Katja Glass Consulting

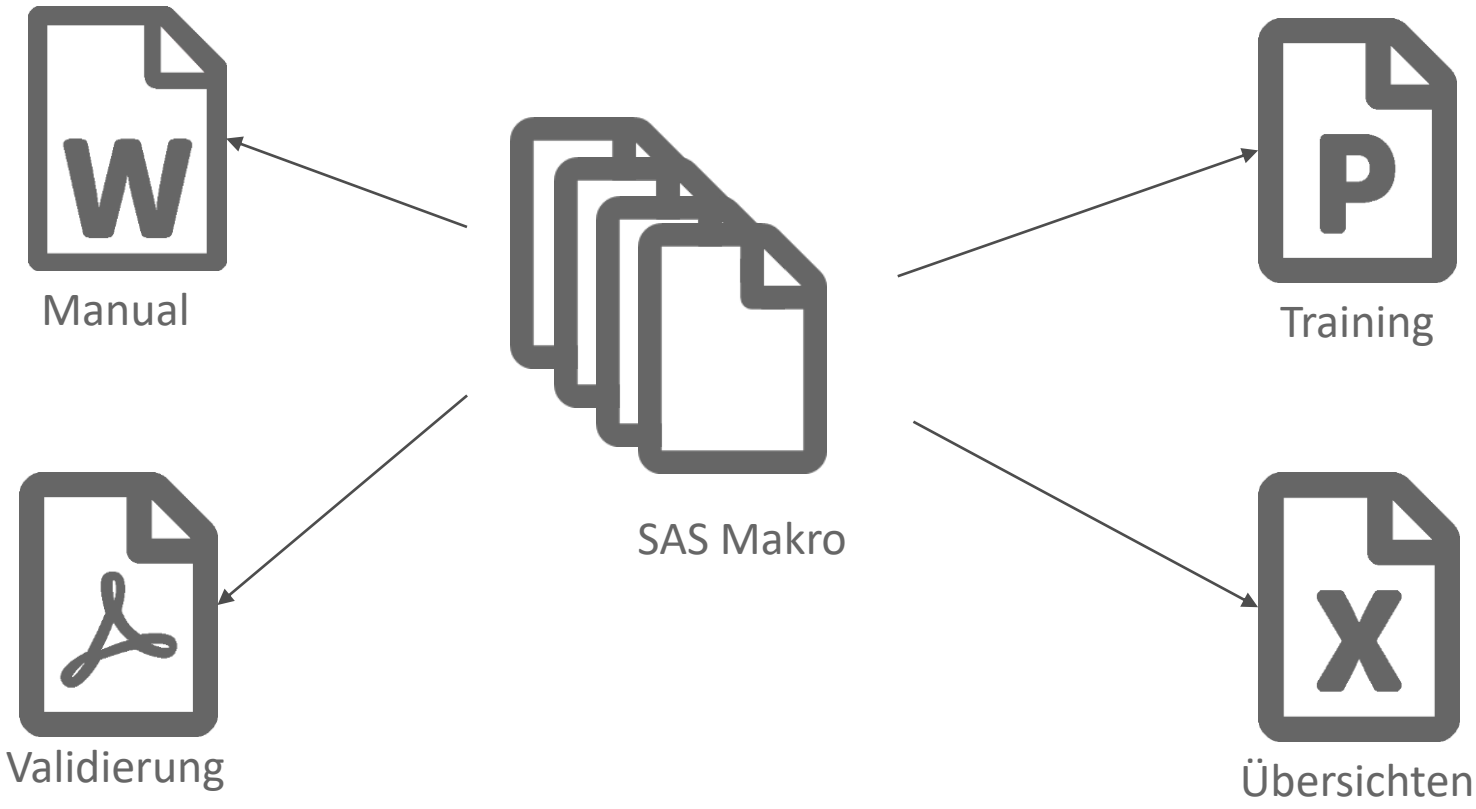
Programminhalte extrahieren und modifizieren mit SAS

