SAS NOTEs: die weniger harmlosen

Stefan Beimel
PAREXEL International GmbH
Am Bahnhof Westend 15
14059 Berlin
stefan.beimel@PAREXEL.com

Zusammenfassung

In sicher jeder Programmierrichtlinie werden eine Reihe von SAS NOTEs beschrieben, die nach Möglichkeit (oder die strengere Variante: unbedingt) vermieden werden sollten. Populäre Beispiele sind u.a.:
- NOTE: Variable XYZ is uninitialized.
- NOTE: MERGE statement has more than one data set with repeats of BY values.
- NOTE: Invalid data for XYZ in line ...
- NOTE: Invalid argument to function ABC.
- NOTE: Character values have been converted to numeric values

Bedenkenträger unter den Programmierern werden immer Gegenargumente finden:
- Wie finde ich diese NOTEs überhaupt? ERRORs und WARNINGs sind im traditionellen SAS Log wenigstens anders eingefärbt.
- Was ist daran überhaupt so schlimm? Was kann denn schiefgehen? Wenn SAS sie nicht als gefährlich ansieht (sonst wäre das ja wenigstens eine WARNING), warum sollte ich das tun?
- Der Aufwand, diese NOTEs zu vermeiden, ist viel zu groß. Und außerdem geht das überhaupt nicht vollständig.

Dieser Beitrag zeigt:
- Wann genau diese NOTEs und andere auftreten.
- Wie man sie findet.
- Warum sie tatsächlich vermieden werden sollten.
- Wie man sie vermeiden kann.

Schlüsselwörter: SAS Note, SAS Log, SAS Warning

1 Einführung

SAS dokumentiert in seinem Log die Ausführung von Programmen und unterscheidet dabei in INFO, NOTE, WARNING und ERROR. Als NOTEs werden Hinweise verstanden, die einfach nur den (erfolgreichen) Programmdurchlauf beschreiben - und sie sind es im allgemeinen auch. Allerdings deuten einige SAS NOTEs auf ein unerwünschtes Verhalten hin und sollten, genau wie ERRORs und WARNINGs, möglichst vermieden werden. Auf einige dieser NOTEs, die so gravierend sein können, dass sie durchaus den Status einer WARNING verdienen, geht dieser Vortrag näher ein.

Welche NOTEs als ‚verdächtig' und damit vermeidenswert gewertet werden, hängt von der Firma, vom Autor der Programmierrichtlinien und auch der Programmierumgebung ab.

In den meisten Fällen ist es ein Programmierfehler, der diese NOTEs erzeugt. Schon aus diesem Grund sollten sie nicht ignoriert werden. Ist die Funktion des Programmes tatsächlich so geplant, sind es oft recht einfache Programmierschritte, die die ‚verdächtigen' NOTEs verhindern können. Dies zu tun sollte eine Selbstverständlichkeit demjenigen gegenüber sein, der die Programme bewerten oder weiterentwickeln muss. Außerdem zeigt es, dass der Programmierer weiß, was er tut. Diese Schritte sind unter Lösung im nächsten Kapitel dargestellt.

2 Ausgesuchte NOTEs

2.1 Variable XXX is uninitialized.

SAS Code und Log

```
data;

    label aa="Label von AA";
    length bb $3;

    num  = cc + 1;
    char = dd || "de";

run;
```

```
NOTE: Variable AA is uninitialized.
NOTE: Variable BB is uninitialized.
NOTE: Variable CC is uninitialized.
NOTE: Variable DD is uninitialized.
```

Was ist passiert?
Es werden Variablen genutzt, denen kein Wert zugewiesen wurde. Das hat ganz unterschiedliche Auswirkungen. AA wird nicht in den Ausgabedatensatz übernommen. BB ist eine Textvariable. CC und DD werden beide als numerische Variablen mit fehlenden Werten im DATA step verarbeitet und auch so in den Ausgabedatensatz übernommen.

Lösung
Die einzige beabsichtigte Situation könnte BB sein, wenn eine leere Variable angelegt werden soll. BB sollte dann explizit ein leerer Text zugewiesen werde. Dann entfällt die NOTE, und man hat klar seine Absicht ausgedrückt.

