Open Source SAS Makros - Was gibt es? Wie nutze ich es? Was sollte ich beachten?

Katja Glaß
Katja Glass Consulting
Biesdorfer Weg 17a
12683 Berlin
Katja.Glass@glacon.eu

Zusammenfassung

Es gibt extrem viele Open Source Lösungen, die bei Problemen einfache Lösungen anbieten. Für Datenauswertungen bieten vor allem R, aber auch Python viele Module und Optionen. Aber auch für SAS gibt es einzelne Open Source Makros und Skripte, die hier vorgestellt werden. Hierbei wird näher auf folgende Fragen eingegangen. Was gibt es für Open Source Lösungen im Bereich der SAS Programmierung? Wo finde ich diese? Wie können diese dann am besten genutzt werden? Und was ist dabei zu beachten?

Schlüsselwörter: Open Source, SAS, Makro, Programme

1 Einleitung

Zunächst stellt sich die Frage, was Open Source ist. Eine Definition findet sich in Wikipedia:

> *„Als Open Source wird **Software** bezeichnet, deren **Quelltext** öffentlich und von Dritten eingesehen, geändert und genutzt werden kann. Open-Source-Software kann meistens kostenlos genutzt werden."*

Der wesentliche Punkt von Open Source ist die Verfügbarkeit des Quelltexts. Im SAS Bereich können unter Software SAS Makros und Programme verstanden werden. Open Source Programme sind dabei flexibel einsetzbar. Die Lizenzbestimmungen sind in jedem Fall zu beachten. Wenn es keine Lizenz gibt, so gilt das „Copyright".

Sehr offene Lizenzen sind die „Unlicense" und die „MIT" Lizenz. Hierbei ist entweder nichts zu beachten, oder die originale Lizenz muss kopiert werden und Erwähnung finden. Die GNU Familie ist ebenso eine sehr beliebte Lizenzgruppe, die auch sehr offen ist. Die kommerzielle Nutzung ist in allen Fällen möglich. Je nach GNU Lizenz muss man mehr beachten, so dass beispielsweise neben dem Beibehalten des originalen Copyrights bei einigen strengeren Lizenzen auch Veröffentlichungspflicht von Veränderungen unter der gleichen Lizenz nötig sind. Darüber hinaus gibt es viele weitere Lizen-

zen, darunter auch einige, die eine kommerzielle Nutzung verbieten. Online gibt es schöne Übersichten über verschiedene gebräuchliche Lizenzen[1].

Open Source Lösungen bieten verschiedene Anwendungsoptionen. Man kann die Lösungen direkt unverändert verwenden, man kann diese verändern und dann nutzen oder man kann komplexere Software entwickeln, welche die Open Source Lösungen in veränderter oder unveränderter Form nutzt.

Open Source wird unheimlich gerne verwendet, weil die Funktionalität einfach vorhanden ist. Man muss ein Problem nicht selbst lösen, sondern kann sich einer fertigen Lösung bedienen! Auch das „Learning by doing" funktioniert bei Open Source Projekten sehr gut und bietet sich an, um zu lernen, wie eine Lösung für ein bestimmtes Problem programmiert werden kann.

Zudem sind Open Source Projekte oft günstiger, sofern sie von der Community oder verschiedenen Firmen unterstützt werden. Die Entwicklungskosten und Trainingskosten werden geteilt. Open Source kann auch viel robuster sein, da durch vielfältige Nutzer und Anwendungen viel mehr Praxistests gemacht werden. Die Community kann mehr Ideen einbringen, da diese üblicherweise größer ist als Einzelentwicklungsteams.

Wenn von Software der Code offen ist, dann kann auch nachgeprüft werden, was genau in dem Programmablauf passiert. Hierbei ist Transparenz gegeben, so dass man weniger Bedenken zu versteckten Abläufen haben muss.

1.1 Das Baukasten-Prinzip

Bei anderen Programmiersprachen wie R, Python, JavaScript und vielen weiteren wird Open Source intensiv genutzt. Dabei bedient man sich meistens dem Baukasten-Prinzips. Es gibt kleinere und größere Funktionsblöcke, sowie komplexe Tools mit viel Funktionalität. Dabei bauen die größeren meist auf den kleineren auf.

Wenn man eine Webseite erstellen will, dann lernt man meist kein einfaches HTML mehr, sondern man überlegt sich, welches Framework man verwendet. Für dieses Framework gibt es dann verschiedenste Komponenten wie filterbare Tabellen, interaktive Grafiken[2], User Interface Themen und vieles mehr. Es werden die einzelnen Funktionsblöcke miteinander kombiniert, um die eigene Webseite entsprechend zu gestalten und mit Funktionalität auszustatten. Dabei sind vielfältigste „Blöcke" als Open Source vorhanden.

[1] Siehe Literaturquelle [2]
[2] Beispiel: Graphikbibliothek D3

In SAS könnte man die Data Steps, Datei Manipulationen und SQL als Basisblöcke sehen. Größere komplexe Komponenten sind zum Beispiel die verschiedenen Prozeduren wie PROC REPORT und ähnliches. Daneben gibt es auch komplexe Vollfunktionen, wie beispielsweise den ODS Graphics Designer, der sich der verschiedenen Einzelkomponenten bedient.

1.2 Problematik

Während bei anderen Programmiersprachen sehr viele Open Source Blöcke vorhanden sind, mangelt es an diesen bei SAS. Die Community stellt leider nicht viele Open Source Lösungen bereit. Warum ist das so? Zum einen ist die Community von SAS kleiner, da weniger Menschen SAS verwenden. Zum anderen wird SAS hauptsächlich von Pharmafirmen, Versicherungen und Banken verwendet. Im privaten Sektor findet SAS kaum Anwendung, da die Softwarekosten hoch sind.

