SAS und LaTeX: Erste Ansätze für eine gute „Zusammenarbeit“

Heiko Zimmermann
Institute of Public Health
Im Neuenheimer Feld 365
Heidelberg
h.zimmermann@uni-heidelberg.de

Andreas Deckert
Institute of Public Health
Im Neuenheimer Feld 365
Heidelberg
a.deckert@uni-heidelberg.de

Zusammenfassung

LaTeX ist ein mächtiges Werkzeug zur Erstellung von Dokumenten mit Hilfe eines auf Makrobefehlen aufbauenden Textsatzsystems. Die Stärke von LaTeX zeigt sich vor allem bei komplexen Dokumenten. Seit Version 9 bietet SAS nun auch eine Schnittstelle zu LaTeX: Grafiken können mit Hilfe von ODS Graphics/Markup erstellt und in LaTeX Dokumente ohne viel Aufwand integriert werden. In dem vorliegenden Dokument werden notwendige Materialien und Schritte zur automatisierten Einbindung von Grafiken direkt aus SAS in LaTeX dargestellt. Des Weiteren wird ein kurzer Überblick über ein kleines Makro gegeben, das unter anderem diese Funktionalitäten zur Verfügung stellt.

Schlüsselwörter: LaTeX, ODS Graphics, ODS Markup

1 Motivation

SAS ermöglicht Grafiken im PostScript Format in LaTeX Dokumente automatisiert einbinden zu lassen. Mit Hilfe von eindeutigen "Platzhaltern" - sog. Flags - im LaTeX Dokument können diese dann durch eine entsprechende Grafik ersetzt werden. Auch zeitversetztes Einbinden ist möglich: Beispielsweise, wenn Daten zu Erstellung einer Grafik noch nicht vorhanden sind. Der Platzhalter im LaTeX Dokument sorgt für die spätere Zuordnung. Durch dieses Prinzip können auch 1 bis n Flags ersetzt werden. Vorteil für LaTeX Nutzer: Grundsätzlich müssen am LaTeX Code keine Änderungen vorgenommen werden. Layout und Stiländerungen an der/den Grafik(en) sind auf SAS Seite und nicht im LaTeX Code selbst durchzuführen. Durch die Erstellung von SAS Makros können die Schritte zwischen LaTeX und SAS automatisiert werden [1].

2 Hintergrund

LaTeX [2] ist ein frei zugängliches Programmpaket, das die Benutzung des Textsatzprogramms TeX mit Hilfe von Makros vereinfacht und sich durch ein hochwertiges Satzsystem auszeichnet.

Durch die Bereitstellung einer großen Vielfalt an unterschiedlichen Paketen bietet es für jedes beliebige typografische Ziel die passenden Werkzeuge. Unter anderem sind Funk-

tionen zur Generierung für jede Art von Dokumenten sowie zur Strukturierung von Texten und Querreferenzierungen vorhanden. LaTeX arbeitet außerdem mit unterschiedlichen Formaten zusammen: Portable Document Format (PDF), PostScript oder DeVice Independent (DVI). Gerade im Bereich mittlerer und größerer technischer Dokumente [3], aber auch bei der Erstellung wissenschaftlicher Fachliteratur ist LaTeX weit verbreitet und sehr beliebt. In großen (wissenschaftlichen) Dokumenten zeigt es seine Stärken: bereits nach einer kurzen Einarbeitungsphase seitens des Anwenders muss dieser kaum noch Anpassungen/Änderungen am Layout und an der Formatierung vornehmen. Zusätzlich bietet LaTeX mit BibTeX ein ausgereiftes und starkes Tool für die Literaturverwaltung und die automatische Referenzierung von Quellen [4].

LaTeX ist ein mächtiges Werkzeug vor allem bei großen Dokumenten (s. Tabelle 1), weil Schritte zur Formatierung und zum Erstellen des Layouts erst bei der Kompilierung[1] des Codes ausgeführt werden. LaTeX arbeitet hierbei mit logischer Struktur nach dem Prinzip WYGIWYM (What you get is what you mean). Struktur-/Formatierbefehle werden im Fließtext integriert und erst am Ende auf den Text angewendet. Hierdurch ist dann das Erstellen von langen komplexen Texten in geringer Zeit leicht möglich. Allerdings ist das fertige Layout des Dokumentes nicht von Anfang an ersichtlich. Word hingegen arbeitet visuell, d.h. das finale Ergebnis ist von Anfang an auf dem Bildschirm zu sehen. Word eignet sich besonders für kurze, einfache Texte (Briefe, Deckblätter, …) und bringt in diesem Fall eine Zeitersparnis mit sich, wenn mit einem WYSIWYG (What you see is what you get) Editor wie Word gearbeitet wird. Bedeutet: Änderungen am Fließtext werden mit Menü- oder Kurzbefehlen vorgenommen.