2.2 MERGE statement has more than one data set with repeats of BY values

SAS Code und Log

```
data MutterUndKind;
    merge MutterUndTochter MutterUndSohn;
    by Mutter;
run;
```

```
NOTE: MERGE statement has more than one data set with repeats of BY
values.
```

Was ist passiert?
Sowohl im Datensatz MutterUndTochter als auch im Datensatz MutterUndSohn kommt der Wert „Margarethe“ mehr als einmal vor. Es gibt viele Möglichkeiten, mit dieser Situation umzugehen. SAS wiederholt in diesem Fall den letzten Wert der Variablen des Datensatzes mit weniger Wiederholungen (siehe im Beispiel „Christopher“).

Mutter	Tochter
Margarethe	Audrey
Margarethe	Sandra
Margarethe	Christina

Mutter	Sohn
Margarethe	Eric
Margarethe	Christopher

↘ ↙

Mutter	Tochter	Sohn
Margarethe	Audrey	Eric
Margarethe	Sandra	Christopher
Margarethe	Christina	Christopher

Lösung
Dieses Verhalten ist in den wenigsten Situationen sinnvoll. Es kann hier deshalb kein allgemeingültiger Workaround gezeigt werden.

2.3 At least one W.D format was too small for the number to be printed.

SAS Code und Log

```
data;
    input a $1-14 b 15-25;
    datalines;
gut            1.2
harmlos        100
schlecht       1234
ganz schlecht 12345678901
;run;
```

```
proc print;
    format b 3.1;
run;
```

NOTE: At least one W.D format was too small for the number to be printed. The decimal may be shifted by the "BEST" format.

Was ist passiert?
Das Format 3.1 ist zu kurz, um alle Stellen der Zahlen darzustellen (3 Stellen insgesamt, davon eine vor dem Dezimalpunkt, der Punkt selbst, und eine Nachkommastelle; im Falle negativer Zahlen sogar noch das Minuszeichen). SAS bemüht sich wirklich sehr, trotzdem so viel wie möglich Information auszugeben. Als erstes werden die Dezimalstellen abgeschnitten, dann wird die Zahl in Exponentialschreibweise dargestellt (1234 als 1E3, d.h. $1x10^3$). Wenn nichts mehr geht, werden nur noch * dargestellt. PROC PRINT gibt also in unserem Beispiel aus:

```
Obs   A              B

1     gut            1.2
2     harmlos        100
3     schlecht       1E3
4     ganz schlecht  ***
```

Lösung
In seltenen Fällen ist dieses Verhalten erwünscht, z. B. wenn Prozente mit zwei Stellen vor dem Komma und einer Nachkommastelle dargestellt werden sollen, die 100% aber ohne Nachkommastelle. Um die NOTE in diesen Fällen zu verhindern, kann dafür ein Format selbst erstellt werden:

```
proc format;
    value pct   100 = "100"
                other = [4.1];
run;
```

2.4 N observation(s) outside the axis range for the XXX * YYY request.

SAS Code und Log

```
data plot;
    input x y;
    datalines;
1   1
2   2.5
3   3
4   5
;run;
```

```
axis1 order=(1 to 4);
symbol1 interpol=join;

proc gplot data= plot;
    plot y*x / haxis=axis1 vaxis=axis1;
run;
```

```
NOTE: 1 observation(s) outside the axis range for the y * x request.
```

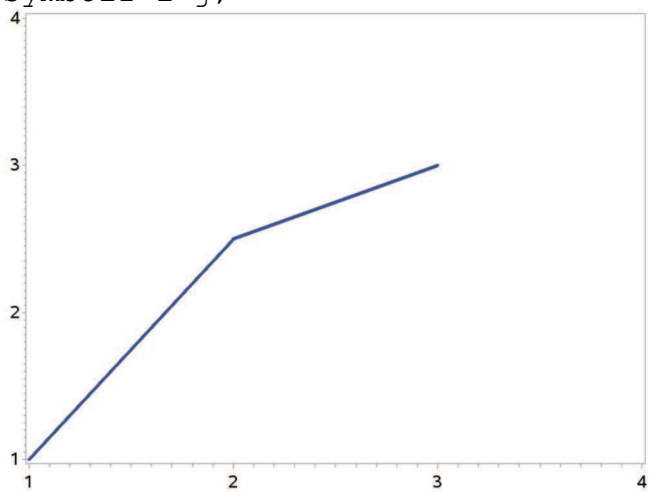
Was ist passiert?