Auf Grund des sehr kommerziellen Einsatzes von SAS ist die Bereitschaft, Open Source für andere zu erstellen, geringer als bei anderen Programmiersprachen. Wenn Funktionsblöcke erstellt werden, so findet die Weiterverbreitung meist nur innerhalb der eigenen Firma statt. Auch Arbeitsverträge und die Auftragsinhalte verhindern, dass Implementierungen – und seien es auch nur kleine Funktionsblöcke – als Open Source zur Verfügung gestellt werden.

Es gibt jedoch auch Möglichkeiten, Open Source im Bereich von SAS zu fördern! Als ersten Schritt sollten Firmen ihren Mitarbeitern und Consultants erlauben, Code zu veröffentlichen. Es gibt auch Arbeitsgruppen, die offen arbeiten. Hier können sich Mitarbeiter einbringen und so aktiv Open Source erstellen. Ein weiterer Punkt ist die Investition in den Open Source Bereich. Wenn man investiert, so ist die Motivation gegeben und es würden mehr Open Source Lösungen zur Verfügung stehen.

1.3 Finden von Open Source

Ein weiteres Problem von Open Source Lösungen ist das Finden vorhandener Lösungen. Für R gibt es Releases in CRAN, so dass es eine Stelle gibt, wo man spezifisch suchen kann. Für SAS gibt es dies derzeit nicht. Natürlich können Lösungen auf Konferenzen wie der KSFE vorgestellt werden, jedoch hat auch dies nur eine begrenzte Reichweite.

Ich habe mich des Problems angenommen und ein Open Source Portal für klinische Studienauswertungen[3] erstellt, wo ebenso diverse SAS Open Source Lösungen zu finden sind. Diese sind sehr generisch und können auch für diverse andere Branchenbereiche verwendet werden.

[3] Siehe Liternaturquelle [3]

Man findet auch eine Übersichtsseite über diverse Tools in dem Bereich. In der Tabellenansicht kann nach verschiedenen Programmiersprachen wie SAS gefiltert werden.

Im „Programs“ Fenster können verschiedene Programme und Makros nach Stichwörtern in der Beschreibung durchsucht werden. Derzeit befinden sich über 400 Programme und Makros eingepflegt. Weitere werden sicher folgen.

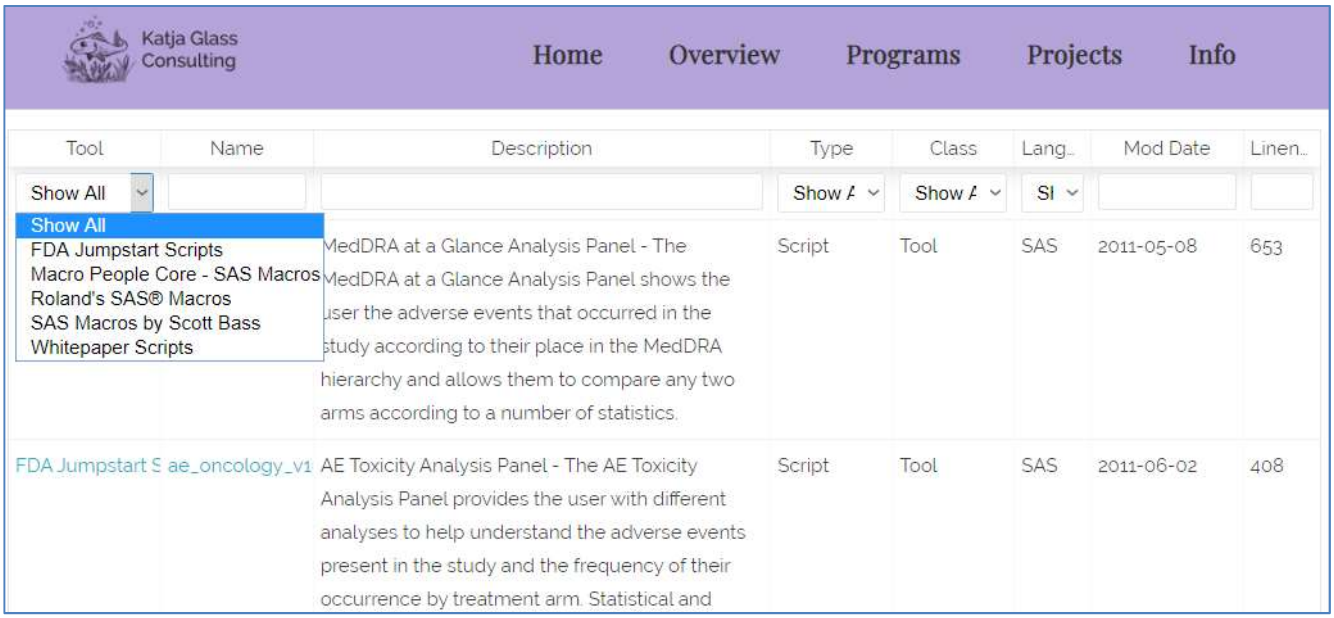

Abbildung 1: Ansicht des Fensters für Makro- und Programmsuche

Wenn die Tools nach SAS gefiltert werden, so finden sich folgende Open Source Lösungen, die im Weiteren erläutert werden.

- FDA Jumpstart Scripts
- SASUnit
- Spectre (Roland's SAS® Macros)
- SAS Macros by Scott Bass
- SAS Macros by Macro People
- PhUSE White Paper Central Tendencies Scripts
- Data Visualization - SAS Blog
- RhoInc Plots
- PhUSE White Paper Utilities
- PhUSE Contributed Scripts & Macros
- Reindeer – Render SAS Results into Word

2 Anwendungsbeispiel

Zunächst soll erläutert werden, wie ein Open Source Programm angewendet werden kann. Bei den PhUSE Whitepaper Scripts handelt es sich um eine Programmsammlung, hauptsächlich in SAS geschrieben, um diverse Tabellen und Graphiken von einer Sammlung – dem sogenannten „White Paper“ – zu erstellen. PhUSE ist dabei eine

Community, welche diese Programme ca. 2013 erstellt und zur Verfügung gestellt hat. Zusätzlich fand 2014 ein Scriptathon auf einer PhUSE Konferenz statt, wo weitere Programme als Open Source veröffentlicht wurden.

