Makrofunktion zur Übernahme von Abfragekriterien aus externen Quellen, z.B. einer Excel Datei, in SAS Code (PROC SQL)

Christian Fauska
Lebkuchen Schmidt GmbH&Co. KG
Zollhausstr. 30
90469 Nürnberg
christian-fauska@lebkuchen-schmidt.com

Zusammenfassung

Häufig werden anhand spezifischer Merkmale Kunden verschiedenen Marketing-Gruppen oder Patienten unterschiedlichen Risikogruppen zugewiesen. Die Anwendungsfälle für die Praxis ließen sich noch erweitern. Die Vorgabe der Kriterien erfolgt meist nicht durch den Datenanalysten bzw. Programmierer. Zusätzlich ist es in der Praxis üblich, dass diese Kriterien sich in mehr oder weniger regelmäßigen Abständen ändern können.
Sind diese nun fest im Programmcode, z.B. innerhalb eines PROC SQL Aufrufes, hinterlegt, so muss bei jeder Anpassung das Programm geändert werden und die Verantwortung zur korrekten Anlage liegt beim Programmierer.
Alle Kriterien lassen sich jedoch auch in einer Liste, z.B. als Excel-Datei, hinterlegen, mittels SAS importieren und über die Verwendung von Makrovariablen und Makrofunktionen innerhalb eines PROC SQL Schrittes abfragen. Die Pflege und Verantwortung dieser Liste liegt beim entsprechenden Fachbereich und Änderungen führen zu keinem neuen Programmieraufwand. Dieser Ablauf wird anhand eines Beispiels und des entsprechenden SAS Codes erläutert.

Schlüsselwörter: Makrofunktionen, Excel-Import, Makrovariablen, PROC SQL

1 Inhalt

Im Folgenden wird anhand eines Praxisbeispiels gezeigt, wie Sie aus externen Quellen Teile eines Programmcodes einlesen, diese Programmteile verschiedenen Makrovariablen zuweisen und anschließend - durch eine Makrofunktion gesteuert - an entsprechender Stelle in Ihrem SAS Programm wieder auflösen.

2 Ausgangslage und Beispiel

In der Praxis kommt es häufig vor, dass Sie anhand einer oder mehrerer Kriterien Ihre Kunden, Patienten, Verkaufsartikel, etc. in unterschiedliche Kategorien einordnen müssen. Angenommen Sie erhalten die Aufgabe Ihrem Kundenstamm anhand zwei verschiedener Merkmale (Umsatz, Mahnung) zwei unterschiedliche Kategorien ('guter

Kunde', 'schlechter Kunde') zuzuordnen, so ist dies noch mit überschaubarem Aufwand verbunden.

1. Kriterium Kunde hatte bereits einmal eine Mahnung erhalten
→ schlechter Kunde
2. Kriterium Umsatz des Kunden liegt über 100,- €
→ guter Kunde
Umsatz des Kunden liegt unter 100,- €
→ schlechter Kunde

Die Reihenfolge der Kriterien soll beibehalten werden. Diese Kriterien können leicht in einem Datastep oder in PROC SQL, was im Verlauf dieses Beispiels verwendet wird, umgesetzt werden.

```
proc sql;
     create table bsp1 as
     select    kdnr,
               (case     when mahnung = 1 then 'schlecht'
                         when umsatz > 100 then 'gut'
                         else 'schlecht'
               end) as kategorie
     from basis1;
quit;
```

Im Normalfall liegen jedoch mehr als zwei Variablen und Kriterien vor, so dass die Zuweisung deutlich umfangreicher ausfällt. Vor allem der Schreibaufwand im Programmcode nimmt zu und die Übersichtlichkeit ab. Im folgenden Beispiel in Tabelle 1 liegt eine Anforderung mit drei Variablen und elf Kriterien vor.