Katja Glaß

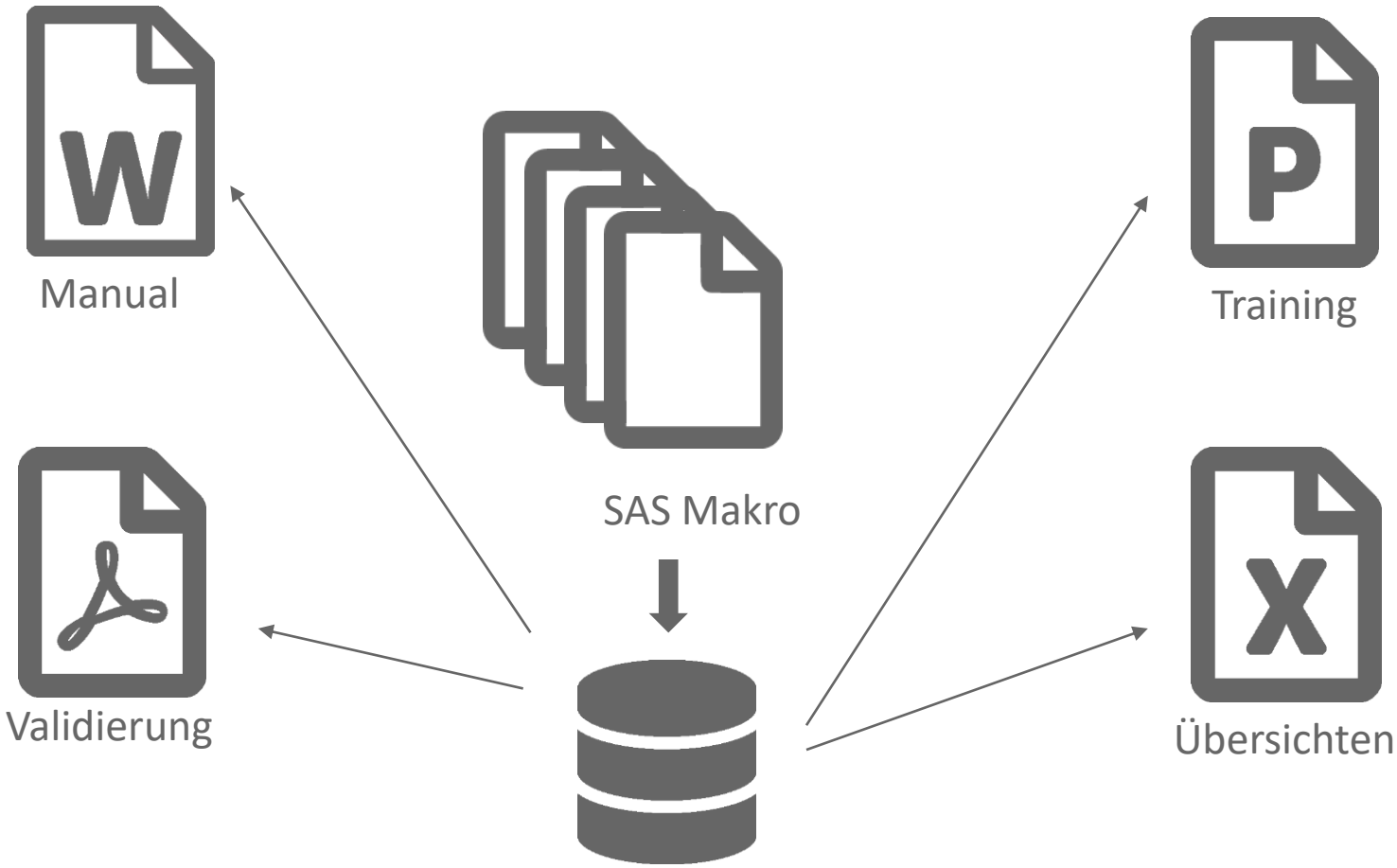
Agenda

- Einleitung
- Fallbeispiel
- Einsatzgebiete

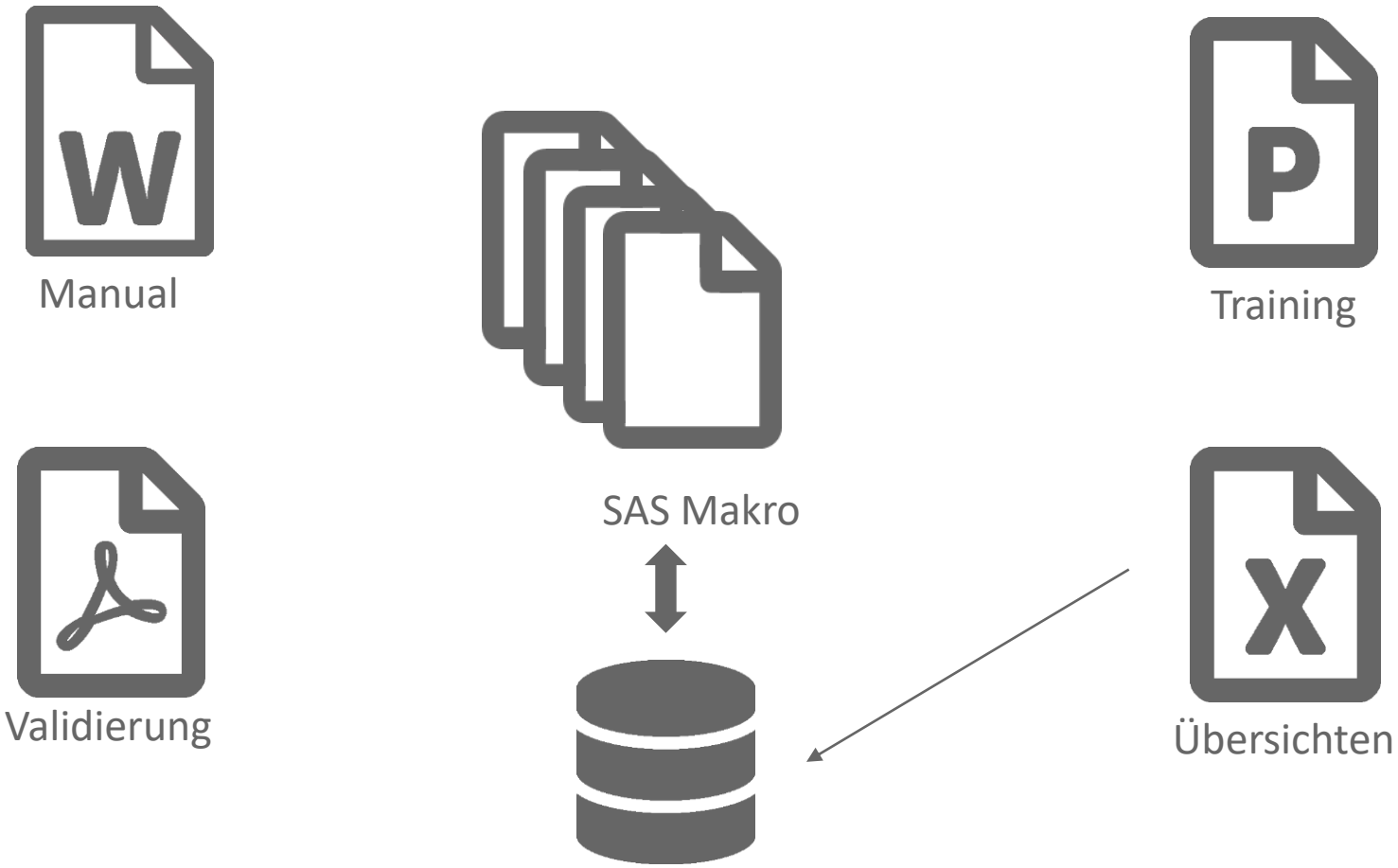
Einleitung



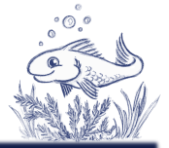
Einleitung



Einleitung



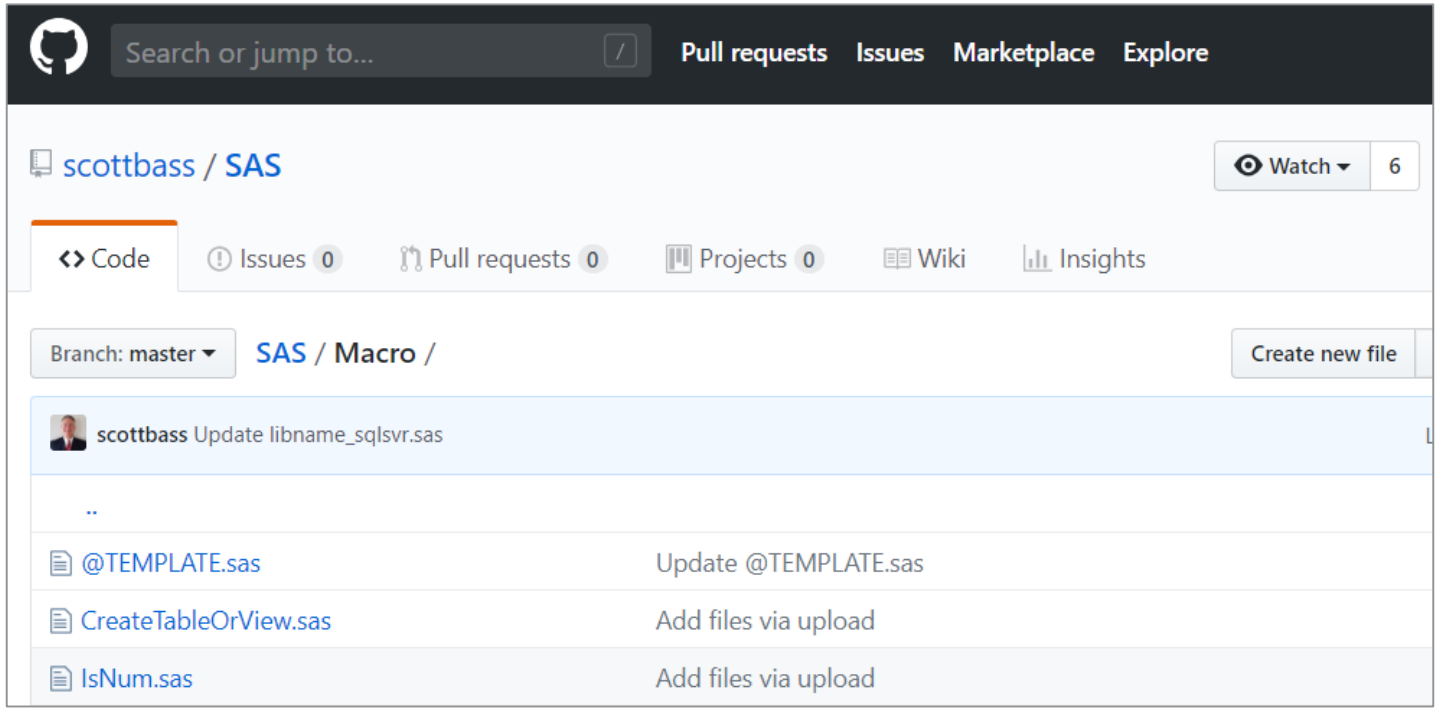
Einleitung



- Dateimanipulation
  - Lesen
  - Ändern
  - Löschen

Einleitung

- „Unlicense“ SAS Makros von Scott Bass

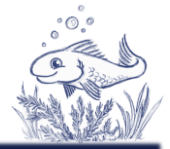


https://github.com/scottbass/SAS

Agenda

- Einleitung
- **Fallbeispiel**
- Einsatzgebiete

Fallbeispiel



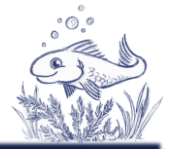
- Informationen aus dem Header extrahieren

```
/*=================================================================
Program Name                : optsave.sas
Purpose                     : Saves current options settings to a SAS
                              dataset
