Modularisierung und Kapselung von SAS-Makros in einem Auswertungspaket für die Evaluation von Klausurergebnissen

Michael Volk, Beate Einsiedler, Rainer Muche
Institut für Epidemiologie und Medizinische Biometrie, Universität Ulm
Schwabstraße 13
89075 Ulm
beate.einsiedler@uni-ulm.de

Zusammenfassung

Bei der Routineauswertung von immer wieder zu erhebenden Daten ist es üblich, entsprechende Standardprogramme als SAS-Makros mit der Übergabe von Standardparametern zu erstellen. Dabei stellt sich bei der Überarbeitung und Erweiterung dieser Makros oft das Problem, dass die Struktur der Parameterübergabe sowie der Beziehungen der entsprechenden Makrovariablen nur schwer nachzuvollziehen ist. Daraus ergibt sich oft ein großer Überarbeitungsaufwand. Für eine Überarbeitung des 2009 auf der KSFE in Halle vorgestellten SAS-Programms zur (teil-) automatisierten Analyse von Klausurergebnissen wurde die Programmierung so umgestellt, dass eine Überarbeitung und Erweiterung in Zukunft viel einfacher möglich ist.

Die dafür eingesetzten Methoden sind Modularisierung und Kapselung. Mit der Hilfe der SAS Macro Language ist es möglich ein Programm in einzelne Module, d.h. Makros, aufzuteilen. Dies hat den Vorteil einer besseren Übersichtlichkeit. Da jedes Makro nur ein Teil des gesamten Programms ist, können hier die einzelnen Schritte besser nachvollzogen werden. Außerdem können Fehler so schneller gefunden werden, da bekannt ist, in welchem Modul er zu suchen ist. Ein wichtiges Werkzeug dazu ist die Definition von lokalen und globalen Makrovariablen (MVAR): in jedem Makro können lokale MVARs definiert werden. Allerdings ist hier sehr darauf zu achten, wann welche MVAR in welchem Makro geändert wird. Um sicher zu sein, dass keine Probleme auftreten, besteht die Möglichkeit alle benötigten Werte mittels Makro-Übergabeparameter in einer gleichnamigen lokalen MVAR zu übergeben, somit kann auf den Wert im aufrufenden Makro nicht mehr zugegriffen werden. Dieses Vorgehen wird auch Kapselung genannt: Jedes Makro hat nur Zugriff auf die eigenen MVARs.Mit Zugriffen auf Datasets ist es etwas anders: Hier gibt es keine Hierarchie wie bei den MVARs, jedes Makro kann auf jedes Dataset in jeder Library zugreifen und die Daten dort manipulieren. Hier hilft die Modularisierung den Überblick zu behalten. Wenn ein Makro abgeschlossen wird, werden z.B. die Ergebnisse in ein existierendes permanentes Dataset gespeichert und anschließend alle während der Bearbeitung im Makro erzeugten Datasets gelöscht. Eine Ausnahme können festgelegte kleine Hilfsmakros sein, die Zwischenwerte in einem temporären Dataset für die weitere Ausarbeitung hinterlegen. In der Phase der Kenngrößenberechnung in unserem Beispiel wird jede Kenngröße von einem eigenen Makro berechnet, was den Vorteil der Austauschbarkeit mit sich bringt: wenn eine Kenngröße nicht mehr benötigt wird bzw. wenn es eine bessere Kenngröße für eine bestimmte Auswertung gibt, kann diese als neues Modul einfach ausgetauscht werden. Es müssen lediglich geringe Teile des Hauptmakros und das Modul für die Ausgabe angepasst werden.

Schlüsselwörter: SAS Makro, Makrovariable, Kapselung, Modularisierung

1 Einleitung

Bei der Routineauswertung von immer wieder zu erhebenden Daten ist es üblich, entsprechende Standardprogramme zu erstellen. In SAS werden diese hauptsächlich als SAS-Makros mit der Übergabe von Standardparametern erstellt. Dabei stellt sich bei der Überarbeitung und Erweiterung dieser Makros oft das Problem, dass die Struktur der Parameterübergabe sowie der Beziehungen der entsprechenden Makrovariablen (MVAR) nur schwer nachzuvollziehen sind. Daraus ergibt sich oft ein immenser Überarbeitungsaufwand. Für eine Überarbeitung des 2009 auf der KSFE in Halle vorgestellten SAS-Programms zur (teil-) automatisierten Analyse von Klausurergebnissen [2] wurde im Rahmen einer Bachelorarbeit [4] die Programmierung so umgestellt, dass eine Überarbeitung und Erweiterung in Zukunft viel einfacher möglich ist. Dabei sind die Programmierprinzipien der Modularisierung und Kapselung zum Einsatz gekommen.

