



# Programmierrichtlinien – unterstützende Hilfe oder lästige Formalität?

Stefan Beimel

Merz Pharmaceuticals GmbH

We care.  
OUR RESEARCH FOR YOUR HEALTH.





## Warum?

- Häufige Fehler lassen sich oft mit einfachen Mitteln vermeiden.
- Historisch gewachsene Programme lassen sich besser überarbeiten.
- SDLC: Zu jeder guten Validierung gehören Programmier-Richtlinien.
- Nebeneffekt  
Tipps und Tricks für Programmierer werden weitergegeben.




## Inhalt


- Dokumentation im Programmcode
  - Kommentare
  - Transparente Programmierung
  - Layout
  
- Fehlervermeidung
  - Batch Modus
  - Wiederholungen vermeiden
  - ERRORS, WARNINGS, NOTES
  - Runden
  - SELECT Statement
  - Keine Schönheitsoperationen



### Kommentare

- Programmkopf benutzen (wer, wann, wozu)
- Sprechende Programm-, Datensatz-, Variablen-, Formatnamen verwenden
- längere Programme gliedern (Zwischenüberschriften)
- komplizierte Abschnitte zeitnah kommentieren
- Ein Programm ist kein Archiv - unnötige Zeilen löschen
- Statt `/* */` besser `* ;` oder `%* ;`



 data ...;  
set demo /\* Demographie \*/  
ae /\* Adverse Events \*/

 data ...;  
set demo %\* Demographie;  
ae %\* Adverse Events;





# Transparente Programmierung im Data Step



## ■ Positiv denken

statt  `if not (age <= 35) then ...;`  
besser  `if age > 35 then ...;`

## ■ 0 / 1 wie Wahrheitswerte benutzen

statt  `if first.patno=0 then ...;`  
besser  `if not first.patno then ...;`

## ■ IN operator statt eine Folge von ORs

statt  `if patno=12 or patno=14 or patno=18 then ...;`  
besser  `if patno in(12, 14, 18) then ...;`



### Layout

- Einrücken (keine Tabs) →
- Ausrichten:  $\updownarrow$  DO/SELECT - END  
 $\updownarrow$  THEN - ELSE

```
data new;  
→ set old;  
→ do i=1 to 13;  
  → output;  
→ end;  
run;
```

- ein Statement pro Zeile
- RUN und Leerzeile nach DATA/ PROC step
- Kleinbuchstaben (außer Strings und Kommentare)



## Inhalt

- Dokumentation im Programmcode
  - Kommentare
  - Transparente Programmierung
  - Layout
  
- Fehlervermeidung
  - Batch Modus
  - Wiederholungen vermeiden
  - ERRORs, WARNINGS, NOTEs
  - Runden
  - SELECT Statement
  - Keine Schöheitsoperationen



# Im Batchmodus ausführen

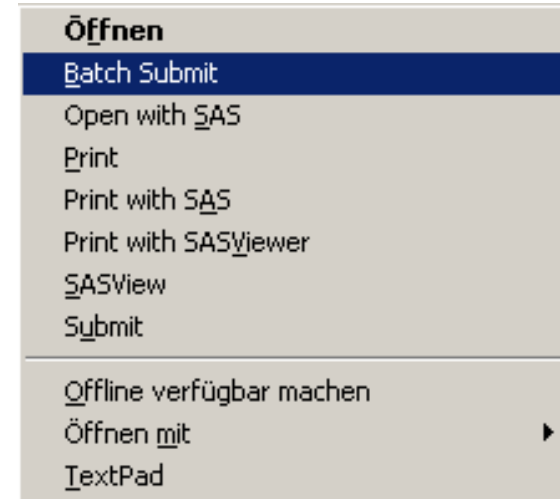
- keine historischen Infos, d.h. ein Programm läuft immer gleich
- Vermeidung von unvollständigen Programmen

```
data b;  
    set a;    *** Wo ist der erzeugt?
```

- Vermeiden von Programmen mit falscher Chronologie

```
data b;  
    set l.a;    *** l ist noch nicht zugewiesen  
  
libname l "C:\daten";
```

