SAS Backstage

Biljana Gigić
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

Ein SAS DATA Step wird grundsätzlich in zwei aufeinanderfolgenden Phasen verarbeitet, Kompilierungsphase und Ausführungsphase. Neben einer Syntaxprüfung des DATA Steps wird während der Kompilierungsphase der Input Buffer, der Program Data Vector (PDV) und der beschreibende Teil der Datei generiert. In der Ausführungsphase wird die Output-Datei erzeugt. Dabei verwaltet der Input Buffer den einzulesenden Datensatz. Die einzelnen Beobachtungen werden im PDV gespeichert und nach jedem Durchgang des DATA Steps in die neu zu erzeugende SAS Datei transferiert. Das Verständnis dieser Prozesse erleichtert den richtigen Umgang beim Einlesen von Rohdaten und Zusammenführen von Tabellen. Anhand von Beispielen soll konkret die Funktion der Buffer-Verwaltung und des PDV aufgezeigt werden.

Schlüsselwörter: SAS DATA Step, Kompilierungsphase, Ausführungsphase, Input Buffer, Program Data Vector, INPUT, LENGTH, MERGE, PUTLOG, DEBUG

1 Verarbeitungsphasen eines SAS DATA Steps

Ein SAS DATA Step wird in zwei Phasen verarbeitet: Zuerst die Kompilierungsphase, dann die Ausführungsphase. Während der Kompilierungsphase wird der DATA Step auf Syntaxfehler geprüft. Nacheinander wird ein Input Buffer, ein Program Data Vector (PDV) und der beschreibende Teil der neu zu erzeugenden SAS Datei generiert.

Folgende Rohdaten-Datei soll eingelesen werden:

`Patient.csv` (Auszug)

```
151,Heidelberg,,Adenocarcinoma
152,Heidelberg,58,Mucinous adenocarcinoma
163,Heidelberg,60,Adenocarcinoma
205,München,72,Adenocarcinoma
217,München,55,Mucinous adenocarcinoma
...
```

Die Comma-Separated-Values-Datei **`Patient.csv`** beinhaltet Daten zu 78 Tumorpatienten und setzt sich aus der Patientenidentifikation, dem behandelnden Zentrum,

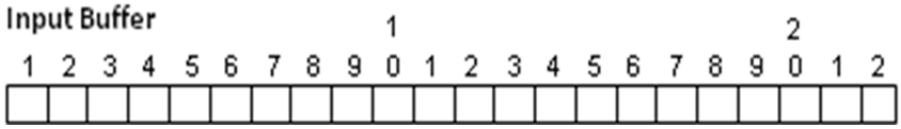
dem Alter und der Tumordiagnose des Patienten zusammen. Die Rohdaten werden listenorientiert [1] eingelesen:

```
data work.Patient;
  infile 'x:\Patient.csv' dsd;
  input Pat_ID Center $ Age Diagnosis $;
run;
```

1.1 Kompilierungsphase

Input Buffer

Beim Einlesen von Rohdaten aus einer externen Datei wird zu Beginn der Kompilierungsphase mit der INFILE Anweisung der Input Buffer generiert. Er hält den gesamten aktuellen Datensatz aus der Rohdaten-Datei.
Der Input Buffer dient als ein logischer „Zwischenspeicher".



Der Input Buffer wird nur beim Einlesen von Rohdaten erzeugt, nicht wenn SAS Dateien eingelesen werden.

Program Data Vector

Nach dem Input Buffer wird der Program Data Vector (PDV) erzeugt. Der PDV erzeugt den jeweiligen SAS Datensatz, und zwar einen Datensatz pro ausgeführten DATA Step. Die Daten erhält der PDV vom Input Buffer. Der PDV wird während der INPUT Anweisung generiert. Dabei erhält er den Namen der Variable, den Typ sowie die Länge. Beim listenorientierten Einlesen ist die Länge der Variablen, sowohl für numerische als auch alphanumerische, standardmäßig auf 8 Zeichen begrenzt. Die einzelnen Variablen werden der Reihe nach, wie sie in der INPUT Anweisung vorliegen, im PDV erzeugt.
Der PDV führt zwei automatische Variablen mit sich: _N_ und _ERROR_. _N_ gibt die Anzahl der Ausführung des DATA Steps an. _ERROR_ kontrolliert fehlerhafte Daten. _ERROR_ ist standardmäßig auf 0 gesetzt. Wird ein ungültiges Datum eingelesen bzw. Fehler innerhalb der Daten detektiert, erhält _ERROR_ den Wert 1. Der DATA Step gibt eine Fehlermeldung im LOG Fenster aus, wird allerdings nicht unterbrochen und führt die Verarbeitung bis zum Ende durch.
_N_ und _ERROR_ werden nicht an die neu zu erzeugende SAS Datei übergeben, sie dienen ausschließlich zur Kontrolle der Verarbeitung der Daten.

