SQL – Brücken zwischen zwei Welten

KSFE 30. März 2023

Renate Scheiner-Sparna, AliraHealth
Jörg Sahlmann, Universitätsklinikum Freiburg,
Institut für
Medizinische Biometrie und Statistik

AliraHealth

At Alira Health, **our mission is to humanize healthcare**

We complement your expertise with a full spectrum of patient-centric data and tech-enabled services to uncover opportunity, accelerate innovation, and improve outcomes for patients around the world.

PRODUCT DEVELOPMENT

REGULATORY

CLINICAL

BIOMETRICS

MARKET ACCESS

MANAGEMENT CONSULTING

TRANSACTION ADVISORY

PATIENT ENGAGEMENT

REAL-WORLD EVIDENCE

Best-of Proc SQL

1. Kenne Deine Daten!

> **“Wie heißt nochmal diese Spalte…?”**

- Man muss nicht gleich die ganze Tabelle abfragen!

```
1  proc sql;
2    describe table sashelp.heart;
3  quit;
```

- Log-Ausgabe:

```
5    proc sql;
6      describe table sashelp.heart;
NOTE: SQL table SASHELP.HEART was created like:

create table SASHELP.HEART( label='Framingham Heart Study' bufsize=65536 )
  (
   Status char(5),
   DeathCause char(26) label='Cause of Death',
   AgeCHDdiag num label='Age CHD Diagnosed',
   Sex char(6),
   AgeAtStart num label='Age at Start',
   Height num,
   Weight num,
   Diastolic num,
   Systolic num,
   MRW num label='Metropolitan Relative Weight',
   Smoking num,
   AgeAtDeath num label='Age at Death',
   Cholesterol num,
   Chol_Status char(10) label='Cholesterol Status',
   BP_Status char(7) label='Blood Pressure Status',
   Weight_Status char(11) label='Weight Status',
   Smoking_Status char(17) label='Smoking Status'
  );

NOTE: Writing HTML Body file: sashtml.htm
7    quit;
NOTE: PROCEDURE SQL used (Total process time):
      real time           0.10 seconds
      cpu time            0.07 seconds
```

- Das funktioniert mit jeder Tabelle! Ich muss sie nicht öffnen oder printen, wenn ich nur den Namen der Variablen brauche.

Brücken zwischen SAS und R

1. Kenne Deine Daten

- Der Umweg über sql und describe ist in R nicht notwendig, um die Variablennamen zu ermitteln.
- Das ist mit names() bzw. sort(names()) zu bewerkstelligen.

```
names(df_heart)
```

```
 [1] "Status"         "DeathCause"     "AgeCHDdiag"     "Sex"
 [5] "AgeAtStart"     "Height"         "Weight"         "Diastolic"
 [9] "Systolic"       "MRW"            "Smoking"        "AgeAtDeath"
[13] "Cholesterol"    "Chol_Status"    "BP_Status"      "Weight_Status"
[17] "Smoking_Status"
```

```
sort(names(df_heart))
```

```
 [1] "AgeAtDeath"     "AgeAtStart"     "AgeCHDdiag"     "BP_Status"
 [5] "Chol_Status"    "Cholesterol"    "DeathCause"     "Diastolic"
 [9] "Height"         "MRW"            "Sex"            "Smoking"
[13] "Smoking_Status" "Status"         "Systolic"       "Weight"
[17] "Weight_Status"
```

Brücken zwischen SAS und R

1. Kenne Deine Daten

> Die unterschiedlichen SQL-Server bieten unterschiedliche Möglichkeiten, die Variablen einer Tabelle abzufragen.

```
con_orsales <- DBI::dbConnect(RSQLite::SQLite(), "data/orsales.sqlite")

rs2 <- dbSendQuery(con_orsales, "
PRAGMA table_info(orsales)
")

dbFetch(rs2)
```

```
##   cid                name type notnull dflt_value pk
## 1   0                Year REAL       0         NA  0
## 2   1             Quarter TEXT       0         NA  0
## 3   2        Product_Line TEXT       0         NA  0
## 4   3    Product_Category TEXT       0         NA  0
## 5   4       Product_Group TEXT       0         NA  0
## 6   5            Quantity REAL       0         NA  0
## 7   6              Profit REAL       0         NA  0
## 8   7 Total_Retail_Price REAL       0         NA  0
```

```
dbDisconnect(con_orsales)
```

Brücken zwischen SAS und R

1. Kenne Deine Daten

- SQLite bietet mit der Tabelle sqlite_schema eine Struktur, die den Dictionary Tables nahekommt.

