main analytics

Nimm 2: SAS und R

KSFE virtuell: 13-April-2021

www.mainanalytics.de

main analytics

Carolyn Cook

- Statistikerin
- 4 Jahre Erfahrung in Statistik und statistischer Programmierung in klinischen Studien
- Erfahrung mit SAS durch die Arbeit und R in der Universität

Sven Wichmann

- Statistischer Programmierer
- Arbeit im Bereich Data Management, Arzneimittelsicherheit und Auswertung klinischer Studien
- Arbeitswerkzeug: SAS

Übersicht

- Gründe für R und SAS
- Nutzung von SAS aus R-Studio
- Beispiele
- Vor- und Nachteile

Fokus

- SAS-Programme von R aus benutzen
  - R Funktion zum Aufruf von SAS Programmen
  - Nutzung der SAS Log-Dateien
  - Ergebnisse von Prozeduren beider Pakete vergleichen
  - Erzeugung klassischer LST-outputs
- Von SAS aus R-Programme mittels PROC IML zu verwenden ist nicht Bestandteil dieses Tipps und Tricks Beitrags

Gründe für R und SAS

- Standard im Universitären- und Wissenschaftlichem Umfeld
- Open Source
- Erweiterbar
- Große Community

- Standard in großen Unternehmen
- Häufig in validierten, ganzheitlichen statistischen Umgebungen
- Support durch SAS und große SAS-Community

Gründe für R und SAS

- Große Bandbreite an Paketen
- Neueste statistische Methoden verfügbar
- Einfache Erzeugung komplexer Grafiken

- Validierte und erprobte statistische Prozeduren
- Datastep für leichte “Anpassungen” und “Modifikationen” von Daten

Nutzung von SAS aus R

```
callSAS <- function (f, path=getwd()) {

  if (grepl(".sas$", f)) {
    exeFile <- "D:/Apps/SASv9/SASFoundation/9.4/sas.exe"
    sasFile <- sprintf("%s/%s", path, f)
    logFile <- sprintf("%s/%s", path, gsub(".sas$", ".log", f))
    cmd <- sprintf("\"%s\" -sysin \"%s\" -log \"%s\" -print \"%s\"",
                   exeFile,
                   sasFile,
                   logFile,
                   logFile
    )
    system(cmd, invisible=FALSE)
    cat(readLines(sasFile), sep="\n")
    if (length(grep("^ERROR:", readLines(logFile)) > 0)) {
      warning(sprintf("\nERROR: See line %s in %s.", grep("^ERROR:", readLines(logFile)), logFile))
    } else {
      message(sprintf("No errors\nSee %s for details.", logFile))
    }
  }
}
callSAS("sasprog.sas")
```

Nutzung von SAS aus R

```
callSAS <- function (f, path=getwd()) {
```


```
}
callSAS("sasprog.sas")
```

Nutzung von SAS aus R

```
callSAS <- function (f, path=getwd()) {

  if (grepl(".sas$", f)) {

  }

}
callSAS("sasprog.sas")
```

Prüfung ob eine Datei mit der Endung “.sas” übergeben wurde*

* Auf ein “else” für den Fall das keine “.sas” Datei übergeben worden ist wurde aus Platzgründen verzichtet.

Nutzung von SAS aus R

```
callSAS <- function (f, path=getwd()) {

  if (grepl(".sas$", f)) {
    exeFile <- "D:/Apps/SASv9/SASFoundation/9.4/sas.exe"

  }
}
callSAS("sasprog.sas")
```

Pfad zur SAS Ausführungsdatei

Hinweis: Innerhalb von R wird generell mit einem “Vorwärts Schrägstrich” gearbeitet. In Windows Umgebungen ist allerdings der “Rückwärts Schrägstrich” üblich. Alternativ kann auch der “\” verwendet werden. Dieser muss jedoch mittels eines weiteren “\” maskiert werden.
Der Pfad mit “\” würde wie folgt aussehen:
“D:\\Apps\\SASv9\\SASFoundation\\9.4\\sas.exe”

Nutzung von SAS aus R

```
callSAS <- function (f, path=getwd()) {

  if (grepl(".sas$", f)) {
    exeFile <- "D:/Apps/SASv9/SASFoundation/9.4/sas.exe"
    sasFile <- sprintf("%s/%s", path, f)

  }
}
callSAS("sasprog.sas")
```