Tabelle 1: Word vs. LaTeX

Word	*LaTeX*
Kurze/einfache Texte	Lange/komplexe Dokumente
WYSIWYG	WYGIWYM
visuell	logisch
Mit Menü- oder Kurzbefehlen visuell verändern	Struktur/Formatierbefehle im Fließtext

Im Gegensatz zu Latex als Open-Source-Projekt handelt es sich bei SAS um eine kommerzielle Software, die vom SAS Institute Inc. angeboten und weiterentwickelt wird (aktuelle Revision 9.2). Diese bietet den Nutzern unter anderem Möglichkeiten zur Dateneingabe, -abfrage und zum Datenmanagement sowie zum Verfassen von Berichten und Erstellen von Grafiken. Weiterhin stellt das Programm Prozeduren für statistische Analysen und Anwendungsentwicklungen zur Verfügung [5]. Seit Version 9 ermöglicht SAS zusätzlich die Erstellung von Grafiken im PostScript Format (.ps) mit einfachen Mitteln. Mit Hilfe von einfachem SAS Code lassen sich generierte .ps Grafiken automatisch in LaTeX Dokumente (und Präsentationen) einbinden.

[1] Kompilierung: Umwandlung von Programmcode in ausführbaren Code

3 Grundlagen

Für die Verwendung von LaTeX werden unterschiedliche Komponenten benötigt: Die MiKTeX Distribution ist ein Textsatzsystem mit eingebauter Makrosprache, das die notwendigen Pakete und Programme zur Arbeit mit LaTeX unter Windows zur Verfügung stellt. Durch das Schnittstellenprinzip lässt sich das Basisprogramm durch jedwede Funktion einfach und unkompliziert erweitern. Des Weiteren wird ein Interpreter benötigt, um Seitenbeschreibungssprachen wie PostScript und Portable Document Format (PDF) benutzen zu können. Hierfür gibt es kostenlose Programme wie GhostScript - steht unter der Gnu Public License - und kann frei verwendet werden. Weite Verbreitung im universitären sowie im industriellen Sektor finden vor allem kommerzielle Produkte, z.B. von Adobe. Um Dokumente formatieren, strukturieren und bearbeiten zu können, wird zusätzlich ein Texteditor für LaTeX Dokumente benötigt. Häufige Anwendung findet hierbei das nicht kommerzielle TeXnicCenter, welches LaTeX Quelltext bearbeitet und mit dem sich LaTeX Befehle direkt im Fließtext integrieren lassen. Weniger Programmcode-orientiert und daher gut für Anwender zum Einstieg geeignet, ist LyX, eine Alternative mit grafischer Oberfläche. Im Gegensatz zu TeXnicCenter verzichtet LyX auf Befehle und Kommandos und ermöglicht es dem Anwender durch verschiedene Menüfunktionen und durch einfache Mausklicks seine gewünschten Ergebnisse darzustellen. Die zugehörigen Befehle sind hierbei hinter verschiedenen grafischen Icons integriert. Wer allerdings tiefer in die Materie einsteigen und individuell seine Dokumente bearbeiten möchte, sollte unbedingt die code-basiertere Variante wählen.

Bei der Statistiksoftware SAS können, falls die notwendigen Lizenzen zur Verfügung stehen, automatisch bei der Installation zusätzlich benötigte Komponenten mit installiert werden. Auch hier bietet sich dem Anwender einerseits die Möglichkeit mehr programmcode-orientiert ans Ziel zu gelangen oder mit Hilfe eines Frameworks (vgl. Enterprise Guide).

4 Methoden

Sind die Produkte installiert und funktionstüchtig, sind alle Voraussetzungen für eine Kombination Beider gegeben:
SAS stellt zwei Pakete zur Verfügung, die eine (in)direkte Verwendung mit LaTeX ermöglichen. Mit *ODS graphics* [6] besteht im Allgemeinen die Möglichkeit, oft verwendete Grafiken in unterschiedlichen Ausgabeformaten automatisiert zu erstellen. Mit der Kombination von *ODS Markup* und *ODS Graphics* in SAS [7] können nun sogar spezielle *tagsets* - LaTeX, ColorLaTeX, SimpleLaTeX, und TablesOnlyLaTeX - verwendet werden. Tagsets sind ODS Kontrollmechanismen, die Methoden zur Verbesserung von ODS-Ausgaben unter SAS zur Verfügung stellen. Sie bieten Vorlagen, die verwendet werden, um jede Art von textbasierter Ausgabe zu generieren: HTML, CSV, XML und auch LaTeX. Beispielsweise bietet ColorLaTeX eine große Vielfalt an unterschiedlichen Stylesheets und Makros für Ausgaben, die in ihrem Aussehen mit HTML oder

RTF vergleichbar sind. Hierfür stellt es vor allem Werkzeuge für farbige Ausgaben zur Verfügung. TablesOnlyLaTeX erlaubt es dem Anwender sogar LaTeX Ausschnitte selbst zu erstellen und in anderen LaTeX Dokumenten zu verwenden.

Im Folgenden (s. Tabelle 2**)** findet sich eine vereinfachte Darstellung notwendiger Schritte, um eine Grafik aus SAS heraus in LaTeX zu integrieren.

