Einführung in effizientes Programmieren mit PROC IML am Beispiel einer Simulation

Biljana Gigic
Nationales Centrum für Tumorerkrankungen /
Deutsches Krebsforschungszentrum
Heidelberg
Im Neuenheimer Feld 350
69120 Heidelberg
biljana.gigic@nct-heidelberg.de

Andreas Deckert
Institute of Public Health /
Institut für Medizinische
Biometrie und Informatik, Heidelberg
Im Neuenheimer Feld 324
69120 Heidelberg
a.deckert@uni-heidelberg.de

Zusammenfassung

Im diesem Beitrag wird eine Einführung in SAS IML gegeben. Neben der Syntax von PROC IML wird auf die wichtigsten Operatoren eingegangen und die Spezifizierung, Generierung und Modifizierung von Matrizen und Vektoren demonstriert. Zum Schluss wird am Beispiel der Simulation „ToxCrit“ die Effizienz dieser Programmiersprache aufgezeigt.

Schlüsselwörter: SAS/IML, PROC IML, Matrizen, Vektoren, Funktionen, Subroutinen, PROC IML-Module, Operatoren, Simulation, Makrovariable

1 SAS IML Software

IML steht für Interaktiv Matrix Language. SAS/IML ist eine eigenständige Programmiersprache innerhalb von SAS, die matrizenorientiert arbeitet. Eine Interaktion mit SAS-Datensätzen ist natürlich möglich. Datensätze können in Matrizen überführt, mit SAS/IML bearbeitet und anschließend die Ergebnisse wieder als Datensätze ausgegeben werden.

Besonders geeignet ist SAS/IML für Matrizenoperation, für die Programmierung statistischer Verfahren, die nicht in SAS als Prozeduren implementiert sind und für aufwändige Simulationen. Eine ausführliche Online-Dokumentation zur Prozedur IML stellt SAS bereit (siehe [2]).

2 Syntax

Mit `proc iml` erfolgt der Aufruf von SAS/IML, mit `quit` wird die Session beendet.

```
proc iml;
...;
< IML-Statements >
...;
quit;
```

Solange `quit` nicht ausgeführt wird, kann SAS/IML interaktiv benutzt werden: Nach dem Aufruf mit `proc iml` können so Anweisungen zeilenweise eingegeben und ausgeführt werden, das jeweilige Zwischenresultat wird angezeigt.

3 Arbeiten mit Matrizen

3.1 Spezifizierung einer Matrix

Im Folgenden wird zunächst auf die Erstellung elementarer Objekte in SAS/IML eingegangen, den Skalar, Zeilen- und Spaltenvektoren und zweidimensionale Matrizen.

Skalar der Dimension 1 x 1:

```
reset print;
a = 10;
```

Im Outputfenster erscheint:

```
a      1 row  1 col  (numeric)

              10
```

Mit der Ausgabe des Ergebnisses mit `reset print` erhält man Informationen über den Skalarnamen (**a**), die Anzahl der Zeilen (**1 row**), die Anzahl der Spalten (**1 col**) und den Typ des Elementes (**numeric**).

Zeilenvektor der Dimension 4 x 1:

```
b = {1 2 3 4};
print b;
```

Mit `print b` erhält man (im Gegensatz zu `reset print`) folgende Outputdarstellung im Output-Fenster:

```
b

1      2      3      4
```

Spaltenvektor der Dimension 1 x 4:

```
reset print;
c = {1,2,3,4};
```

Output:

```
 c        4 rows   1 col    (numeric)

                   1
                   2
                   3
                   4
```

Zweidimensionale Matrix der Dimension 2 x 4:

```
reset log print;
d = {1 2 3 4,5 6 7 8};
```

Mit `reset log print` wird der Output in das Log-Fenster geschoben. Folgende Ausgabe erscheint:

```
reset log print;
d = {1 2 3 4,5 6 7 8};

d        2 rows   4 cols   (numeric)

1        2        3        4
5        6        7        8
/*Zweidimensionale Matrix der Dimension 2 x 4*/
```

3.2 Generierung einer Matrix

SAS/IML stellt zur Generierung von Matrizen eine große Zahl an Funktionen zur Verfügung. Im Folgenden werden einige davon erläutert.