SAS Version                 : SAS 9.1.3
Input Data                  : None
Output Data                 : SAS options dataset

Macros Called               : parmv

Originally Written by       : Scott Bass
Date                        : 14MAY2010
Program Version #           : 1.0
=================================================================
```

Fallbeispiel



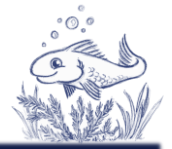
- DATA Step mit _INFILE_

```
%LET path = <path>;

DATA content;
     INFILE "&path/Macro/varlist.sas";
     INPUT;
     line = _INFILE_;
RUN;
```

	line
1	/*==========
2	Program Name : varlist.sas
3	Purpose : Returns a string containing a space separated
4	list of variables in a dataset

Fallbeispiel

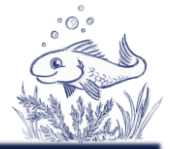


- Variablen anlegen, ATTRIB

```
ATTRIB line          FORMAT = $255.      LABEL="File content line"
   name              FORMAT = $50.       LABEL="Macro Name"
   purpose           FORMAT = $500.      LABEL="Purpose"
   sas_version       FORMAT = $10.       LABEL="SAS Version"
   input_data        FORMAT = $50.       LABEL="Input Data"
   output_data       FORMAT = $50.       LABEL="Output Data"
   mac_called        FORMAT = $200.      LABEL="Macros Called"
   origin_by         FORMAT = $50.       LABEL="Originally Written By"
   date              FORMAT = $50.       LABEL="Date"
   mac_version       FORMAT = $10.       LABEL="Program Version";
```

line	name	purpose	sas_version	input_data	output_data	mac_called	origin_by	date	mac_version
/*===========									
Program Name : varlist.sas									
Purpose : Returns a string containing a space separated									
list of variables in a dataset									
SAS Version : SAS 9.3									
Input Data : N/A									