PDV

_N_	_ERROR_	Pat_ID N 8	Center $ 8	Age N 8	Diagnosis $ 8
1	0				

Beschreibender Teil der SAS Datei
Am Ende der Kompilierungsphase wird der beschreibende Teil der neu auszugebenden SAS Datei generiert. Es handelt sich hier um die Metadaten der SAS Datei, die beispielsweise mit einem PROC CONTENTS ausgegeben werden können.

Data Set Name	WORK.Patient
Engine	V9
Created	Montag, 27. Februar 2012 13.17 Uhr
Observations	78
Variables	4
...	

_N_	_ERROR_	Pat_ID N 8	Center $ 8	Age N 8	Diagnosis $ 8
1	0				

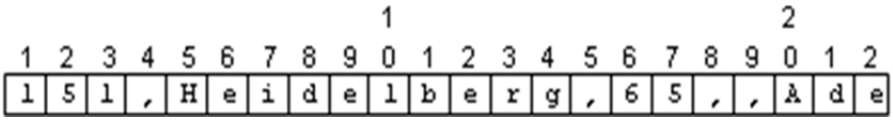
1.2 Ausführungsphase

Die Ausführungsphase beginnt mit einer DATA Anweisung. Dabei werden die Iterationen des DATA Steps gezählt und von der Variable _N_ festgehalten.
Die Werte der einzulesenden Variablen im PDV werden auf „missing“ gesetzt.

PDV

_N_	_ERROR_	Pat_ID N 8	Center $ 8	Age N 8	Diagnosis $ 8
1	0	.		.	

Im nächsten Schritt wird die INPUT Anweisung verarbeitet. Dabei wird in der Rohdaten-Datei geprüft, ob ein Datensatz zum Einlesen vorliegt. Trifft dies zu, werden die Rohdaten des aktuellen Datensatzes vom Input Buffer gehalten und nacheinander vom PDV eingelesen.




```
data work.Patient;
  infile 'x:\Patient.csv' dsd;
  input Pat_ID Center $ Age Diagnosis $;
run;
```

PDV

_N_	_ERROR_	Pat_ID N 8	Center $ 8	Age N 8	Diagnosis $ 8
1	0	151	Heidelbe	.	Adenocar

Am Ende des DATA Steps, vor der RUN Anweisung, erfolgen ein impliziter OUTPUT und ein impliziter RETURN. Der erzeugte Datensatz wird an die SAS Datei übergeben, das System springt zurück an den Anfang des DATA Steps. Der PDV wird reinitialisiert und der nächste Datensatz analog zum ersten DATA Step eingelesen. Dieser Prozess wird bis zum Ende der Rohdaten-Datei durchgeführt. Nach dem Einlesen des letzten

Rohdatensatzes befindet sich das System durch den impliziten RETURN erneut am Anfang des DATA Steps. Die Ausführungsphase läuft wie bereits beschrieben weiter, bis von der INPUT Anweisung ein neuer Rohdatensatz zum Einlesen gefordert wird. Ist dies nicht gegeben, wird der DATA Step mit der RUN Anweisung geschlossen und die nächsten DATA oder PROC Steps werden ausgeführt.

Output der ersten 5 Datensätze aus der SAS Datei **`work.Patient`**

Pat_ID	Center	Age	Diagnosis
151	Heidelbe	.	Adenocar
152	Heidelbe	58	Mucinous
163	Heidelbe	60	Adenocar
205	München	72	Adenocar
217	München	55	Mucinous

Die Werte der Variablen **`Center`** und **`Diagnosis`** wurden standardmäßig mit der Länge 8 eingelesen und sind unvollständig.