Generierung einer Matrix:

Zur Generierung einer Matrix aus identischen Elementen wird die J-Funktion angewandt.

Eingabeparameter: Zeilenanzahl (`2`), Spaltenanzahl (`3`), Element (`0`).

```
e = j(2,3,0);
```

Im Output erscheint:

```
e

0        0        0
0        0        0
```

Vervielfachen einer Matrix:
Eine Matrix kann mit der REPEAT-Funktion vervielfacht werden.
Eingabeparameter: Matrix ({**1 2 3,4 5 6**}), zeilenweise Wiederholung (**2**), spaltenweise Wiederholung (**1**).

```
f = repeat({1 2 3,4 5 6},2,1);
```

Im Output erscheint:

```
f

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

Generierung eines Indexvektors:
Ein Indexvektor wird mit der DO-Funktion generiert.
Eingabeparameter: Startwert (**-8**), Endwert (**2**), Inkrement (**2**).

```
g = do(-8,2,2);
```

Im Output erscheint:

```
g

-8      -6      -4      -2      0       2
```

3.3 Modifizierung einer Matrix

Mit sogenannten Subroutinen können innerhalb von PROC IML Matrizen bearbeitet bzw. modifiziert werden.
Anhand einiger Beispiele soll die Modifizierung der Matrix d gezeigt werden.

```
d

1       2       3       4
5       6       7       8
```

Beispiel 1: Die ersten drei Elemente der zweiten Zeile sollen um 3 erhöht werden.

```
do i = 1 to 3;                          i = do(1,3,1);
    d[2,i] = d[2,i]+3;      ODER        d[2,i] = d[2,i]+3;
end;
```

Um die nacheinander folgenden Elemente anzusprechen und zu modifizieren, kann zum einen eine DO-Schleife verwendet werden, zum anderen kann stattdessen ein ganzer Vektor, in diesem Fall ein Indexvektor von 1 bis 3 an die Stelle i gesetzt werden. Soll eine solche Anweisung ohne Abbruchkriterien durchgeführt werden, empfiehlt es sich, einen Indexvektor zu verwenden. Besteht die Möglichkeit, dass die Ausführung an einer bestimmten Stelle abgebrochen werden muss, ist die DO-Schleife anzuwenden. Im Fall des Indexvektors ist ein Abbruchkriterium nicht möglich.

Output:

```
d

1       2       3       4
8       9       10      8
```

Beispiel 2: Alle Elemente der zweiten Zeile um 3 erhöhen.

Um eine komplette Zeile bzw. Spalte anzusprechen wird die Element-by-Element-Funktion verwendet:

```
d[2,] = d[2,]+3;
```

Hierbei wird die Matrix d an der Stelle „2. Zeile“ um 3 erhöht.

4 Operatoren in SAS/IML

Für Matrizenoperationen bietet SAS/IML eine umfangreiche Operatorenbibliothek. Die hier Aufgeführten stellen lediglich einen Auszug der wichtigsten Operatoren dar.

- Transponieren einer Matrix '
- Matrixmultiplikation *
- Potenzierung quadratischer Matrizen **
- Elementweise Multiplikation #
- Elementweise Potenzierung ##
- Elementweise Division /
- Horizontale Konkatenation ||
- Vertikale Konkatenation //
- Arithmetisches Mittel :
- Inversion inv

5 Syntax eines Moduls innerhalb PROC IML

In SAS/IML können Unterprogramme, die bestimmte Anweisungen unter bestimmten Voraussetzungen ausführen, implementiert werden. Man spricht von Modulen. Wird dabei ein Rückgabewert erzeugt, spricht man von Funktionen.