Vollständiger Pfad zum SAS Programm gebildet aus “path” und “f”.
Z.B.: “C:\Users\wichmann\Rwork\sasprog.sas“

Path ist in der Funktionsdefinition mit „`getwd`“ definiert. „getwd“ gibt das R Arbeitsverzeichnis wieder.
Beispielsweise „C:\Users\wichmann\Rwork“

Nutzung von SAS aus R

```
callSAS <- function (f, path=getwd()) {

  if (grepl(".sas$", f)) {
    exeFile <- "D:/Apps/SASv9/SASFoundation/9.4/sas.exe"
    sasFile <- sprintf("%s/%s", path, f)
    logFile <- sprintf("%s/%s", path, gsub(".sas$", ".log", f))

  }
}
callSAS("sasprog.sas")
```

Vollständiger Pfad zum SAS Log file, gebildet aus “path” und “f”. Die Funktion gsub ersetzt die Endung “.sas” durch “.log”

Ergebnis:
“C:\Users\wichmann\Rwork\sasprog.log“

Nutzung von SAS aus R

```
callSAS <- function (f, path=getwd()) {

  if (grepl(".sas$", f)) {
    exeFile <- "D:/Apps/SASv9/SASFoundation/9.4/sas.exe"
    sasFile <- sprintf("%s/%s", path, f)
    logFile <- sprintf("%s/%s", path, gsub(".sas$", ".log", f))
    cmd <- sprintf("\"%s\" -sysin \"%s\" -log \"%s\" -print \"%s\"",
                   exeFile,
                   sasFile,
                   logFile,
                   logFile
    )









  }
}
callSAS("sasprog.sas")
```

Erstellung des SAS Aufrufs basierend auf den vorher erstellten Parametern von SAS Programm Pfad, SAS Programm und Log-Datei.
Ergebnis:
\"D:/Apps/SASv9/SASFoundation/9.4/sas.exe\" -sysin \"C:\Users\wichmann\Rwork\sasprog.sas\" -log \"C:\Users\wichmann\Rwork\sasprog.log\" -print \"C:\Users\wichmann\Rwork\sasprog.log\"

Nutzung von SAS aus R

```
callSAS <- function (f, path=getwd()) {

  if (grepl(".sas$", f)) {
    exeFile <- "D:/Apps/SASv9/SASFoundation/9.4/sas.exe"
    sasFile <- sprintf("%s/%s", path, f)
    logFile <- sprintf("%s/%s", path, gsub(".sas$", ".log", f))
    cmd <- sprintf("\"%s\" -sysin \"%s\" -log \"%s\" -print \"%s\"",
                   exeFile,
                   sasFile,
                   logFile,
                   logFile
    )
    system(cmd, invisible=FALSE)

  }
}
callSAS("sasprog.sas")
```

Systemaufruf des vorher definierten Befehls (cmd).

Nutzung von SAS aus R

```
callSAS <- function (f, path=getwd()) {
```

Einlesen des SAS-Logs (Zeile für Zeile).
Für den Fall, dass im SAS-Log am Zeilen-Anfang ein “ERROR:” vorhanden ist, wird die Zeilennummer, in der die Fehlermeldung steht, in der R-Konsole mit ausgegeben.
Für den Fall, dass das SAS Programm keine Fehler enthält, wird der Text „No errors“ und ein Verweis auf die Log-Datei ausgegeben.

```
    cat(readLines(sasFile), sep="\n")
    if (length(grep("^ERROR:", readLines(logFile)) > 0)) {
      warning(sprintf("\nERROR: See line %s in %s.", grep("^ERROR:", readLines(logFile)), logFile))
    } else {
      message(sprintf("No errors\nSee %s for details.", logFile))
    }
  }
}
callSAS("sasprog.sas")
```

Nutzung von SAS aus R

- Direkte Verfügbarkeit des SAS-Logs (`readLines(logFile)`)
- Erweiterung der Auswertung des SAS-Logs möglich
  - Analyse von Rückgabewerten (Nutzung der globalen SAS Variable “sysinfo”)
  - Analyse benutzerdefinierter Log-Ausgaben über put statements

