Zugriff auf eine externe Dynamic Link Library zur Berechnung von Personenjahren und Standardisierten Mortalitäts Ratios mit SAS

Dirk Taeger
Institut für Epidemiologie und Sozialmedizin
Universität Münster
Domagkstr. 3
48129 Münster

Zusammenfassung

Die Standardisierte Mortalitäts Ratio (SMR) ist ein in der Epidemiologie häufig verwendeter Effektschätzer in Kohortenstudien. Besonders die hierfür benötigte Berechnung von Personenjahren kann sich aufwändig gestalten. SAS bietet hierbei keine direkte Hilfestellung an. Seit einiger Zeit existiert das kostenlose Windows Programm PAMCOMP (Person-Years and Mortality Computation Program) zur Berechnung von Personenjahren, SMRs und deren exakten Konfidenzintervallen. Es wird ein SAS-Macro vorgestellt, das die Dynamic Link Library (DLL) von PAMCOMP verwendet, um dessen Ergebnisse in SAS verfügbar zu machen. Dabei wird besonders auf die „SASCBTBL Attribute Table“ und die „MODULExy Routinen“ zum Zugriff auf externe DLLs näher eingegangen. Das vorgestellte Macro zeigt, dass es sich lohnen kann, DLLs von externen Anbietern in SAS zu integrieren, um das Spektrum von SAS zu erweitern.

Keywords: Dynamische Bibliotheken, DLL, Personenjahre, Standardisierte Mortalitäts Ratio.

1 Einleitung

Dynamische Bibliotheken oder auch Dynamic Link Libraries (DLLs) genannt, sind wichtige Elemente des Betriebssystems Microsoft Windows. In ihnen sind die Funktionen des Betriebssystems gekapselt. So wird z. B. der „Datei-Öffnen“–Dialog von einer Windows-DLL bereitgestellt. Dies ermöglicht den Programmierer auf bereits vorgefertigte Routinen zuzugreifen und spart somit einen erheblichen Entwicklungsaufwand.

Darüber hinaus ist es zudem möglich eigene Bibliotheken zu programmieren. Dort werden dann häufig benötigte Funktionen ausgelagert, auf die dann das Programm dynamisch zugreifen kann. Dynamisch bedeutet hier, dass diese Funktionen zur Laufzeit eines Programms geladen und auch wieder entladen werden können. So besteht auch SAS aus mehreren hundert dynamischen Bibliotheken.

Üblicherweise werden diese DLLs in C oder C++ programmiert, aber auch andere Programmiersprachen wie Delphi bieten diese Möglichkeit. Sind die Funktionen einer DLL bekannt und die Schnittstellen mit denen auf diese Funktionen zugegriffen wird, so können diese in SAS eingebunden werden.

Seit einiger Zeit existiert das kostenlose Windows Programm PAMCOMP [1] (Person-Years and Mortality Computation Program) zur Berechnung von Personenjahren, SMRs und deren exakten Konfidenzintervallen. Die benötigten Algorithmen sind dabei in der dynamischen Bibliothek pamcomp.dll ausgelagert worden, die in Microsoft Visual C++ geschrieben wurde. Anhand dieser DLL soll gezeigt werden, wie auf dynamische Bibliotheken von externen Anbietern in SAS zugegriffen werden kann.

2 Die dynamischen Bibliotheken und ihre Schnittstellen

Bevor SAS auf dynamische Bibliotheken zugreifen kann, müssen deren Schnittstellen bekannt sein. Dies ist bei kommerzieller Software normalerweise nicht der Fall. Da das Programm PAMCOMP jedoch kostenlos ist, und der Quellcode zur Verfügung steht, bereitet dies keinerlei Probleme. Es existiert zum Beispiel in der pamcomp.dll eine Funktion zur Berechung des Multiplikators für das untere Konfidenzintervall einer SMR, die folgendermaßen in C++ kodiert ist:

```
//Computes the Multiplier for the Wald CI_LOW
EXPORT double CALLBACK MultiplierWaldLow(double deaths, double alpha)
{
  return 1/(exp(alpha/sqrt(deaths)));
}
```

Diese Funktion heißt also MultiplierWaldLow und hat als Rückgabetyp eine Gleitkommazahl mit doppelter Genauigkeit, darüber hinaus erwartet sie zwei Argumente mit Namen deaths und alpha, die ebenso Double-Variablen sind. Nun muss bekannt sein, dass mit deaths die Anzahl der Toten gemeint ist und mit alpha das (1-alpha/2)-Quantil der Standardnormalverteilung, zudem ergibt sich das untere Konfidenzintervall erst nach Multiplikation der SMR mit dem Rückgabewert der Funktion. Ohne diese Angaben ist ein Zugriff auf diese Funktion nicht oder nicht sinnvoll möglich. Eine Schwierigkeit ergibt sich zudem in der Verwendung von verschiedenen Programmiersprachen, hier SAS und in der DLL z.B. C/C++. Da Arrays in SAS mit dem Index 1 beginnen in C/C++ allerdings mit dem Index 0, kann es zu unerwünschten Nebeneffekten und damit zu inkorrekten Ergebnissen kommen, falls dies nicht mit berücksichtigt wird.

