

Introduction to SASPairs

This paper introduces SASPairs, a package of software routines in SAS (Statistical Analysis System) for analyzing data on pairs of relatives. The routines are a series of SAS macros and IML (Interactive Matrix Language) code that analyze genetically informative data in a SAS session. There are both notable advantages and disadvantages to SASPairs, but it is easiest to explain these by referencing a specific example.

Figure 1 provides the code that will fit six models involving an additive genetic covariance matrix (V_A), a common environment covariance matrix (V_C) and the Cholesky factors of a unique environment covariance matrix (V_U). SASPairs requires a *SASPairs Data Set*, a valid SAS data set in which each “observation” is a character string containing SASPairs commands. In Figure 1, the SASPairs Data Set is called `example_mds1` and it is constructed from the first four lines of SAS code in the Figure.

Figure 1. Example of a SASPairs Model Data Set.

```

data example_mds1;
  length card $80;
  input card $char80.;
datalines4;

BEGIN DATASETS
  PHENOTYPIC DATA SET = spothstf.twindata1
  FAMILY ID VARIABLE = twinpair
  RELATIONSHIP VARIABLE = zygotity1
  PHENOTYPES FOR ANALYSIS = iq reading writing
  RELATIONSHIP DATA SET = spothstf.twins_sex_differences
END DATASETS

BEGIN MODEL DATA = spmdslib.acu_models
;;;
run;
%saspairs(example_mds1);

```

A SASPairs data set requires two major sets of commands, the first to specify the data to be processed and the second, the model(s) to be fitted to the data. The first set of commands is called the *Dataset Definitions*, and they are included between the `Begin Datasets` and `End Datasets` phrases in Figure 1. The meaning of most of the dataset definition commands in Figure 1 should be obvious. They give the SAS data set to be processed (`testsp.twindata1`), the name of a variable identifying families in the data set (`twinpair`), the name of a relationship code variable (`zygotity1`), and the names of the variables to be analyzed (`iq reading writing`). (The command `RELATIONSHIP DATA SET` will be discussed later).

The second major set of commands, *Model Definitions*, is included between a `Begin Model` and an `End Model` statement (if a user is writing a specific model) or in the argument to a `Begin Model DATA=` statement (if the model is to be read from

an existing SAS data set). In Figure 1, the model will be read from the SAS data set `spmdslib.acu_models`. (We will see an example of a user-written model later).

One of the major advantages of SASPairs comes from the fact that processing of the data set, constructing a model, and fitting that model to the data are all performed within a single package—SAS. For example, in Figure 1, three phenotypes are specified, so matrices V_A , V_C , and V_U must all be (3 by 3) symmetric matrices. Consequently, there is no need for a user to dimension these matrices or to specify their types. Similarly, the fact that the data are processed within SAS permits the calculation of sample sizes and automatic starting values for some matrices. These features then permit a user to write code in SASPairs that can apply to *any* data set, providing of course, that the data contain genetically informative information. This code may be stored as a SAS data set in a SAS library and called upon demand.

Figure 1 illustrates these features. The command `BEGIN MODEL DATA = spmdslib.acu_models` reads in models from the SASPairs model data set `acu_models` which is a SAS data set in the SAS library `spmdslib`. Figure 2 gives the contents of `acu_models`.

Figure 2. The SASPairs Model Data Set `spmdslib.acu_models` Called by the Code in Figure 1.

```
data spmdslib.acu_models;
    length card $80;
    input card $char80.;
datalines4;
begin model data=spmdslib.acu;
begin model data=spmdslib.acu_noa;
begin model data=spmdslib.acu_noc;
begin model data=spmdslib.acu_noacorr;
begin model data=spmdslib.acu_nocorr;
begin model data=spmdslib.acu_noucorr;
; ; ; ;
```

Notice that the Model Definition statements in `acu_models` themselves call other *Model Data Sets* within the same library. In this way, one can construct a hierarchy of calls to Model Data Sets. In the present case, the six models to be fitted to the data are fairly obvious by the names of the data sets. For example, data set `acu` contains the code to fit V_A , V_C , and V_U ; data set `acu_noc` sets V_C to 0; and data set `acu_noacorr` tests whether the genetic correlations can be set to 0.

These features can be very helpful to researchers who gather data over a long period of time. Early in the project, SASPairs code can be developed, tested, and then stored in a SAS data set. As more data come in, then a whole series of models can be fitted to the data by just submitting to SASPairs a file analogous to the one in Figure 1. There is no need to compute covariance matrices, change code to reflect the increase in sample size or, for that matter, add code for an addition group because you want to analyze adoption data with your twin data—SASPairs automates these tasks.

Part of the reason for this simplicity is the use of a *Relationship Data Set*. Table 1 gives the Relationship Data Set `twins_sex_differences` referred to in Figure 1.

The Relationship Data Set gives the relationship codes for the two members of a pair (along with their labels) and then three coefficients that are used by SASPairs to construct predicted covariance matrices. These coefficients are:

- `Gamma_A`: The correlation between additive genetic values of the pair.
- `Gamma_C`: The correlation between common environment values of the pair.
- `Gamma_D`: The correlation between dominance values of the pair.