Tabelle 2: Sequenzielles Vorgehen

LaTeX	SAS
Neues LaTeX Dokument mit einem/mehreren eindeutigen Flag(s) (Platzhalter(n)) erstellen	Daten (für die Erstellung einer .ps Grafik) zur weiteren Verwendung im entsprechenden Format zur Verfügung stellen
	Mit ODS graphics/markup Grafiken anhand der verfügbaren Daten als .ps generieren
	Einlesen des ursprünglichen LaTeX Dokumentes, ersetzen des/der Flags
	Neues LaTeX Dokument mit dem ersetzten Code erzeugen
Kompilieren und das fertige Resultat im pdf-Format erhalten	

5 Ergebnisse

Als Einstieg wird ein einfaches LaTeX-Dokument benötigt. Ein solches könnte folgendermaßen aussehen:

```
%LaTeX-Ausschnitt 1: Flags (Platzhalter) im LaTeX Code
\documentclass{article}
%Praeambel
\usepackage[T1]{fontenc}
\usepackage{longtable}
\usepackage{graphicx}
\usepackage{times}
\usepackage{color}
%Dokument-Körper
\begin{document}
\begin{center}
\huge{Wie binde ich .ps-Dateien in LATEX ein?}
\end{center}
\section{ODS Grafik mit SAS}
\paragraph{Diese Grafik: In SAS mit ODS erstellt\\\\}
\textcolor{red}{FLAG1}
\end{document}
```

Jedes LaTeX-Dokument beginnt mit der Definition einer Dokumentenklasse `\documentclass{article}` (s. LaTeX-Ausschnitt 1). Hierbei stellt LaTeX für fast jeden Dokumenttyp eine entsprechende Klasse bereit. Somit lassen sich sowohl kurze Briefe, als auch z.B. längere Dissertationen oder ganze Bücher erstellen. In der anschließenden `Praeambel` werden globale, d.h. für das gesamte Dokument gültige Eigenschaften gesetzt (z.B. `\usepackage[T1]{fontenc}, \usepackage{color}`). Erst dann folgt der eigentliche Dokumentenkörper, in den der Fließtext integriert wird (`\begin{document}…\end{document}`). Es gilt noch zu erwähnen, dass in LaTeX interne Befehle mit einem `\` und anschließendem Befehl gekennzeichnet sind. Diese werden dann auf den in `{}` eingeschlossenen Fließtext beim Kompiliervorgang angewendet. Kommentare in LaTeX werden mit `%` eingeleitet.
Das Resultat des Kompilierens unter LaTeX und das finale Resultat mit eingebundener .ps Grafik (aus SAS) sind im Folgenden zu sehen:

Abbildung 1: LaTeX-Dokument mit Platzhalter

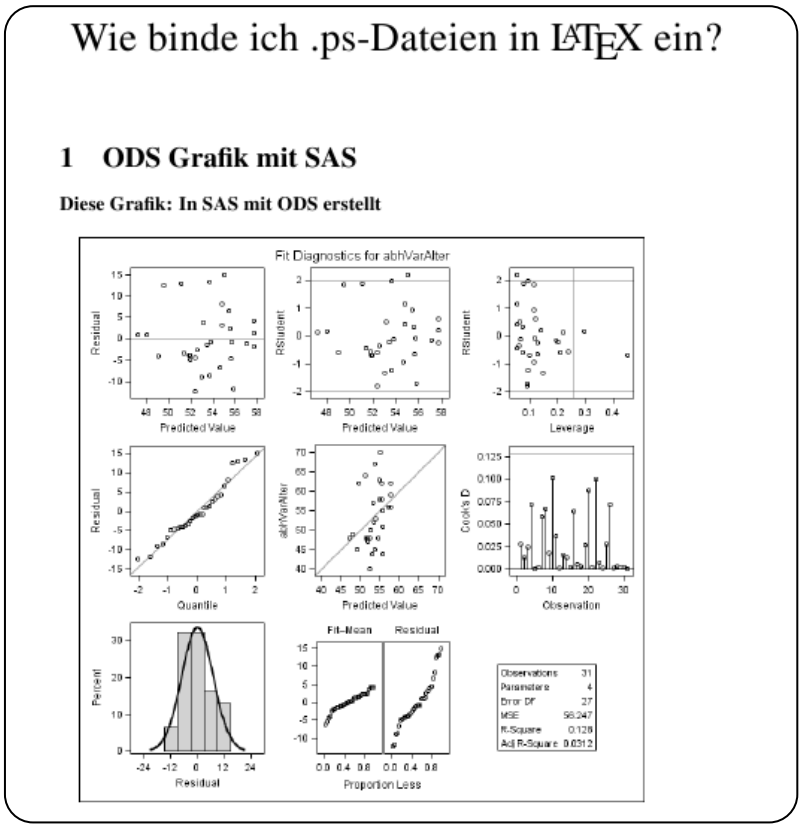

Abbildung 2: LaTeX-Dokument mit ersetztem PostScript

Hier wurde nun die Markierung mit dem eindeutigen Namen „FLAG1“ (s. Abbildung 1) durch die mit SAS und dem Makro `%SAS2LaTeX` erstellte Grafik ersetzt (s. Abbildung 2).