2 LENGTH Anweisung

Damit die Daten vollständig eingelesen werden, wird die LENGTH Anweisung hinzugefügt.

```
data work.Patient;
  infile 'x:\Patient.csv' dsd;
  length Center $ 12 Diagnosis $ 26;
  input Pat_ID Center $ Age Diagnosis $;
run;
```

Center	Diagnosis	Pat_ID	Age
Heidelberg	Adenocarcinoma	151	.
Heidelberg	Mucinous adenocarcinoma	152	58
Heidelberg	Adenocarcinoma	163	60
München	Adenocarcinoma	205	72
München	Mucinous adenocarcinoma	217	55

Der Output:

Ein weiterer Effekt, den die LENGTH Anweisung mit sich bringt, ist das Umsortieren der Variablen. Der Grund hierfür ist die Reihenfolge der LENGTH und INPUT Anweisung. Zunächst werden im PDV die Variablen **`Center`** und **`Diagnosis`** aus der LENGTH Anweisung erstellt und erst im nächsten Schritt die restlichen Variablen während der INPUT Anweisung ergänzt.

3 MERGE Anweisung

Im Folgenden werden anhand der MERGE Anweisung die SAS Dateien **work.Phase1** und **work.Phase2** über die Variable **Pat_ID** zu **work.Phases** verknüpft. Beide Dateien sind nach **Pat_ID** aufsteigend sortiert.

work.Phase1

Obs	Pat_ID	Age	Gender
1	151	55	male
2	151	.	female
3	152	58	female
4	163	60	male

work.Phase2

Obs	Pat_ID	Age	Screened
1	151	.	yes
2	151	63	no
3	163	61	no

Die Merge Anweisung:

```
data work.Phases;
  merge work.Phase1 work.Phase2;
  by Pat_ID;
run;
```

Die BY-Anweisung erzeugt zwei temporäre Variablen im PDV, **FIRST.Pat_ID** und **LAST.Pat_ID**.

PDV

_N_	_ERROR_	Pat_ID N 8	Age N 8	Gender $ 8	Screened $ 8	FIRST. Pat_ID	LAST. Pat_ID
1	0						

1. DATA Step: Die Ausführungsphase beginnt mit der Initialisierung des PDV, d.h. alle Werte der Variablen werden auf „missing“ gesetzt.
Im nächsten Schritt wird die MERGE Anweisung verarbeitet. Die BY-Variablen der ersten Beobachtungen in beiden Tabellen werden verglichen. Stimmen die Werte überein, wird jeweils eine Beobachtung aus **work.Phase1** und **work.Phase2** eingelesen. Dabei werden zunächst die Werte aus **work.Phase1**, anschließend aus **work.Phase2** eingelesen. Existieren in beiden Dateien gleichnamige Variablen, werden die Werte aus der zuerst eingelesenen Datei durch die Werte der zuletzt eingelesenen Datei überschrieben. Die Reihenfolge kann in der MERGE Anweisung vom SAS Anwender selbst bestimmt werden.

PDV

_N_	_ERROR_	Pat_ID N 8	Age N 8	Gender $ 8	Screened $ 8	FIRST. Pat_ID	LAST. Pat_ID
1	0	151	.	male	yes	1	0

Nach einem impliziten OUTPUT und impliziten RETURN werden die Werte in die neue SAS Datei **work.Phases** geschrieben. Das System springt zurück an den Anfang des SAS DATA Steps.

work.Phases

Pat_ID	Age	Gender	Screened
151	.	male	yes

2. DATA Step: Die nächste Beobachtung in den Dateien wird geprüft. Die Werte der BY-Variablen stimmen überein, die Beobachtungen werden in den PDV eingelesen.

PDV

_N_	_ERROR_	Pat_ID N 8	Age N 8	Gender $ 8	Screened $ 8	FIRST. Pat_ID	LAST. Pat_ID
2	0	151	63	female	no	0	1

Die zweite Beobachtung wird in die SAS Datei **work.Phases** transferiert und das System befindet sich erneut am Anfang des DATA Steps.

work.Phases

Pat_ID	Age	Gender	Screened
151	.	male	yes
151	63	female	No

3. DATA Step: Bei den nächsten beiden Beobachtungen stimmen die Werte der BY-Variable nicht überein. Es wird zunächst geprüft, ob eine der BY-Variablen im PDV bereits vorliegt. Da sie nicht mit dem Wert im PDV übereinstimmt, wird die Beobachtung mit dem niedrigeren Wert (aufsteigende Sortierung) eingelesen, d.h. der dritte Datensatz aus der SAS Datei **work.Phase1**.