Relative1	Relative2	Label1	Label2	Gamma_A	Gamma_C	Gamma_D
1	1	mz_f	mz_f	1.0	1.0	1.0
2	2	dz_f	dz_f	0.5	1.0	0.25
3	3	mz_m	mz_m	1.0	1.0	1.0
4	4	dz_m	dz_m	0.5	1.0	0.25
5	6	dzos_f	dzos_m	0.5	1.0	0.25

A user must construct a Relationship Data Set to reflect the types of codes used in his/her lab. This, however, is not an onerous chore, and once it is done the Relationship Data Set can be saved as a permanent SAS data set to be called when needed.

Let us return to discussion of the Model Data Set. Figure 3 gives the actual SASPairs code used to fit the model testing that the genetic correlations are 0. Readers familiar with Mx (Neale et al., 2002) will notice a similarity in structure and syntax of SASPairs code to that of the more general program. In this case, the phrase on the `Begin Model` statement gives a title to the model. The title has no functional significance; it only labels output.

Statements between a `Begin Matrices` and an `End Matrices` statement define the matrices for the problem. All four of the matrices here are “default matrices” in the sense that SASPairs recognizes their names and sets their types and (usually) their dimensions. Matrices `VA`, `VC`, and `VU` will all default to symmetric matrixes with dimensions equal to the number of phenotypes. Matrix `FU` is also a default matrix (it is used to fit factor models) but its type must be specified¹. In this case, `L` denotes a lower diagonal matrix. The number of rows for a default factor matrix is set to the number of phenotypes, but the number of columns must be specified. The phrase `&np` in Figure 3 tells SASPairs to set the number of columns to the number of phenotypes².

Statements between `Begin Mx` and `End Mx` place constraints on the parameters and provide starting values. With one exception, the commands here are a subset of Mx commands that perform the same tasks. In Figure 3, the command `CO VA FU` is that one exception. It uses the observed data to compute starting values for the additive genetic

¹ For two reasons, I prefer the notation “U” to the customary “E” to refer to the unique or idiosyncratic environment. First, undergraduates are much less confused by this terminology. Second, in the world of genetic epidemiology where a dominance covariance matrix is `VD` and a standard deviation is an `STD`, there is great need for an `FU` matrix.

² Because matrix `FU` is square and because the number of rows default to the number of phenotypes, there is no need to specify the number of columns. They are given here in order to illustrate the use of `&np`.

covariance matrix and for the Cholesky factors for the unique environment covariance matrix.

Figure 3. SASPairs Code for a Model Setting All Genetic Correlations to 0.

```

data spmdslib.acu_noacorr;
  length card $80;
  input card $char80.;
datalines4;
Begin Model  VA=Diag, VC, VU, no genetic correlations
  Begin Matrices
    VA
    VC
    VU
    FU L
  End Matrices

  Begin Mx
    CO VA FU
    FI 0 Offdiag(VA) VU
  End Mx

  Begin IML
    if pair_number=1 then do;
      VU = FU * t(FU);
      P1 = VA + VC + VU;
      P2 = P1;
    end;
    R12 = gamma_a * VA + gamma_c * VC;
  End IML
End Model
;;;
run;

```

The command `FI Offdiag(VA) VU` fixes the off diagonal elements of `VA` to 0 and also sets all elements of `VU` to 0. (It is not necessary to include `VU` in the model; it is present here only so that its value can be printed later in the program).

Statements between `Begin IML` and `End IML` are analogous to those between the `Mx Begin Algebra` and `End Algebra` commands. They provide code written in PROC IML (Interactive Matrix Language) of SAS that give blocks of the predicted covariance matrix for a pair of relatives. (The reason for the SAS statement `datalines4` to read in the data and the four semicolons ending the data stream that may have puzzled readers in Figures 1 through 3 is now apparent—the IML code contains semicolons and this is the only way to read in such statements.) In the IML code, a user must calculate three blocks of a predicted covariance matrix using the parameter matrices previously defined:

- (1) `P1`, the predicted within-individual phenotypic covariance matrix for the first relative of a pair;
- (2) `P2`, the phenotypic covariance matrix for the second relative of the pair;
- (3) `R12`, the covariance matrix between the phenotypes of the first and the second relatives.

To understand this code, let us examine the logic used by SASPairs in minimizing a function value. In the present case, there will be five groups, each one corresponding to the types of relatives given in Table 1. When the minimizer requests the function value for a set of parameters, then the IML code in Figure 3 will be called five times. The first call will be for MZ females and the values of `gamma_a`, `gamma_c`, (and `gamma_d`, if it were used) will be taken from the values given in the Relationship Data Set for MZ females (see Table 1). The second call will be for DZ females and their values of `gamma_a` and `gamma_c` will be used, and so on. Hence, there is no need to write special code for each group in the analysis.

SASPairs refers to a group by the variable called `pair_number`. Hence, in Figure 3, when the function is evaluated for MZ females, `pair_number = 1`. The `if-then` statement in Figure 3 restricts calculation of the unique environment matrix `VU` from its Cholesky factors and the two phenotypic covariance matrices to the first call. The same values of `P1` and `P2` will be used for the other four groups, so some computational time is saved.