Allgemein sind „Flags“ als Platzhalter zu verstehen, die später automatisch mit Code für die Grafik oder einem Text ersetzt werden. Auch das Setzen mehrerer Flags ist möglich (s. LaTeX-Ausschnitt 2).

```
%LaTeX-Ausschnitt 2: Setzen mehrerer Flags im LaTeX Code
…
Auch der Text kann mit einem
FLAG1 ersetzt werden.
\section{Teil2}
FLAG2
FLAG3
\section{Teil3}
FLAG4
\end{document}
```

Stehen entsprechende Daten zur Verfügung - im Anhang unter A findet sich ein beispielhaftes SAS-Makro zum Einlesen von Daten - kann in SAS (z.B. mit `proc reg`) die Grafik als PostScript in einem SAS Data-Step mit Hilfe von `tablesonlylatex` erstellt werden (s. Anhang B). Gleichzeitig erzeugt dieser SAS Code eine zugehörige LaTeX

Datei `mygraph.tex`, die dem `%SAS2LaTeX` SAS-Makro als Basis für weitere Schritte und zum Einbinden in das finale LaTeX Dokument dient. Es wird davon ausgegangen, dass dem Anwender Grundprinzipien und Methodiken zum Einlesen von Daten und das Arbeiten mit Arrays in SAS, auch in Makros, im Allgemeinen bekannt sind. Das SAS-Makro zum Einlesen von Daten `TabelleEinlesen` (s. Anhang A) sei nur der Vollständigkeit wegen erwähnt.

Neu für den Anwender ist vermutlich die Möglichkeit, direkt aus SAS heraus mit LaTeX zu arbeiten und Grafiken in PostScript zu erstellen (s. SAS-Data-Step-Ausschnitt 1), die anschließend in LaTeX-Dokumente eingebunden werden können. Der folgende SAS-Codeausschnitt generiert - anhand der zuvor eingelesenen Daten - eine neue PostScript Grafik und eine zugehörige LaTeX Datei, die den LaTeX-Code zum Einbinden dieses Grafen beinhaltet (`mygraph.tex`). Hierfür ist in SAS vor allem das *tagset* `tablesonlylatex` zuständig. Ein ähnliches Ergebnis einer fertigen .ps Grafik findet sich beispielsweise auf der SAS Homepage [8].

```
/*SAS-Data-Step-Ausschnitt 1: SAS Code zum Erstellen einer Grafik
mit tablesonlylatex*/
data sas.TabelleNeuWerte;
     set sas.datenWerte;
     abhVarAlter=einzulesendeDaten4*1;
     var1=einzulesendeDaten6*1;
     var2=einzulesendeDaten1*1;
     var3=einzulesendeDaten5*1;
ods tagsets.tablesonlylatex
file="mygraph.tex" (notop nobot) newfile=table style=journal;
ods select diagnosticspanel;
ods graphics on / imagefmt=png imagename="datenWerte" reset;
proc reg data=sas.TabelleNeuWerte plots=diagnosticspanel;
     model abhVarAlter = var1 var2 var3;
     OUTPUT OUT=sas.dignosticpanel;
     ods select diagnosticspanel;
     title'Übersicht: Testwerte';
quit;
ods graphics off;
ods tagsets.tablesonlylatex close; run;
```

Für die weitere Verarbeitung von `mygraph.tex` soll nun ein Makro verwendet werden. Dieses soll die Grafik in bestehenden LaTeX-Code einbinden. Das Makro `%SAS2LaTeX` benötigt dafür folgende Übergabeparameter:

```
%SAS2LaTeX(path=mygraph.tex,nameFlag=FLAG,
nameOriginalTexDatei=SASwithLaTeX,anzahlFlags=4);
```

Dem SAS-Makro werden der Name der im SAS Data-Step erzeugten LaTeX Datei (`mygraph.tex`, die den Code zum Einbinden der Grafik enthält), der Name des im LaTeX Code verwendeten Flags (`FLAG`), der ursprüngliche Name der LaTeX Datei (`SASwithLaTeX`) sowie die variable Anzahl der Flags übergeben (`4`). Diese Bedingungen sind auch für die folgenden SAS-Code Ausschnitte geltend.

Innerhalb des Makros wird zunächst die LaTeX Datei `mygraph.tex`, die den Code zum Grafen beinhaltet, in einen SAS Datensatz geladen (s. SAS-Ausschnitt 2).

```
/*SAS-Ausschnitt 2: SAS-Code zum Einbinden der Grafik*/
data sas.mygraph ;
     infile tex(&fileMyGraph.) length=linelong;
     input wort $varying50. linelong ;
     CALL SYMPUT('wort', wort);
run;
```