Um ein Programm auszuprobieren, könnte zum Beispiel nach „Boxplot“ in dem Open Source Portal gesucht werden. Im Weiteren soll das Programm vorgestellt werden, welches eine Boxplot Graphik erstellt (Abb. 2). Das zugehörige Programm befindet sich im PhUSE GitHub Repository[4].

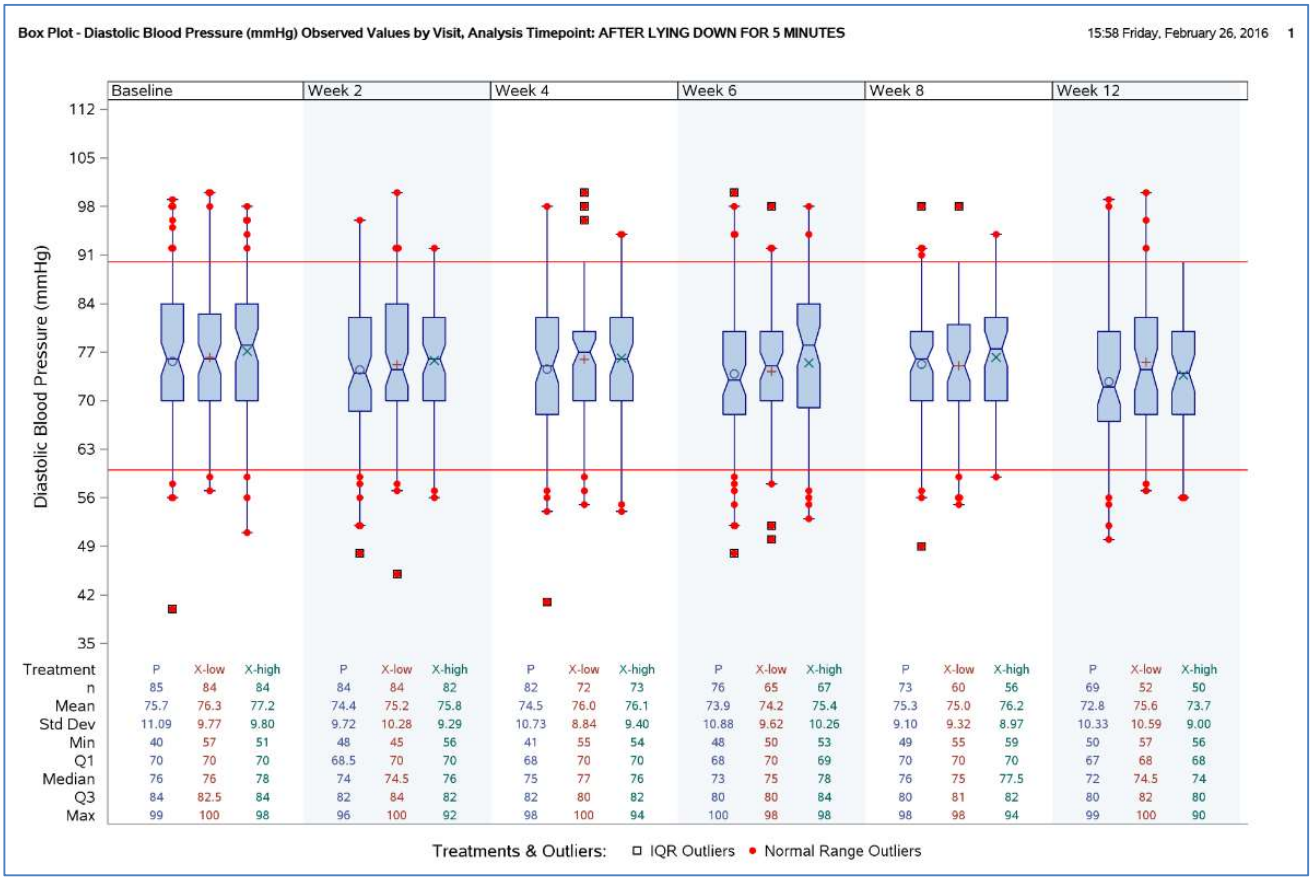

Abbildung 2: BoxPlot als Anwendungsbeispiel

Das Skript Repository, welches sich in GitHub befindet, muss heruntergeladen werden. Dies kann entweder über das Klonen oder ein Zip-Download geschehen. Das Programm wird gewählt, geöffnet und angepasst. Schließlich wird das Programm ausgeführt und der Output erscheint hoffentlich.

Bei älteren Programmen kann es häufiger passieren, dass diverse Fehler im Programm auftauchen. Diese Fehler müssen als nächstes analysiert und korrigiert werden.

Wofür kann nun dieses Anwendungsbeispiel verwendet werden? Zum einen kann man über solche Beispiele lernen, wie man eigene ähnliche Graphiken erstellt. Im Beispiel

[4] Siehe Literaturquelle [5]

wurde mit PROC TEMPLATE und SGRENDER gearbeitet. Der Code kann analysiert und entsprechend den eigenen Bedürfnissen angepasst werden.

Vorhandene Beispiele können auch genutzt werden, um diese auf eigene Daten anzuwenden, um diese beispielsweise in einen Studienreport einzufügen. Da im Bereich der Studienauswertung hohe Validierungsanforderungen bestehen, müssen natürlich die lokalen Validierungsvorschriften berücksichtigt werden. Open Source ist üblicherweise nicht validiert, und so obliegt es dem Verwender, die Validierung sicher zu stellen. Dies ist aber auch der Fall, wenn ein Programm von einem Kollegen kopiert wird.

Darüber hinaus kann das Programm auch für vieles weiteres genutzt werden. Man könnte ein Makro erstellen, was generische Boxplots erstellt. Man könnte ein komplexes Tool entwickeln, was verschiedene Graphiken und unter anderem solche Boxplots erstellt und vieles mehr.