SASPairs produces considerable output in the SAS Output window, but the final part of the output from the Model Data Set in Figure 1 is given in Figure 4. In addition to the χ^2 goodness-of-fit, SASPairs also provides the Akaike Information Criterion (AIC), corrected AIC (CAIC), Schwarz' Bayesian criterion (SBC), and McDonald's Measure of Centrality (MMoC).

SASPairs also calculates the likelihood-ratio χ^2 for all models fitted in a Model Data Set. There is no intelligence to this. The program compares every pair of models fitted, and if their degrees of freedom differ, then it computes a χ^2 and its level of significance. The user must examine the output for nested models to guarantee which of these statistics are valid.

SASPairs can also fit models to raw data and means and to covariates. Its most notable limitation is that deals only with pairs of relatives. In some circumstances, this limitation can be overcome. In a large twin sample, for instance, treating the odd triplet as three independent pairs will not seriously compromise substantive conclusions. SASPairs also has a macro that permits one to calculate a weighted, pairwise covariance matrix for variable-sized sibships. With some types of data, this could provide a suitable approximation. Finally, if SASPairs is here, could SASPeds be far behind?

SASPairs, however, cannot deal with—and should not be used to deal with—general pedigrees. Recoding data on, say, nuclear families in terms of relative pairs is not statistically justified (at least until someone arrives at a suitable scheme for weighting the relative pairs). SASPairs also deals only with quantitative phenotypes, so it cannot fit threshold models. Finally, SASPairs cannot perform linkage or association tests unless the problem conforms to SASPairs requirements and a user writes his/her own IML definitions.

A final limitation is execution time. The SAS Institute is forthright in declaring PROC IML and its optimization routines as programming conveniences, not as speed demons. SASPairs works reasonably well on moderately sized problems on desktops (using a 1.5 Ghz Athlon, it takes a little over a minute to find the solution to a 108 parameter model fitted to twin covariance matrices). For larger models or for fitting models to raw data with a large number of pedigree types, then batch submission is recommended.

Figure 4. Summary Output from SASPairs.

```

-----
Summary of Model Fit Indices
-----

Model Number and Title

1: VA, VC, VU
2: VA=0, VC, VU
3: VA, VC=0, VU
4: VA=Diag, VC, VU, no genetic correlations
5: VA, VC=Diag, VU, no common environment correlations
6: VA, VC, VU=Diag, no unique environment correlations

Model:                RC      Chi_2   df      p      AIC      CAIC      SBC      MMoC

1: VA, VC, VU          6      93.73   87     0.292  -80.27  -579.08  -492.08  0.996
2: VA=0, VC, VU        6     214.81  93     0.000   28.81  -504.39  -411.39  0.930
3: VA, VC=0, VU        6     108.63  93     0.128  -77.37  -610.57  -517.57  0.991
4: VA=Diag, VC, VU, no g  6     142.79  90     0.000  -37.21  -553.22  -463.22  0.969
5: VA, VC=Diag, VU, no c  6     104.52  90     0.141  -75.48  -591.49  -501.49  0.991
6: VA, VC, VU=Diag, no u  3     181.76  90     0.000   1.76  -514.24  -424.24  0.947

-----
Likelihood Ratio Statistics
***** NOTE WELL: THESE STATISTICS ARE INAPPROPRIATE FOR MODELS THAT
***** ARE NOT NESTED
-----

More General Model:   Constrained Model:   LR_df   LR_Chi2   LR_p

1: VA, VC, VU        2: VA=0, VC, VU      6       121.09   0.000
1: VA, VC, VU        3: VA, VC=0, VU      6        14.91   0.021
1: VA, VC, VU        4: VA=Diag, VC, VU, no g  3        49.06   0.000
1: VA, VC, VU        5: VA, VC=Diag, VU, no c  3        10.79   0.013
1: VA, VC, VU        6: VA, VC, VU=Diag, no u  3        88.04   0.000
4: VA=Diag, VC, VU, no g  2: VA=0, VC, VU      3        72.03   0.000
5: VA, VC=Diag, VU, no c  2: VA=0, VC, VU      3       110.30   0.000
6: VA, VC, VU=Diag, no u  2: VA=0, VC, VU      3        33.05   0.000
5: VA, VC=Diag, VU, no c  3: VA, VC=0, VU      3         4.11   0.249

```

If you are interested in SASPairs, a beta test version is available at <http://psych.colorado.edu/SASPairs>.

If you are interested in a completely debugged and fully guaranteed version of SASPairs, then please send US\$100,000 in unmarked currency to:

Gregory Carey
Department of Psychology
University of Colorado
Boulder CO USA 80309-0345

phone: 303-492-1658
fax: 303-492-2967
email: gregory.carey@colorado.edu

References

Neale MC, Boker SM, Xie G, Maes HH. (2002). *Mx: Statistical Modeling*. VCU Box 900126, Richmond, VA 23298: Department of Psychiatry, 6th Edition.