Der Inhalt der Datei wird dann in der Makrovariablen `wort` gespeichert. Hierbei beinhaltet die Variable `wort` den LaTeX Befehl mit eingeschlossener Referenz auf die PostScript Grafik: `\includegraphics{DiagnosticsPanel2.ps}`. Um eine variable Zeilenlänge berücksichtigen zu können, wird im SAS Code die Option `$varying50.` benutzt. Danach wird in einem weiteren Data-Step die Ersetzung der Flags im LaTeX-Code vorgenommen:

```
/*SAS-Ausschnitt 3: Ursprüngliches LaTeX Dokument einlesen und Neues
  erstellen*/
data sas.saspgm ;
     infile tex(&filename..tex) length=linelong ;
     input @ ;
     input line $varying500. linelong ;
     file tex(&filename._new.tex) ;
```

Dazu lesen die ersten Zeilen SAS-Code die ursprüngliche LaTeX Datei zeilenweise ein, auch wieder unter Berücksichtigung variabler Zeilenlänge, und es wird eine neue LaTeX Datei mit dem Namen `&filename._new.tex` erzeugt. Der Name der ursprünglichen LaTeX Datei ist in `&filename` referenziert und wird entsprechend aufgelöst (s. SAS-Ausschnitt 3). Somit wird `&filename._new.tex` schließlich aufgelöst in `SASwithLaTeX_new.tex`.

Das Ersetzen der Flags im LaTeX Code wird unter zu Hilfenahme von Perl basierten Regulären Ausdrücken (P.R.A.), einem sehr mächtigen Werkzeug zur Textmanipulation, erreicht. Im Allgemeinen stellen P.R.A.'s eine Art Filterkriterium für unterschiedliche Texte dar [9] (s. auch "Anwendung von (Perl) Regular Expressions für die Mustersuche in Strings" im selben Konferenzband). Ein regulärer Ausdruck - in Form des gewünschten Musters - wird hierbei mit dem vorhandenen Text abgeglichen und bietet dann Funktionen zur Veränderung des Textes [10].

```
/*SAS-Ausschnitt 4: Flag(s) durch den Code ersetzen und in neues
  Dokument schreiben*/
pattern = "m/&name.[1-&anzahlFlags.]/i";
     if prxmatch(pattern, _infile_) > 0 then do;
               temp=prxmatch(pattern,line);
     end;
     if temp>0 then do;
          rx1=prxparse("s/&name.[1-&anzahlFlags.]/&wort./i");
          call prxchange(rx1,-1,line);
```

```
     put line;
end;
else do;
     put line $varying500. linelong ;
end;
```

Anhand des Musters des P.R.A. `pattern="m/&name.[1-&anzahlFlags.]"` wird überprüft, ob in den eingelesenen Zeilen der folgende Ausdruck `&name.[1-&anzahlFlags.]` vorkommt. Hierbei findet eine interne Auflösung der vorhandenen Variablen statt: `&name.[1-&anzahlFlags.]` wird dann intern aufgelöst in FLAG1, FLAG2, FLAG3, FLAG4 – abhängig von den dem Makro übergebenen Parametern. Durch die Option `m` im Muster des P.R.A. wird auf Übereinstimmung (`m`=Matching) geprüft. Die `i` Option verhindert die genaue Unterscheidung zwischen Groß- und Kleinschreibung – flag1, flag2, … ist somit auch möglich. Ist die Bedingung erfüllt (`prxmatch(pattern, _infile_) > 0`) wird die entsprechende Zeile in einer Variablen `temp` gespeichert. Nur bei erfüllter Bedingung werden mit `rx1=prxparse("s/&name.[1-&anzahlFlags.]/&wort./i")` dann die Variablen wieder aufgelöst. Bei `prxparse` mit Option `s` (Substitute) weiß SAS dann, dass der vordere Teil des Musters `&name.[1-&anzahlFlags.]` mit dem in der Variablen `wort` gespeicherten Inhalt (`\includegraphics{DiagnosticsPanel2.ps}`) ersetzt werden soll und mit `call prxchange(rx1,-1,line)` findet schließlich der eigentliche Austausch statt. `Put line` schreibt die ersetzte Zeile in die Datei `SASwithLaTeX_new.tex` (`&filename._new.tex`). Für alle Zeilen, für die keine Übereinstimmung gefunden wird, wird die ursprüngliche Zeile ohne Änderung übernommen und in die neue LaTeX Datei geschrieben (s. SAS-Ausschnitt 4).
Beide Makros mit detaillierten Kommentaren finden sich im Anhang.

6 Diskussion

Ein Ansatz, der notwendige Schritte zur Kombination von SAS und LaTeX mit Hilfe eines Makros überführt, ist im Anhang C zu finden. Dieser Ansatz gilt allerdings nur für das `tagset tablesOnlyLaTeX` und soll eine Idee zur Integration für zukünftige, verbesserte Makros darstellen.
Das Makro ist auf der Basis von SAS Codebeispielen mit allgemeinen Ansätzen zum Einbinden von Code aus SAS in LaTeX entstanden [7]. Es dient als Einstieg, die Verbindung zwischen SAS und LaTeX herzustellen. Da es sich nur um einen Prototypen handelt, bedarf es weiterer Anpassungen in naher Zukunft. Zur Orientierung und Weiterführung ergeben sich hieraus vielleicht funktional starke SAS-Makros. Gerade jegliches individuelle und automatisierte Einbinden aus SAS in LaTeX wäre wünschenswert, beispielsweise die Möglichkeit der direkten Überführung jeglicher SAS-Output-Tabellen in LaTeX Tabellen-(Code). Anregung: Warum nicht zukünftig die Möglichkeit bieten, einen Output (wie wir ihn aus dem Log-Fenster von SAS kennen) direkt im LaTeX Tabellen Format zu überführen; hiermit wäre ein individuelles und automatisiertes Einbinden möglich.