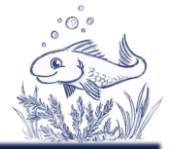
Fallbeispiel



- Füllen der Variablen

line	name	purpose	sas_version	input_data
/*========== ========== ==========				
Program Name : varlist.sas →	varlist.sas			
Purpose : Returns a string containing a space separated list of variables in a dataset		...		
SAS Version : SAS 9.3			SAS 9.3	
Input Data : N/A				N/A

Fallbeispiel

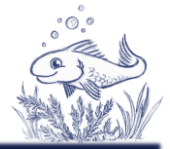


- Variableninhalte merken, RETAIN

```
RETAIN name purpose sas_version input_data output_data
       mac_called origin_by date;
```

line	name	purpose	sas_version	input_data
/*========== ========== ==========				
Program Name : varlist.sas →	varlist.sas			
Purpose : Returns a string containing a space separated	varlist.sas	...		
list of variables in a dataset	varlist.sas	...		
SAS Version : SAS 9.3	varlist.sas	...	SAS 9.3	
Input Data : N/A	varlist.sas	...	SAS 9.3	N/A

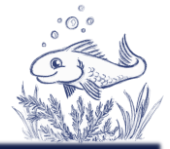
Fallbeispiel



- Informationen ermitteln, INDEX & SUBSTR, (RXMATCH)
  - Ermitteln des Schlüsselwortes in der Zeile (Program Name / Purpose / …)
  - Inhalt ist der Wert nach „:“

line
Program Name : varlist.sas
Purpose : Returns a string containing a space separated
list of variables in a dataset
SAS Version : SAS 9.3
Input Data : N/A

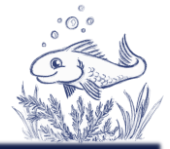
Fallbeispiel



- Informationen ermitteln, INDEX & SUBSTR, (RXMATCH)

```
IF       INDEX(line,"Program Name") > 0
   THEN name = SUBSTR(line,INDEX(line,":"));
ELSE IF INDEX(line,"Purpose") > 0
   THEN purpose = SUBSTR(line,INDEX(line,":"));
ELSE IF INDEX(line,"SAS Version") > 0
   THEN sas_version = SUBSTR(line,INDEX(line,":"));
...
```

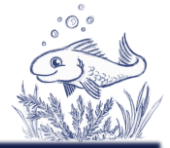
Fallbeispiel



- Ausgabe & Einlesen stoppen, OUTPUT & STOP

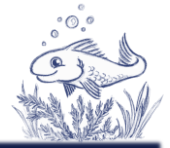
```
IF _N_ > 5 AND
INDEX(line,"====================") > 0
THEN DO;
     OUTPUT;
     STOP;
END;
```

Fallbeispiel



Variable	Value
Macro Name	optsave.sas
Purpose	Saves current options settings to a SAS
SAS Version	SAS 9.1.3
Input Data	None
Output Data	SAS options dataset
Macros Called	parmv
Originally Written By	Scott Bass
Date	14MAY2010
Program Version	1.0

Fallbeispiel

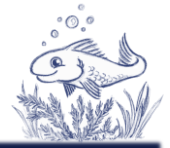


Variable	Value
Macro Name	optsave.sas
Purpose	Saves current options settings to a SAS
SAS Version	SAS 9.1.3

```
/*=================================================================
Program Name                : optsave.sas
Purpose                     : Saves current options settings to a SAS
                              dataset
SAS Version                 : SAS 9.1.3
...
```

Program Version	1.0

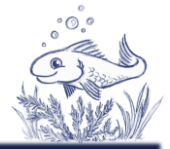
Fallbeispiel



- Mehrzeilen einlesen

line
Program Name : varlist.sas
Purpose : Returns a string containing a space separated
list of variables in a dataset
SAS Version : SAS 9.3
Input Data : N/A

Fallbeispiel

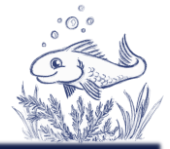


- Mehrzeilen einlesen
  - mit RETAIN „Parameter“ merken

```
DATA content;
     ...
     ATTRIB parameter FORMAT = $100. LABEL="Current Parameter";
     RETAIN parameter;

     IF INDEX(line,"Program Name") > 0
     THEN DO;
          name = STRIP(SUBSTR(line,INDEX(line,":") + 1));
          parameter = "Program Name";
     END;
     ...
RUN;
```