Nutzung von SAS aus R

```r
library(SASxport)
write.xport(rdata, file="U:/r-test/rdata.xpt")
```

Daten aus R herausschreiben

```
* Get R data *;
libname rdata xport "U:/r-test/rdata.xpt";

* Create lst output *;
ods listing file="U:/r-test/compare-output.lst" gpath="U:/r-test/";

* Compare data from SAS and R *;
proc compare data=sasdata comp=rdata.rdata;
   informat _all_;
   format _all_;
run;
%let compreturn=&sysinfo.;
%put %str(COMPARE)%str( RETURN CODE:);
%put &compreturn.;
```

R Daten in SAS einlesen

SAS Output in lst-Datei schreiben

SAS und R Daten mit Proc Compare vergleichen

Proc Compare Rückgabecode in log schreiben

Nutzung von SAS aus R

```
if (length(grep("^COMPARE RETURN CODE:", readLines(logFile)) > 0)) {
  #message(sprintf("\nFound in Line: %s", grep("^COMPARE RETURN CODE:", readLines(logFile)), logFile))
  messageline <- as.numeric(grep("^COMPARE RETURN CODE:", readLines(logFile)))
  returncode <- scan(file=logFile, what=character(), skip=messageline+1, nlines=1)
  #returncode <- readLines(logFile,n=messageline)
  message(sprintf("SAS compare return code is %s.", returncode))
}
```

Erweiterung der R-Funktion callSAS zum Einlesen des PROC COMPARE Rückgabecodes

Ausgabe der R-Konsole:
Der von PROC COMPARE erzeugte Rückgabewert wird in R ausgegeben und kann im weiteren Verlauf verwendet werden


```
Read 1 item
SAS compare return code is 0.
No errors
See U:/r-test//compare-method.log for details.
```

Stärken beider Pakete nutzen - Beispiele

- Datenaustausch zwischen SAS und R
  - Siehe u.a. 20. KSFE Vortrag von Luzia Tinten „Synergien einer Verbindung von SAS und R“
- SAS Datastep zur Aufbereitung der Daten
- PROC COMPARE (SAS) zum Vergleich von Ergebnisse
  - PROC COMPARE return codes
- Weitere Möglichkeiten der Erweiterung:
  - R-Shiny für interaktive Dashboards

Stärken beider Pakete nutzen - Beispiele

- R zur Erzeugung von Grafiken verwenden
- PROC REPORT (SAS) zur Erzeugung von klassischen LST-Dateien
- Kritische Prozeduren in SAS und R unabhängig voneinander programmieren
  - Siehe u.a. PHUSE Vortrag von Elvira Erhardt, Corinna Miede und Christian Stock: „Joint Model Implementations in SAS and R“

Vorteil 1: Statistische Robustheit

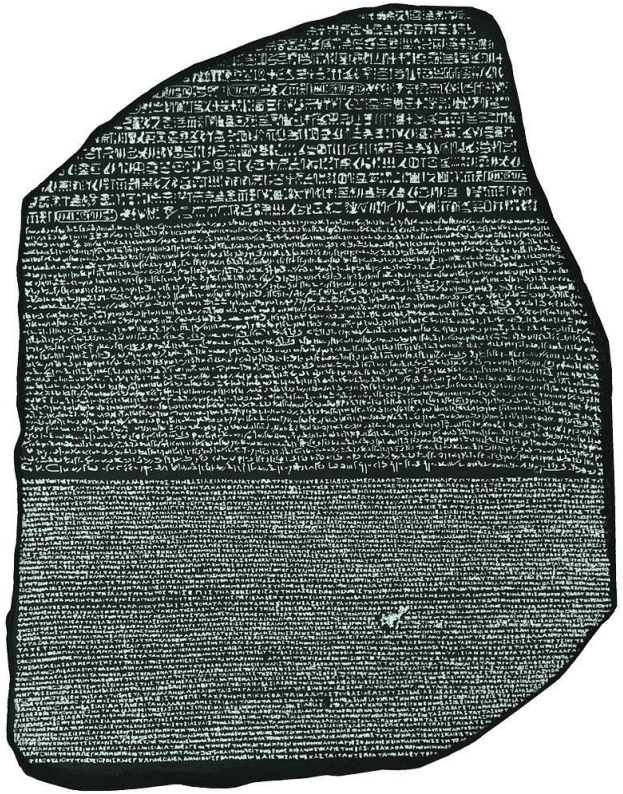
- Falls eine Analyse gemacht und validiert werden muss wo:
  - Ein (unvalidiertes) R Paket für das Hauptprogramm benutzt werden soll, bestimmte Teile könnten aber auch mit SAS validiert werden