Mit dem AXIS Statement und den Optionen HAXIS and VAXIS ist der Zeichenbereich festgelegt worden. Der Punkt (4;5) liegt außerhalb dieses Bereiches. Dieser Punkt wird weder beim Zeichnen von Linien (z. B. INTERPOL=JOIN) noch beim Berechnen (INTERPOL=BOX|HILO|STD) berücksichtigt (siehe Abbildung 1, linkes Bild). Bei Berechnungen wird statt einer NOTE eine WARNING ausgegeben.

Lösung

Kann der Zeichenbereich nicht angepasst werden, ist es möglich, dem SYMBOL Statement die Option MODE=INCLUDE hinzuzufügen. Das führt dazu, dass der außenliegende Punkt in die Berechnungen, z. B. eines Boxplots, mit aufgenommen wird. Außerdem werden Linien bis zum Rand der Zeichenfläche dargestellt (siehe Abbildung 1, rechtes Bild). Die NOTE wird immer noch ausgegeben, ist damit aber bewusst verarbeitet. Mit einem PUT Statement ist es möglich, zusätzlich eine erklärende NOTE ins Log zu schreiben.

```
symbol1 i=j;
```




```
symbol1 i=j mode=include;
```

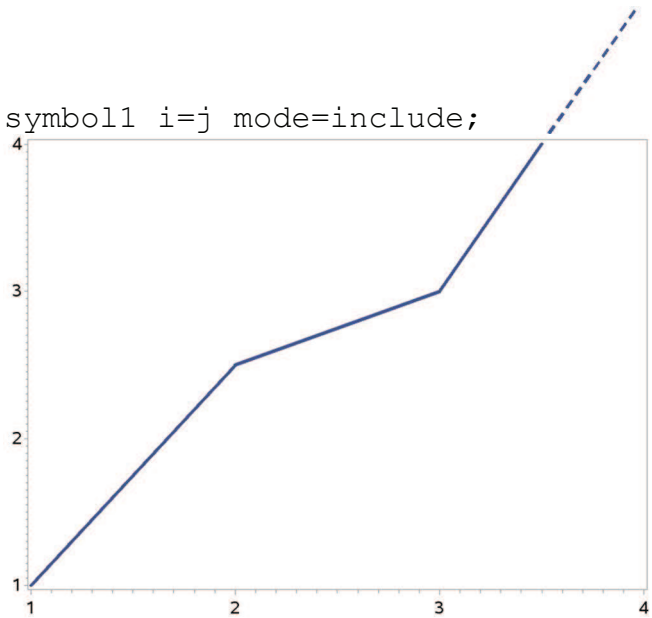


Abbildung 1: Einfluss der MODE=INCLUDE Option im SYMBOL Statement. Ohne die Option wird der Punkt (4;5) ignoriert (links). Im Bild rechts wird die Linie, die zu diesem Punkt gehört, bis zum Rand der Zeichenfläche gezeichnet.

2.5 Format XXX is already on the library.

SAS Code und Log

```
proc format;
    value noyes 1="no" 2="yes";
run;


proc format;
    value noyes 1="yes" 2="no";
run;
```

```
NOTE: Format NOYES is already on the library.
```

Was ist passiert?
Das Format NOYES, das im ersten PROC FORMAT erzeugt wurde, wird im zweiten PROC FORMAT überschrieben. Alle Variablen, denen das erste NOYES zugewiesen wurde, werden nun mit dem zweiten NOYES dargestellt – möglicherweise falsch.