- Das ist dann kein Feature von R.

```
con_orsales <- DBI::dbConnect(RSQLite::SQLite(), "data/orsales.sqlite")

rs2 <- dbSendQuery(con_orsales, "
SELECT sql
FROM sqlite_schema
WHERE name = 'orsales';
")

dbFetch(rs2)
```

```
##
sql
## 1 CREATE TABLE `orsales` (\n  `Year` REAL,\n  `Quarter` TEXT,\n  `Product_Line` TEXT,\n
`Product_Category` TEXT,\n  `Product_Group` TEXT,\n  `Quantity` REAL,\n  `Profit` REAL,\n
`Total_Retail_Price` REAL\n)
```

Best-of Proc SQL

2. Abfrage von Dictionary Tables

- **SAS Metadaten stehen dem Programmierer zur Verfügung.**
- **Das Original liegt im Libname Dictionary, der automatisch zur Verfügung steht.**
- **Die Kopie liegt als View (mit den bekannten Eigenschaften eines Views) im Libname SASHELP.**

- **Mit Proc SQL haben wir eine besonders günstige Möglichkeit des Zugriffs.**
  - **Das geübte Auge sieht sofort, dass “D i c t i o n a r y” zu lang für einen Libname ist!**

    1 2 3 4 5 6 7 8 9 10

  - **Spezialfall, und sogar besonders effektiv.**

    → Hier ist kein Data Step möglich!

    → Die Runtime ist sehr competetiv!

Best-of Proc SQL

2. Abfrage von Dictionary Tables

Abfrage von SASHELP Views:

```
>     proc sql noprint;
        create table TAB1 as select * from SASHELP.VTABLE
          where libname='WORK';
      quit;
```

```
real time          0.01 seconds
cpu time           0.01 seconds
```

- Die real time und cpu time sind abhängig von der Menge der vorhandenen Metadaten und der Komplexität der Abfrage.
- In diesem Beispiel wurde jeweils eine neu geöffnete SAS Sitzung mit denselben Serverbedingungen verwendet.
- Diese Performance ist schon nicht schlecht!

Best-of Proc SQL

2. Abfrage von Dictionary Tables

- **Abfrage von DICTIONARY Table**

```
proc sql noprint;
  create table TAB1 as select * from DICTIONARY.TABLES
    where libname='WORK';
quit;
```

```
real time              0.00 seconds
cpu time               0.00 seconds
```

→ Die Runtime ist sehr competetiv!

→ Hier ist kein Data Step möglich!

- **Versuchen wir noch den Data Step**:

```
1    data TAB1;
2      set DICTIONARY.TABLES;
ERROR: Libref 'DICTIONARY' exceeds 8 characters.
3      where libname='WORK';
4    run;

NOTE: The SAS System stopped processing this step because of errors.
WARNING: The data set WORK.TAB1 may be incomplete.  When this step was stopped there were 0 observations and 0 variables.
NOTE: DATA statement used (Total process time):
      real time           0.01 seconds
      cpu time            0.01 seconds
```

Brücken zwischen SAS und R

2. Abfrage von Dictionary Tables

- Dictionary Tables als solche gibt es in R nicht, weil die Objekte in R unterschiedliche Strukturen haben, die nicht in eine Metastruktur passen.
- Es gibt unterschiedliche Befehle, mit denen man Objekte analysieren und näher beschreiben kann.

```
# Gib alle Object aus
ls()
# Finde die Dimensionen eines Datensatzes
dim(df_heart)
# Untersuche die Struktur
str(df_heart)
# Schau die ersten 6 Zeilen an
head(df_heart)
# Schau die letzten 6 Zeilen an
tail(df_heart)
```

```
# Anzeige der Variablennamennames
(df_heart)
# Alle Ausprägungen einer kategoriellen Variablen darstellen
unique(df_heart$DeathCause)
# Deskriptive Statistik einer numerischen Variablen darstellen
summary(df_heart$AgeAtStart)
# View the data set
View(df_heart)
```