Tabelle 1: Kriterienübersicht

Kunden-kategorie	Kunde seit	Umsatz	Kategorie
Consumer	max. seit 12 Monaten		Neukunde
Consumer	max. seit 24 Monaten	max. 500 €	Bronze
Consumer	max. seit 24 Monaten	501 € bis 1.000 €	Silber
Consumer	max. seit 24 Monaten	über 1.000 €	Gold
Consumer	länger als 24 Monate	max. 500 €	Silber
Consumer	länger als 24 Monate	über 500 €	Gold
Business	max. seit 12 Monaten		Neukunde
Business	länger als 12 Monate	max. 1.000 €	Silber
Business	länger als 12 Monate	über 5.000 €	Platin
Business	max. seit 24 Monaten	1.001 € bis 5.000 €	Gold
Business	länger als 24 Monate	1.001 € bis 5.000 €	Platin

Hier könnte die Zuweisung in einem SQL Statement wie folgt aussehen:

```
proc sql;
     create table bsp2 as
     select    kdnr,
               (case     when kunde = 'consumer'
                              and dauer <= 12
                         then 'Neukunde'
                         when kunde = 'consumer'
                              and dauer <= 24
                              and umsatz <= 500
                         then 'Bronze'
                         when kunde = 'consumer'
                              and dauer <= 24
                              and umsatz between 501 and 1000
                         then 'Silber'
                         when kunde = 'consumer'
                              and dauer <= 24
                              and umsatz > 1000
                         then 'Gold'
                         …
                         when kunde = 'business'
                              and umsatz > 5000
                         then 'Platin'
               end) as kategorie
     from basis2;
quit;
```

Falls zusätzlich noch der Hinweis gegeben wird, dass sich diese Kriterienliste im Wochenrhythmus ändern kann und von Ihnen der Code entsprechend angepasst werden soll, haben Sie zwei Möglichkeiten:

1. Sie lassen sich über Änderungen informieren und passen diese selbst in Ihrem Programm an.
   Dieser Weg kann für Sie aufwändig sein und weißt ein geringes Spaßpotential auf.
2. Sie überlegen sich einen Weg, wie Ihr Auftraggeber seine Liste selber pflegen kann, ohne dass er Zugriff auf Ihr Programm erhält.
   Die Programmierung ist zwar im ersten Schritt etwas arbeitsintensiver, lohnt sich aber anschließend.
   Diesen Weg werde ich im folgenden Verlauf zeigen.

2.1 Erstellen einer Kriterienliste

Wird der SQL Code noch einmal genauer betrachtet, so fällt auf, dass alle Kriterien immer nach folgendem Schema aufgebaut sind:

```
when      <Variablenname_1> <Operator_1> <Wert_1>
```

```
        and <Variablenname_2> <Operator_2> <Wert_2>
        and <Variablenname_3> <Operator_3> <Wert_3>
    then    <Ergebnis>
```

Die fehlenden Textbausteine können aus einer externen Tabelle eingelesen werden. Diese sollte so aufbereitet werden, dass

1. sie für Ihren Auftraggeber verständlich ist und daher durch ihn gepflegt werden kann.
2. Sie diese in SAS importieren und weiterverarbeiten können.

Zu obigem Beispiel sieht die Kriterienliste wie folgt aus:

Tabelle 2: Kriterienübersicht zum Einlesen in SAS

var_1	op_1	wert_1	var_2	op_2	wert_2	var_3	op_3	wert_3	ergebnis
kunde	=	'consumer'	dauer	<=	12				'Neukunde'
kunde	=	'consumer'	dauer	<=	24	umsatz	<=	500	'Bronze'
kunde	=	'consumer'	dauer	<=	24	umsatz	<=	1000	'Silber'
kunde	=	'consumer'	dauer	<=	24	umsatz	>	1000	'Gold'
kunde	=	'consumer'	dauer	>	24	umsatz	<=	500	'Silber'
kunde	=	'consumer'	dauer	>	24	umsatz	>	500	'Gold'
kunde	=	'business'	dauer	<=	12				'Neukunde'
kunde	=	'business'	dauer	>	12	umsatz	<=	1000	'Gold'
kunde	=	'business'	dauer	>	12	umsatz	>	5000	'Platin'
kunde	=	'business'	dauer	<=	24				'Gold'
kunde	=	'business'	dauer	>	24				'Platin'