Hinsichtlich der Kompilierung des neuen LaTeX Dokumentes:
Es gilt einige aufgetretene Probleme zu berücksichtigen. Die Anzeigegröße der einzulesenden Grafik muss (noch) manuell im LaTeX Dokument über entsprechende LaTeX Befehle nachträglich angepasst werden, ansonsten wird die einzubindende Grafik immer komplett auf eine darauffolgende Seite gesetzt. Dies bedeutet, dass `\includegraphics{DiagnosticsPanel2.ps}` stattdessen z.B. entsprechend zu `\includegraphics[width=0.50\textwidth]{DiagnosticsPanel2.ps}` geändert werden muss, um eine Anpassung auf die halbe Größe der Textbreite zu erreichen. Eine Möglichkeit, dieses Problem zu automatisieren, bestünde vielleicht darin, in SAS festgelegte Ausgabegrößen der Grafiken mit als Eingabeparameter an das Makro zu übergeben und dann die Grafikbreite in `includegraphics` individuell anzupassen.

Bei der Verwendung von TeXnicCenter wird standardmäßig mit pdflatex.exe aus dem MiKTeX Paket kompiliert. Da dieses standardmäßig keine PostScript Dateien unterstützt, sollte stattdessen XeLaTeX.exe (ebenfalls in MiKTeX enthalten) verwendet werden. Die notwendige Änderung von pdflatex auf XeLaTeX (über die Definition eines neuen Ausgabeprofils) unter TeXnicCenter ist online gut beschrieben [11]. Des Weiteren wurde nur die automatische Einbindung von .ps Dateien ermöglicht bzw. getestet. Sinnvollerweise sollte auch die Einbindung von .eps standardmäßig bereitgestellt werden.

Weitere Einschränkung des Makros: Das verwendete Makro enthält aktuell nur die Möglichkeit mehrere Flags im LaTeX Dokument automatisiert durch die gleiche PostScript Grafik ersetzen zu lassen. Die Prozedur der Grafikerstellung ist in einem SAS Data-Step ausgelagert und ist unabhängig vom Makro (s. SAS-Data-Step-Ausschnitt 1). Somit lassen sich auch andere Grafikprozeduren zur Erstellung einer .ps Datei verwenden. Dem `%SAS2LaTeX` Makro wird nur der finale Name der neuen LaTeX Datei (in diesem Fall: `mygraph.tex`), die den Code zum Einbinden der Grafik enthält, übergeben.
Der Nutzer sollte nach jeder Grafik in SAS das Makro anwenden und die aktuelle Grafik und das dazugehörige Flag übergeben.

Die hier vorgestellte Methode behandelt vor allem das tagset: `tablesonlylatex`. Es gibt (s. o.) jedoch noch drei weitere packages, die zusätzliche Optionen bieten.

Es bleibt abzuwarten, inwieweit in neueren SAS-Versionen die LaTeX Unterstützung voranschreitet. Zusätzliche, vereinfachte und dennoch umfangreiche Pakete zur gemeinsamen und automatisierten Verarbeitung sind auf jeden Fall erstrebenswert.
Insgesamt wäre eine bessere Kontrolle und direkte Ausgabe jeglicher SAS Prozeduren in LaTeX wünschenswert.

Es gilt zu berücksichtigen, dass es sich bei dem Makro nur um einen Prototypen handelt, der vorhandene Möglichkeiten in kleinem Rahmen aufzeigt und als eine Art Ideen-

geber dienen soll. Aufgrund der geringen Komplexität der Beispiele (aber im Hinblick auf komplexe wissenschaftliche Dokumente) wurde auf die P.R.A.'s zurückgegriffen, die vor allem die Laufzeit bei der Flag-Suche und -Ersetzung bei großen Dokumenten drastisch verringern. Gleichzeitig lassen sich hiermit komplexe `substr` und `index` Funktionen kombiniert mit vielen `if` Bedingungen im SAS Code gezielt und gekonnt vermeiden - bei dennoch gleichbleibender Funktionalität. Somit kann das Makro relativ einfach erweitert, verändert, verbessert und vor allem an eigene Bedürfnisse angepasst werden.

7 Schlussfolgerung

Die vorgestellten Möglichkeiten, die durch Kombination von SAS und LaTeX zur Verfügung gestellt werden, stellen einen ersten wichtigen Schritt dar, um SAS mit LaTeX zu kombinieren und somit auch die Anwender von LaTeX bei der Einbindung von SAS-Output zu unterstützen. Gerade wenn zukünftig umfangreichere Kombinationsmöglichkeiten in der Produktpalette von SAS-Paketen für die Integration in LaTeX zur Verfügung stehen, werden auch den LaTeX Nutzern mehr Möglichkeiten gegeben. In Zukunft erwarten wir noch mehr und vor allem bessere Kontrollen und einen direkten Input jedweder SAS Prozedur in ein LaTeX Dokument.

