

## Efficiency Issues in Evaluating Pharmacogenetics Data With SAS

Volker Harm, Bayer Schering Pharma AG, Berlin, Germany

Stephan Lehr, Bayer Schering Pharma AG, Berlin, Germany

### ABSTRACT

In evaluating pharmacogenetics data one has to be extremely careful to cope with the enormous amount of data and especially the high number of variables. Thus presumably simple operations can become extremely challenging. We present typical data management and basic evaluation examples of Single Nucleotide Polymorphism (SNP) data and some strategies to handle them considering the optimal choice of SAS<sup>®</sup> options.

### INTRODUCTION

Thanks to the recent revolutionary developments in genomic technology it is now possible to measure a huge number of markers covering the whole human genome. The availability of this comprehensive information holds promise for the identification of genetic factors that cause, e.g. differing responses to treatments. On the other hand it poses substantial challenges to the data analyst, who has to cope with the enormous amount of data, the multiplicity problem, and the choice of the validation strategy. In this paper we focus on the data handling, data storage and the preparation of a data matrix that can readily be used for the statistical analysis with SAS procedures.

The amount of data requires all data management and analysis steps to be well planned because of the time taken to do even simple analyses. Therefore we start to develop kind of an analysis plan template by briefly outlining the workflow of a pharmacogenetic study and pointing out how the data analyst's part fits in. In the following section we will describe how to perform the tasks of genetic data management and analysis with SAS, and in the third section we will go into detail about some efficiency issues that arise.

### ANALYZING PHARMACOGENETICS DATA

#### PHARMACOGENETIC STUDIES

The following figure shows the sequence of steps that have to be taken in a pharmacogenetic study using genome wide SNP arrays:

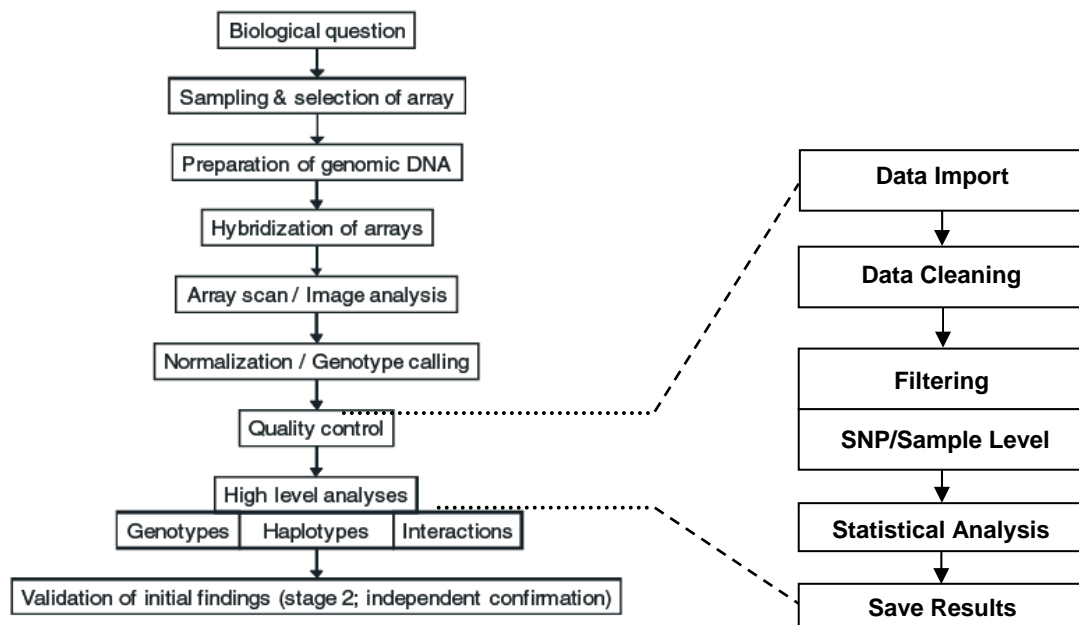


Figure 1: Sequence of steps in a pharmacogenetic analysis using genome-wide SNP arrays (left part of the figure is taken from Ziegler et al, 2008)

## PhUSE 2008

The objective for the study will be derived from a biological question. The steps from sampling and selection of array to normalization/genotype calling are done in the genetics laboratory and by bioinformatics. Genetic data management and analysis, the data analyst's part, comprises the steps of quality control and serves as basis for high level analyses. The right part of figure 1 shows these steps in detail. The results of these steps are saved in appropriate formats and delivered to the scientist as basis for further high level analyses and biological interpretation. Findings of these analyses then have to be validated.