Wenn Probleme im Programmablauf festgestellt werden, so ist es sehr förderlich, diese zu reporten oder zu beheben. Hierbei muss berücksichtigt werden, ob die Open Source Lösung eine aktive Wartung hat oder nicht. Probleme können in GitHub einfach als „Issue“ berichten werden. Wenn man weiß, dass ein Repository aber keine Wartung hat oder von der Wartung durch andere lebt, so ist es sinnvoll, Änderungen selbst durchzuführen. Bei der PhUSE können sehr leicht die Rechte angefragt werden und Änderungen im Repository vorgenommen werden.

3 Beispiele

Im Folgenden sollen nun weitere Open Source Lösungen im SAS vorgestellt werden.

3.1 PhUSE Whitepaper und Scriptathon Programme

Die PhUSE Whitepaper Scripte und Programme vom “Scriptathon“ bieten viele Programmbeispiele, um diverse Graphiken und Tabellen für klinische Studien zu erstellen.

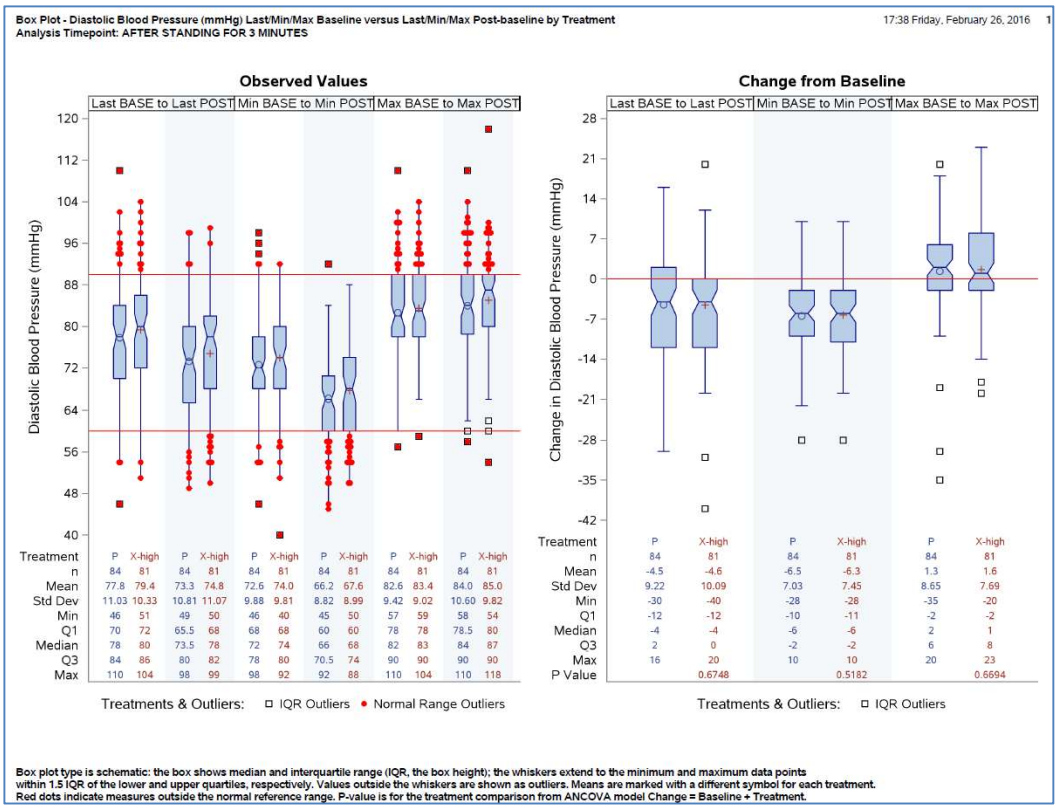

Abbildung 3: BoxPlot Beispiel

&under

		&trtname1 (N=86)	trtname1 (N=84)	trtname1 (N=84)
Sex n(%)	n[a]	86 (100.0%)	84 (100.0%)	84 (100.0%)
	F	53 (61.6%)	50 (61.6%)	40 (61.6%)
	M	33 (38.4%)	34 (38.4%)	44 (38.4%)
Age (years)	n[a]	86	84	84
	Mean	75.2	75.7	74.4
	SD	8.6	8.3	7.9
	Median	76.0	77.5	76.0
	Q1, Q3	69.0, 82.0	71.0, 82.0	70.5, 80.0
	Min, Max	52, 89	51, 88	56, 88
Age categories n(%)	65-80	42 (48.8%)	47 (48.8%)	55 (48.8%)
	<65	14 (16.3%)	8 (16.3%)	11 (16.3%)
	>80	30 (34.9%)	29 (34.9%)	18 (34.9%)
	n[a]	86 (100.0%)	84 (100.0%)	84 (100.0%)
Race n(%)	n[a]	86 (100.0%)	84 (100.0%)	84 (100.0%)
	White	78 (90.7%)	78 (90.7%)	74 (90.7%)
	Black or African American	8 (9.3%)	6 (9.3%)	9 (9.3%)
	American Indian or Alaska Native	0	0	1
Ethnicity n(%)	n[a]	86 (100.0%)	84 (100.0%)	84 (100.0%)

Abbildung 4: Beispielausgabe zu Demographie

		Post-Baseline Result							
		Low		Normal		High		Total	
Treatment	Baseline Result	n	%	n	%	n	%	n	%
Placebo (N = 81)	Low	2	(2.5)	2	(2.5)	0	(0.0)	4	(4.9)
	Normal	4	(4.9)	66	(81.5)	1	(1.2)	71	(87.7)
	High	0	(0.0)	4	(4.9)	2	(2.5)	6	(7.4)
	Total	6	(7.4)	72	(88.9)	3	(3.7)	0	(0.0)
Xanomeline Low Dose (N = 72)	Low	0	(0.0)	1	(1.4)	0	(0.0)	1	(1.4)
	Normal	2	(2.8)	66	(91.7)	0	(0.0)	68	(94.4)
	High	0	(0.0)	3	(4.2)	0	(0.0)	3	(4.2)
	Total	2	(2.8)	70	(97.2)	0	(0.0)	0	(0.0)
Xanomeline High Dose (N = 73)	Low	1	(1.4)	2	(2.7)	0	(0.0)	3	(4.1)
	Normal	2	(2.7)	64	(87.7)	0	(0.0)	66	(90.4)
	High	0	(0.0)	3	(4.1)	1	(1.4)	4	(5.5)
	Total	3	(4.1)	69	(94.5)	1	(1.4)	0	(0.0)