Literatur

[1] Juha-Pekka Perttola and F. Hoffmann-La Roche AG: SAS and LATEX - a Perfect Match?. 2008. http://www.phuse.eu/download.aspx?type=cms&docID=587

[2] LaTeX Project Team: LaTeX - A document preparation system. 2010. http://www.latex-project.org/

[3] D. Arnold: Writing Scientific Papers in LATEX. 2001. http://online.redwoods.cc.ca.us/instruct/darnold/linalg/latex/project_latex.pdf

[4] M. Bärwolff: LATEX leicht gemacht. 2004.

http://www.ig.cs.tu-berlin.de/oldstatic/materialien/docs/latex_leicht_gemacht.pdf

[5] SAS Institute Inc.: SAS - Statistical Analysis System. 2010. http://support.sas.com/

[6] SAS Institute Inc.: Managing Your Graphics. 2010. http://support.sas.com/rnd/app/da/stat/odsgraph/examples.html

[7] A. Dauchy & S.L. Guennec: Professional outputs with ODS LATEX. 2008. http://www.phuse.eu/download.aspx?type=cms&docID=594

[8] SAS: Institute Inc.: Modifying Individual Plots in a Diagnostic Panel http://support.sas.com/documentation/cdl/en/grstateditug/61951/HTML/default/n056f76ygmmiqmn16wi864zz84o2.htm

[9] Daniel Fett: Tutorial Reguläre Ausdrücke, Online Tutorial http://www.danielfett.de/internet-und-opensource,artikel,regulaere-ausdruecke

[10] M. Kappler, Carina Ortseifen, Grischa Pfister, Heinrich Stürzl: KSFE 2005, http://saswiki.org/images/a/ae/9.KSFE-2005-Kappler-Tipps-und-Tricks-f%C3%BCr-den-leichteren-Umgang-mit-der-SAS-Software.pdf

[11] XeTeX Setup + unicode-math, 2010. http://anotherthought.de/blog/?p=17

Anhang A: SAS Makro - TabelleEinlesen

```
options symbolgen mprint mlogic;
/* Festlegen des globalen Pfades zum Ordner, der verwendet werden
soll und Speichern in einer Makrovariablen Pfad. */
%LET Pfad=C:\KSFE\SASundLaTeX;
/* Makro zum Einlesen der Daten aus einer Datei */
%macro
TabelleEinlesen(path=,delimiter=,startSchleife=,dimArray=,schrittWei
te=);
/* Pfad für "Current Folder" individuell definieren'*/
/*x "cd &Pfad.\Current Folder";*/
x "cd &Pfad.";
/* Eigenen Library definieren */
libname sas "&Pfad.";
/*Zuordnung der Übergabeparameter zu den zu verwendenden Variablen*/
%LET path=&path;
%LET delimiter=&delimiter;
%LET numberArray=&dimArray;
%LET step=&schrittWeite;
/* Erstellung von Tabellen zur besseren Übersicht*/
data sas.datenWerte;
     infile "&path";
     delimiter="&delimiter";
     numberArray="&numberArray";
     /* Tabelle noch als Character-Spalten. Dies kann/sollte später
     angepasst werden (direkte numerische Werte) */
     length einzulesendeDaten1 $20;
     /* Erstellen eines Arrays in Abhängigkeit der vorgesehenen
     Elemente, gegeben durch den Übergabeparameter für das Makro */
     array einzulesendeDaten_A {1:&numberArray. } $20
einzulesendeDaten1-einzulesendeDaten&numberArray.;
/* Satz für nächsten INPUT-Befehl halten */
input einzulesendeDaten1 $ @ ;
     /* Schleifenzähler, delimiter und Größe des Arrays NICHT
ausgeben */
     drop i;
     drop delimiter;
     drop numberArray;
     /* Schleife durch Übergabeparameter festgelegt.
     Einlesen der Daten*/
     do i=&startSchleife. to &numberArray. by &step.;
     /* Feld i einlesen, Satz für nächsten INPUT-Befehl halten */
input einzulesendeDaten_A(i) $  @ ;
end;
```

```
     %LET AnzVar=0;
     %DO AnzVar=1 %TO 5 %BY 1;
          %LET einzulesendeDaten&AnzVar. = einzulesendeDaten_A(i);
          /*Ausgabe der Variable(n)*/
          %PUT einzulesendeDaten&AnzVar=&&&einzulesendeDaten&AnzVar.;
    %END;
     run;
/*Ausgabe der Daten - Tabelle datenWerte begrenzt auf 10 Zeilen
(obs=10) zur bessern Übersichtlichkeit*/
proc print data=sas.datenWerte(obs=10);
run;
%mend TabelleEinlesen;

Aufruf:
/*Makro zum Einlesen von Daten aus einer Datei.
Übergabeparameter: Pfad zur Text-Datei, der Delimiter für die
Spaltentrennung, Startparameter für Schleifendurchlauf,
Größe des Arrays (# Variablen) sowie die Schrittweite für den
Schleifendurchlauf*/
%TabelleEinlesen(path=&Pfad.\test.txt,delimiter=' ',
startSchleife=1, dimArray=6, schrittWeite=1);
```