### DATA TO BE ANALYZED

The SNP data we had to deal with were obtained from the Affymetrix Genome-Wide Human SNP Array 6.0. This array contains probes for more than 906.000 SNPs. After array scanning and image analysis the SNP array raw intensities are preprocessed by normalization and genotype calling. This was performed using the Birdseed algorithm (v2) within the Affymetrix Power Tool apt-1.8.5. The Birdseed algorithm performs a multiple-chip analysis to estimate a signal intensity for each allele of each SNP, fitting probe-specific effects to increase precision. It then makes genotype calls by fitting a Gaussian mixture model in the two-dimensional A-signal vs. B-signal space, using SNP-specific models to improve accuracy. The SNP genotype call is stored as AA, BB (homozygous subject for the A, B allele respectively) or AB (heterozygous subject) or the genotype is missing.

The letters A and B are used for all SNPs. The concrete bases for the SNPs as well as other information about the SNPs are given in a separate annotation file.

### GENETIC DATA MANAGEMENT

The matrix delivered after genotype calling, the *genotype calls matrix*, has the following structure:

	Sample <sub>1</sub>	...	Sample <sub>n</sub>
SNP <sub>1</sub>	<i>Genotype calls matrix</i>		
SNP <sub>k</sub>			

The genotype calls matrix is the starting point for genetic data management. First the data have to be cleaned from laboratory controls. There are control SNPs and control samples. After removing this data quality control can be started. Typical quality criteria for filtering out SNPs are based on the relative frequency of missing genotypes across samples, the minor allele frequency and on tests for Hardy-Weinberg equilibrium. For filtering out samples the most basic criteria are the call rate, i.e. the percentage of successfully genotyped SNPs per sample, as well as on heterozygosity and relatedness of individuals.

### DATA STRUCTURE FOR STATISTICAL ANALYSIS

After completing quality control data sets ready for statistical analyses can be created. The complete set of data for a SNP array experiment generally comprises three data sets:

- The *genetics data matrix*, which contains the measurements, in our case the genotype calls. It contains data of  $n$  samples and  $k$  SNPs. In a clinical environment the  $n$  samples correspond to  $n$  subjects. For statistical analyses the data are arranged in a  $n \times k$  matrix with  $n$  rows and  $k$  columns.
- The *annotation matrix*, which contains metadata about the SNPs. For  $k$  SNPs the matrix contains 26 attributes, such as the chromosome position, the associated gene, the minor allele etc. These data are typically arranged in a  $k \times 26$  matrix with  $k$  rows and 26 columns.
- The *phenotype matrix*, which contains data about the samples. It contains data for  $n$  samples and  $p$  variables. In a clinical environment these are the typical clinical data such as treatment groups, demographic variables, clinical endpoints, outcomes etc. for  $n$  subjects. These data are typically arranged in a  $n \times p$  matrix with  $n$  rows and  $p$  columns.

The row names of the genetics data matrix provide a link to the row names of the phenotype matrix, the column names provide a link to the row names of the annotation matrix as illustrated in the following figure.

