KM 9(1983)
pag 85 - 95

UNIFORMERING VAN PROGRAMMA-AANSTURING DOOR SAS ?

Toevoegen van eigen programma's.

W. Maessen[1]

J. Mulder[2]

Samenvatting

Er is zo'n groot aantal statistische pakketten en losse programma's, dat er gedacht moet worden over een uniforme manier van aansturing. Een systeem waarin dit mogelijk is, moet aan een aantal eisen voldoen qua toegankelijkheid en uitbreidingsmogelijkheden. Een mogelijke aanpak wordt hier beschreven aan de hand van het programmapakket SAS. Met name de koppeling van eigen programma's aan het pakket en de eisen die daaruit voortvloeien, worden behandeld.

[1]Instituut voor Massacommunicatie,
[2]Mathematisch-Statistische Adviesafdeling,
Medewerkers KU-Nijmegen.

1. Inleiding

Voor statistische verwerking van onderzoeksresultaten is er een skala aan programmatuur voorhanden. Deze programmatuur varieert van normale univariate tot de meest geavanceerde multivariate analyse-methoden.
Reeds lang is geprobeerd, al deze methoden te vangen in een universele aansturing. SPSS is daar het oudste voorbeeld van.
Echter, komplete pakketten hebben het nadeel, dat ze nooit alle specifieke programmatuur, die door onderzoekers vereist wordt, kunnen herbergen. Het ligt dan voor de hand eigen programma's te gaan schrijven of te gaan zoeken in het grote aantal stand-alone programma's, dat er op de wereld te vinden is. De eerste mogelijkheid kost tijd (en geld), de tweede brengt vaak aansturingsproblemen met zich mee. Met name als het gaat om onervaren komputergebruikers geeft dit grote problemen.
Meestal zal dit betekenen, dat de analyses dan toch maar met inadekwate, maar wel bekende, programmatuur worden verricht. Dit zal zeker gebeuren als de betreffende afdeling, waar de onderzoekers werkzaam zijn, qua statistisch-methodologische ondersteuning onderbezet is. Tijd om nieuwe zaken uit te zoeken, is er dan niet.
Alle reden dus, om eens te bekijken of er niet een standaardisatie van programma's mogelijk is, waarbij ook onervaren onderzoekers maar 1 'taal' hoeven te leren om alle voorhanden programma's te kunnen aansturen en benodigde datamanipulaties te kunnen uitvoeren.
Zo'n taal zal aan een aantal voorwaarden moeten voldoen:

1. Inzichtelijk voor onervaren komputergebruikers. Geen gezeur met abstrakte kommando's als bijvoorbeeld allokaties (waarom FT08F001 ?), compilers (welke moet ik nu gebruiken ?), enz.

2. Makkelijk in te groeien. Naarmate men ingewikkelder zaken uit wil voeren, leert men een kleinigheidje bij.

3. Uniforme aansturing van alle op het komputer-systeem aanwezige programmatuur.

4. De te gebruiken bestanden moeten makkelijk toegankelijk zijn, zowel qua aanmaak als qua veranderen.

5. Het toevoegen van nieuwe (eigen) programma's moet door enkele 'deskundigen' in relatief kort tijdsbestek kunnen gebeuren.