Lösung
Formatfehler sind oft schwer zu entdecken. Oft ist es übersichtlicher, das zweite Format im Beispiel anders zu benennen. Sollte ein Format tatsächlich überschrieben werden müssen, kann es vorher gelöscht werden - um die NOTE zu verhindern, aber auch, um die Absicht ganz klar darzustellen.

```
proc catalog cat=work.formats;
    delete noyes /entrytype=format;
run;
```

Mit der Option NOREPLACE im PROC FORMAT Aufruf wird ein versehentliches Überschreiben verhindert.

2.6 Character values have been converted to numeric values

SAS Code und Log

```
data;

    dateV1="20170212";
    dateV2="20170312";
    dateDiff=dateV2-dateV1;

run;
```

```
NOTE: Character values have been converted to numeric values at the
      places given by: (Line):(Column).
      29:14   29:21
```

Was ist passiert?
Hier wurde mit Text gerechnet. Die Differenz der Textvariablen dateV1 und dateV2 wurde mit numerischen Operatoren (-) erzeugt. Dafür hat SAS die Textvariablen zunächst in Zahlen umgewandelt. Da im Beispiel oben die Variablen Datumsangaben beinhalten, ist das Ergebnis falsch. Im Beispiel hat die Variable dateDiff den Wert 100 statt wie eigentlich beabsichtigt 28.

Lösung
Ist eine Textvariable tatsächlich für numerische Operationen vorgesehen, sollte die Variable mit der Funktion INPUT explizit umgewandelt werden. Dadurch wird die NOTE verhindert, aber auch die Absicht des Programmierers deutlich gemacht. In unserem Beispiel hilft das YYMMDD Format:

```
dateDiff=input(dateV2, yymmdd8.) - input(dateV1, yymmdd8.);
```

In den meisten Fällen ist das BEST-Informat ausreichend:

```
input(TextVar, best.)
```

2.5 Weitere verdächtige NOTEs

Die Liste bemerkenswerter NOTEs ist lang. Die folgenden Fragmente (und viele mehr) können auf unerwünschtes Verhalten des SAS Programmes hinweisen:

```
NOTE: ... LOST CARD ...
NOTE: ... 0 variables ...
NOTE: No statistics are computed ...
NOTE: ... ignored ...
NOTE: ... stopped processing ...
NOTE: Invalid argument ...
NOTE: Invalid data ...
NOTE: ... Oracle execute error ...
NOTE: Numeric values have been converted ...
NOTE: Division by zero detected ...
NOTE: ... will be overwritten by data set ...
NOTE: SAS went to a new line ...
One or more lines were truncated
NOTE 49-169: The meaning of an identifier after a quoted string ...
```

3 Entdecken

Wenn NOTEs automatisch gesucht werden sollen, ist es wichtig, das richtige Fragment einer NOTE zu suchen. Die genaue Ausprägung hängt von vielen Faktoren ab:

- SAS Version
- Betriebssystem
- Log-Sprache
- Options
- Groß-/ Kleinschreibung
- Variable Texte in einer NOTE
- Zeilenlänge des Log
- Position zum Zeilenanfang

Es ist daher wichtig, Instrumente, die NOTEs suchen, regelmäßig zu testen. Empfehlenswert sind kurze Programme, die die jeweilige NOTE provozieren.

Nach dem Programmlauf im Batch, bei dem das Log automatisch als Textdatei gespeichert wurde, gibt es viele Möglichkeiten, das Log nach verdächtigen NOTEs zu scannen. Es gibt viele Beispiele von SAS Programmen, die eine Suche nach ERRORs, WARNINGs und NOTEs durchführen.

Unix (grep) und Windows (findstr) bieten in ihren Scriptsprachen Möglichkeiten der Textsuche. Das Ergebnis kann sofort am Bildschirm angezeigt werden oder zur späteren Weiterverarbeitung in einer Datei gespeichert werden. Es können einzelne Dateien oder ganze Verzeichnisse durchsucht werden. Die Programmierung des Scripts um die Textsuchfunktion herum hängt ganz von den Erfordernissen und der Programmierumgebung ab. Empfehlenswert ist es, diese Scripts automatisch ablaufen zu lassen und den Programmierer ‚ungefragt' auf die verdächtigen NOTEs aufmerksam zu machen.