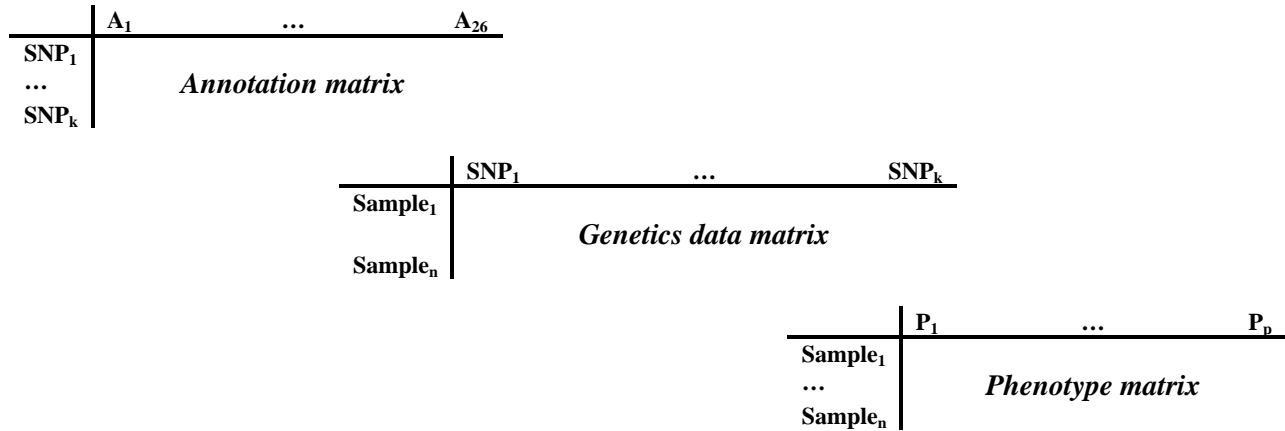


Figure 2: Structure of data sets in an genome-wide SNP array experiment

**STATISTICAL ANALYSIS**

For the remainder of this paper, we assume that we have data from a study that was carried out to identify genetic factors (SNPs) which predict, for example, the occurrence of a clinical symptom, or the response to a study treatment. For this analysis, phenotypic data are used in addition to the genetic information. As a typical example, it is assumed that the statistical evaluation is performed using Cochran-Armitage trend tests. It has reasonable power under all genetic models.

Result of this statistical evaluation is a list with a p-value for each SNP, which then can be linked to the annotation matrix.

**GENETIC DATA MANAGEMENT AND ANALYSIS WITH SAS**

In the following we describe how the basic tasks in genetic data management and analysis are performed in SAS. Assuming we have test data that poses no efficiency issue we will show that these basic tasks are quite simple and straightforward to handle with SAS data steps and procedures.

**THE CALLS FILE**

The genotype calls matrix is delivered from the bioinformatics group as a tab delimited text file, the “calls” file. The different genotype calls are coded as 0 (AA), 1 (BB), 2 (AB) and -1 (missing). Thus, an output file has the appearance as indicated in this table.

<i>probeset_id</i>	86090289334211	87986747397110	73608486866936	Control_57B
SNP_A-2131660	2	2	2	0
SNP_A-1967418	2	2	1	2
SNP_A-1969580	2	2	-1	2
AFFX_C-4263484	1	2	2	0

The first line gives as column headers the keys for the samples, the first column gives the keys for the SNPs.

**IMPORT CALLS**

To import the calls file we start with a basic call of proc import:

```
%macro ImportCalls;
  proc import datafile = Calls out = Data.Calls dbms = tab replace;
  run;
%mend;
```

Proc import creates a data set Calls as illustrated below:

## PhUSE 2008

<i>probeset_id</i>	<i>_6090289334211</i>	<i>_7986747397110</i>	<i>_3608486866936</i>	<i>Control_57B</i>
SNP_A-2131660	2	2	2	0
SNP_A-1967418	2	2	1	2
SNP_A-1969580	2	2	-1	2
AFFX_C -4263484	1	2	2	0

Proc import replaces the first digit of the sample id with an underscore. In our context this is no problem. But later on as mentioned above the genotype data have to be merged with the phenotype data. For this the sample ids will be needed as keys.

### CLEAN CALLS

The calls file has to be cleaned removing the samples and SNPs that are only used for laboratory quality control. This is done in the next step. Furthermore '-1' entries are replaced by SAS missing values.