6. Beperking van de aan de programma's toe te voeren parameters tot de meest essentiele.

7. De output uit de programma's moet goed leesbaar gemaakt kunnen worden d.m.v. labels en titels.

8. Alle kommando's buiten deze taal moeten bereikbaar zijn.

9. De taal moet zowel interaktief als 'batch' aan te spreken zijn.

10. Hoe meer programma's in de taal kunnen worden opgenomen, hoe beter.

De meest belangrijke programmapakketten zijn momenteel SPSS en BMD(P), die voor de meeste grote komputers beschikbaar zijn, en SAS, (voorlopig ?) alleen beschikbaar op IBM-komputers.
SPSS wordt momenteel het meest gebruikt. Studenten, wetenschappelijk medewerkers en onderzoekers worden het eerst met dit pakket bekend gemaakt. Deze wijde verbreiding levert het voordeel op, dat de output op brede schaal bekend is, aldus de uitwisseling van gegevens vergemakkelijkend.
Beschouwen we SPSS echter op bovenvermelde kriteria, dan valt het pakket tegen. Het aantal programma's is beperkt, menige data- manipulatie is er niet mee mogelijk, de system-files zijn voor de meesten een gesloten boek en toevoegen van eigen programma's is moeilijk, zoniet onmogelijk. Dit niet in het laatst wegens gebrek aan medewerking van de SPSS-producent.
BMD(P) is een pakket, dat bestaat uit een groot aantal losse programma's. Deze programma's zijn, statistisch gezien, erg goed. Echter, elk programma heeft zijn eigen aansturing. Van een taal kan daarom niet gesproken worden. Bij beschouwing van bovengenoemde kriteria scoort dit pakket dan ook zeer laag.
Zoals boven al vermeld, is SAS alleen nog maar op IBM-komputers beschikbaar en dat levert uitwisselingsproblemen op. Op korte termijn kan daar verandering in komen. De eerste geluiden over het herschrijven van de ASSEMBLER-routines voor andere machines hebben ons inmiddels bereikt. Met de aanpassing voor de VAX is men op het moment bezig en enige tijd geleden is er een kontrakt afgesloten met DATA GENERAL, waarvoor de aanpassing tweede helft '83 wordt verwacht.
Wat de bovenstaande kriteria betreft, skoort SAS echter erg hoog. SAS laat gestruktureerd programmeren toe en lijkt daardoor veel op een hogere programmeertaal als ALGOL of PL/1, zonder dat de taal te ingewikkeld wordt. De meest

ingewikkelde zaken en manipulaties zijn er mee uit te voeren en er is een skala van voorgeprogrammeerde funkties.
Dankzij de multiple-file database struktuur van SAS is het koppelen van bestanden (MERGING, SETTING en MATCH-MERGING) een erg eenvoudige zaak. Tussenkomst van FORTRAN o.i.d. (zoals bij SPSS vaak nodig is) is daardoor overbodig geworden.
SAS-datasets (cf. SPSS - systemfiles) kennen, naast de voordelen van dokumentatie van variabelen, ook nog de mogelijkheid van verandering via EDIT- en Full-screen edit-procedures en zijn daardoor minder een black box dan de SPSS-systemfiles.
Ook op de andere kriteria kan men vanuit SAS een positief antwoord verwachten. Het is bijvoorbeeld mogelijk vanuit SAS rechtstreeks BMD(P)-programma's aan te roepen, of SPSS/BMD(P)/OSIRIS - (system)files te lezen of te konverteren. Het aantal programma's in de SAS-bibliotheek (inklusief de plotrogramma's) gaat de 100 te boven.
Met name de mogelijkheid om eigen programma's aan het totaal van SAS-programma's toe te voegen is een sterk punt en daarop willen we in dit artikel dan ook dieper ingaan.

Enerzijds hopen we hiermee IBM/SAS-gebruikers een dienst te bewijzen, c.q. op een idee te brengen, anderzijds kan de programmeervorm, zoals die door SAS gebruikt wordt (m.n. de dynamische geheugenallokatie), gebruikers van andere komputer-systemen een idee geven van de mogelijkheden van een dergelijk systeem.

2. Het principe

Om eigen programma's, die door SAS als procedure worden aangeroepen, te kunnen toevoegen, is het zinvol iets over de struktuur van SAS te weten.
SAS beschikt over een Supervisor, die de gang van zaken tijdens de uitvoering van een programma regelt. Alle procedures worden door de Supervisor herkend, doordat ze beginnen met het keyword PROC en daarna de naam van de procedure. De Supervisor zoekt dan naar twee modules, een "parser" en een procedure-module.
De "parser" wordt verkregen met behulp van een eenvoudige ASSEMBLER-macro en het eigenlijke PROC-module op de normale manier via compiler en linkage-editor. Door het toevoegen van een bibliotheek met "parsers" en PROC-modules, kunnen eigen programma's vanuit SAS aangeroepen worden. In de parser wordt aan SAS doorgegeven, hoe het PROC-statement voor de procedure er uit moet zien: VAR-statement, WITH-statement, een output-dataset of niet, enzovoort.