Abbildung 5: Beispiel zum Labor

3.2 FDA Jumpstart Scripts

Die amerikanische Gesundheitsbehörde FDA hat der PhUSE Organisation SAS Makros, die für klinische Studiendaten verwendet werden, als Open Source zur Verfügung gestellt. Hierbei handelt es sich um reine SAS Programme, die nichts mit der Software „Jump“ zu tun haben.

Die Programme sind generisch und unterstützen den CDISC-SDTM Datenstandard, der für klinische Studien verwendet wird. Es werden Excel Outputs erstellt, jedoch geschieht dies durch direkte XML-Anpassungen und ist daher sehr komplex. Die Makros sind auch sehr umfangreich und komplex, bieten aber eine ideale Möglichkeit der Weiterentwicklung. Da hier die MIT Lizenz verwendet wurde, ist dies problemlos möglich.

Zu diesem Programmpaket gibt es auch viel Dokumentation sowie Spezifikationsdokumente. Darüber hinaus wurde durch eine PhUSE Arbeitsgruppe eine Qualifikation der Programme durchgeführt.

Odds Ratio & 95% Confidence Interval Analysis
Top 30 Odds Ratio Observations

NDA/BLA:
Study: CDISC
Analysis run date: 2016-03-07 11:15:12 AM

Placebo vs. Xanomeline High Dose

Body System	Dictionary-Derived Term	Placebo Count	Placebo %	Xanomeline High Dose Count	Xanomeline High Dose %	Odds Ratio	Lower 95% CI	Upper 95% CI
Gastr	Diarrhoea	9	10,5	4	4,8	2,3	(0,6,	10,8)
Infec	Upper Respiratory Tract Infec	6	7,0	3	3,6	2,0	(0,4,	12,9)
Genrl	Pyrexia	2	2,3	1	1,2	2,0	(0,1,	118,0)
Infec	Urinary Tract Infection	2	2,3	1	1,2	2,0	(0,1,	118,0)
Inj&P	Excoriation	2	2,3	1	1,2	2,0	(0,1,	118,0)
Inv	Electrocardiogram T Wave In	2	2,3	1	1,2	2,0	(0,1,	118,0)
Nerv	Somnolence	2	2,3	1	1,2	2,0	(0,1,	118,0)
Psych	Agitation	2	2,3	1	1,2	2,0	(0,1,	118,0)
Psych	Confusional State	2	2,3	1	1,2	2,0	(0,1,	118,0)
Card	Supraventricular Extrasystole	1	1,2	1	1,2	1,0	(0,0,	77,6)
Gastr	Abdominal Pain	1	1,2	1	1,2	1,0	(0,0,	77,6)
Genrl	Application Site Reaction	1	1,2	1	1,2	1,0	(0,0,	77,6)
Genrl	Asthenia	1	1,2	1	1,2	1,0	(0,0,	77,6)
Genrl	Chills	1	1,2	1	1,2	1,0	(0,0,	77,6)
Infec	Cystitis	1	1,2	1	1,2	1,0	(0,0,	77,6)
Infec	Influenza	1	1,2	1	1,2	1,0	(0,0,	77,6)
Inj&P	Fall	1	1,2	1	1,2	1,0	(0,0,	77,6)

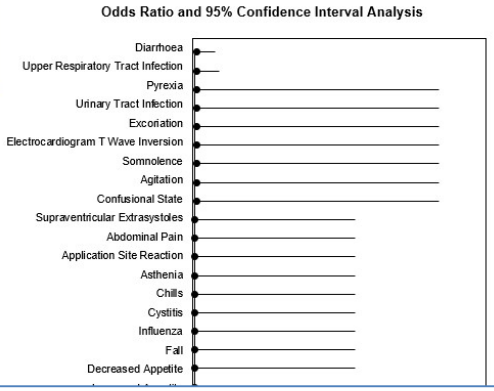

Abbildung 6: Beispielausgabe zu Konfizenzintervallen

Demographic Baseline Characteristics		Treatment A N=302		Treatment B N=159		Overall N=461	
Age	Mean (SE)	61.8 (8.5)		60.4 (10.1)		61.3 (9.1)	
	Min	39		29		29	
	Q1	57		55		56	
	Median	61		62		61	
	Q3	68		67		68	
	Max	83		85		85	
		Count	%	Count	%	Count	%
Age Groupe	between 1 year and 35 years	0	0,0	3	1,9	3	0,7
	Age between 35 and 65	186	61,6	102	64,2	288	62,5
	Age 65 and over	116	38,4	54	34,0	170	36,9
Sex	F	128	42,4	66	41,5	194	42,1
	M	174	57,6	93	58,5	267	57,9
Race	erican Indian Or Alaska Native	2	0,7	0	0,0	2	0,4
	Asian	4	1,3	2	1,3	6	1,3
	Black Or African American	17	5,6	9	5,7	26	5,6
	Native Hawaiian Or Other Pacific Islander	1	0,3	0	0,0	1	0,2
	White	278	92,1	148	93,1	426	92,4
Ethnicity	Hispanic Or Latino	54	17,9	25	15,7	79	17,1
	Not Applicable	35	11,6	19	11,9	54	11,7
	Not Hispanic Or Latino	213	70,5	115	72,3	328	71,1

Abbildung 7: Beispielausgabe zur Demographie

3.3 SAS Blogs

SAS als Softwareanbieter bietet in Blogs einige nützliche Beispiele. Da es sich um Blogs handelt, gibt es Beschreibungen und Erklärungen. Leider wird hier keine Softwarelizenz erwähnt, so dass die Weiterverwendung fragwürdig ist. Darüber hinaus sind die Programme nicht für alle Bespiele vorhanden, aber für einige.

