The SHARE package: **S**NP-**H**aplotype **A**daptive **RE**gression

Ting-Yuan Liu, James Y. Dai

January 22, 2010

1 Introduction

Association studies have been widely used to identify genetic liability variants for complex diseases. While scanning the chromosomal region one SNP at a time may not fully explore linkage disequilibrium (LD), haplotype analyses tend to require a fairly large number of parameters, thus potentially losing power. Clustering algorithms, such as the cladistic approach, have been proposed to reduce the dimensionality, yet they have important limitations. We propose the SHARE algorithm (Dai et al., 2009) that seeks the most informative set of SNPs for genetic association in a targeted region by growing/shrinking haplotypes with one more/less SNP in a stepwise fashion, and comparing prediction errors of different models via cross-validation. This is the manual of the SHARE package.

2 Prequisites

Both haplo.stats and MASS packages are required for using the SHARE package. They will be loaded automatically when loading the SHARE package.

3 SHARE Analysis

SHARE can handle both phased haplotype data and unphased genotype data. For unphased genetic data, SHARE adopts the *haplo.em* function in haplo.stats to estimate the probability of haplotype pairs. Users can also phase the genotypic data upfront by using existing software such as PHASE or fastPHASE (Stephens et al., 2001; Stephens and Donnelly, 2003), then use SHARE to search for the most informative haplotypes. We illustrate the usage of SHARE mainly on the genotypic data in Section 4, and briefly show how to input the phased data in Section 5.

We use the Cystic Fibrosis data in Browning (2006) for illustration. The original R dataset contains 186 haplotypes based on 23 SNPs, which can be found in HapVLMC package (`http://www.stat.auckland.ac.nz/~browning/HapVLMC/index.htm`). Subject-level diploid genetic information, either haplotype or genotype sequences, and the clinical status (e.g., case or control) are required to perform the SHARE algorithm. Therefore, we constructed the diploid level genotypic data (`keremRandAllele`) and haplotype data (`keremRandSeq`) by randomly pairing two haplotypes and stored them in the package.

```
> data(keremRand)
> ls()

[1] "keremRandAllele" "keremRandSeq"    "keremRandStatus"
```

The *data.frame* object `keremRandSeq` contains 186 sequences with 23 SNPs. The row names show the subject id and the sequence id within this subject. The SNPs are coded as 1 referring to the large allele of the RFLP, and 2 referring to the smaller allele. The vector object `keremRandStatus` provides the CF/control status of each subject. 1 indicates subjects in case group (i.e., CF), and 0 indicates control group. There are 47 subjects in CF group and 46 in control group. `keremRandAllele` is a *matrix* object containing allelic data for 23 SNPs, coded as 0, 1, and 2 as the number of minor alleles.

4 Unphased Genotype Sequences

Starting from allelic data such as `keremRandAllele`, we first estimate the haplotypes and their frequencies. SHARE performs the EM algorithm to obtain these parameter estimates.

4.1 *genoSet*

We first construct a *genoSet* object from our genotype sequences and the phenotype information:

```
> unphasedGeno <- new("genoSet", genoSeq = data.frame(keremRandAllele),
+     phenoData = data.frame(CF = keremRandStatus))
> unphasedGeno

An object of class "genoSet"
# of SNP: 23
# of genotype sequence:  93
Phenotype:
 CF
```

4.2 Haplotype Reconstruction

The method *haplo* performs the EM-based haplotype reconstruction algorithm on *genoSet* objects. The reconstructed haplotypes as well as other information will be stored in the object of class *haplo*.

```
> unphasedHaplo <- haplo(unphasedGeno)
> unphasedHaplo

An object of class "haplo" (extended from class "haploSet")
# of SNP: 23
# of haplotype sequence:  90
```

4.3 SHARE Algorithm

The next step is to search for the most informative set of SNPs and its associated haplotype model. The default method is to compare prediction deviance via ten-fold cross-validation (ModSelMethod=”Cross-Val”). Alternatively, we also provide the BIC criterion for faster computation (ModSelMethod=”BIC”). The mode of inheritance for haplotype effects can be specified to be additive (Minherit=”additive”), dominant (Minherit=”dominant”), or recessive (Minherit=”recessive”). The cshare function will search for the best subset up to a user-defined maximum number of SNPs (default 6 SNPs).

The cshare function outputs a *share* object, which contains the best size of SNPs, and either the prediction deviance or BIC. If the best subset is null set (i.e., the model with intercept only), there is no genetic association in the candidate region. If the best subset contains at least 1 SNP, cshare also outputs the haplotype frequencies, the estimated haplotype effect, nominal p-values in the final model, and the experiment-wise p-values that has corrected for multiple tests.