3 Die SASCBTBL Attribut Tabelle

Will man in SAS auf die externen Routinen oder Funktionen von dynamischen Bibliotheken zugreifen, so muss man SAS einige Informationen über diese bereitstellen, wie im vorherigen Abschnitt erläutert. Dies geschieht am besten mit der SASCBTBL Attribut Tabelle. Für die folgende C++ Routine, die für die Personenjahr-Berechung innerhalb der pamcomp.dll zuständig ist

```
//Computes Person-Years
EXPORT double CALLBACK PYCOMPUTE(double dblEPS, double dblTPS,
double dblDOB, double *py,int pyAnzahl, double *ageclass,
short ageAnzahl, double *yearclass, short yearAnzahl);
```

sieht die SASCBTBL Attribut Tabelle folgendermaßen aus:

```
PUT "ROUTINE PYCOMPUTE";
     PUT " MINARG=9";
     PUT " MAXARG=9";
     PUT " STACKPOP=CALLED";
     PUT " MODULE=PAMCOMP";
     PUT " RETURNS=DOUBLE;";
     PUT " ARG 1 INPUT  NUM BYVALUE FORMAT=RB8.;"   * dblEPS;
     PUT " ARG 2 INPUT  NUM BYVALUE FORMAT=RB8.;"   * dblTPS;
     PUT " ARG 3 INPUT  NUM BYVALUE FORMAT=RB8.;"   * dblDOB;
     PUT " ARG 4 OUTPUT NUM BYADDR  FORMAT=RB8.;"   * *py;
     PUT " ARG 5 INPUT  NUM BYVALUE FORMAT=PIB4.;"  * pyAnzahl;
     PUT " ARG 6 INPUT  NUM BYADDR  FORMAT=RB8.;"   * *ageclass;
     PUT " ARG 7 INPUT  NUM BYVALUE FORMAT=PIB2.;"  * ageAnzahl;
     PUT " ARG 8 INPUT  NUM BYADDR  FORMAT=RB8.;"   * *yearclass;
     PUT " ARG 9 INPUT  NUM BYVALUE FORMAT=PIB2.;"  * yearAnzahl;
```

Hier werden also die Informationen übergeben, die SAS braucht, um auf die Funktion zur Personenjahr-Berechnung zuzugreifen.

Zunächst einmal wird der Routinenname „PYCOMPUTE“ übergeben. MINARG und MAXARG geben die minimale und maximale Anzahl der von der DLL-Routine erwarteten Parameter an. STACKPOP=CALLED legt fest, dass die aufgerufene Routine den Stack aufruft. Dies ist in MS-Visual C++ die übliche Methode. MODULE=PAMCOMP zeigt SAS an, dass sich die Routine PYCOMPUTE in der DLL pamcomp.dll befindet. RETURNS=DOUBLE kennzeichnet den Rückgabewert der Routine als Gleitkommazahl mit doppelter Genauigkeit.

Nach diesem allgemeinen Teil, werden die einzelnen, zu übergebenen Parameter, näher definiert. Die allgemeine Syntax lautet:

```
ARG argnum
     <INPUT|OUTPUT|UPDATE>
     <NUM|CHAR>
     <NOTREQ|REQUIRED>
     <BYADDR|BYVALUE>
     <FDSTART>
     <FORMAT=format>
     ;
```

Alle Parameter der Routine PYCOMPUTE sind numerische Werte, die entweder nur an die Routine übergeben werden, ohne verändert zu werden (INPUT) oder von der Routine verändert werden dürfen (OUTPUT). Die einzelnen Werte sind in diesem Beispiel numerisch und werden entweder mit ihrem Wert (BYVALUE) oder mit ihrer Adresse (BYADDR) übergeben. Letzteres ist der Fall, wenn nicht der Variablenwert, sondern ein Zeiger darauf übergeben wird, wie es bei Vektoren oder Matrizen der Fall ist. Wichtig ist noch die FORMAT Anweisung, die angibt, welches Format die übergebenen Variablen besitzen. Tabelle 1 listet einige Formate für C-DLLs auf.

Tabelle 1: C Formate

C	**SAS Format/Informat**
double	RB8.
float	RB4.
unsigned int	PIB2.
unsigned short	PIB4.

Da die Attribut Tabelle eine sequentielle Textdatei sein muss, die über den SASCBTBL *fileref* aufgerufen werden kann, bietet es sich an, dies folgendermaßen in SAS einzubinden:

```
FILENAME SASCBTBL CATALOG "work.temp.attrfile.source";

DATA _NULL_;
FILE SASCBTBL;
PUT "ROUTINE PYCOMPUTE";
...
RUN;
```

4 Der Aufruf einer DLL-Routine

Nachdem die aufzurufenden Routinen in der Attribut Tabelle definiert wurden, kann auf diese zugegriffen werden. Je nach Rückgabewert geschieht dies im Data Step entweder durch

```
CALL MODULE(module,arg-1,...arg-n)
num=MODULEN(module,arg-1,...arg-n)
char=MODULEC(module,arg-1,...arg-n)
```

und in PROC IML durch

```
CALL MODULEI(module,arg-1,...arg-n)
num=MODULEIN(module,arg-1,...arg-n)
char=MODULEIC(module,arg-1,...arg-n)
```

