `zeus$ module load math/sas`

5. Aufruf von SAS:
   `zeus$ sas`

Unter Windows müssen Putty *und* XMing installiert sein. Dann muss im Putty zunächst unter Connection/SSH/X11 die Option „Enable X11 forwarding“ gesetzt werden. Dann wird unter Session als Host Name zeus.rz.uni-ulm.de angegeben. Jetzt wird auf dem Button „Open“ geklickt.

In die Konsole die sich nun öffnet, müsst ihr zunächst Euren KIZ-Benutzernamen eingeben, danach geht es wie unter Linux weiter.

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.

Nach dem Start 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.

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 + H	Markierung aufheben
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 Extras->Optionen->Funktionstasten

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.

Hilfe zu SAS

Zur auf den KIZ-Rechner installierten SAS-Version 9.3 gibt es eine Seite, auf der die gesamte Dokumentation verlinkt ist:
http://support.sas.com/documentation/93/index.html

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 */`.

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 nicht immer 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.

Erzeugen von Daten

Manuelles Einlesen der Daten: Beispiel:

```
DATA vorlesung1;
INPUT Name$ Benotung Punkte1 Punkte2 Gesamtpunkte M;
CARDS;
Gunter 5 4 6 10 5
Erika 5 10 3 13 6.5
;
RUN;
```

Syntax:

- `DATA`-Zeile: Name der (temporäreren) SAS-Datei
- `INPUT`-Zeile: Spaltennamen, für String-Spalten mit `$` am Schluss;
- `CARDS;` zeigt den Beginn der Daten;
- `RUN;` dient (fast) nur zur Übersicht.

Achtung: Das Semikolon nach dem letzten Dateneintrag muss in einer neuen Zeile stehen.

Während wir uns mit dem PROC-Step erst nächstes Mal befassen, besprechen wir hier zwei Anweisungen, die benötigt werden, um die Korrektheit unserer Eingaben zu überprüfen.

```
PROC PRINT;
RUN;
```

gibt die zuletzt erzeugte Datei aus.

```
PROC PRINT data=vorlesung1;
RUN;
```

gibt die Datei *vorlesung1* aus.

Einlesen von Dateien:

Wir betrachten die Datei *lehre.dat* mit folgendem Inhalt:
Die Datei hat 4 Spalten, die erste ist ein String, dann noch 3 Zahlen. Spaltentrenner sind Tabs. Dabei soll die erste Zeile ignoriert werden, sie enthält die Spaltennamen:

```
Name Note Uebungspunkte1 Uebungspunkte2
Martin 1 16 23
Max 1 18 25
Bene 1 20 20
Susan 1 15 18
Charles 1 20 30
...
```

Hier der SAS-Code zum Einlesen:

```
DATA vorlesung_d;
INFILE "Verzeichnis/lehre.dat" FIRSTOBS=2 DLM="09"x;
INPUT Name$ Benotung Punkte1 Punkte2;
RUN;
```

Syntax der `INFILE`-Zeile:

- Dateiname
- `DLM`: Datentrenner, per Default Leerzeichen, Tab kriegt man mit „09“x
- `FIRSTOBS`: erste relevante Zeile
- `OBS`: letzte relevante Zeile (wird selten gebraucht)

Speichern von SAS-Daten

Nach Eingabe von

```
LIBNAME lib "~/SAS";
```

werden alle Dateien, deren Name mit `lib.` beginnt, in `~/SAS` gespeichert.

Bsp.:

```
DATA lib.choc;
    ...
RUN;
```

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

Einlesen von SAS-Dateien

```
LIBNAME lib "~/SAS";
DATA choc2;
    SET lib.choc;
RUN;
```

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

Einfache Funktionen

Das Schlüsselwort `SET` ist auch hilfreich, wenn bereits vorhandene Dateien verändert, z.B. um weitere Spalten ergänzt, werden sollen.

```
DATA vorlesung2;
SET vorlesung_d;
Gesamtpunkte = Punkte1+Punkte2;
M = MEAN(Punkte1, Punkte2);
M2 = MAX(Punkte1, Punkte2);
M3 = MIN(Punkte1, Punkte2);
RUN;
```

Es werden also 4 weitere Spalten zur Datei *vorlesung_d* hinzugefügt. Allerdings bleibt die Datei *vorlesung_d* unverändert und die erweiterte Datei wird unter dem neuen Namen *vorlesung2* gespeichert.

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)

Logischen Operationen:

```
Vergleiche (NE EQ LE)
& = AND
| = OR
```

Das Ergebnis ist „1“, wenn die Operation richtig ist, ansonsten „0“.

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'

Arrays

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

Treten innerhalb eines DATA-Steps folgende Anweisungen auf

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

so können diese abgekürzt werden zu:

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

Beispiel zu Arrays

```
DATA Tiere;
INPUT Tier1$ Tier2$ Tier3$;
CARDS;
Hund Katze Maus
Hase Hamster Wellensittich
;
RUN;

DATA Haustier;
SET Tiere;
ARRAY Tierliste (*) $ Tier1-Tier3;
RUN;
```

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):

```
holiday
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 */
```

DROP, KEEP, OUTPUT und co

Soweit bisher besprochen, werden alle Variablen, die bei der Erstellung einer Datei definiert wurden, in der Datei gespeichert. Da dies aber nicht unbedingt erwünscht ist, gibt es die Befehle `DROP` und `KEEP`. Bsp.:

```
DATA Vorlesung3;
SET Vorlesung1;
M2=MAX(Punkte1, Punkte2);
M3=MIN(Punkte1, Punkte2);
KEEP Name M2 M3;
RUN;
```

Äquivalent kann man die Zeile

```
KEEP Name M2 M3;
```

auch durch

```
DROP Benotung Punkte1 Punkte2 Gesamtpunkte M;
```

ersetzen.

Bei mehereren Datensätzen gleichzeitig:

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

Verbinden von Datensätzen (spaltenweise):

```
DATA Vorlesung4;
MERGE Vorlesung1 Vorlesung3;
RUN;
```

Bem.: Es ist wichtig, dass die Datensätze gleich sortiert sind (also z.B. alphabetisch nach Namen). Es kann zwar mit

```
BY Name;
```

der Name einer gemeinsamen Spalte zum Verbinden der Datensätze angegeben werden.

Allerdings führt dies zu einer Fehlermeldung, falls die Spalte `Name` nicht in beiden Datensätzen alphabetisch sortiert ist. Der Sinn dieser Option besteht im spaltenweisen Zusammenführen von Dateien, deren Spalten zwar nicht disjunkt, aber auch nicht identisch sind.

Aneinanderhängen von Datensätzen (zeilenweise):

```
DATA Vorlesung5;
SET Vorlesung4 Vorlesung2;
RUN;
```

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.

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;
```

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;
```

Bem.: Wird in einem Schleifendurchlauf immer der selbe Wert als seed für den Zufallszahlengenerator verwendet, erhält man trotzdem jeweils neue Zufallszahlen.

- 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, n, p)

Stetige Verteilungen

- Gleichverteilung in [a, b]
  $f(x) = \frac{1}{b-a} \cdot 1_{[a,b]}(x), \qquad a < b$
  Simulation: RANUNI(seed)
  Bem.: Keine Berechnung der Verteilungsfunktion, keine Simulation für $(a, b) \neq (0, 1)$. SAS hält seine Anwender offensichtlich für intelligent genug.

- 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
  $X \sim \mathsf{N}(0,1), U_n \sim \chi^2_n$ 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