Abbildung 8: SAS Blogs

3.4 RhoInc Plots

Die Firma RhoInc ist sehr Open Source freundlich und hat ihre Mitarbeiter diverse Open Source Lösungen in ihrem GitHub veröffentlichen lassen. Einige davon beinhalten nützliche generische Makros, welche verschiedene Plots erzeugen können.

Hierbei wird meistens auch die freundliche MIT Lizenz verwendet. Neben dem Quellcode und Beispielen gibt es meistens auch ein Paper zu dem entsprechenden Makro.

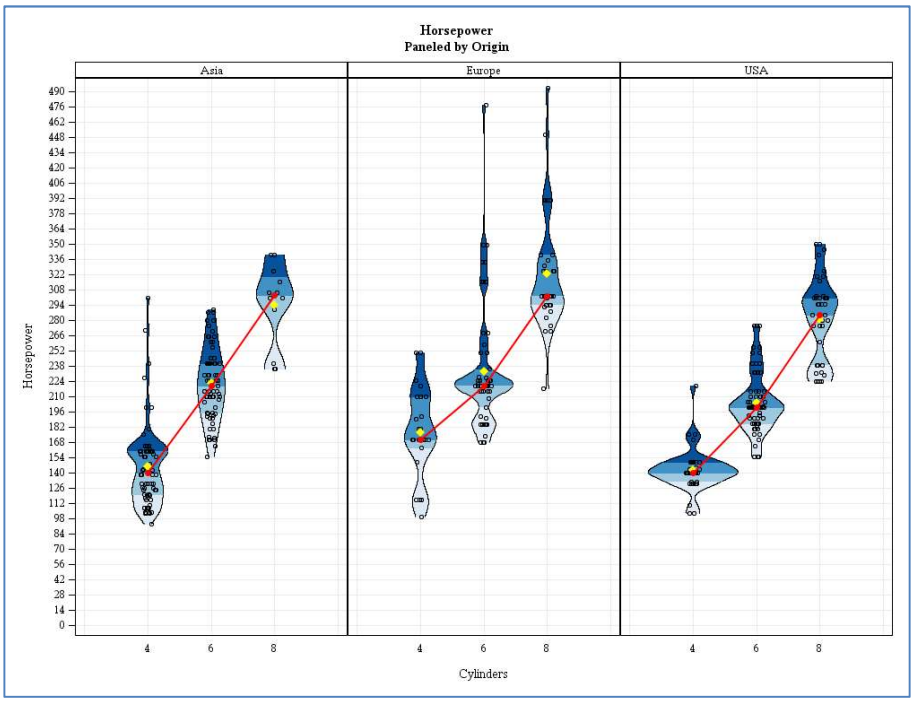

Abbildung 9: Beispiel eines Violin-Plots

3.5 Makro Kollektionen

Es gibt auch verschiedene Sammlungen von Makros, die extrem nützlich für verschiedene Situationen sein können. Hierbei handelt es sich zumeist um kleinere Hilfsmakros für unterschiedliche Zielsetzungen.

Die SAS Makros von „**Macro People**“ werden hauptsächlich von Allen Bowe entwickelt und gewartet. Es gibt hierbei sogar eine separate Dokumentationsseite. Es gibt hier beispielsweise Makros, um Verzeichnisse anzulegen, Dateien binär zu kopieren, Verzeichnisse und Dateien zu listen, einen Datensatz in CARDS Anweisungen umzuwandeln und vieles mehr.

Scott Bass ist ein Selbstständiger, der in Australien lebt und seine Makros auch als Open Source zur Verfügung stellt. Auch hier findet eine aktive Wartung statt. Es handelt sich hierbei auch um diverse Hilfsmakros. So finden sich Makros, um Zahlen dezimal auszurichten, Dateien zu löschen, Excel Dateien einzulesen, diverse Makros zum Exportieren, Datensätze vergleichen und viele mehr.

Roland hat ein komplexes Open Source System “Spectre” für klinische Studienauswertungen zur Verfügung stellt, welches allerdings spezifische Input Datensätze erfordert, die nicht dem aktuellen CDISC Standard entsprechen. Darüber hinaus hat er eine sehr große Anzahl verschiedener Hilfsmakros veröffentlicht. Roland ist inzwischen leider verstorben, so dass es keine aktive Wartung gibt.

Bei den 243 Hilfsmakros sind Makros für Altersberechnungen aus Datumsangaben, Umwandlungen von Formaten, Löschen von Makros, Datensatzreduktion, Datenvergleiche und vieles weiteres.

Die **PhUSE White Paper Utilities** sind hauptsächlich für spezifische Anwendungsfälle erstellt worden, um beispielsweise diverse Checks durchzuführen. Im PhUSE Repository selbst können auch weitere nützliche Programme und Makros gefunden werden, jedoch sind diese teilweise sehr gut versteckt.

3.6 SASUnit

Bei SASUnit handelt es sich um ein komplexes Validierungsframework, was wahrscheinlich eher als Tool verwendet wird und weniger, um dieses selbst anzupassen. Es ist seit Jahren in aktiver Wartung und beinhaltet sehr viel Funktionalität und Dokumentation. Es ist ein sehr nützliches Tool für Qualitätschecks und Validierungen von SAS Programmen und Makros.