```
start meinModul(argument1, argument2...);
     ...;
       < IML Statements >
     ...;
     return();
finish mein Modul;
```

Mit `start` beginnt das Modul, mit `finish` wird es beendet. `Argument1, argument 2…` sind die Eingabeparameter. Mit `return` wird der Ausgabewert zurückgegeben.

6 Simulation ToxCrit

Im Folgenden soll anhand einer Simulation ein ausführliches IML-Programm vorgestellt werden. In frühen Phasen klinischer Studien kommt es häufig zu toxischen Reaktionen der Patienten auf die Prüfsubstanz. Aufgrund inakzeptabel hoher Toxizitätsraten kann eine Studie abgebrochen werden. In diesem Zusammenhang wurde die Simulation ToxCrit implementiert. Die Simulation berechnet zum einen für jede Patientenzahl die Anzahl der kritischen Toxizitätsfälle, bei denen die Studie abgebrochen werden muss, zum andern die Wahrscheinlichkeit, dass es zum Abbruch kommt ([1]).

6.1 Ein- und Ausgabeparameter

Eingabeparameter für die Simulation:

- Npat → Anzahl Studienteilnehmer
- ptoxcrit → kritische Toxizitätsrate
- pabbruch → Wahrscheinlichkeitsgrenze für inakzeptable Toxizität
- ptox → angenommene Wahrscheinlichkeit für Toxizität bei einem Patienten
- nsim → Anzahl der Simulationsläufe
- seed → Zufallsanker für die Reproduzierbarkeit der Simulation

Ausgabe:

- Anzahl Toxizitätsfälle, bei denen die Studie abgebrochen wird
- Abbruchrisiko der Studie

6.2 Struktur des Programms

ToxCrit ist innerhalb eines Makros eingebettet. Zur Implementierung wurde SAS/IML angewandt.

```
%macro toxcrit(...);
     proc iml;
          start Pat_Array(...);
               ...;
               return(crit);
          finish Pat_Array;
     start sim(...);
               ...;
               return(stop);
     finish sim;
     < Aufruf der Funktionen Pat_Array und sim>
     < Berechnung der Wahrscheinlichkeiten zum Studienabbruch >
     < Ausgaben >
     quit;
%mend toxcrit;
```

Das Programm führt zwei Funktionen aus, `Pat_Array` und `sim`. Innerhalb `Pat_Array` wird die kritische Anzahl der Toxizitäten für einen Studienabbruch berechnet. `Sim`

schätzt die Anzahl der abgebrochenen Studien. Beide Funktionen erzeugen einen Ausgabewert.

Im zweiten Teil des Programms werden die Funktionen aufgerufen und die Return-Werte (ganze Vektoren) übergeben. Anschließend werden die Wahrscheinlichkeiten zum Studienabbruch berechnet. Zum Schluss werden die Ergebnisse zum einen als Vektoren ausgegeben, zum anderen werden diese zu einer Matrix zusammengeführt und als Data-Set an SAS übergeben.

6.3 Effizienz von SAS/IML

Natürlich ist die Implementierung der Simulation mit SAS Datasteps ebenfalls umsetzbar, jedoch bietet SAS/IML in einigen kritischen Fällen eine schnellere und komfortable Lösung an.

Hier einige Vorteile von SAS/IML:

- Wie im Makro kann auch in SAS/IML ein `goto` in verschachtelten Schleifen oder IF-Anweisungen verwendet werden und die Anweisung, unter bestimmten Abbruchkriterien, verlassen werden. Im SAS Datastep ist dies nicht möglich.

```
start Pat_Array(npat,pabbruch,ptoxcrit); /*Funktionsaufruf*/
  crit = j(npat,1,npat+1); /*Initialisierung des Vektors crit*/
  do u = 1 to npat;
         do l = 0 to u;
         wk = betainv((1-pabbruch),1+l,1+u-l);
              if (wk>ptoxcrit) then do;
              crit[u,1] = l;
                goto exit;
            end;
    end;
     exit:
  end;
```

- Durch die Unterprogramme (Funktionen) können Berechnungen bzw. Modifikationen durchgeführt werden und die Rückgabewerte jederzeit übergeben und verwendet werden. Die Ergebnisse können neben einzelnen Ergebnissen auch Vektoren und Matrizen sein.