Diese Liste kann zum Beispiel in Excel hinterlegt werden. Die Pflege übernimmt der Auftraggeber und sie wird anschließend in SAS importiert. Die Variablennamen *var_1, op_1* und *wert_1* werden unter *Abfrage 1* zusammengefasst. Entsprechend *var_2* - *wert_3*.

Da es in einem späteren Schritt notwendig wird, alle *varnames*, *operator* und *werte* jeweils in ein Array zu lesen, sollten alle das gleiche Format haben, d.h. Stringwerte sein. Dies wird in der PROC IMPORT Prozedur einfach durch die Option *getnames = no* erreicht.

```
proc import
     datafile = "&vz.\ksfe_kriterienliste.xlsx"
     out = &out_dat.
     dbms = xlsx
     replace;
     sheet = 'liste';
     getnames = no;
run;
```

Da alle Variablen hierdurch einen vom SAS System vorgegebenen Namen erhalten haben (*A, B, C, ...*), müssen anschließend natürlich wieder alle Variablen umbenannt werden um ihren ursprünglichen Variablennamen zu erhalten. Dies kann durch einen PROC TRANSPOSE Aufruf erfolgen, wodurch man ein Dataset erhält, das eine Spalte mit den aktuellen Variablennamen (Spalte *_NAME_*)und eine Spalte mit den ursprünglichen Namen (diese stehen im Dataset in der ersten Zeile, Spalte *COL1*) enthält.

Diese Werte können, verknüpft mit einem '=' - Zeichen, in einem *_NULL_* - Datastep jeweils einer nummerierten Makrovariable zugewiesen werden. Diese beinhaltet anschließend die entsprechenden Werte zu folgendem Ausdruck:
NAME = COL1

```
proc transpose
     data = &out_dat.
     out = trans_dat;
     var _all_;
run;

data _null_;
     set trans_dat;
     call symput(   compress('rename_' || _n_),
                    compress(_name_ || '=' || col1)
          );
run;
```

In einem *rename* Statement können nun alle Makrovariablen aneinander gefügt werden, so dass folgender Befehl entsteht:

```
rename &rename_1. &rename_2. &rename_3.…;
```

Da dies allerdings manuelle Programmanpassungen nach sich ziehen würde, falls sich die Anzahl der Variablen verändert, kann die Zuweisung über eine Makrofunktion erfolgen.
Die Anzahl der aktuell verwendeten Variablen wird durch die systemseitig angelegte Makrovariable *&sysnobs.* direkt nach dem letzten *_NULL_* - Datastep abgerufen. Im anschließenden Datastep, in dem das ursprüngliche Dataset ab der zweiten Zeile übernommen wird, wird nur noch diese Funktion aufgerufen.

```
%let anz_rename = &sysnobs.;
%macro rename_vars;
     %local i;
     %do i = 1 %to &anz_rename.;
          &&rename_&i.
     %end;
%mend rename_vars;
data kriterien_2;
     set &out_dat.;
     rename %rename_vars;
     if _n_ >= 2; /*erste Zeile mit ursprünglichen Variablennamen*/
run;
```

Genau das gleiche Prinzip - Zuweisung nummerierter Makrovariablen und Auflösen dieser über eine Makrofunktion - können wir uns auch für unsere eigentliche Fragestellung zunutze machen.