Wenn als Beispiel die C++ Routine in der DLL so aussieht

```
EXPORT double CALLBACK MultiplierWaldLow(double deaths, double alpha)
```

dann wird sie in PROC IML folgendermaßen aufgerufen:

```
ci_low=smr*MODULEIN('MultiplierWaldLow',dths,1.96);
```

wobei `smr` die zuvor berechnete SMR ist und multipliziert mit dem Ergebnis der Routine das untere 95%-Konfidenzintervall ergibt. dths ist die Anzahl der berechneten Todesfälle und 1.96 das 97,5%-Quantil der Standardnormalverteilung.

5 Das Macro zur Berechnung von Personenjahren und SMRs

Das nun beschriebene Macro zur Berechnung von Personenjahren und SMRs besteht tatsächlich aus zwei Macros. Das erste `%pamcomp_int` legt die Attribut-Tabelle an, und es genügt dieses Macro einmal pro SAS-Sitzung aufzurufen. Das eigentliche Macro wird folgendermaßen aufgerufen

```
%macro pamcomp(cohort,dob,eps,tps,icd,agecls,yearcls,rates,icdvalue);
```

Folgende Parameter werden erwartet:

cohort	SAS-Datei, die die Kohortendaten enthält
dob	Name der Variablen für das Geburtsdatum der Kohortenmitglieder in cohort
eps	Name der Variablen für das Eintrittsdatum in die Kohorte in cohort
tps	Name der Variablen für das Austrittsdatum aus der Kohorte in cohort
icd	Name der Variablen, die angibt, ob Kohortenmitglieder gestorben sind in cohort
agecls	SAS-Datei mit den Altersklassen
yearcls	SAS-Datei mit den Kalendarjahrklassen
rates	SAS-Datei mit den Referenzraten
icdvalue	Der Wert der Variablen icd, die angibt, dass das Kohortenmitglied verstorben ist.

Die Variablen dob, eps und tps müssen als SAS-Datumswerte vorliegen (z.B. `date9.`), die Variable icd als numerische Ganzzahl.

Möchte man die SMR nach den Altersklassen [10-14], [15-19] und [20-24] stratifizieren, so muss die SAS-Datei agecls aus folgenden Beobachtungen bestehen:

10
15
20
25

Möchte man zudem die SMR nach den Kalenderjahren [1.1.75-31.12.79] und [1.1.80-31.12.86] stratifizieren, so muss die SAS-Datei yearcls aus folgenden Beobachtungen bestehen:

1975
1980
1987

Die SAS-Datei rates beinhaltet die entsprechenden Referenzraten für die zu untersuchende Todesursache, d.h. Anzahl der Verstorbenen an dieser Ursache geteilt durch die Anzahl der Personen unter Risiko für jede Kombination aus Altersklasse und Kalenderjahrklasse. Die Datei muss so aufgebaut sein, dass in den Zeilen die Alterklassen stehen und in den Spalten die Kalenderjahrklassen. Letztendlich werden die Macros dann folgendermaßen aufgerufen

```
libname c 'c:\ksfe2002';
%pamcomp_init;
%pamcomp(c.cohort2,dob,eps,tps,icd,c.ageclass,c.yearclass,c.rates,1);
```

Nach dem Aufruf wird im Output-Fenster die Personenjahr-Matrix, die Matrix der Verteilung der Todesfälle, die Matrix der erwarteten Fälle sowie die SMR mit exakten 95% -Konfidenzintervallen ausgegeben.

6 Zusammenfassung und Ausblick

Es wurde beschrieben wie von SAS aus auf externe dynamische Bibliotheken zugegriffen werden kann. Als Beispiel diente die pamcomp.dll zur Berechnung von Personenjahren und SMRs. Es wurde gezeigt, dass das SAS-System eine einfache Schnittstelle zum Zugriff auf externe DLLs zur Verfügung stellt. Damit lässt sich das Spektrum von SAS erweitern. Allerdings ist eine umsichtige Programmierung vonnöten, denn Fehler beim Zugriff auf DLLs können zu Systemabstürzen führen oder Ergebnisse verfälschen. Nicht alle Möglichkeiten die zur Verfügung stehen, konnten hier dargestellt werden. Dazu existiert allerdings ein SAS-Dokument [2], das weitere Einzelheiten beschreibt.

Das vorgestellte Macro befindet sich zur Zeit noch in der weiteren Entwicklung. Sobald eine komfortable Version zur Verfügung steht, wird diese mit Beispieldaten auf der PAMCOMP-Homepage unter

`http://medweb.uni-muenster.de/institute/epi/pamcomp/pamcomp.html`

zur Verfügung gestellt.

Literatur

[1] Taeger D., Sun Y., Keil U., Straif K.: A Standalone Windows Application for Computing Exact Person-Years, Standardised Mortality Ratios and Confidence Intervals in Epidemiological Studies. Epidemiology 2000;11:607-608.

[2] SAS Technical Support Document TS-322.