Best-of Proc SQL

3. Join mit BETWEEN

- **Numerische Wertebereiche können während des Joins berücksichtigt werden.**
- **Sehr nützlich, z.B. zum Berücksichtigen von Zeitfenstern andere Abhängigkeiten von Datumsvariablen.**
- **Hier ein Beispiel, das nur das Jahr verwendet:**

```
10 data new;
11    newvar=1;
12    year1=2000;
13    year2=2001;
14 run;
15
16 proc sql noprint;
17    create table orsales as select a.*, b.newvar
18    from sashelp.orsales as a
19    left join new as b on (a.year BETWEEN b.year1 and b.year2)
20    order by a.year;
21 quit;
```

VIEWTABLE: WORK.ORSALES

	Year	Product_Group	Quantity	newvar
225	1999	Tracker Clothes	2624	.
226	1999	T-Shirts	679	.
227	1999	Baseball	770	.
228	1999	Street Wear	749	.
229	2000	Green Tomato	421	1
230	2000	N.D. Gear, Kids	1676	1
231	2000	Eclipse Clothing	7332	1
232	2000	Ypsilon, Kids	310	1
...				
682	2001	Anoraks & Parkas	3311	1
683	2001	Tracker Shoes	2116	1
684	2001	Shoes	1759	1
685	2002	Winter Sports	2577	.
686	2002	Soccer	2983	.
687	2002	Basket Ball	470	.

Brücken zwischen SAS und R: SAS > R

3. Join mit BETWEEN

- Der SQL-Code, der in PROC SQL genutzt worden ist, kann in diesem Fall ohne Änderungen in R übernommen werden.
- Das erleichtert den Transfer von Teilen von SAS-Programmen in R-Programme.

```
new <- data.frame("newvar" = 1, year1 = 2000, year2 = 2001)
con_orsales <- DBI::dbConnect(RSQLite::SQLite(), "data/orsales.sqlite")
dbWriteTable(con_orsales, "new", new, overwrite = TRUE)
src_dbi(con_orsales)

rs <- dbSendQuery(con_orsales, "
create table os as
select a.*, b.newvar
from orsales as a
left join new as b on (a.year BETWEEN b.year1 and b.year2)
order by a.year
")
dbFetch(rs)

rs2 <- dbSendQuery(con_orsales, "
PRAGMA table_info(orsales)
")
dbFetch(rs2)

dbDisconnect(con_orsales)
```

Brücken zwischen SAS und R : R > SAS

3. Join mit BETWEEN

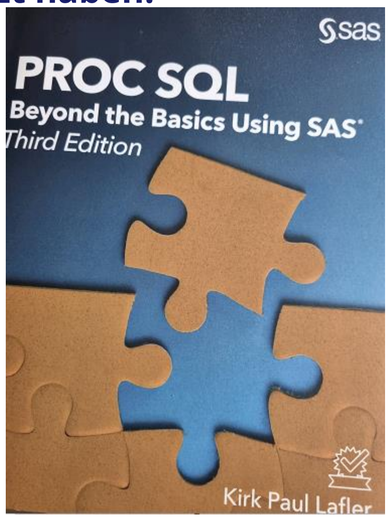
- Der tidyverse/dplyr-Code, der in R genutzt worden ist, kann in diesem Fall ohne Änderungen in PROC SQL in R übernommen werden.
- Das erleichtert den Transfer von Teilen von R-Programmen in SAS-Programme.

```
> df %>%
+   group_by(DeathCause) %>%
+   summarize(cnt = n()) %>%
+   ungroup() %>%
+   arrange(desc(cnt))
# Source:     SQL [6 x 2]
# Database:   sqlite 3.39.4 [E:\projects\ksfe_2023\sql_R\data\heart.sqlite]
# Ordered by: desc(cnt)
  DeathCause                    cnt
  <chr>                       <int>
1 ""                           3218
2 "Coronary Heart Disease"      605
3 "Cancer"                      539
4 "Cerebral Vascular Disease"   378
5 "Other"                       357
6 "Unknown"                     112
> df %>%
+   group_by(DeathCause) %>%
+   summarize(cnt = n()) %>%
+   ungroup() %>%
+   arrange(desc(cnt)) %>%
+   show_query()
<SQL>
SELECT `DeathCause`, COUNT(*) AS `cnt`
FROM `heart`
GROUP BY `DeathCause`
ORDER BY `cnt` DESC
```