2.2 Abfragekriterien Makrovariablen zuweisen

Die *Abfragen 1, 2* und *3* sind jeweils mit *AND* verknüpft, so dass eine neue Variable berechnet werden kann, die sich aus der Zusammenführung aller Variablen ergibt.

```
code      =    varname_1 || operator_1 || wert_1 || ' and ' ||
               varname_2 || operator_2 || wert_2 || ' and ' ||
               varname_3 || operator_3 || wert_3;
```

Schon für die erste Zeile ergibt sich allerdings für Abfrage 3 ein Problem, da sie nicht hinterlegt ist. Das Ergebnis sieht in diesem Fall wie folgt aus:

```
code      =    kunde = 'consumer' and dauer <= 12 and ;
```

Dieser Ausdruck würde zu einer Fehlermeldung führen, da nach dem letzten *and* keine weitere Zuweisung mehr erfolgt.

Daher müssen zuvor alle Missing-Values so ersetzt werden, dass trotzdem ein fehlerfreier Programmcode entsteht. Dies wird durch Ersetzung aller *varnames* und *werte* mit *'1'* und aller *operator* mit *'='* erreicht.Es sind jegliche Ausdrücke möglich, die zu einer wahren Aussage führen. Da alle Variablen als Strings hinterlegt sind, muss die Ersetzung ebenfalls durch Strings erfolgen.

Um die Variablen nicht mehr einzeln aufgrund ihres Formates prüfen zu müssen, wurden zu Beginn alle als Character Format eingelesen. Somit kann die Ersetzung der Missing-Values innerhalb eines Array erfolgen.

```
data kriterien_3;
     set kriterien_2;
     drop i j;

     array ersetzen_var(*) $ varname_: wert_:;
     do i = 1 to dim(ersetzen_var);
          if ersetzen_var(i) = '' then ersetzen_var(i) = '1';
     end;
     array ersetzen_op(*) $ operator_:;
     do j = 1 to dim(ersetzen_op);
          if ersetzen_op(j) = '' then ersetzen_op(j) = '=';
     end;
run;
```

Für die erste Zeile ergibt sich anschließend folgender Text:

```
code      =    kunde = 'consumer' and dauer <= 12 and 1 = 1;
```

Wird dies vorab um das Wort *when* und im Anschluss um den Ausdruck *then ergebnis* erweitert, also

```
code      =    when
               varname_1 || operator_1 || wert_1 || ' and ' ||
               varname_2 || operator_2 || wert_2 || ' and ' ||
               varname_3 || operator_3 || wert_3
               then ergebnis;
```

dann erhält man eine Spalte *code* mit folgendem Inhalt:

Tabelle 3: Inhalt der Variable *code*

code
when kunde='consumer' and dauer<=12 and 1=1 then 'Neukunde'
when kunde='consumer' and dauer<=24 and umsatz<=500 then 'Bronze'
when kunde='consumer' and dauer<=24 and umsatz<=1000 then 'Silber'
when kunde='consumer' and dauer<=24 and umsatz>1000 then 'Gold'
when kunde='consumer' and dauer>24 and umsatz<=500 then 'Silber'
when kunde='consumer' and dauer>24 and umsatz>500 then 'Gold'
when kunde='business' and dauer<=12 and 1=1 then 'Neukunde'
when kunde='business' and dauer>12 and umsatz<=1000 then 'Gold'
when kunde='business' and dauer>12 and umsatz>5000 then 'Platin'
when kunde='business' and dauer<=24 and 1=1 then 'Gold'
when kunde='business' and dauer>24 and 1=1 then 'Platin'

Diese Stringwerte können jetzt ganz einfach den Makrovariablen *&kriterium_1 - &kriterium_11* zugewiesen werden. Der entsprechende Datastep Code zu diesen beiden Schritten sieht wie folgt aus:

```
data _null_;
     set kriterien_3;
     code =    compbl(
               'when ' ||
               varname_1 || operator_1 || wert_1 || ' and ' ||
               varname_2 || operator_2 || wert_2 || ' and ' ||
               varname_3 || operator_3 || wert_3 ||
               'then ' || ergebnis
               );
     call symput(compress('kriterium_' || _n_), code);
run;
```