  Oder

  - Es gibt nur eine Möglichkeit, die Analyse in SAS zu machen, diese muss aber unabhängig validiert werden
    - Siehe u.a. PHUSE Vortrag „Joint modelling implementation in SAS and R: apples and oranges or unequal twins?„
    - Durch die Validierung werden Programmierer gezwungen, die jeweiligen Prozesse auf einer detaillierteren Ebene zu verstehen

Vorteil 2: R lernen durch Validierung

- Falls jemand mit SAS programmiert und ein Kollege mit R das Programm validiert, lernt der SAS Programmierer nebenbei R
- Die jeweils unbekanntere Sprache kann Stück für Stück weiter erlernt werden

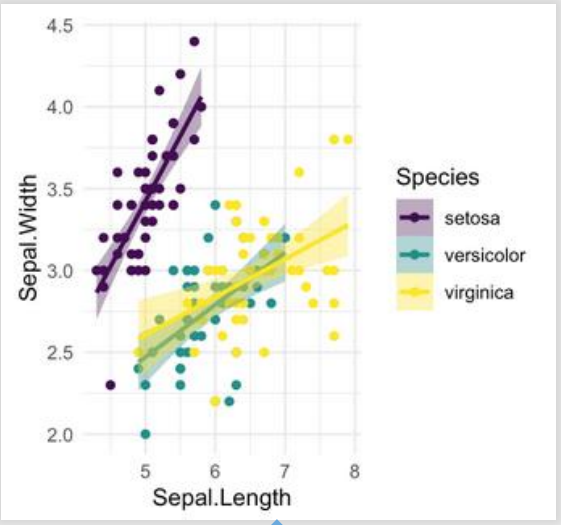


Vorteil 3: SAS und R für ein Ziel

- Gilt nicht unbedingt, wenn mit R und SAS unabhängig programmiert wird
- Aufteilung von einem Programm in einen SAS und einen R Teil, um die Stärken der jeweiligen Sprache zu nutzen
  - Z.B. Daten mit SAS analysieren und nachher Graphiken von den Daten in R erstellen
  - "Best of both worlds"

```
DATA TEMP;
INPUT ID $ NAME $ SALARY DEPARTMENT $;
comm = SALARY*1.25;
LABEL ID = 'Employee ID' comm = 'COMMISION';
DATALINES;
1 Tom 5000 IT
2 Harry 6000 Operations
3 Michelle 7000 IT
4 Dick 8000 HR
5 John 9000 Finance
;
RUN;
PROC MEANS;
RUN;
PROC PRINT DATA = data_set;
OPTIONS;
```

	ids	bday	Gender	Rank	State
1	23643	22NOV1990	0	.	
2	30953	23AUG1995	1	1	In state
3	20531	29DEC1994	0	1	In state
4	22416	.	0	1	In state
5	41227	19APR1994	1	2	In state
6	37301	06JUN1993	1	2	Out of state
7	39181	17MAY1992	0	3	In state
8	22652	04DEC1989	1	3	In state
9	35684	29JUN1991	0	4	In state
10	43344	26MAR1993	0	.	In state




```
# Discrete color. use the argument discrete = TRUE
ggplot(iris, aes(Sepal.Length, Sepal.Width))+
  geom_point(aes(color = Species)) +
  geom_smooth(aes(color = Species, fill = Species), method = "lm") +
  scale_color_viridis(discrete = TRUE, option = "D")+
  scale_fill_viridis(discrete = TRUE)
```

Nachteile

- Zwei System müssen gepflegt werden
- Der Datenaustausch zwischen den Plattformen kann zu Problemen führen
- Know-How beider Welten wird benötigt
- Komplexität kann übermäßig ansteigen
- Nachvollziehbarkeit
  - (R Log-Datei)
  - (Zwei Log-Dateien)

main analytics

Viel Spaß beim Abtauchen in neue Welten!