Best-of Proc SQL

4. Erzeugen von Makrovariablen in parallelen Serien

- **Verwendung in Macro Loops oder Arrays**
- **Sehr einfaches Beispiel aus dem Buch, etwas abgewandelt auf die ORSALES Tabelle, die wir schon genutzt haben:**




```
33  proc sql noprint;
34     select trim(PRODUCT_LINE), QUARTER, QUANTITY
35       into :_PRODUCT_LINE separated by '#',
36            :_QUARTER separated by '#',
37            :_QUANTITY separated by '#'
38         from orsales;
39  quit;
40  %put &=_PRODUCT_LINE;
41  %put &=_QUARTER;
42  %put &=_QUANTITY;
```

```
230  %put &=_PRODUCT_LINE;
_PRODUCT_LINE=Children#Clothes & Shoes#Outdoors#Sports#Children#Clothes & Shoes#Outdoors#Sports
231  %put &=_QUARTER;
_QUARTER=2000Q1#2000Q1#2000Q1#2000Q1#2000Q2#2000Q2#2000Q2#2000Q2#2000Q3#2000Q3#2000Q3#2000Q3#20
232  %put &=_QUANTITY;
_QUANTITY=90#355#722#3462#1676#4226#2018#1182#310#421#4243#1199#239#2933#1072#2729
```

Brücken zwischen SAS und R

4. Erzeugen von Makrovariablen in parallelen Serien

- Da es in R Variablen außerhalb von Strukturen wie Data Frames gibt, ist die Erzeugung von Makrovariablen wie bei SAS nicht unbedingt notwendig.
- Nichtsdestotrotz macht sich das Fehlen von Textersetzungen, wie sie mit Makrovariablen gemacht werden können, manchmal schmerzlich bemerkbar.
- Es gibt Möglichkeiten, aus Strings Variablennamen zu generieren und damit Variablen und ihre Namen programmatisch zu erzeugen.

```
> a <- "variable_name"
> assign(a, 3)
> print(variable_name)
[1] 3
```

Best-of Proc SQL

5. Join ohne vorherige Sortierung

- Nützlich vor allem bei großen Datenmengen: Jede Sortierung kostet Zeit im Programmlauf.
- Sehr einfach und effektiv kann man sogar am Ende die Daten in Proc SQL sortieren, oft zeitsparender.

```
55 proc sort data=sashelp.zipcode out=zip;
56    by STATECODE COUNTYNM CITY;
57 run;
```

```
242  proc sort data=sashelp.zipcode out=zip;
243    by STATECODE COUNTYNM CITY;
244  run;

NOTE: There were 41140 observations read from the data set SASHELP.ZIPCODE.
NOTE: The data set WORK.ZIP has 41140 observations and 21 variables.
NOTE: PROCEDURE SORT used (Total process time):
      real time           0.42 seconds
      cpu time            0.09 seconds
```

- Fiktive Group Variable
- Unsortierte Datasets

VIEWTABLE: WORK.GROUPVAR

	CITY	STATECODE	COUNTYNM	GROUP
1	Holtsville	NY	Suffolk	A
2	Holtsville	NY	Suffolk	A
3	Adjuntas	PR	Adjuntas	B
4	Aguada	PR	Aguada	B
5	Aguadilla	PR	Aguadilla	B
6	Aguadilla	PR	Aguadilla	B
7	Aguadilla	PR	Aguadilla	B
8	Maricao	PR	Maricao	B
9	Anasco	PR	Anasco	B

Best-of Proc SQL

5. Join ohne vorherige Sortierung

- **Es geht ohne Sortierung!**

```
67  proc sql noprint;
68     create table zip2 as select a.*, b.GROUP
69     from sashelp.zipcode as a
70     left join groupvar as b
71     on a.STATECODE=b.STATECODE and
72        a.COUNTYNM=b.COUNTYNM and
73        a.CITY=b.CITY
74     ;
75  quit;
```

```
275  proc sql noprint;
276    create table zip2 as select a.*, b.GROUP
277    from sashelp.zipcode as a
278    left join groupvar as b
279    on a.STATECODE=b.STATECODE and
280       a.COUNTYNM=b.COUNTYNM and
281       a.CITY=b.CITY
282    ;
NOTE: Table WORK.ZIP2 created, with 475082 rows and 22 columns.

283  quit;
NOTE: PROCEDURE SQL used (Total process time):
      real time           0.50 seconds
      cpu time            0.51 seconds
```