Ter illustratie van een aanpassing gaan we in dit artikel uit van een FORTRAN-programma dat de toets van Kruskal-Wallis uitvoert.
Het programma werd aan SAS aangepast, omdat de SAS-procedure, die deze toets ook uitvoert, te weinig informatie geeft en, ons inziens, een inkomplete tie-korrektie toepast.
Voor de toetsingsprocedure moeten de volgende gegevens aan het programma doorgegeven worden:

1. De variabelen waarop de toets moet worden uitgevoerd.

2. De indelingsvariabele(n).

3. Een parameter voor het opvragen van extra statistics (bijv. de Wilcoxon toetsen twee aan twee).

De parser gaat er nu als volgt uitzien:

```
        PRINT   NOGEN
PROC    SASPROC NAME=MSAKRUSK,LOADMOD=MSAKRUS2,DEFLIST=1,DEFMODE=NUMERIC
PARMS   SASLIST TOETS,1,MODE=NUMERIC
LISTS   SASLIST VAR,1,MODE=NUMERIC
LISTS   SASLIST CLASSES,2,CLASS,2,MODE=NUMERIC
        SASEND
        END
```

In het PROC-statement wordt aangegeven, hoe de naam is waarmee de procedure aangeroepen moet worden (NAME=), hoe het PROC-module heet (LOADMOD=) en een default-list voor de data (DEFLIST=). DEFLIST=1 zorgt ervoor dat een VAR-statement gegenereerd wordt, als bij de aanroep van de procedure dit statement niet wordt opgegeven. Hierin komen dan alleen de variabelen die niet in een andere list (bijv. CLASSES, WITH) staan.
In de andere statements wordt aan de Supervisor doorgegeven, hoe de bovengenoemde drie punten aan de procedure worden bekendgemaakt.
Bij de compilatie van deze macro worden de statements geexpandeerd tot een assembler-programma, welk SAS de instrukties doorgeeft, die geraadpleegd worden bij de aanroep van het programma.
Het PROC-load-module wordt verkregen door compilatie van het eigenlijke FORTRAN of PL/1-programma. Dit programma moet op een speciale manier worden opgebouwd. Alle READ's worden bijvoorbeeld vervangen door CALL's naar SAS-subroutines. Na

het maken van de "parser" en het PROC-module wordt de nieuwe procedure vanuit een SAS programma als volgt aangeroepen:

```
PROC MSAKRUSK  TOETS = parameter;
     VAR lijst van te toetsen variabelen;
     CLASSES lijst van indelingsvariabele(n);
```

3. De aanpassing van het programma.

SAS maakt gebruik van dynamische geheugen-allokatie. Dat brengt met zich mee, dat de eigenlijke bewerkingen in een subroutine moeten gebeuren. Het hoofdprogramma dient alleen tot het binnenhalen van de parameters en het aan de hand van die parameters vaststellen van de geheugengrootte.
Een aantal CALL's die in een hoofdprogramma moeten voorkomen, zijn de volgende:

SASFT of SASPLF	: Doorgeven of het om een FORTRAN of PL/1-programma gaat.
NOVAR	: Funktie die het aantal variabelen van een input-list (bijv. de VAR-list) oplevert.
VARERR	: Indien het aantal variabelen uit een list 0 is.
SASLOG	: Foutboodschappen in de SASLOG.
SETERR	: Om aan SAS door te geven dat er een fout is gekonstateerd.
GETSTR (F)/GETMEM (PL/1)	: Na berekening van de te gebruiken geheugenruimte oproepen van die ruimte.
MEMERR	: Doorgeven, dat er niet genoeg ruimte verkrijgbaar was.
proc-call	: Roept de eigenlijke subroutine aan en geeft de geheugenruimte door.

In de subroutine zullen de volgende CALL's minimaal moeten voorkomen:

VARX en INPUT	: Haalt de data binnen in een van te voren gedefinieerd array.
NAMES	: Haalt de namen van de variabelen binnen.

STITLE : Verzorgt de titels boven de output en geeft het aantal geschreven regels op een pagina.

OBSERR : Als blijkt dat er geen data doorgegeven zijn.

Naast deze CALL's zijn er nog verschillende verfraaiingen mogelijk. Zo kunnen de labels van de variabelen en de waarden binnengehaald worden, output datasets gekreeerd in de vorm van SAS-datasets en meer van dit moois. De algemene programma-flow ziet er als volgt uit:

```
-------------------------------------
1) Definieer language-environment    |
2) Informeer naar het aantal vari-   |
   abelen in de verschillende lists. |
3) Bepaal of deze = 0                |
4) Bepaal hoeveelheid te gebruiken   |
   geheugen aan de hand van 2) .     |
5) Eventueel eerste pass door data   |
   voor de bepaling van de hoeveel-  |> Hoofdprogramma.
   heid data                         |
6) Eventueel berekening geheugen-    |
   ruimte voor eigenlijke programma. |
7) CALL naar eigenlijke programma    |
   met doorgeven van gegevens en     |
   geheugenruimte uit voorgaande     |
   stappen.                          |
8) EINDE PROGRAMMA.                  |
-------------------------------------
1) Deklareer geheugenruimte          |
2) Zet variabelen op beginwaarde     |
3) Haal een observatie binnen        |
4) Voer eventueel berekening uit     |
5) Doe stap 3) tot EOF               |> Subroutine
6) Kontroleer of aantal observaties  |
   ongelijk aan 0                    |
7) Voer eventueel berekeningen uit   |
8) Print output                      |
9) EINDE SUBROUTINE                  |
-------------------------------------
```