2 Fragestellung

In diesem Kapitel werden Informationen über den Hintergrund der Auswertung von Prüfungen und die entsprechenden Kenngrößen dargestellt.

2.1 Notwendigkeit der quantitativen Evaluation von Prüfungen

Seit Änderung der Ärztlichen Approbationsordnung im Jahr 2003 für das Humanmedizinstudium werden vermehrt benotete Prüfungen an den Universitäten durchgeführt, die die Leistungen der Studierenden bescheinigen (sollen). Diese Prüfungen sollen natürlich fair, objektiv, reliabel und valide sein, um die einzelnen Studierenden gerecht zu bewerten. Zur Untersuchung dieser Eigenschaften können verschiedene statistische Auswertungen herangezogen werden. Es gibt dazu entsprechende Vorschläge, die diese Auswertungen festlegen (s. nächster Abschnitt).
Im Studierendenunterricht werden heute einige Evaluationen durchgeführt. Schon lange gibt es die Evaluation der Lehrveranstaltungen als Akzeptanzevaluation. Die Ergebnisse geben Aufschluss über den Lehrerfolg sowie die Umsetzung der Lehrveranstaltung. Zunehmend werden aber auch die Prüfungen evaluiert, um einen genaueren Einblick in die Details der Klausuren zu bekommen. Wie komplex waren die Fragen, welche Fehler wurden wie oft gemacht, wie ist der Zusammenhang zwischen Lehre und Prüfung? Zur Untersuchung dieser Eigenschaften der Prüfungen im Pflichtunterricht Biometrie haben wir diese vorgeschlagenen Auswertungen mit SAS programmiert und 2009 auf der KSFE dargestellt [2].

2.2 Statistische Kenngrößen zur Beurteilung von Klausurergebnissen

Es gibt einige Vorschläge zur Untersuchung von Prüfungsergebnissen, um deren Eigenschaften insgesamt oder aufgabenspezifisch untersuchen zu können. Um die Prüfungen in der Humanmedizin zu unterstützen und vergleichbar zu machen, wurde in Baden-Württemberg das „Kompetenzzentrum für Prüfungen in der Medizin“ an der Medizini-

schen Fakultät der Universität Heidelberg eingerichtet[1]. Ein Vorschlag für entsprechende Auswertungen wurde von den Mitarbeitern des Kompetenzzentrums als Vorschlag veröffentlicht [1], den wir für unser Programm übernommen haben.

Folgende Aspekte werden ausgewertet:

Gesamtergebnis:

- Darstellung der Gesamtpunktzahlen:

Hier wird eine tabellarische Auflistung der Häufigkeiten (inklusive Histogramm), ergänzt durch Bestehensgrenzen und Notenschlüssel, dargestellt.

Aufgabenschwierigkeitsgrad

- **Aufgabenschwierigkeit** bzw. Itemschwierigkeit sowie
  Normierte Aufgabenschwierigkeit:

Diese sind definiert als relativer (prozentualer) Anteil der maximal erreichbaren Punktzahl der Aufgabe (Mittelwert der erreichten Punkte / Maximum der erreichbaren Punkte dieser Aufgabe).

Trennschärfe

Für korrigierte Maße werden bei jeder Aufgabe die Summe aller anderen Aufgaben der Prüfung als Gesamtpunktzahl verwendet.

- **Korrigierter Diskriminationsindex** = D:

Nach Einteilung der Prüflinge bzgl. der Gesamtpunktzahl in drei Gruppen ⇒ gut / mittel / schlecht wird anhand der Terzile (unteres und oberes 33 %-Perzentil) die Differenz der normierten mittleren Schwierigkeit der Gruppe der "Guten" und der Gruppe der "Schlechten" Studierenden miteinander verglichen.

- **Korrigierte Trennschärfe über Korrelationsmaße**:

Die Korrelationskoeffizienten (Pearson-Bravais, Spearman, Kendall), der in der Aufgabe erreichten Punktzahlen mit der Gesamtpunktzahl in allen anderen Aufgaben, werden als Maß für die Trennschärfe herangezogen.