Anhang B: Grafik als PostScript

```
/*Es wird eine Grafik in PostScript anhand der eingelesenen Daten
generiert*/
data sas.TabelleNeuWerte;
     set sas.datenWerte;
     /* Da die Werte als Character in der Tabelle stehen, diese nun
        in numerische Variablen umwandeln */
     abhVarAlter=einzulesendeDaten4*1;
     var1=einzulesendeDaten6*1;
     var2=einzulesendeDaten1*1;
     var3=einzulesendeDaten5*1;
     /*Verwendung des tagsets: tablesonlylatex*/
     ods tagsets.tablesonlylatex
     file="mygraph.tex" (notop nobot) newfile=table style=journal;
     ods select diagnosticspanel;
     ods graphics on / imagefmt=png imagename="datenWerte" reset;
     proc reg data=sas.TabelleNeuWerte plots=diagnosticspanel;
          model abhVarAlter = var1 var2 var3;
          OUTPUT OUT=sas.dignosticpanel;
          ods select diagnosticspanel;
          title'Übersicht: Testwerte';
     quit;
     ods graphics off;
     ods tagsets.tablesonlylatex close;
run;
```

Anhang C: SAS Makro - SAS2LaTeX

```
/* Makro, das die Daten für LaTeX verarbeitet. Die LaTeX Datei
(mygraph.tex), die den Code zur Grafik enthält, wird in einen SAS
Datensatz geladen. Das originale LaTeX Dokument wird geladen.
Ersetzen des/der "flags"(Markierung(en)) durch den neuen LaTeX Code.
Neues LaTeX Dokument mit finalem Code wird bereitgestellt*/
%macro
SAS2LaTeX(path=,nameFlag=,nameOriginalTexDatei=,anzahlFlags=);
/*Variablen: fileMyGraph, nameflag, filename erhalten die Übergabe-
parameter, die dem Makro mitgegeben wurden*/
%LET fileMyGraph=&path;
%LET filename=&nameOriginalTexDatei;
%LET anzahlFlags=&anzahlFlags;
%LET name=&nameFlag;
/* Pfad zum Ordner der LaTeX Dateien: Der Befehl "tex" ist hierbei
durch das Package vorgegeben */
filename tex "&Pfad.";
;
/* Laden der LaTeX Datei, die den Grafen beinhaltet, in einen SAS
Datensatz */
data sas.mygraph ;
     /*Einlesen der LaTeX Datei mygraph und speichern des
       beinhaltenden Textes in der Variablen wort.*/
     infile tex(&fileMyGraph.) length=linelong;
     input wort $varying50. linelong ;
     CALL SYMPUT('wort', wort);
     run;
/* Originale LaTeX Datei laden.
Die neue LaTeX Datei erstellen mit dem zu ersetzenden Code */
data sas.saspgm ;
     infile tex(&filename..tex) length=linelong ;
     input @ ;
     input line $varying500. linelong ;
     file tex(&filename._new.tex) ;
/*Die Regular expression überprüft auf matching zwischen dem
eingelesenen Text und dem Übergabeparameter (name). Hierbei
werden mehrere Fälle anhand der Laufvariablen berücksichtigt;
Der Buchstabe "m" in der Regularexpression ist für das Überprüfen
auf Matching zuständig. Die "i" Option verhindert die genaue
Unterscheidung zwischen Groß-/Kleinschreibung*/
     pattern = "m/&name.[1-&anzahlFlags.]/i";
     if prxmatch(pattern, _infile_) > 0 then do;
          temp=prxmatch(pattern,line);
     end;
/*Die Regular Expression überprüft auf matching zwischen dem
  eingelesenen Text und dem Übergabeparameter (name). Hierbei werden
  mehrere Fälle anhand der Laufvariablen berücksichtigt"*/
  if temp>0 then do;
          rx1=prxparse("s/&name.[1-&anzahlFlags.]/&wort./i");
          /*Bei Treffer: Austauschen durch neuen Wert*/
          call prxchange(rx1,-1,line);
```

```
          put line;
     end;
      else do;
          put line $varying500. linelong ;
      end;run;
%mend SAS2LaTeX;

Aufruf:
/*Makro zur Erstellung des LaTeX Dokumentes. Übergabeparameter:
Name der Datei, die den includegraphics LaTeX-Befehl für das
erstellte PostScript enthält.
 2. Name des/der Flags im LaTeX-Dokument.
 3. Der originale Name der LaTeX Datei,
 4. Die Anzahl der Flags des LaTeX Dokumentes*/
%SAS2LaTeX(path=mygraph.tex,nameFlag=FLAG,nameOriginalTexDatei=SASwi
thLaTeX,anzahlFlags=4);
```