PDV

_N_	_ERROR_	Pat_ID N 8	Age N 8	Gender $ 8	Screened $ 8	FIRST. Pat_ID	LAST. Pat_ID
3	0	152	58	female		1	1

work.Phases

Pat_ID	Age	Gender	Screened
151	.	male	yes
151	63	female	No
152	58	female	

4. DATA Step: In **work.Phases1** wird der vierte, in **work.Phase2** der dritte Datensatz beobachtet. Die Werte der BY-Variablen stimmen überein und werden in den PDV geschrieben und in **work.Phases** transferiert:

work.Phases

Pat_ID	Age	Gender	Screened
151	.	male	yes
151	63	female	no
152	58	female	
163	61	male	no

5. DATA Step: In diesem letzten Schritt wird in beiden SAS Dateien das End of File (EOF) erkannt, der DATA Step wird beendet.

4 DEBUGGING

Wird während der Kompilierungsphase ein Syntaxfehler detektiert, erfolgt im SAS Log eine Fehlermeldung. Der Grund hierfür ist das nicht einhalten der Regeln der SAS Sprache. Ist die Syntax korrekt, die Ergebnisse aber fehlerhaft, wird im Log keine Meldung ausgegeben. Solche logischen Fehler sind sehr schwer zu entdecken und bleiben bei großen Datenmengen oft verborgen. In diesem Kapitel sollen zwei Möglichkeiten des Debuggings in SAS vorgestellt werden.

4.1 PUTLOG Anweisung

Die PUTLOG Anweisung ermöglicht unter anderem das Schreiben des aktuellen oder eines bestimmten Inhaltes des PDV im SAS Log.
Zur Veranschaulichung soll das Beispiel aus Kapitel 3 verwendet werden. Dabei sollen die Werte aller Variablen (_ALL_) aus dem PDV im SAS Log ausgegeben werden. Die Syntax hierfür sieht folgendermaßen aus:

```
data work.Phases;
  merge work.Phase1 work.Phase2;
  by Pat_ID;
  putlog _ALL_;
run;
```

Ausgabe im SAS Log:

```
Pat_ID=151 Age=. Gender=male Screened=yes FIRST.Pat_ID=1
LAST.Pat_ID=0 _ERROR_=0 _N_=1
Pat_ID=151 Age=63 Gender=female Screened=no FIRST.Pat_ID=0
LAST.Pat_ID=1 _ERROR_=0 _N_=2
Pat_ID=152 Age=58 Gender=female Screened=  FIRST.Pat_ID=1
LAST.Pat_ID=1 _ERROR_=0 _N_=3
Pat_ID=163 Age=61 Gender=male Screened=no FIRST.Pat_ID=1
LAST.Pat_ID=1 _ERROR_=0 _N_=4
```

