SAS and R

-The Odd Couple-

PhUSE - Single Day Event Frankfurt - 04. Mai 2011
Company

• HMS Analytical Software is a specialist for Information Technology in the field of Data Analysis and Business Intelligence Systems

• Profile
  – 40 employees in Heidelberg, Germany
  – SAS Institute Silver Consulting Partner for 14 years
  – Doing data oriented software projects for more than 20 years

• Technologies
  – Analytics and Data Management:
    SAS, JMP, R, Microsoft SQL Server
  – Application Development: Microsoft .NET, Java
Agenda

• Concepts behind SAS and R
• Installation and support
• How to develop programs?
• SAS-Macro vs. R functions
• Performance comparison using large data sets
• Interfaces from SAS to R and vice versa
• Software quality
• Conclusion
Concepts Behind SAS

- 1970 presentation of the Statistical Analysis Software at the NC State University
- 1976 foundation of the SAS Institute, Cary, North Carolina
- commercial software product
- largest privately-held software company in the world[1]
  - approx. 11,800 employees
  - 2010 Revenue: $2.43 billion
  - customers at more than 50,000 sites in 126 countries
- Focus:
  - data integration / data warehousing
  - business intelligence / data mining
  - research and development
  - various and compatible software components
  - license model→ you can not „buy“ SAS
Concepts Behind R

• 1993 first occurrence in a mailing list[1]
• Open Source Software under the General Public License (GPL)
  – freely available
  – around 2 million users[2]
• Focus:
  – statistical computing
  – graphics
• The R Foundation for Statistical Computing
  – official voice of the R project
  – non-profit organization seated in Vienna, Austria
  – consists of ordinary and supporting members, mainly the R Development Core-Team
  – holds and administers the copyrights of the R software and documentation.
• huge amount of freely available additional packages
  – not developed by the R Development Core-Team but by programmers from throughout the world
Motivation of the Comparison SAS vs. R

• Focus
  – SAS language and the SAS-windowing environment (Display Manager)
  – R environment

• Not included
  – e.g. SAS Enterprise Guide
  – other frontends of the SAS-Plattform
## Installation and Support

### SAS
- download-center / DVD
- long installation process
- SAS Foundation
  - PROC CORR, PLOT, ...
  - MACRO
- other modules
  - SAS STAT, GRAPH, ...

- **support:**
  - online
  - hotline (+49 6221 415 - 200)
  - training
  - newsletter

### R
- download from [CRAN](https://cran.r-project.org)
- short installation process
- R „Base“
  - recommended / base packages
- other packages
  - BioConductor, ...

- „support“ / help:
  - [mailing-list](https://lists.r-project.org)
  - [R-journal](https://journal.r-project.org)
  - [bug-tracking](https://bugs.r-project.org)
  - [R-search](https://search.r-project.org)
How to Develop Programs?

SAS windowing environment

R environment
How to Develop Programs?

SAS Code

- code blocks
- data types
  - numerical values
  - alphanumerical values
  - tables
  - matrices (only in SAS IML)
  - macro variables

- DATA STEP
  - key element
  - e.g. generation of data
  - concatenation of data

- PROC MEANS
  - key element
  - special functionalities
How to Develop Programs?

R Code

- „line based“ programming
- object oriented data structures
  - vector
  - matrix
  - dataframe
  - ....
- optimized for matrix manipulation
- R-functions
  - special functionalities
SAS-Macros vs. R-Functions

SAS-macro language

```sas
%MACRO BerechneMean (p_s_LibRef = , p_sTableName = , p_sOutTableName = )
  %LET l_sTableName = &p_s_LibRef..&p_sTableName.;
  PROC MEANS DATA = &l_sTableName. MEAN ;
  VAR Height;
  CLASS Sex;
  OUTPUT OUT=&p_sOutTableName.;
RUN;
%MEND BerechneMean:
```

- storing recurring program sequences
- macros are executed by the macro-processor
- macros base on text replacement
- use of macro-libraries
- macro variables store strings (local or global scope)
SAS-Macros vs. R-Functions

R-functions

```r
> berechneMean <- function(p_vector){
+   l_result <- mean(p_vector)
+   return(l_result)
+ }
```

```r
> berechneMean( c(4,6,7,8,3,6,10))
[1] 6.285714
```

- storing recurring program sequences
- several functions are usually stored as packages (help pages, manuals etc.)
- internal variables have local scope
Performance Comparison Using Large Data Sets

Pseudocode:

```
start  = system time;

Create data by concatenation of data1 and data2; /*512 byte/record*/

Sort by row1 & row2;

Select * where row1 = "ABCD";

end    = system time;

duration = end - start;
```
SAS Options

• **tagsort:**
  – stores only the BY variables and the observation numbers in temporary files
  – requires less temporary storage space and memory
  – processing time is usually higher

• **compress:**
  – observations in a newly created data set are compressed (variable-length records) by using “Run Length Encoding” (RLE) (compresses observations by reducing repeated consecutive characters (including blanks) to two-byte or three-byte representations)

• **memlib**
  – library is processed in memory
  – files are kept in memory until SAS is terminated or the files are deleted.