The names of control sample variable are collected into a macro variable by using proc contents, ods output, and proc sql. The quality controls then easily can be excluded using data step options. Using an array over all numeric variables -1 is converted to missing.

```

%macro CleanCallsDataSet;
* searches for Control variables and cleans the Calls data set;
* search for control variables;
ods output Variables = CallsVars;
proc contents data = Data.Calls;
run;
proc sql noprint;
select Variable into: ControlSamples
separated by ' '
from CallsVars
where ((index (Variable, '_') ne 1)and (Variable ne 'probeset_id'));
proc datasets;
delete CallsVars;
run;
quit;
* clean the data set;
data Data.SNPs;
* deletes control samples;
* deletes Affymetrix Control SNPs;
set Data.Calls (drop = &ControlSamples
where =(substr (probeset_id, 1, 3) eq 'SNP')) end = last;
* replaces -1 by missings;
array tab _numeric_;
drop ii;
do ii = 1 to dim (tab);
if tab[ii] = - 1 then tab [ii] = .;
end;
run;
%mend;

```

This cleaning reduces the data set in the following way:

<i>probeset_id</i>	<i>_6090289334211</i>	<i>_7986747397110</i>	<i>_3608486866936</i>
SNP_A-2131660	2	2	2
SNP_A-1967418	2	2	1
SNP_A-1969580	2	2	.

## PhUSE 2008

### FILTER SNPS

Here we focus on the quality criteria related to missing values. The calls are counted across SNPs and across samples. Using these frequencies validity conditions for SNPs and samples are defined to build up a data set of valid SNPs and samples.

For convenience we start with filtering the SNPs. After counting the calls the percentage of missing values (across samples) and the minor allele frequency (MAF) are determined for each SNP. SNPs with more than 3% of missings and a MAF less than 5% are excluded.

```
%macro FilterSNPs;
  data Data.ValidSNPs (drop = n nmiss n0 n1 n2 i term1 term2 PctMiss MAF)
    Data.ExcludedSNPs (keep = probeset_id)
    Data.CallCounts (keep = probeset_id nmiss n0 n1 n2 PctMiss MAF);
  set Data.SNPs;
  * Calculate frequencies of calls for each SNP;
  array snp _numeric_;
  nmiss = 0;
  n0 = 0;
  n1 = 0;
  n2 = 0;
  do i = 1 to dim (snp);
    if snp {i} = . then nmiss + 1;
    if snp {i} = 0 then n0 + 1;
    if snp {i} = 1 then n1 + 1;
    if snp {i} = 2 then n2 + 1;
  end;
  * Calculate percent missings per SNP;
  PctMiss = (nmiss/dim (snp))*100;
  * Calculate the MAF for each SNP;
  n = n0 + n1 + n2;
  if n > 0 then do;
    term1 = (2*n0 + n1)/(2*n);
    term2 = (n1 + 2*n2)/(2*n);
    MAF = min (term1, term2);
  end;
  * Select the valid SNPs;
  if (PctMiss le 3.0) and (MAF gt 0.05) then do;
    * An ID for the SNPs is added to support the creation
      of the sample matrix;
    SNPID + 1;
    output Data.ValidSNPs;
  end;
  else do;
    output Data.ExcludedSNPs;
  end;
  output Data.CallCounts;
run;
%mend;
```

Valid SNPs, excluded SNPs and the counts are saved in different data sets. An Id is added to the data set ValidSNPs. This is needed in the next step.

### FILTER SAMPLES

To filter the samples in the same way as the SNPs the data set of valid SNPs has to be transposed. This is also the structure that is needed for further statistical analysis. Due to the fact that the data set contains only numerical data this can be done with the following simple statement.

## PhUSE 2008

```
%macro CreateSamples;
* transpose the SNPs data set to samples data set;
* removes the leading underscores from the sample names;
proc transpose data = SNPs out = Samples name = probeset_id prefix = SNP;
  id SNPID;
  idlabel probeset_id;
run;
data Samples (rename = (probeset_id = SampleID));
  set Samples;
run;
%mend;
```

The SNPID is used for the new variable names, the SNP names are stored as labels. The samples data set has the following appearance:

<i>SampleID</i>	<i>SNP1</i>	<i>SNP2</i>	<i>SNP3</i>
_6090289334211	2	2	2
_7986747397110	2	2	2
_3608486866936	2	1	.