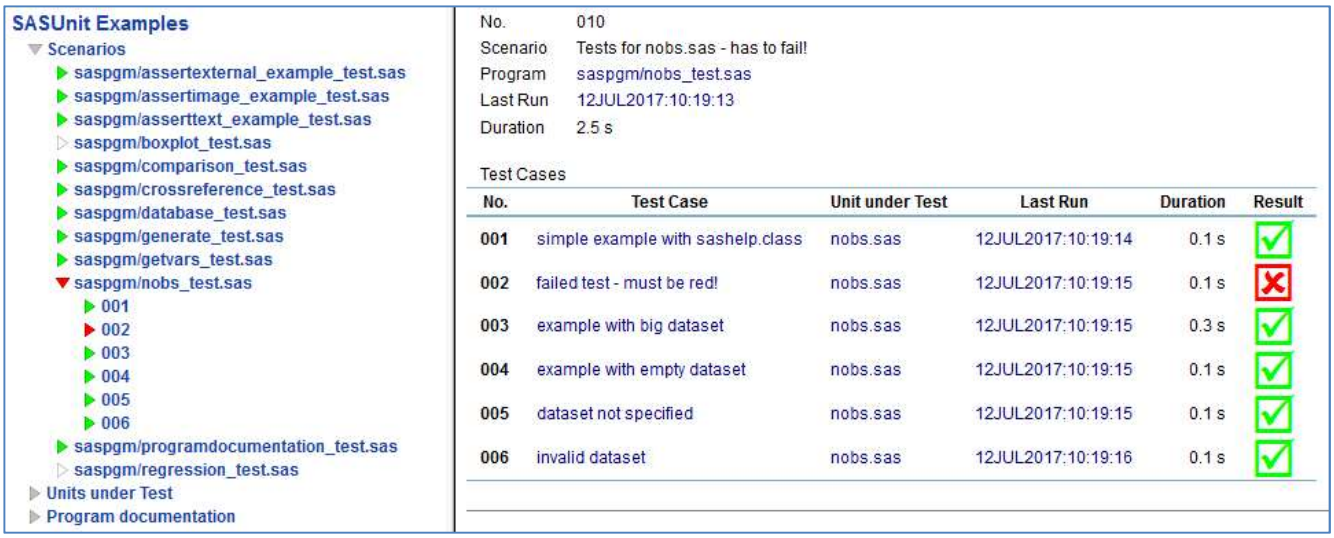

Abbildung 10: Beispiel von SASUnit

3.7 Reindeer – Render SAS Results

Das Tool Reindeer ist ein Tool, um SAS Ergebnisse nach Word zu rendern. Listings, RTFs und Graphiken, die mit SAS erzeugt wurden, können mit diesem VBA Tool sehr einfach in ein vorhandenes und formatiertes Word Template zu überführen. Dieses Open Source Tool wurde von Katja Glass Consulting entwickelt und von ClinStat GmbH gesponsort.