Einzelantwort- oder Distraktorenanalyse bei MC-Aufgaben

- Häufigkeitsanalyse der Antworten:

Alle Antwortalternativen, ihre absoluten und relativen Häufigkeiten werden angegeben.

- Trennschärfenanalyse von Antworten/Distraktoren:

Erstellung einer Indikator-Variablen für jede Antwortmöglichkeit und Berechnung der Trennschärfemaße mit dieser Indikatorvariablen.

Reliabilität (Zuverlässigkeit oder Reproduzierbarkeit)

Die Reliabilität der gesamten Prüfung wird durch Cronbachs α gemessen, wobei Cronbachs α eine untere Grenze für die Reliabilität darstellt.

Für verschiedene Zwecke werden allerdings weitere Analysen notwendig sein und/oder es werden in Zukunft unterschiedlich umfangreiche Auswertung und Berichte verlangt.

[1] http://www.medizinische-fakultaet-hd.uni-heidelberg.de/Kompetenzzentrum-fuer-Pruefungen-KomP-Med.3081.0.html

Dafür war unser programmtechnischer Ansatz aus dem Jahr 2009 [2] nicht flexibel genug. Deshalb haben wir eine Umsetzung der Programmierung mittels Modulen und Kapselung vorgenommen, die im Folgenden dargestellt wird.

3 Modularisierung und Kapselung

In diesem Abschnitt werden die Prinzipien der Modularisierung und Kapselung beschrieben. Im folgenden Kapitel 4 werden anhand von Beispielen aus unserem Makropaket die Prinzipien dargestellt. Dabei werden Elemente der Modularisierung durchgehend unterstrichen und Elemente der Kapselung gestrichelt markiert.

Modularisierung
Die systematische Aufteilung von komplexen Aufgaben in logische Teilblöcke wird als **Modul** bezeichnet. Die Aufteilung in Module ist in SAS mit der **SAS Macro Language** möglich [3]. Die wesentlichen Vorteile bei diesem Vorgehen sind, dass jedes Modul, d.h. jedes SAS-Makro, unabhängig arbeitet. Dadurch wird eine bessere Übersichtlichkeit über das Programm sowie eine bessere Nachvollziehbarkeit der Programmschritte erreicht. Bei der Programmierung wird darüber hinaus eine Fehlersuche erleichtert.

Kapselung
Bei der objektorientierten Programmierung mit Modulen sollen die Module, hier SAS-Makros, den internen **Zustand anderer Module nicht in unerwarteter Weise lesen oder ändern können**. Auf die anderen Module soll **nur über definierte Schnittstellen** (Übergabeparameter) zugegriffen werden können. Diese Vorgehensweise wird als Kapselung (der Programmmodule) bezeichnet [5].
Bei der Verwendung von SAS-Makros werden üblicherweise Makrovariablen (MVARs) als Übergabeparameter verwendet. Innerhalb des aufgerufenen Makros werden die jeweiligen MVARs, z.B. mit Angaben von Dateipfad, Dataset und Dataset-Variablen, zur Steuerung bzw. bei der Bearbeitung eingesetzt. Die MVARs, sowohl Übergabeparameter als auch neue MVARs sind im Makro ‚lokal' definiert, d.h. sie sind speziell im definierten Makro gültig. Jedes Makro hat eine eigene ‚local symbol table' und die MVARs stehen nur während der Bearbeitung des jeweiligen Makros zur Verfügung. Auf diese Weise ist nach Abschluss des jeweiligen Makro-Programmlaufs die Kapselung gewährleistet, da die Übergabeparameter eines Makros keinen Einfluss auf weitere Makros des SAS-Programms haben. Grundsätzlich dürfen allerdings SAS-Makros beliebig oft ineinander verschachtelt werden. Hier kann es zu Problemen bei der Kapselung kommen.

Bei ineinander verschachtelten SAS-Makros können in jedem Makro lokale MVARs definiert werden. Wenn z.B. Makro A das Makro B aufruft, hat Makro B vollen Zugriff auf alle lokalen MVARs von Makro A. Die Bearbeitung im Makro A ist bei Programmlauf von Makro B noch nicht beendet. Abbildung 1 zeigt das Vorgehen des Makroprozessors bei der Suche der MVAR ‚&DS' (aus [4]). Der Vorteil dieser üblichen Programmierweise ist, dass vorbereitete lokale MVAR im untergeordneten Makro ver-

fügbar sind. Der Nachteil ist allerdings, dass Änderungen der lokalen MVAR in einem Makro zu Fehlern in den anderen Makros führen können!