- **Der Zeitaufwand für die Sortierung in Proc SQL ist im wahrsten Sinne des Wortes zu vernachlässigen**

```
254  proc sql noprint;
255    create table zip2 as select a.*, b.GROUP
256    from sashelp.zipcode as a
257    left join groupvar as b
258    on a.STATECODE=b.STATECODE and
259       a.COUNTYNM=b.COUNTYNM and
260       a.CITY=b.CITY
261    order by STATECODE, COUNTYNM, CITY;
NOTE: Table WORK.ZIP2 created, with 475082 rows and 22 columns.

262  quit;
NOTE: PROCEDURE SQL used (Total process time):
      real time           0.50 seconds
      cpu time            0.51 seconds
```

Best-of Proc SQL

6. Join über Variablen mit unterschiedlichen Namen

- **MYCITY, CITY**

VIEWTABLE: WORK.GROUPVAR

	MYCITY	STATECODE	COUNTYNM	GROUP
1	Holtsville	NY	Suffolk	A
2	Holtsville	NY	Suffolk	A
3	Adjuntas	PR	Adjuntas	B
4	Aguada	PR	Aguada	B

VIEWTABLE: Sashelp.Zipcode (US Zipcodes; Source: zipcodedownload.com Jan 2017)

	CITY	STATECODE	COUNTYNM
1	Holtsville	NY	Suffolk
2	Holtsville	NY	Suffolk
3	Adjuntas	PR	Adjuntas
4	Aguada	PR	Aguada

```
71  proc sql noprint;
72     create table zip2 as select a.*, b.GROUP
73     from sashelp.zipcode as a
74     left join groupvar as b
75     on a.STATECODE=b.STATECODE and
76        a.COUNTYNM=b.COUNTYNM and
77        a.CITY=b.MYCITY
78     ;
79  quit;
```

Brücken zwischen SAS und R

5. Join ohne vorherige Sortierung
6. Join über Variablen mit unterschiedlichen Namen

- Durch dplyr mit seinen Join-Möglichkeiten ist man in R nicht darauf angewiesen, nach den by-Variablen zu sortieren und by-Variablen gleich lauten zu lassen.
- Sie können unterschiedlich heißen.
- by = (“var.x” = “var.y”)
- Bei gleichlautenden Namen kann man sich das Gleichheitszeichen sparen.
- by = (“var”)

- Alternativ kann man auch direkt mit SQL arbeiten.
- Auch dieser SQL-Code konnte ohne Änderungen übernommen werden.

```
DROP TABLE IF EXISTS groupvar;

CREATE TABLE groupvar AS
  SELECT city AS "mycity",
         statecode,
         countynm,
         CASE
           WHEN Random() > 0 THEN "A"
           ELSE "B"
         end  AS "group2"
  FROM   zipcode


SELECT a.statecode,
       a.countynm,
       a.city,
       b.group2 AS "group"
FROM   zipcode AS a
       LEFT JOIN groupvar AS b
              ON a.statecode = b.statecode
                 AND a.countynm = b.countynm
                 AND a.city = b.mycity
```

	mycity	STATECODE	COUNTYNM	group2
	Filtern	Filtern	Filtern	Filtern
1	Holtsville	NY	Suffolk	B
2	Holtsville	NY	Suffolk	B
3	Adjuntas	PR	Adjuntas	A
4	Aguada	PR	Aguada	B
5	Aguadilla	PR	Aguadilla	A
6	Aguadilla	PR	Aguadilla	B
7	Aguadilla	PR	Aguadilla	A
8	Maricao	PR	Maricao	A
9	Anasco	PR	Anasco	A
10	Angeles	PR	Utuado	A

Zusammenfassung

Es gibt Brücken zwischen SAS und R in beide Richtungen

> Bestimmte Sonderlösungen mit SQL, die bei SAS notwendig sind, braucht man bei R aufgrund des Sprachumfangs nicht.

> Der SQL-Code kann zwischen SAS und R in beide Richtungen ausgetauscht werden.

AliraHealth

Thank You

LET'S KEEP IN TOUCH

www.alirahealth.com

info@alirahealth.com

NORTH AMERICAN OFFICES

Toronto | Canada
Boston | US
San Francisco | US

EUROPEAN OFFICES

Vienna | Austria
Paris, Bordeaux | France
Munich | Germany
Bologna, Milan, Verona | Italy
Zevenbergen | Netherlands
Barcelona | Spain
Basel | Switzerland
London | UK

ASIAN PACIFIC OFFICES

Sydney | Australia
Singapore