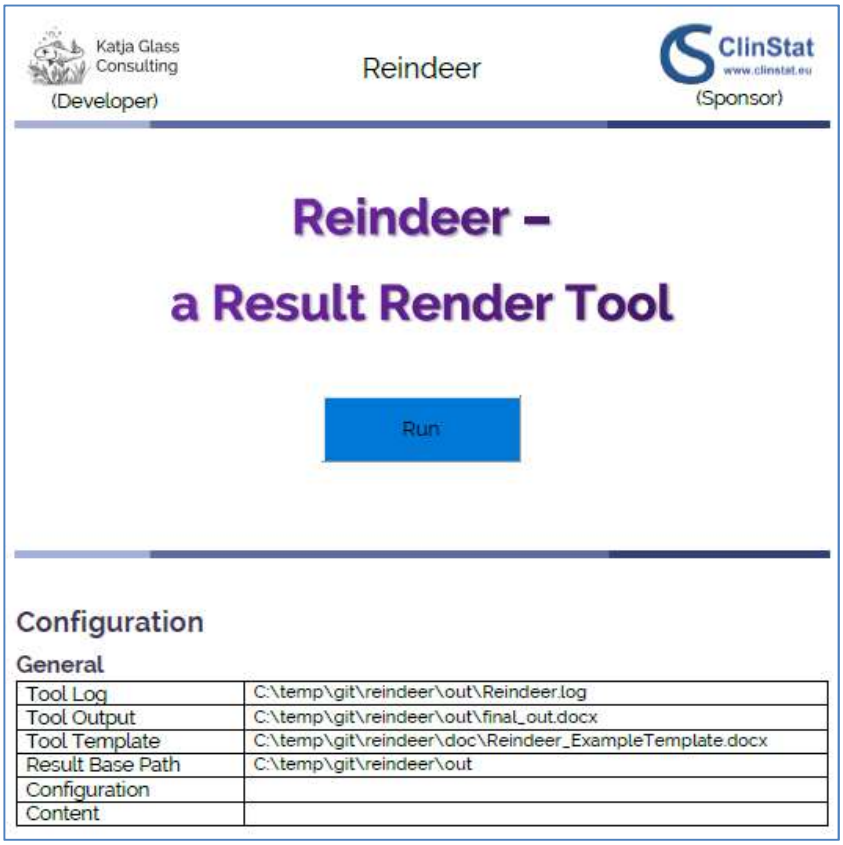

Abbildung 11: Reindeer User Interface in Word

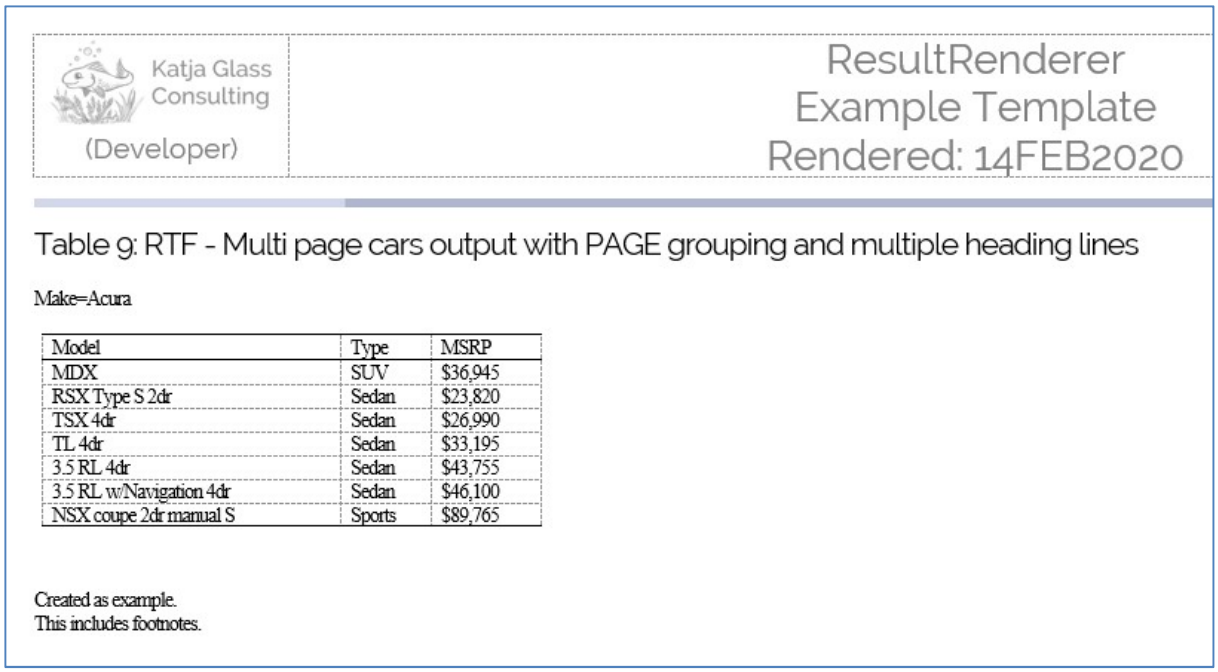

Abbildung 12: SAS Ergebnisse nach Word eingezogen

4 Überlegungen

Es gibt verschiedene Open Source Lösungen, die gefunden und verwendet werden können. Es wäre ideal, wenn noch weitere entwickelt und veröffentlicht werden.

4.1 Anwendungen von Open Source

Open Source Lösungen können meistens sehr leicht verwendet werden. Die jeweilige Lizenz ist zu berücksichtigen. Es gibt diverse Übersichten, wo die verschiedenen Lizenzen näher erläutert werden[5].

Wenn die Lizenz nicht gegeben ist, beispielsweise, weil der Quellcode auf Internetseiten wie Blogs oder in SAS Papers oder Posts zu finden ist, so gelten hier die Copyright Regeln. Hier müsste also der Autor nach der Berechtigung gefragt werden, ob der Code verwendet werden kann.

Jedoch kann man auch eine „Idee“ verwenden und diese selbst implementieren. Besonders die Antworten in Posts sind hier sehr hilfreich. Wenn es sich um „Basis-Implementierungen“ handelt, also davon ausgegangen werden kann, dass es keine Kreativität erfordert, so kann dieses auch direkt verwendet werden. Ein PROC SORT Beispiel kann üblicherweise als einfach angesehen werden.

[5] Siehe Literaturquelle [2]

4.2 Herausforderungen bei Open Source

Die Erstellung von Funktionalität ist bei Open Source meistens nicht die Schwierigkeit. Die Herausforderungen liegen eher im Bereich der meist weniger vorhandenen Dokumentation, Kommunikation, der Wartung, Validierung und der Qualität.

Dies liegt vor allem an der Motivation. Da Open Source meistens in der Freizeit erstellt wird, ist das Interesse für die Dokumentationserstellung sehr gering. Möchte man hier bessere Dokumentationen und Qualitätsprüfungen haben, so könnte dies allerdings mit Businesskonzepten verknüpft werden.

4.3 Open Source Fördern

Um mehr Open Source Lösungen zu erhalten, müsste folgendes realisiert werden:

- Internen & Externen Open Source gestatten
- Studentische Arbeiten veröffentlichen
- Open Source Arbeitsgruppen gründen und mitarbeiten
- In Open Source investieren

Derzeit ist unklar, ob Mitarbeiter Source Code, selbst wenn sie diesen in ihrer Freizeit schreiben, einfach veröffentlichen können. Hierfür wäre es ideal, wenn Firmen dies aktiv gestatten. Dies gilt ebenso für externe Mitarbeiter.

Es ist ebenso günstig, weitere Arbeiten öffentlich zu machen. Hierzu gehört neben Studienarbeiten auch Inhalte von Open Source Arbeitsgruppen, die immer mehr gebildet werden. Hierbei ist besonders wichtig, neben der Funktionalität aber auch Dokumentation mit anzubieten.

Und letztlich öffnen Investitionen in dem Bereich die höchsten und effizientesten Möglichkeiten, mehr Open Source zu erstellen. Besonders wenn die Motivation auch kommerzieller Natur ist, wird die Motivation für Problembereiche wie Dokumentation und Qualitätssicherung erhöht.

5 Ausblick

Open Source bietet viele Möglichkeiten der Arbeitserleichterung. Das Erlernen, das Verwenden oder das Weiterentwickeln von Lösungen und Methoden wird erheblich von Open Source unterstützt. Es gibt vielfältige Lösungen die mit dem Portal[6] einfach gefunden werden können. Es liegt nun an jedem einzelnen, dieses Potential auszuschöpfen.

[6] Siehe Literaturquelle [3]

Literatur

[1] Open Source Beschreibung, Wikipedia, https://de.wikipedia.org/wiki/Open_Source

[2] Licenses, Choose an open source license, https://choosealicense.com/licenses/

[3] Such Portal, Open Source Portal for Clinical Study Evaluations, https://www.glacon.eu/portal/

[4] PhUSE GitHub, https://github.com/phuse-org/phuse-scripts

[5] PhUSE GitHub Boxplot Beispielprogramm, https://github.com/phuse-org/phuse-scripts/blob/master/whitepapers/WPCT/WPCT-F.07.01.sas