Hierna volgt het hoofdprogramma van de Kruskal-Wallis procedure. Anders dan bij de meeste andere statistische programma's wordt hier de komplete data matrix in het geheugen gezet.
Let op het gebruik van de dynamische geheugen-allokatie. Er wordt een array ND gedeklareerd. Bij de eerste aanroep van de subroutine GETSTR, wordt het aantal beschikbare geheugen plaatsen opgevraagd. Bij de tweede CALL wordt dit aantal N000 werkelijk opgevraagd en geeft de routine het startpunt van het array terug.Voor elk van de benodigde arrays (in dit geval 0-11=12) wordt het aantal geheugen plaatsen bepaald. De array's worden na elkaar over het grote array gelegd. De startpunten worden berekend en staan in N0,N1,....,N11. Deze startpunten worden in de CALL naar de subroutines LDATA en KRWAL doorgegeven. In deze subroutines kan nu met andere namen worden gewerkt, zoals in een eenvoudig programma gebruikelijk is.

```
C
C     PROGRAMMA VOOR HET UITVOEREN VAN DE TOETS VAN
C              KRUSKAL-WALLIS
C     MET WILCOXON TOETSEN  TWEE AAN TWEE
C
C     NVAR  AANTAL VARIABELEN
C     NIND  AANTAL INDELINGSVARIABELEN
C     KTOVAR  NVAR + NIND
C     NTOT  AANTAL WAARNEMINGEN
C
      COMMON /IOUNIT/IP
      REAL*8 ND(1),Z
      INTEGER MIS
      EQUIVALENCE (Z,MIS)
C
C     TOTAAL BENODIGD GEHEUGEN:
C           INLEZEN DATA + INDELINGSVARIABELEN      KTOVAR*NTOT
C           HULPARRAY BIJ HET INLEZEN               KTOVAR
C           RANGNUMMEREN                            NTOT
C           OPSLAAN VAN HET MAXIMALE RANGNUMMER     NVAR
C           VOOR NGIV                               7
C           SUBROUTINE KW                           6*NTOT
C           VOOR DE NAMEN VAN VARIABELEN            NVAR
C
      CALL SASFT
```

```
      CALL SASLOG('        NAME AND ADRESS OF AUTHOR:',33)
      CALL SASLOG('        J. MULDER',16)
      CALL SASLOG('        M.S.A.',13)
      CALL SASLOG('        TOERNOOIVELD',19)
      CALL SASLOG('        NIJMEGEN',15)
      NTOT=0
C
C     DOORGEVEN VAN DE EERSTE PARAMETER: TOETS
C
      NPAR=PARM(1)
      Z=PARM(1)
      IF(Z.EQ.0.DO.AND.MIS.NE.0) NPAR=0
C
C     BEPALEN VAN HET AANTAL VARIABELEN
C
      NVAR=NOVAR(1)
      IF(NVAR.GT.0) GOTO 50
      CALL VARERR
      STOP
C
C     BEPALEN VAN HET AANTAL KLASSE INDELINGEN
C
 50   NIND=NOVAR(2)
      IF(NIND.GT.0)GOTO 10
      CALL SASLOG(' ER IS GEEN INDELINGSVARIABELE GESPECIFICEERD',45)
      STOP
 10   KTOVAR=NVAR+NIND
C
C     BEPALEN VAN GEHEUGENGROOTTE EN HET MAXIMAAL AANTAL IN TE LEZEN
C     GEVALLEN
C
      CALL GETSTR(-1,ND,INDY,N000,4000)
      CALL GETSTR(N000,ND,INDX,NNN,4000)
      IF (NNN.EQ.0) GOTO 51
      CALL MEMERR((N000-NNN)*8)
      STOP
C
C     BEREKEN NOVER: AANTAL GEVALLEN DAT MAXIMAAL VERWERKT KAN WORDEN.
C
 51   NOVER=(N000-7-2*NVAR-NIND)/(7+KTOVAR)
```

```
      N0=INDX
      N1=N0+7
      N2=N1+KTOVAR*NOVER
      N3=N2+NVAR
      N4=N3+NOVER
      N5=N4+NOVER
      N6=N5+NOVER
      N7=N6+NOVER
      N8=N7+NOVER
      N9=N8+NOVER
      N10=N9+NOVER
      N11=N10+NVAR
C
C     HET INITIALISEREN VAN DE PARAMETERS
C
      ND(INDX+2)=3
      ND(INDX+3)=NPAR
      ND(INDX+4)=1
C
C     HET AANROEPEN VAN DE INLEES SUBROUTINE EN HET REKENPROGRAMMA
C
      CALL LDATA(ND(N1),NVAR,NIND,NOVER,NTOT,ND(N10),ND(N11),KTOVAR)
      CALL KRWAL(ND(N1),ND(N0),ND(N2),ND(N3),ND(N4),ND(N5),ND(N6),
     *ND(N7),ND(N8),ND(N9),NOVER,NVAR,KTOVAR,NGI,NIND,NTOT)
      STOP
      END
```