```
     return (crit);
finish Pat_Array; /*Funktionsende*/
```

- Der Zugriff auf Vor- und Nachfolgewerte in einem Vektor erfolgt durch sogenannte Subroutinen. Hier kann auf jedes beliebige Element mit wenig Aufwand zugegriffen werden. Im SAS Datastep ist dieser Schritt mit der LAG-Funktion durchführbar, aber nicht innerhalb von DO-Schleifen. Hierfür müssten Arrays verwendet werden, was jedoch dann nicht trivial ist. In SAS-IML können die Vektorwerte dagegen einfach manipuliert werden:

```
start sim(npat,nsim,ptox,crit,seed);
...
     else total[z,1] = total[z-1,1]+tox; /*Zugriff auf Vorwert*/
...
finish sim;
```

- Um auf die Rückgabewerte der Funktionen Pat_Array und sim zuzugreifen, werden die Funktionen aufgerufen und die Return-Werte einem neuen Vektor übergeben:

```
...
_crit = Pat_Array(&npat,&pabbruch,&ptoxcrit);
_stop = sim(&npat,&nsim,&crit,&seed);
...
```

- Die Ergebnisse können sowohl in Vektor- bzw. Matrixform ausgegeben werden, als auch in Datasets überführt werden.

```
print _Pat _crit _wkm _wkmkum; /*Ausgabe der vier Vektoren*/
```

Horizontale Konkatenation, die vier Vektoren werden zu Matrix `_toxcrit` zusammengeführt:

```
_toxcrit = _Pat || _crit || _wkm || _wkmkum;
varnames = {PatientNo ToxPatients InterruptionProb
InterruptionCum}; /*Variablennamen*/
```

Matrix `_toxcrit` wird an Dataset `toxcrit` übergeben und die Variablennamen vergeben:

```
create toxcrit from _toxcrit [colname=varnames];
append from _toxcrit;
close toxcrit;
```

_Pat	_crit	_wkm	_wkmkum
1	1	7.9895	7.9895
2	1	7.3188	15.3083
3	2	0	15.3083
4	2	0.5484	15.8567
5	2	0.9804	16.8371
6	2	1.3628	18.1999
7	2	1.6718	19.8717
8	3	0	19.8717
9	3	0.1556	20.0273
10	3	0.3155	20.3428
11	3	0.4619	20.8047
12	3	0.6107	21.4154
13	3	0.788	22.2034
14	4	0	22.2034
15	4	0.0727	22.2761
16	4	0.1552	22.4313
17	4	0.2388	22.6701
18	4	0.3146	22.9847
19	4	0.4055	23.3902
20	5	0	23.3902

Abbildung 1: Ausgabe der vier Vektoren im Output-Fenster

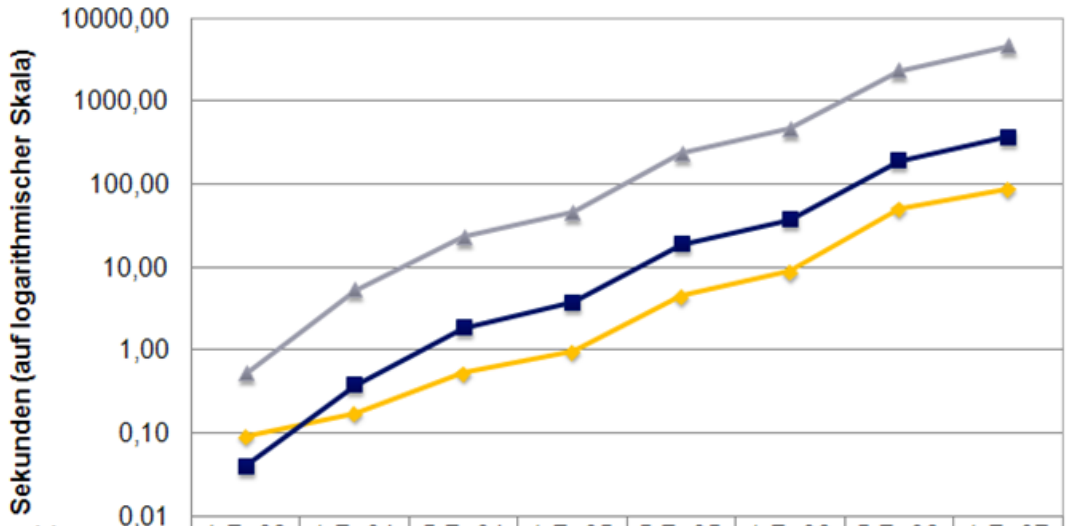
	PATIENTNO	TOXPATIENTS	INTERRUPTIONPROB	INTERRUPTIONCUM
1	1	1	7.9895	7.9895
2	2	1	7.3188	15.3083
3	3	2	0	15.3083
4	4	2	0.5484	15.8567
5	5	2	0.9804	16.8371
6	6	2	1.3628	18.1999
7	7	2	1.6718	19.8717
8	8	3	0	19.8717
9	9	3	0.1556	20.0273
10	10	3	0.3155	20.3428
11	11	3	0.4619	20.8047
12	12	3	0.6107	21.4154
13	13	3	0.788	22.2034
14	14	4	0	22.2034
15	15	4	0.0727	22.2761
16	16	4	0.1552	22.4313
17	17	4	0.2388	22.6701
18	18	4	0.3146	22.9847
19	19	4	0.4055	23.3902
20	20	5	0	23.3902

Abbildung 2: Ausgabe des Ergebnisses als Dataset

- Generell müssen innerhalb eines Makros die Steuerbefehle und Parameter eines SAS/IML-Moduls nicht mit einem %-Zeichen versehen werden.
- Innerhalb eines Makros werden keine %sysevalf- oder %eval-Funktionen für die SAS/IML-Operationen benötigt.

6.4 Simulationszeiten im Vergleich zu R und Java

ToxCrit wurde ursprünglich in R implementiert. Das SAS-Programm wurde zu Validierungszwecken erstellt, mit Java konnte eine benutzerfreundliche Oberfläche zur Verfügung gestellt werden. Abbildung 3 stellt die Simulationszeiten grafisch dar:



Anzahl Sim. →	1,E+03	1,E+04	5,E+04	1,E+05	5,E+05	1,E+06	5,E+06	1,E+07
Java - Mittelwert	0,09	0,18	0,54	0,98	4,51	8,86	51,52	88,92
SAS -Mittelwert	0,04	0,38	1,86	3,80	18,76	37,38	191,77	375,34
R - Mittelwert	0,53	5,29	23,41	45,76	233,41	465,81	2354,67	4626,00

Abbildung 3: Simulationszeiten von ToxCrit ([3])

Betrachtet man die Simulationszeiten für 1.000.000 Simulationsläufe, so liegt das Java-Programm mit ca. 9 Sekunden ganz vorne. SAS/IML benötigt für die Berechnung ungefähr das 5-fache von Java und R über das 10-fache von SAS/IML. Sicherlich muss man hier beachten, dass die verschachtelten DO-Schleifen das R-Programm deutlich verlangsamen, allerdings sind diese für dieses Simulationsprogramm unverzichtbar.

Literatur

[1] H. Aamot, B. Gigic, U. Abel, I. Schenkel: Continuous monitoring of toxicity in clinical trials – simulating the risk of stopping prematurely. Int J Clin Pharmacol Ther, 48(7):476-7, 2010.

[2] http://support.sas.com/documentation/cdl/en/imlug/59656/HTML/default/imlstart.htm, SAS/IML® 9.2 User's Guide, zuletzt besucht am 15.02.2011.

[3] H. Aamot, B. Gigic, U. Abel, I. Schenkel: ToxCrit – Ein Simulationsprogramm zum Studienabbruch aufgrund des kontinuierlichen Monitorings der Toxizität; 54. GMDS-Jahrestagung, Essen 2009.