One criterion for excluding samples, referred to as the call rate, is the percentage of missings in a sample. Samples with a call rate of less than 97% are excluded.

```
%macro FilterSamples;
data Data.ValidSamples (drop = i nmiss CallRate)
  Data.ExcludedSamples (keep = SampleID)
  Data.CallRates (keep = SampleID CallRate);
set Data.Samples;
* calculate Call rate;
array Sample _numeric_;
nmiss = 0;
do i = 1 to dim (Sample);
  if Sample {i} = . then nmiss + 1;
end;
CallRate = 1 - nmiss/(dim (Sample) - 1);
* select valid samples;
if CallRate ge 0.97 then do;
  output Data.ValidSamples;
end;
else do;
  output Data.ExcludedSamples;
end;
output Data.CallRates;
run;
%mend;
```

Valid samples, excluded samples and the call rates are saved in different data sets. The data set of valid samples is now cleaned and filtered, and serves as the genetics data matrix introduced above.

### MERGE PHENOTYPES

The phenotype matrix is merged to the genetics data matrix to perform the statistical analysis. In the example given below, the only phenotypic information is given by a variable Outcome, which can be thought of as an indicator of response to a certain drug or not.

```
%macro MergePhenotypes;
data AnalysisDataSet;
  merge Samples Phenotypes;
run;
%mend;
```

## PhUSE 2008

Merging of the phenotypes results in a data set with the following appearance:

<i>SampleID</i>	<i>SNP1</i>	<i>SNP2</i>	<i>SNP3</i>	<i>Outcome</i>
_6090289334211	2	2	2	1
_7986747397110	2	2	2	0
_3608486866936	2	1	.	1

### COCHRAN-ARMITAGE TREND TEST

Finally we can start the statistical analysis. The Cochran-Armitage trend test is implemented in proc freq and for our test data the statistical evaluation can be done in one step for all SNPs.

```
%macro DoTrendTest;
ods output TrendTest = TrendTest;
proc freq data = AnalysisDataSet;
    tables Outcome*(SNP1--SNP&NoOfSNPs) / trend;
run;
data results (drop = name1 rename = (table = SNPID));
    set TrendTest (keep = table name1 nvalue1
        where = (name1 = 'P2_TREND') rename = (nvalue1 = P_trend));
    table = substr (table, length ('Table Outcome*') + 2, length (table) - 1);
run;
proc datasets;
    delete TrendTest;
run;
quit;
%mend;
```

The results are stored for subsequent analysis (e.g. adjustment for multiplicity) or biological interpretation in the following form:

<i>SNPID</i>	<i>P_trend</i>
SNP1	0.248213
SNP2	0.345617
SNP3	0.546494

Fur further analysis this data set can be merged to the annotation matrix.

### EFFICIENCY ISSUES

#### THE CHALLENGE

The calls file we had to deal with contains calls for 909.622 SNPs and 1446 samples. The size of the calls file is approximately 2.5GB. Working in standard environment for clinical development this file size is the first efficiency issue. Copying the file is not done in seconds, and even a good text editor has problems to handle such a big file. In fact all typical first steps to analyze data as data listings, variable lists, frequency tables and ad hoc analyses are quite impossible.

#### THE ENVIRONMENT

We start the evaluation with some global settings:

```
%macro SetEnvironment;
* define analysis directory;
%global AnalysisDir;
%let AnalysisDir = C:\ginseng\SNP\Calls\;
* assign data library;
libname Data "&AnalysisDir.Data";
* define filename for calls file;
%global CallsFileName;
%let CallsFileName = Calls;
```

## PhUSE 2008

```

filename Calls "&AnalysisDir.Background\&CallsFileName..txt";
* set output options;
options pagesize = max;
proc printto new log = "&AnalysisDir.Programs\&CallsFileName..log"
              print = "./nul";

run;
ods results off;
%mend;

```

The settings for an analysis directory and the filename statement are quite standard. As we work with very large files, it should be avoided to access the data via a slow network connection. In our environment we had the best results putting the analysis directory on the workstation where the SAS process runs. Access via our network was not feasible.

The output options are specially selected for our purposes:

- Option pagesize is set to max. This prevents page breaks in the log file and saves space.
- Proc printto is used to redirect the log output to a file. Otherwise, when working in display manager, all too often there is a window buffer overflow. The print destination is deactivated.
- Ods results are set to off. This prevents tracking the output objects and saves time.