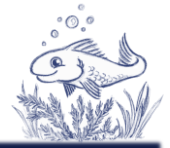
Fallbeispiel



- Mehrzeilen einlesen
  - mit RETAIN „Parameter“ merken
  - mit CATX Inhalte hinzufügen

```
ELSE DO;
   SELECT (STRIP(parameter));
      WHEN ("Program Name")  name = CATX(" ",name,line);
      WHEN ("Purpose")       purpose = CATX(" ",purpose,line);
      WHEN ("SAS Version")   sas_version = CATX(" ",sas_version,line);
      WHEN ("Input Data")    input_data = CATX(" ",input_data,line);
      ...
      OTHERWISE;
    END;
END;
```

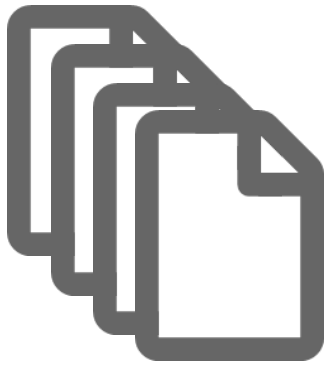
Fallbeispiel



- Werkzeugkasten
  - INFILE, _INFILE_, INPUT zum Einlesen
  - INDEX, reguläre Ausdrücke etc. zum Erkennen
  - STRIP, SUBSTR, SCAN, CATX zum Extrahieren

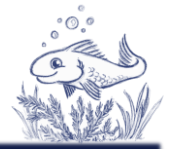
Fallbeispiel

- Mehrere Makros verarbeiten
  - SAS Macro
  - PIPE
  - „Wildcards“



SAS Makro

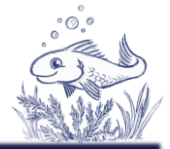
Fallbeispiel



- Mehrere Makros verarbeiten
  - „Wildcards“ *.sas

```
DATA allContent;
     LENGTH filename fname $128;
     INFILE "&path/Macro/*.sas" FILENAME=fname;
     INPUT;
     line = _INFILE_;
     filename = fname;
RUN;
```

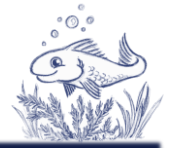
Fallbeispiel



- Mehrere Makros verarbeiten
  - EOV – Indikator für neue Datei (ab 2.)
  - Zeilennummer der aktuellen Datei

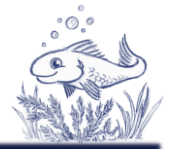
```
...
RETAIN lineno 0;
INFILE "&path/Macro/*.sas" FILENAME=fname
EOV=eov END=end;
…
lineno = lineno + 1;
IF eov OR _N_=1 THEN DO;
      PUT "Read new file: " filename;
      eov=0;
      lineno = 1;
END;
```

Fallbeispiel



- Mehrere Makros verarbeiten
  - Programmmodifikationen
    - statt STOP „ignore“-Logik
    - statt _N_, „lineno“ verwenden
    - CALL MISSING(<var1>, <var2>, …)

Fallbeispiel



- Mehrere Makros verarbeiten

name	purpose
age.sas	Determines a person's age in <units> based on a reference date
align_decimals.sas	Aligns the decimal points of numeric or character variables, optionally adding additional bracketing characters (usually parentheses).
attrib.sas	Generate attrib statements from a template dataset
check_if_empty.sas	Checks if the source dataset is empty.
compare.sas	PROC COMPARE either two datasets or two libraries
create_datetime_range.sas	Create min and max datetime values for a given period
create_directory.sas	Creates a directory using the dlcreatedir option, without requiring the ALLOWXCMD option to be active.

Fallbeispiel

Informationen Extrahieren

Dateien Modifizieren

Fallbeispiel

- Header anpassen

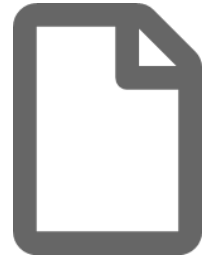
```
%*************************************************************************;
%* Project                : Tools Macro Package
%* Program name           : <name>
%* Author                 : <author>
%* Date created           : <date>
%* Purpose                : <purpose>
%* Origin                 : https://github.com/scottbass/SAS
%*************************************************************************;
```

Dateien Modifizieren

Fallbeispiel

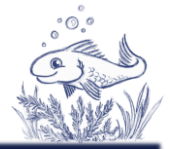
- Header anpassen
  - Datensatz lesen
  - Datei lesen
  - neue Datei schreiben

name	purpose	origin_by	date	mac_version
age.sas	Determines a person's age in <units> based on a reference date	Scott Bass	30MAY2006	1.0