1. SAS tagsort off, compress off, memlib off
2. SAS tagsort on, compress off, memlib off
3. SAS tagsort off, compress on, memlib off
4. SAS tagsort on, compress on, memlib off
5. SAS tagsort on, compress on, memlib on
Performance Comparison Using Large Data Sets

- R
- 1. SAS tagsort off, compress off, memlib off
- 2. SAS tagsort on, compress off, memlib off
- 3. SAS tagsort off, compress on, memlib off
- 4. SAS tagsort on, compress on, memlib off
- 5. SAS tagsort on, compress on, memlib on

Dr. Peter Bewerunge
© 2011 HMS Analytical Software GmbH
Performance Comparison Using Large Data Sets

- R
- 1. SAS tagsort off, compress off, memlib off
- 2. SAS tagsort on, compress off, memlib off
- 3. SAS tagsort off, compress on, memlib off
- 4. SAS tagsort on, compress on, memlib off
- 5. SAS tagsort on, compress on, memlib on

Dr. Peter Bewerunge
© 2011 HMS Analytical Software GmbH
Performance Comparison Using Large Data Sets– 1.3 GB

Computing Time in Percentage

- **R**
- 1. SAS tagsort off, compress on, memlib off
- 2. SAS tagsort on, compress off, memlib on
- 3. SAS tagsort on, compress off, memlib off
- 4. SAS tagsort on, compress on, memlib off
- 5. SAS tagsort on, compress on, memlib on

- **Select**
- **Sort**
- **Concatenate**
Performance Comparison Using Large Data Sets

- Intel Core Duo CPU, 2.53 GHz, 2GB RAM
- SAS 9.2
- R 2.11.1

→ R is slightly faster
  - data are loaded into the main memory
  - limitation given by the size of the main memory

→ SAS processes large data sets
  - reads and writes data to the hard disk for each step
  - accordingly computation time is higher
  - almost as fast as R by the use of memlib (data are loaded to the main memory - also limitation given by the size)
Interface SAS IML 9.22 to R

• IML must be installed and comes with an extra license
• SAS must be started with Option RLANG
• at least compatible with R Version R-2.10.1

```r
PROC IML;
SUBMIT / R;
#R-Code
rx <- matrix(1:3, nrow=1)
rm <- matrix(1:9, nrow=3,
byrow=TRUE)

rq <- rm %*% t(rx)
print(rq)
ENDSUBMIT;
QUIT;
```
## Interface SAS IML 9.22 to R

### Data export from SAS to R:

<table>
<thead>
<tr>
<th>Subroutine</th>
<th>SAS source</th>
<th>R target object</th>
</tr>
</thead>
<tbody>
<tr>
<td>ExportDataSetToR</td>
<td>SAS data set</td>
<td>R data frame</td>
</tr>
<tr>
<td>ExportMatrixToR</td>
<td>SAS/IML matrix</td>
<td>R matrix</td>
</tr>
</tbody>
</table>

**Example SAS code:**

```
PROC IML;
RUN ExportDataSetToR("Sashelp.Class", "df");
submit / R;
  names(df)
endsubmit;
QUIT;
```

**R output:**

```
[1] "Name"  "Sex"  "Age"  "Height"  "Weight"
```

### Data import from R to SAS:

<table>
<thead>
<tr>
<th>Subroutine</th>
<th>R source</th>
<th>SAS target object</th>
</tr>
</thead>
<tbody>
<tr>
<td>ImportDataSetFromR</td>
<td>R expression</td>
<td>SAS data set</td>
</tr>
<tr>
<td>ImportMatrixFromR</td>
<td>R expression</td>
<td>SAS/IML matrix</td>
</tr>
</tbody>
</table>
Interface from R to SAS

- no direct interface to SAS implemented
- calling SAS with system commands:

```
> system{
  +   "C:\\Programme\\SAS\\SASFoundation\\9.2\\sas.exe
  +   C:\\HMS\\Projekte\\KSFE2011\\ProcMeansExport.sas"
  + }
```

- calling SAS-program:

```
PROC MEANS DATA = SAShelp.CLASS MEAN SUM;
  VAR Height;
  CLASS Sex;
  OUTPUT OUT=Results;
RUN;
PROC EXPORT DATA= Results
  OUTFILE= "C:\\tmp\\Results.csv"
  DBMS=CSV REPLACE;
  PUTNAMES=YES;
RUN;
```

- Data import to R:

```
> sasData <- read.table{
+   file = "C:\\tmp\\Results.csv"
+   , sep = ","," , header = TRUE }

> sasData
     Sex X_TYPE_ X_FREQ_ X_STAT_   Height
   1     0      19     N      19.000000
   2     0      19     MIN    51.300000
   3     0      19     MAX    72.000000
   4     0      19     MEAN   62.336842
   5     0      19     STD     5.127075
```
Software Quality

**SAS**
- customer orientated
- installation / operational – qualification tool
- developer work together locally
- well defined test procedures

**R**
- developer orientated
- installation qualification-process could be improved (no operational qualification tool)
- developers are spread world wide
- tests of additional packages could be improved
Conclusion

• Focus was on the SAS language and R environment
  – SAS platform has many frontends, servers, meta-data administration, middleware-components

• Differences
  – history of origin
  – user requirements
  – performance using large data sets
  – installation and support

• Similarities
  – development environment
  – structure of recurring code

• Common
  – interface

• Personal impression
  – SAS comes along with a higher proof of quality
  – R packages implement the state of the art algorithms
  – R is well integrated into SAS IML 9.22
  – there is no general better or worse
Thank you for your attention

Dr. Peter Bewerunge
Software Engineer

HMS Analytical Software GmbH
Rohrbacher Str. 26 • 69115 Heidelberg
Telefon +49 6221 6051-0
Peter.Bewerunge@analytical-software.de
www.analytical-software.de