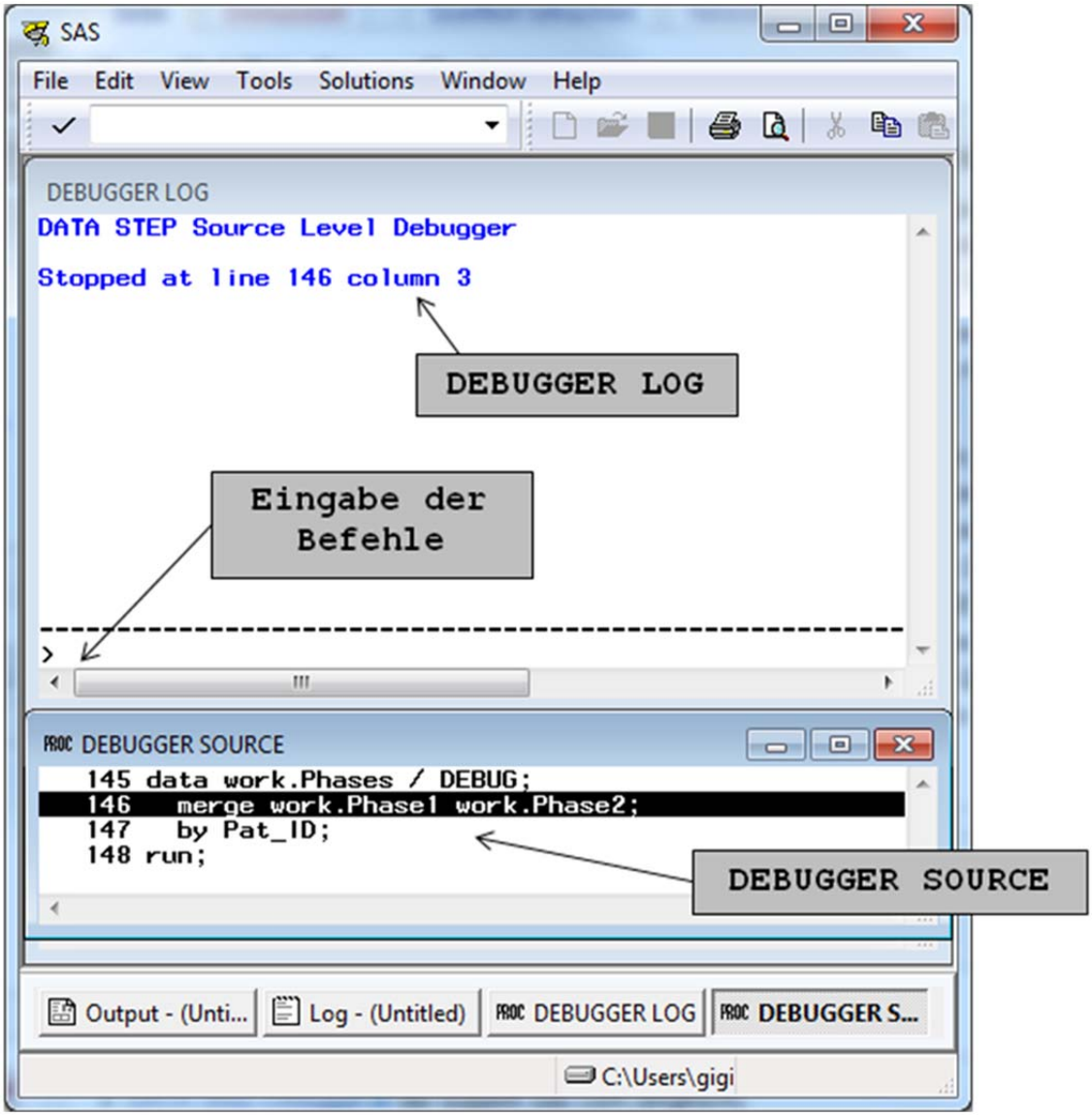
4.2 DEBUG Option

Eine weitere Möglichkeit zur Beobachtung des PDVs und zur Kontrolle der Datenverarbeitung bietet die DEBUG Option.
Der SAS DEBUGGER [2] ist eine interaktive Schnittstelle zum SAS DATA Step.

Syntax der DEBUG Option:

```
data work.Phases / DEBUG;
  merge work.Phase1 work.Phase2;
  by Pat_ID;
  putlog _ALL_;
run;
```

Beim Ausführen des DATA Steps wird die DEBUGGING Session gestartet.



Das Fenster „DEBUGGER SOURCE“ zeigt dabei den Code des DATA Steps an. „DEBUGGER LOG“ zeigt den LOG des DEBUGGERS an und enthält eine Eingabezeile für DEBUGGER-Befehle [3].

Unter anderem stehen folgende Befehle zur Verfügung:

- STEP (Schrittweise Ausführung von SAS DATA Steps)
- EXAMINE (Prüfen des Wertes einer oder mehrerer Variablen im PDV)
- WATCH (Änderungen der Werte einer oder mehrerer Variablen im PDV beobachten)
- DESCRIBE (Metadaten ausgeben)

Mit dem Befehl QUIT wird die DEBUGGER Session beendet.

Schlussfolgerung

Warum wurden meine Daten nicht vollständig eingelesen? Wie werden die Variablen _ERROR_ und _N_ erzeugt? Wie kann ich die Verarbeitung des DATA Steps schrittweise beobachten und kontrollieren?
Das Verständnis der Hintergrundprozesse eines SAS DATA Steps erleichtert den richtigen Umgang u.a. beim Einlesen von Rohdaten oder Zusammenführen mehrerer Dateien. Gerade beim Verarbeiten großer Datenmengen ist es von Vorteil, diese Prozesse zu kontrollieren und somit logische Fehler oder Fehler innerhalb der Daten zu erkennen. Die Bedienung und Anwendung der verschiedenen Debugging-Möglichkeiten führt zur höheren Qualität und Validität der verarbeiteten Daten.

Literatur

[1] SAS Help: Starting with Raw Data: The Basics

[2] SAS Help: DATA Step Debugger

[3] J. Spilke, C. Becker, E. Schuhmacher: Proceedings der 13. KSFE. Shaker-Verlag, 2009.