- Vollständiges, automatisch gespeichertes Log vorhanden



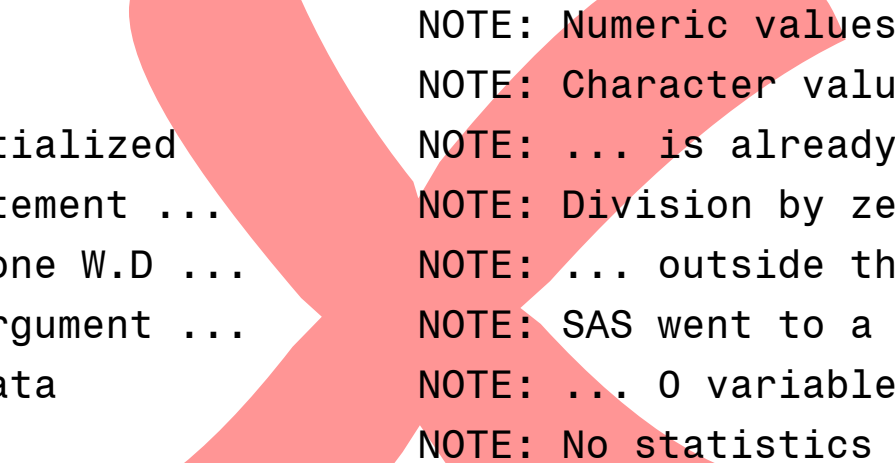


## Wiederholungen vermeiden (Code und Programmlauf)

- Auf vorhandenen, standardisierten Code zurückgreifen
  - AutoExec.sas                      allgemeine SAS Umgebung  
`options mergenoby=warn validvarname=upcase nocenter nodate;`
  - StudyInit.sas                      studienspezifische Umgebung  
`libname data "L:\proj4711\RawData";`
  - AutoCall-Bibliothek              (validierte) Standardmakros
- ARRAYS oder MACROS für sich wiederholende Schritte
- Wiederholt benötigte oder komplizierte Berechnungen nur einmal durchführen und in permanenten Datensätzen ablegen



# ERRORs, WARNINGs, verdächtige NOTEs vermeiden



ERROR	NOTE: Numeric values have been converted...
WARNING	NOTE: Character values have been converted
NOTE: ... uninitialized	NOTE: ... is already on the library
NOTE: MERGE statement ...	NOTE: Division by zero detected
NOTE: At least one W.D ...	NOTE: ... outside the axis range ...
NOTE: Invalid argument ...	NOTE: SAS went to a new line ...
NOTE: Invalid data	NOTE: ... 0 variables ...
NOTE: LOST CARD	NOTE: No statistics are computed...



## 'Werterhaltendes' Runden von numerischen Variablen

$$4.6 + 0.1 + 0.1 = 4.7999999999999999 \\ \neq 4.8$$

- Probleme beim Vergleichen, Sortieren, Ranking
- z.B. - Vergleich von Laborwerten mit Normalbereichen  
- PROC NPAR1WAY WILCOXON
- numerische Variablen mit Nachkommastellen sollten gerundet werden, bevor sie verglichen, sortiert oder gerankt werden:

**ROUND(num\_var, 10E-12)**

- beim Importieren aus Datenbanken runden



## SELECT Statement

**X** `if sex='male' then index=0.5;           * male;  
                  else index=0.6;           * female;`

- Problem: Alle Werte außer 'male' erhalten den Wert 0.6, also auch 'Male' und ''.

**✓** `select (sex);  
      when ('male')       index = 0.5;  
      when ('female')    index = 0.6;  
      otherwise put 'WARNING: Unerwarteter Wert ' sex=;  
end;`

- Es müssen alle Observations behandelt werden.



# Keine Schönheitsoperationen





## Weiterführende Regeln

- Tests während der Programmierung
- SAS-Makros
- Performance
- Programmierumgebung
- Checkliste
- und viele andere, für die keine Zeit war