2.3 Aufruf der Makrovariablen innerhalb des SQL Statement

Um die Makrovariablen innerhalb des PROC SQL aufzulösen, können sie natürlich einzeln aufgerufen werden.

```
proc sql;
     create table ergebnis as
```

```
     select    *,
               (case     &kriterium_1.
                         &kriterium_2.
                         &kriterium_3.
                         &kriterium_4.
                         &kriterium_5.
                         &kriterium_6.
                         &kriterium_7.
                         &kriterium_8.
                         &kriterium_9.
                         &kriterium_10.
                         &kriterium_11.
                end) as kategorie
     from beispiel;
quit;
```

Hierzu muss allerdings immer die Anzahl der Kriterien bekannt sein. Wird die vorab importierte Kriterienliste verändert, so muss ebenfalls der Programmcode angepasst werden. Genau dies soll jedoch vermieden werden. Die Auflösung der Makrovariablen kann über eine Makrofunktion durchgeführt werden. Hierzu muss ebenfalls die Kriterienanzahl bekannt sein. Sie ist nach dem Aufruf des letzten *_NULL_* - Datastep in der systemseitig angelegten Makrovariable *&sysnobs.* hinterlegt und sollte einer neuen Makrovariable zugewiesen werden, die anschließend zur Anwendung kommt.

```
%let anz_var = &sysnobs.;

%macro zuweisung;
     %local i;
     %do i = 1 %to &anz_var.;
          &&kriterium_&i.
     %end;
%mend zuweisung;
```

Wird nun das Makro *%zuweisung* aufgerufen, so werden nacheinander die Makrovariablen *&kriterium_1 - &kriterium_11* aufgelöst und deren Inhalt als Bestandteil des SAS Code geschrieben und ausgeführt.

```
proc sql;
     create table ergebnis as
     select    *,
               (case %zuweisung end) as kategorie
     from beispiel;
quit;
```

Durch vorheriges Setzten der Optionen *mprint* und *mlogic* werden im *LOG* Fenster folgende Inhalte ausgegeben, die zeigen, dass die Makrovariablen aufgelöst und als Code innerhalb des PROC SQL Schrittes ausgeführt wurden.

```
...
MLOGIC(ZUWEISUNG):  Ausführung beginnt.
MLOGIC(ZUWEISUNG):  %LOCAL  I
MLOGIC(ZUWEISUNG):  %DO-Schleife beginnt; Indexvariable I;
                    Startwert ist 1;
                    Stoppwert ist 11;
                    By-Wert ist 1.
MLOGIC(ZUWEISUNG):  %DO-Schleife Indexvariable I ist jetzt 2;
          Schleife wird  wiederholt.
...
MLOGIC(ZUWEISUNG):  %DO-Schleife Indexvariable I ist jetzt 12;
          Schleife wird nicht wiederholt.

MPRINT(ZUWEISUNG):
     when kunde = 'consumer' and dauer <= 12 and 1 = 1
     then 'Neukunde'
     when kunde = 'consumer' and dauer <= 24 and umsatz <= 500
     then 'Bronze'
     when kunde = 'consumer' and dauer <= 24 and umsatz <= 1000
     then 'Silber'
     when kunde = 'consumer' and dauer <= 24 and umsatz > 1000
     then 'Gold'
     when kunde = 'consumer' and dauer > 24 and umsatz <= 500
     then 'Silber'
     when kunde = 'consumer' and dauer > 24 and umsatz > 500
     then 'Gold'
     when kunde = 'business' and dauer <= 12 and 1= 1
     then 'Neukunde'
     when kunde = 'business' and dauer > 12  and umsatz <= 1000
     then 'Gold'
     when kunde = 'business' and dauer > 12 and umsatz > 5000
     then 'Platin'
     when kunde = 'business' and dauer <= 24 and 1 = 1
     then 'Gold'
     when kunde = 'business' and dauer > 24 and 1 = 1
     then 'Platin'

MLOGIC(ZUWEISUNG):  Ausführung wird beendet.
126       from    beispiel;
...
```

3 Hinweise

Die Möglichkeit verschiedene Textbausteine miteinander zu verknüpfen und in eine Makrovariable zu laden ist natürlich durch die Maximallänge an Zeichen, die eine Makrovariable enthalten kann, beschränkt. Dies sind allerdings 65.534 Zeichen (SAS V9.1) und somit für die meisten Anwendungsfälle ausreichend.