Opm. : Het COMMON-block IOUNIT is verplicht. De variabele IP bevat de ddname van de SAS-listing output-file.
Wat verder nog opvalt is de missing data-handling. Missing data worden in SAS opgeslagen als decimale punt (.). D.m.v. een kombinatie van EQUIVALENCE en IF-statement worden missing values herkend. Het eerste gedeelte van de subroutine die met de echte berekeningen start ziet er als volgt uit:

```
      SUBROUTINE KRWAL(NRO,NGIV,NGR,NROH,NRIND,NROG,NROD,IVEC
     C,IVEC2,NAAM,NDCAS,NVAR,KTOVAR,NGI,NIND,NTOT)
      REAL*8 NRO(NDCAS,KTOVAR),NGIV(7),NGR(NVAR),NROH(NDCAS)
     C,IVEC(NDCAS),NRIND(NDCAS),IVEC2(NDCAS),NROD(NDCAS)
     C,NROG(NDCAS),NAAM(NDCAS),A,Z
      INTEGER MIS
      EQUIVALENCE (Z,MIS)
      COMMON /IOUNIT/IP
      DO 100 I=1,NVAR
      CALL NAMES(1,I,A,ITYPE,LENGTH)
      NAAM(I)=A
100   CONTINUE
```

Vanaf hier worden de arrays normaal behandeld. De input van de data geschiedt bijvoorbeeld met de volgende subroutine:

```
      SUBROUTINE LDATA(NRA,NVAR,NIND,NMAX,NTOT,VAR,VAR2,KTOVAR)
C
C     LEZEN VAN DATA
C
      REAL*8 NRA(NMAX,KTOVAR),VAR(NVAR),VAR2(NIND)
      COMMON /IOUNIT/IP
      NVAR1=NVAR+1
      NEIND=NVAR+NIND
      I=1
   10 CALL INPUT(IEND)
      IF(IEND.EQ.1) GOTO 99
      CALL VARX(1,VAR)
      CALL VARX(2,VAR2)
      DO 20 J=1,NVAR
      NRA(I,J)=VAR(J)
   20 CONTINUE
      DO 30 J=NVAR1,NEIND
      K=J-NVAR
      NRA(I,J)=VAR2(K)
30    CONTINUE
      I=I+1
      IF(I.LE.NMAX) GOTO 10
      WRITE(IP,40) NMAX
40    FORMAT(' GEHEUGEN TE KLEIN, INGELEZEN ',I8,' GEVALLEN')
      STOP
99    NTOT=I-1
      IF(NTOT.GT.0)GOTO 100
      CALL OBSERR
      STOP
100   RETURN
      END
```

4. Procedures voor het maken van de modules.

Voor het maken van de modules zijn een aantal procedures voor de operating systemen MVS en CMS gemaakt. Aangezien de opbouw van zo'n procedure van installatie tot installatie kan verschillen, willen we hier echter volstaan met te verwijzen naar de SAS PROGRAMMER'S GUIDE (1981 edition), waarin hierover suggesties aan de hand worden gedaan.
Voor verdere inlichtingen kan men zich wenden tot de auteurs.