### IMPORTING THE CALLS FILE

As we have 1446 columns in the calls file we saw no other chance than to use proc import to get the process started. Proc import used 14 minutes real time, the Calls data set had a size of 11GB. This is a long time and a huge file. So let us have a look at the log to see which statements the External File Interface (EFI) created:

```

37      data DATA.CALLS                                ;
38      %let _EFIERR_ = 0; /* set the ERROR detection macro variable */
39      infile 'C:\ginseng\SNP\Calls\Background\birdseed-dev.calls.bc2.txt'
              delimiter='09'x MISSOVER DSD lrecl=32767 firstobs=2 ;
40      informat probeset_id $13. ;
41      informat _6090289334211 best32. ;
...
1484     informat _7012657675880 best32. ;
1485     format probeset_id $13. ;
1486     format _6090289334211 best12. ;
...
2929     format _7012657675880 best12. ;
2930     input
2931         probeset_id $
2932         _6090289334211
...
4375     _7012657675880
4376     ;
4377     if _ERROR_ then call symputx('_EFIERR_',1); /* set ERROR detection macro
variable */
4378     run;

```

As can be seen the data were read in with a length of 12 bytes. Actually a genotype call can have only the values 0, 1, 2 and -1, i.e. four distinct values. This could be stored using two bits. Compared to 12 bytes this is pretty much waste of space. So we modified the data step and tested two alternatives:

- Minimize the length of the numerical variables: The minimal length for numerical variable in SAS is three bytes. So we added a length statement for all numerical variables.
- Reading the data in as character: The minimal length for character variable in SAS is one. As we have -1 in the calls file as token for missing, we changed informat, format and length to \$2. Later after replacing the -1 as missing value the length of the genotype calls can even be reduced to one.

The following table summarizes the results :

Input Method	Time needed (on a standard notebook)	Size of calls file
Proc import	14:19.85	10.9 GB
Reduced to length 3	5:10.13	4.9 GB
Read in as character 2	3:53.21	2.9 GB
Reduced to length 1	7:51:11	1.3GB

(Our standard notebook has a Intel® Core™2 Duo Mobile Processor T7200 and 2 GB memory.)

## PhUSE 2008

The most efficient way to store the data in a SAS data set is the character representation. Fortunately it is no problem to count the calls as character values for filtering the data and the Cochran-Armitage trend test in proc freq also works for character values. By converting the genotype calls to length 1 the genetics data set after cleaning and filtering can be reduced to a size of about 1GB.

### MEMORY PROBLEMS

Even with this reduction in size it is still a challenge to handle these data sets with proc freq's trend test. Using proc freq for all SNPs gives an out of memory error. Therefore we had to find a way to test the SNPs in subsets.

We decided to use two macros for the evaluation of all SNPs: one which evaluates a range of SNPs in one step and one which calls the first macro in a loop and collects the results.

The macro DoTrendTest is used to perform the Cochran-Armitage trend test for a range of SNPs and add the relevant p-values to a result data set:

```
%macro DoTrendTest (from =, to = );
ods output TrendTest = TrendTest;
proc freq data = Data.AnalysisDataSet;
tables Outcome*(SNP&from.--SNP&to.) / trend;
run;
data TrendTest(drop = name1 rename = (table = SNPID));
set TrendTest (keep = table name1 nvalue1
               where = (name1 = 'P2_TREND')
               rename = (nvalue1 = P_trend));
table = substr (table, length ('Table Outcome*') + 2, length (table) - 1);
run;
%if &from. = 1 %then %do;
data Data.Results;
set TrendTest;
run;
%end;
%else %do;
proc append base = Data.Results data = TrendTest;
run;
%end;
%mend;
```

Test evaluations with this macro showed that the time elapsed per evaluation of one SNP decreases with the number of SNPs evaluated per proc freq call up to a certain limit and then increases again until the out of memory situation is reached. In each environment some test evaluation have to be run, to find a reasonable range of SNPs to be evaluated in one step.

As an example in the following tables evaluation times are given for numeric length 3 representation and the character length 1 representation of genotype calls.