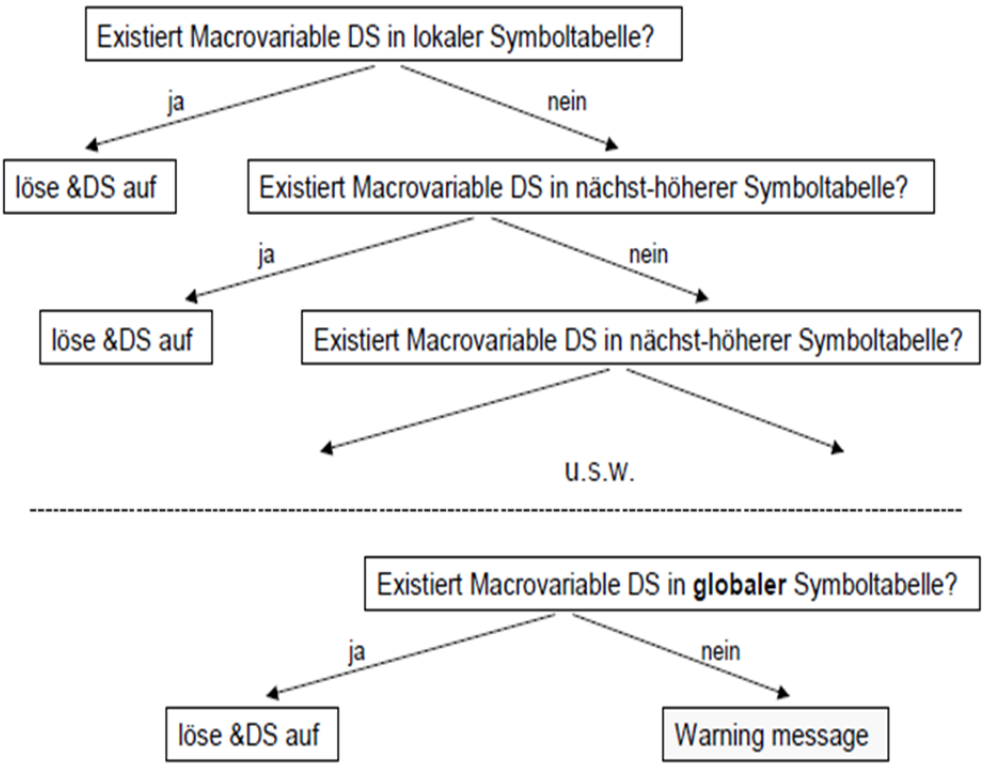

Abbildung 1: Gültigkeitsbereiche von Makrovariablen [4]

Die Lösung der Kapselung:
Die benötigten MVARs werden mittels **Übergabeparameter** in gleichnamige lokale MVARs übergeben. Die gleichnamige MVAR wird auch im untergeordneten Makro lokal definiert, dadurch kann die MVAR mit gleichem Namen im übergeordneten Makro nicht mehr geändert werden (Übergabe mit Call by Value) [3]. Somit ist das **Makro „gekapselt", d.h. nur definierter Zugriff ist möglich!**

Als Schnittstellen zwischen den Modulen sind auch die notwendigen Datasets anzusehen. Bei Zugriffen auf Datasets gibt es keine Hierarchie wie bei den MVARs, jedes Makro kann auf jedes Dataset in jeder Library zugreifen und die Daten dort manipulieren. Hier hilft die Modularisierung, den Überblick zu behalten. Wenn ein Makro abgeschlossen wird, werden z.B. die Ergebnisse der statistischen Kenngrößen in ein existierendes permanentes Dataset gespeichert und anschließend alle selbst erzeugten Datasets gelöscht [4].

4 Umsetzung im Makro-Paket

Im Folgenden werden zwei Makros aus dem Evaluationspaket dargestellt: einmal das übergeordnete Makro mit Aufruf der Module und ein Beispiel-Modul-Programm, um die Kapselungselemente aufzeigen zu können. Durch die verschiedenen Varianten des

Unterstrichs werden die Elemente der Modularisierung und Kapselung in den Makros dargestellt.

Das Gesamtpaket zur Auswertung der Prüfungsevaluation umfasst insgesamt 9 Makros, die in der folgenden Tabelle kurz beschrieben sind.