Die Anzahl der Makrovariablen, die angelegt werden können, ist anscheinend nicht beschränkt. Zumindest wurde kein Hinweis darauf gefunden und beim Testen war die An-

lage von einer Million Makrovariablen möglich. Bei dieser Anzahl steigt natürlich der CPU Aufwand deutlich an.

4 Kompletter Programmcode

```
options mprint mlogic;
%let vz = C:\temp\praesentation_ksfe;
%let out_dat = kriterien;

/*******************************************************
Erstellung Beispieldaten
*******************************************************/

data beispiel;
     do kdnr = 1 to 100;
          /*Kunde*/
          if ranuni(0) < 0.5 then kunde = 'consumer';
          else kunde = 'business';
          dauer = int(ranuni(0)*100/2);
          umsatz = int(ranuni(0)*10000);
          output;
     end;
run;

/*******************************************************
Import Kriterienliste
*******************************************************/

proc import
     datafile = "&vz.\ksfe_kriterienliste.xlsx"
     out = &out_dat.
     dbms = xlsx
     replace;
     sheet = 'liste';
     getnames = no;
 run;

/*******************************************************
Variablennamen umbenennen
*******************************************************/
/*
Transpose
     ->   eine Spalte mit aktuellen Variablennamen (_NAME_)
          eine mit gewünschten Labels (COL1)
*/

proc transpose
     data = &out_dat.
     out = trans_dat;
     var _all_;
run;
```

```
data _null_;
     set trans_dat;
     call symput(   compress('rename_' || _n_),
                    compress(_name_ || '=' || col1));
run;

/*Anzahl Variablen zum umbenennen*/
%let anz_rename = &sysnobs.;

/*Aufruf der zuvor zugewiesenen Makrovariablen*/
%macro rename_vars;
     %local i;
     %do i = 1 %to &anz_rename.;
          &&rename_&i.
     %end;
%mend rename_vars;

data kriterien_2;
     set &out_dat.;
     rename %rename_vars;
     if _n_ >= 2;
run;

/*********************************************************
leere Felder ersetzen durch wahren Ausdruck, hier: 1 = 1
*********************************************************/
data kriterien_3;
     set kriterien_2;
     drop i j;

     array ersetzen_var(*) $ varname_: wert_:;
     do i = 1 to dim(ersetzen_var);
          if ersetzen_var(i) = '' then ersetzen_var(i) = '1';
     end;
     array ersetzen_op(*) $ operator_:;
     do j = 1 to dim(ersetzen_op);
          if ersetzen_op(j) = '' then ersetzen_op(j) = '=';
     end;
run;

/*********************************************************
Zuweisung und Aufruf der Kriterien
*********************************************************/

/*
Zuweisung je Kriterium eine Makrovariable,
durch Nummerierung bleibt Reihenfolge erhalten
*/

data _null_;
     set kriterien_3;
     code =    compbl(
                    'when ' ||
                    varname_1 || operator_1 || wert_1 || ' and ' ||
```

```
                    varname_2 || operator_2 || wert_2 || ' and ' ||
                    varname_3 || operator_3 || wert_3 ||
                    'then ' || ergebnis
               );
     call symput(compress('kriterium_' || _n_), code);
run;

/*Anzahl Kriterien*/
%let anz_var = &sysnobs.;

%macro zuweisung;
     %local i;
     %do i = 1 %to &anz_var.;
          &&kriterium_&i.
     %end;
%mend zuweisung;

proc sql;
     create table ergebnis as
     select    *,
               (case %zuweisung end) as kategorie
     from beispiel;
quit;
```