Call of DoTrendTest (numeric length 3)	Real time needed (on a standard notebook)	CPU time needed (on a standard notebook)	Real time needed per SNP
DoTrendTest (from = 1, to = 1)	1:35.65	3.04	1:53.65
DoTrendTest (from = 1, to = 100)	2:32.92	3.56	1.53
DoTrendTest (from = 1, to = 500)	3:00.71	4.28	0.36
DoTrendTest (from = 1, to = 1000)	4:25.25	8.24	0.25
DoTrendTest (from = 1, to = 10000)	12:43.93	8:19.39	0.076
DoTrendTest (from = 1, to = 20000)	46:23.17	38:12.51	0.14
DoTrendTest (from = 1, to = 100000)	out of memory		

Call of DoTrendTest (character length 1)	Real time needed (on a standard notebook)	CPU time needed (on a standard notebook)	Real time needed per SNP
DoTrendTest (from = 1, to = 1)	14.34	0.81	14.34
DoTrendTest (from = 1, to = 100)	14.43	0.90	0.14
DoTrendTest (from = 1, to = 500)	19.62	2.43	0.04
DoTrendTest (from = 1, to = 1000)	20.26	5.98	0.02
DoTrendTest (from = 1, to = 10000)	12:43.93	8:19.39	0.076
DoTrendTest (from = 1, to = 20000)	8:18.64	7:57.57	0.05
DoTrendTest (from = 1, to = 100000)	out of memory		

Looking at the tables above the character representation not only uses less memory, but is also much faster.

## PhUSE 2008

The macro DoTrendTests is a wrapper to the proc freq calls, which does all the tests with a constant increment. On our workstations as seen in the table above 10000 tests per proc freq call gave a reasonable stable result for the numeric length 3 representation, for the character length 1 representation an increment of 1000 seems appropriate.

```
%macro DoTrendTests (StartSNPID =, EndSNPID =, Increment = 10000);
ods results off;
ods _all_ close;
%let low = &StartSNPID.;
%do %while (%eval (&low. + &Increment.) le &EndSNPID.);
    %let high = %eval (&low. + &Increment. - 1);
    %DoTrendTest (from = &low., to = &high.);
    %let low = %eval (&low. + &Increment.);
%end;
%DoTrendTest (from = &low., to = &EndSNPID.);
proc datasets;
    delete TrendTest;
run;
quit;
%mend;
```

### WRITING EFFICIENT CODE

Throughout the code snippets it can be seen that we made substantial effort to ensure that only data that is needed is read or written. This is done by clever use of data set options and the use of proc contents, ODS output statements, proc sql and macro variables. See the Recommended Reading section for the strategies applied.

### CONCLUSION

The large amount of data seemed to prevent the use of simple data steps and procedures in SAS. In this paper, we described how to overcome the issues that arise because of the large datasets. We identified some critical issues, for example the fact that proc import replaces the first digit of a numerical with an underscore, instead of adding the underscore in order to obtain a valid variable name.

Still, we are looking at several points that were not yet clarified. For example, even a simple merge of the dataset containing the SNPs and a simple dataset containing phenotypic data does not work in a straightforward way.

As a conclusion it is evident that, whilst working with SAS is possible, a careful planning of the programming tasks is essential in this environment.

### REFERENCES

Ziegler A, König IR, and Thompson JR.. 2008. "Biostatistical Aspects of Genome-Wide Association Studies", *Biometrical Journal* 50,1:1-21.

### RECOMMENDED READING

McQuown, G..2005. "PROC IMPORT with a Twist", Paper 038-30, SUGI 30

Whitlock, I..2007. "How to think through the SAS® data step", Paper 208-2007, SAS Global Forum 2007

Zdeb, M. S..1999."Creating Macro Variables via Proc SQL", Paper CC107, NESUG 99

### CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Volker Harm  
Bayer Schering Pharma AG  
Müllerstraße 178  
13342 Berlin  
Work Phone: +49-30-468-11208  
Fax: +49-30-468-91208  
Email: volker.harm@bayerhealthcare.com  
Web: <http://www.bayerscheringpharma.de>

Brand and product names are trademarks of their respective companies.