SAS Makro

Fallbeispiel



- Header anpassen
  - Datensatz & Datei einlesen

```
DATA process;
     SET ageInfo;
     INFILE "&path/Macro/age.sas“;
     INPUT;
     line = _INFILE_;
RUN;
```

Fallbeispiel

- Header anpassen
  - Datensatz & Datei einlesen
    - Nur 1 Zeile

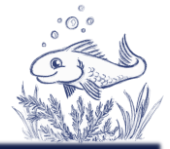
Beob.	name	purpose	origin_by	date	line
1	age.sas	Determines a person's age in <units> based on a reference date	Scott Bass	30MAY2006	/*========== ========== ==========

Fallbeispiel

- Header anpassen
  - Datensatz & Datei einlesen
    - Nur 1 Zeile

Datensatz					Datei
Beob.	**name**	**purpose**	**origin_by**	**date**	**Line**
1	age.sas	Determines a person's age in <units> based on a reference date	Scott Bass	30MAY2006	/*================== ========== ==========
					Program Name : age.sas
					Purpose : Determines a person's age in <units>
					…

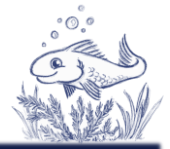
Fallbeispiel



- Header anpassen
  - Datensatz (300 Duplikate) & Datei einlesen
    - 99 Zeilen (Datei-Zeilen)

Datensatz					Datei
Beob.	**name**	**purpose**	**origin_by**	**date**	**line**
…	…	…	…	…	…
97	age.sas	Determines a person's …	Scott Bass	30MAY2006	%mend;
98	age.sas	Determines a person's …	Scott Bass	30MAY2006	
99	age.sas	Determines a person's …	Scott Bass	30MAY2006	/******* END OF FILE *******/

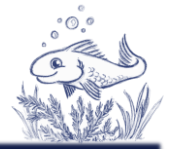
Fallbeispiel



- Header anpassen
  - Datensatz & Datei sollten gleichviele Zeilen haben
  - Besser: DO LOOP verwenden

```
DATA content;
     SET ageInfo;
     ATTRIB line FORMAT = $255. LABEL="File line";
     INFILE "&path/Macro/age.sas" END=eof;
     DO UNTIL (eof);
          INPUT;
          line = _INFILE_;
          OUTPUT;
     END;
RUN;
```

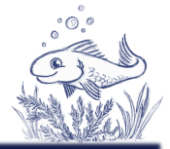
Fallbeispiel



- Header anpassen

Datensatz					Datei
Beob.	**name**	**purpose**	**origin_by**	**date**	**line**
1	age.sas	Determines a person's …	Scott Bass	30MAY2006	/*==========…=
2	age.sas	Determines a person's …	Scott Bass	30MAY2006	Program Name : age.sas
3	age.sas	Determines a person's …	Scott Bass	30MAY2006	Purpose : Determines a person's age in <units>
4	age.sas	Determines a person's …	Scott Bass	30MAY2006	based on a reference date
…	…	…	…	…	…

Fallbeispiel



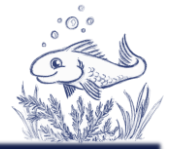
- Datei schreiben
  - FILE & PUT

```
FILE "&path/Macro/Mod/age.sas";
PUT line;
```

```
31 data _null_;
32    dob = "25Dec60"d;
33    end = "25Dec05"d;
34
35    age1 = %age(dob);
36    age2 = %age(dob,end);
37    age3 = %age(dob,end,units=mc
38    age4 = %age(dob,units=day);
39
40    put (dob end) (=date7. +1) (
41 run;
```

```
31 data _null_;
32 dob = "25Dec60"d;
33 end = "25Dec05"d;
34
35 age1 = %age(dob);
36 age2 = %age(dob,end);
37 age3 = %age(dob,end,units=montl
38 age4 = %age(dob,units=day);
39
40 put (dob end) (=date7. +1) (ag
41 run;
```

Fallbeispiel



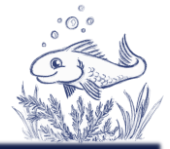
- Datei schreiben
  - FILE & PUT & @ & NOTSPACE

```
FILE "&path/Macro/Mod/age.sas";
PUT @(NOTSPACE(line)) line;
```

```
31 data _null_;
32    dob = "25Dec60"d;
33    end = "25Dec05"d;
34
35    age1 = %age(dob);
36    age2 = %age(dob,end);
37    age3 = %age(dob,end,units=mo
38    age4 = %age(dob,units=day);
39
40    put (dob end) (=date7. +1) (
41 run;
```

```
31 data _null_;
32    dob = "25Dec60"d;
33    end = "25Dec05"d;
34
35    age1 = %age(dob);
36    age2 = %age(dob,end);
37    age3 = %age(dob,end,units=mo
38    age4 = %age(dob,units=day);
39
40    put (dob end) (=date7. +1) (
41 run;
```