Tabelle 1: Übersicht über die SAS-Makros des Evaluationspakets für Prüfungen

Programmaufbau	
Makros/Module	**Funktion**
%controller	Hauptmakro: Steuerung der Makros [Programm-Beispiel 1 in Abb.2]
%aufbereitung	Datenaufbereitung
%plausibility	Plausibilitätscheck
Kenngrößen:	
%aufgabenschwierigkeit	Normierte Aufgabenschwierigkeit [Programm-Beispiel 1 in Abb.2]
%diskriminationsindex	Korrigierter Diskriminationsindex
%trennschaerfe	Korrigierte Trennschärfemaße
%cronbachs_alpha	Reliabilität
%multiple_choice	Distraktorenanalyse: Analyse der Antworten jeder MC-Aufgabe: Häufigkeiten, Aufgabenschwierigkeit, Diskriminationsindex und Trennschärfe
%output	Ausgabe der Ergebnisse von Gesamt-, Aufgaben- und MC-Fragen

Makro %controller [Programm-Beispiel 1 in Abb.2]:
Dieses SAS-Makro steuert übergeordnet den Programmablauf von Datenaufbereitung, Plausibilitätsprüfung, Kenngrößen-Berechnung bis zur Ausgabe der Ergebnisse. Dabei werden die Aufgaben streng in Module, d.h. Makros gegliedert. Beispielhaft wird jede Kenngröße separat in einem eigenen Modul berechnet. Treten Fehler bei der Berechnung einer Kenngröße auf, so ist durch die Modularisierung sofort ersichtlich, wo er zu suchen ist. Ist eine neue Kenngröße erforderlich, so wird die Berechnung in einem neuen Makro programmiert und anschließend im Makro %controller eingefügt.

```
%MACRO controller;
%local i;
%do i=1 %to &count_tests;
    /*------Phase 1: Datenaufbereitung--------------*/
     %aufbereitung(test&i.);
```

```
    /*------Phase 2: Plausibilitätscheck------------*/
     %plausibility(test&i.);
    /*------Phase 3: Kenngrößen berechnen-----------*/
     %if &plausibility_complete=0 %then
     %do;
          %put Keine Fehler in den Daten.;
          /*Kenngrößen für alle Aufgabentypen*/
          %aufgabenschwierigkeit(test&i.);
          %diskriminationsindex(test&i.);
          %trennschaerfe(test&i.);
          %cronbachs_alpha(test&i.);
          /*Distraktorenanalyse der MC Aufgaben*/
          %multiple_choice(test&i.);
          /*------Phase 4: Ausgabe ---------------*/
          %output(test&i.);
     %end;
...
%MEND controller;
```

Abbildung 2: Programm-Beispiel 1 (Makro %controller)

Makro %aufgabenschwierigkeit:
Im Programm-Beispiel 2 (Abb. 3) wird die Berechnung der Aufgabenschwierigkeit (s. Kapitel 2.2) dargestellt. Als Übergabeparameter wird der Pfad (MVAR &lib) für die zu bearbeitende Datei angegeben und die MVAR &mc wird für die Steuerung zwischen Aufgabenanalyse und Distraktorenanalyse eingesetzt. Erforderliche Angaben für die Berechnung der Kenngröße werden als lokale MVARs mit PROC SQL bereitgestellt und sind nach Abschluss des Makros nicht mehr verfügbar. So wird für jede Prüfungsaufgabe (&mc=0) die Aufgabenschwierigkeit berechnet und das Ergebnis im Data &lib.gesamt gespeichert. Für die Berechnung der Kenngröße bei einer MC-Aufgabe wird für die MVAR &mc die Anzahl der Antwortmöglichkeiten angegeben und der zweite Programmteil des Makros bearbeitet. Für jede mögliche Antwort einer MC-Aufgabe wird die Aufgabenschwierigkeit bestimmt und das Ergebnis im zugehörigen Data (&lib.mc_aufgabe&k) gespeichert.

Die Kenngrößenberechnung jeder Prüfungsaufgabe wird vom Makro %controller und jeder MC-Aufgabe (Distraktorenanalyse) vom Makro %multiple_choice aufgerufen. Dabei sind die Makros %multiple_choice und %aufgabenschwierigkeit ineinander verschachtelt. Im Makro %multiple_choice wird für den Pfad ebenfalls die MVAR &lib verwendet und im Aufruf des untergeordneten Makros %aufgabenschwierigkeit gleichnamig mit &lib übernommen (siehe Kapitel 3 Kapselung). Die Makros zur Berechnung weiterer Kenngrößen sind ebenfalls in zwei Programmteile getrennt für Aufgabenanalyse und MC-Antworten untergliedert.