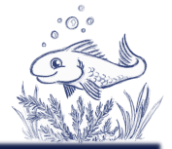
Fallbeispiel



- Datei schreiben

```
DATA _NULL_;
      SET ageInfo;
      ATTRIB line FORMAT = $255. LABEL="File line";
      INFILE "&path/Macro/age.sas" END=eof;
      FILE "&path/Macro/Mod/age.sas";
      ...
```

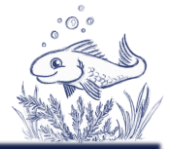
Fallbeispiel



- Datei schreiben
  - PUT – neuen Header einfügen

```
...
* Put new header;
PUT "%*****************************************************;";
PUT "%* Project        : Tools Macro Package";
PUT "%* Program name   : " name;
PUT "%* Author         : " origin_by;
PUT "%* Date created   : " date;
PUT "%* Purpose        : " purpose;
PUT "%* Origin         : https://github.com/scottbass/SAS";
PUT "%*****************************************************;";
```

Fallbeispiel



- Datei schreiben
  - PUT – neuen Header einfügen
  - Don‘t PUT – den alten Header nicht schreiben

```
. . .
inHead = 1;
DO UNTIL (eof);
   INPUT;
   line = _INFILE_;
   IF inHead = 0 THEN PUT @(NOTSPACE(line)) line;
   ELSE DO;
            IF INDEX(line,"=======") > 0 AND INDEX(line,"/*") = 0
            THEN DO;   inHead = 0; PUT "/*==================================";
            END;
END;END;RUN;
```

Fallbeispiel

- Erweiterungen / Optimierungen
  - Mehrere Dateien bearbeiten
    - Empfehlung SAS Makro
  - Mehrzeilige Inhalte

- Zusammenfassung:
  - Dateien Lesen, Ändern, Schreiben mit SAS?

Fallbeispiel

- Modifikationen „vor“ einer Zeile?

Beob.	name	line	Indicator
…	…	…	
96	age.sas		
97	age.sas	%mend;	
98	age.sas		
99	age.sas	/******* END OF …	davor

Beob.	name	line	Indicator
99	age.sas	/******* END OF …	davor
98	age.sas		hier
97	age.sas	%mend;	
96	age.sas		
…	…	…	

Beob.	name	line	Indicator
…	…	…	
96	age.sas		
97	age.sas	%mend;	
98	age.sas		hier
99	age.sas	/******* END OF …	davor

Agenda

- Einleitung
- Fallbeispiel
- **Einsatzgebiete**

Einsatzgebiete

- Informationssammlung (Header)
  - Autor
  - Grund für Programm/Makro
  - SAS Versionen
  - Validierungsstatus
  - Beispielaufrufe

Einsatzgebiete

- Informationssammlung (Inhalt)
  - Verwendung von Pfaden, SAS Datensätze
  - Makro Verwendung
  - Programmcodezeilen
  - Kommentare
  - Spezifische Sektionen für Dokumentationen

Einsatzgebiete

- Automatisierte Änderungen
  - Header & Pfade automatisch anpassen
    - Kopieren von Programmen & Templates
  - Migration von SAS Version
  - Update von Makro Systemen mit geänderten Parametern
  - Code formatieren (andere Tools)
  - Neue Parameter in vielen Makros einführen
    - Log Ausgaben oder Optionen aktivieren / deaktivieren
  - Generierte Parameterprüfungen hinzufügen
  - Log Ausgaben vereinheitlichen

Einsatzgebiete

- Weitere Ausgabeformate
  - Ausgaben (HTML, RTF, ASCII)
    - Ersetze alle Zahlen durch „#“
  - Java Script in HTML5 einfügen
  - XML schreiben / ändern (z.b. define.xml)
  - Komplexe JSON erstellen
  - RDF Triples erstellen (für Linked Data)
  - ...

Zusammenfassung

- Lesen, Schreiben, Ändern
- Automatisierungsmöglichkeiten

Vielen Dank

Katja Glass Consulting

Info@glacon.eu