Im Programm-Beispiel 2 wird die Kapselung erreicht durch:

- Verwendung weniger festgelegter Übergabeparameter als lokale MVARs.
- Alle MVARs im Modul zur Berechnung werden lokal definiert und sind nach Abschluss des Makros nicht mehr definiert und können somit nicht mehr verändert werden.
- Bei verschachtelten Makros wird die Verwendung gleichnamiger MVARs bei der Datenübergabe eingesetzt.

```
%MACRO aufgabenschwierigkeit (lib, mc=0);
%local k students_count answer_count;

/* Aufgabenschwierigkeit der Aufgaben */
%if &mc=0 %then
%do;
     proc sql noprint;
     select count(*) into: students_count trimmed
     from &lib..aufgabenpunkte;
     quit;
     data &lib..gesamt;  set &lib..gesamt;
     P_norm = round(MEAN(of student1-student&students_count)
               /maximum, .001);
     keep variable fach semester maximum typ anz_antwort_mc
           ergebnis_mc p_norm;
     run;
%end;

/* Aufgabenschwierigkeit der MC-Aufgaben */
%else
%do;
     %do k=1 %to &mc;
          proc sql noprint;
          select count(%scan(&keep_cols_mc,&k))
          into : answer_count trimmed
          from mc_trans
          where %scan(&keep_cols_mc,&k) ^= ' ';
          quit;

          data &lib..mc_aufgabe&k;
          set &lib..mc_aufgabe&k;
          P_norm= round(freq/&answer_count., .001);
          run;
     %end;
%end;
%MEND aufgabenschwierigkeit;
```

Abbildung 3: Programm-Beispiel 2 (Makro %aufgabenschwierigkeit)

5 Zusammenfassung und Ausblick

Modularisierung und Kapselung sind bei Verwendung eines systematischen Programm-Aufbaus sowie definierten Schnittstellen bei der Übergabe von Parametern, in der SAS-

Makro-Programmierung gut umsetzbar, wie das Beispiel der Prüfung von Klausuren zeigt. Jedes Modul ist unabhängig und kann bei gleicher Aufgabenstellung ggf. mehrfach genutzt werden.
Da jedes Makro unabhängig arbeitet, sind alle Funktionen von Daten- und MVARs-Bereitstellung, über Berechnung bis zur Ergebnisspeicherung direkt in jedem Makro durchzuführen. In manchen Fällen, wie z.B. bei Bootstrap-Verfahren, mag dies nachteilig sein, da sich dadurch die Programmlaufzeiten verlängern.
Großer Vorteil dieser Methoden bleibt die Übersichtlichkeit, einfache Wartung und Fehlersuche der Programme und dies führt zu Arbeitserleichterung und nicht zuletzt Arbeits(-zeit-)ersparnis bei der weiteren Programmpflege. Man kann eine neue Kenngröße einfach als Makro in das Programm einfügen bzw. eine neue Reihenfolge festlegen. Besonders bei Aufgabenstellungen mit systematischer Struktur, die nicht abhängig von der Laufzeit sind, ist die modulare Programmierung mit Kapselung sehr empfehlenswert.

Literatur

[1] Möltner A, Schnellberg D, Jünger J: Grundlegende quantitative Analysen medizinischer Prüfungen. GMS Z. Med. Ausbild. 2006; 23(3): Doc53

[2] Muche R, Janz B, Einsiedler B: Quantitative Analysen medizinischer Prüfungen mittels eines (teil-) automatisierten SAS-Programms. In: Spilke J, Becker C, Haerting J, Schumacher E (Hrsg.): KSFE 2009 – Proceedings der 13. Konferenz für SAS-Anwender in Forschung und Entwicklung, Shaker Verlag, Aachen, 2009, 187-194

[3] Pfister G: Fortgeschrittene Makro-Programmierung. Tutoriumsunterlagen, KSFE 2013, 2013

[4] Volk M: Automatisierte Analyse medizinischer Prüfungen mit SAS. BSc.-Arbeit, Hochschule Ulm; 2013

[5] Wikipedia: Datenkapselung (aufgerufen am 4.4.2014)