Here is an example that demostrates the SHARE algorithm by the cross-validation method on an additive model:

```
> unphasedKerem <- list()
> unphasedKerem[["Cross-Val"]] <- cshare(unphasedHaplo, status = "CF",
+     nfold = 20, maxsnps = 5, ModSelMethod = "Cross-Val", Minherit = "additive")
> unphasedKerem[["Cross-Val"]]

An object of class "share"
Given nFold=20 cross-validation and maxSNP=5, the best number of SNP to select is 2.

# of selected SNP: 2
# of haplotype sequence:  4
```

```
Haplotype sequences in the final subset:
     locus_11 locus_17 Frequency
hap1        1        1  0.037634
hap2        1        2  0.430108
hap3        2        1  0.381720
hap4        2        2  0.150538

Testing:
                  Coefficient Standard Error   z-Score  p-Value
hap 2 (Intercept)    4.172075       1.064413  3.919600 0.000044
hap 1               -4.367063       1.475469 -2.959780 0.001539
hap 3               -3.725943       0.919607 -4.051670 0.000025
hap 4               -2.735133       0.719839 -3.799646 0.000072

Global p-value: 0.00045
```

Here is an example using the BIC method on an additive model:

```
> unphasedKerem[["BIC"]] <- cshare(unphasedHaplo, , status = "CF",
+     maxsnps = 5, ModSelMethod = "BIC", Minherit = "additive",
+     verbose = 1)
> unphasedKerem[["BIC"]]
```

```
An object of class "share"
Given BIC criterion and maxSNP=5, the best number of SNP to select is 2.

# of selected SNP: 2
# of haplotype sequence:  4

Haplotype sequences in the final subset:
     locus_11 locus_18 Frequency
hap1        1        1  0.370968
hap2        1        2  0.048387
hap3        2        1  0.435484
hap4        2        2  0.145161

Testing:
                  Coefficient Standard Error   z-Score  p-Value
hap 3 (Intercept)    4.579789       1.225504  3.737065 0.000093
hap 1               -3.778493       0.923236 -4.092661 0.000021
hap 2               -6.211824       1.616834 -3.841967 0.000061
hap 4               -2.984438       0.824179 -3.621103 0.000147

Global p-value: 0.000292
```

4.4 Deviance/BIC Plot

We can also check the path of model searching by creating a deviance/BIC plot (Figure 1 and Figure 2) by using the method *dplot*. The minimum deviance in the plot indicates the best size of SNP (e.g., the best size in Figure 1 is 2 and in Figure 2 is 2).

```
> dplot(unphasedKerem[["Cross-Val"]])
```

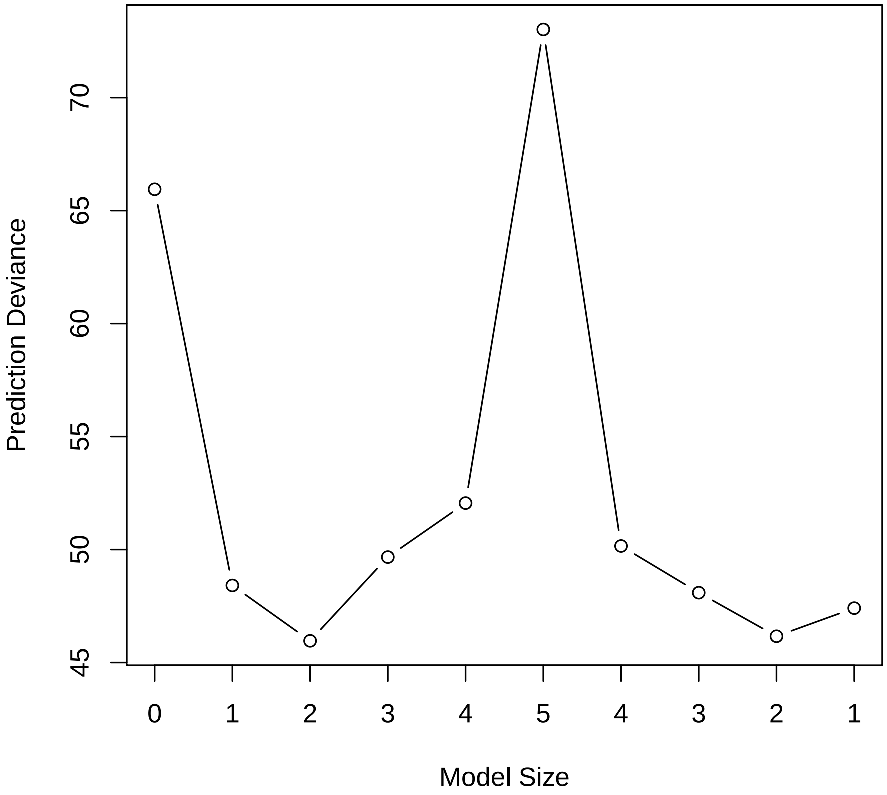

Figure 1: Deviance Plot

```
> dplot(unphasedKerem[["BIC"]])
```

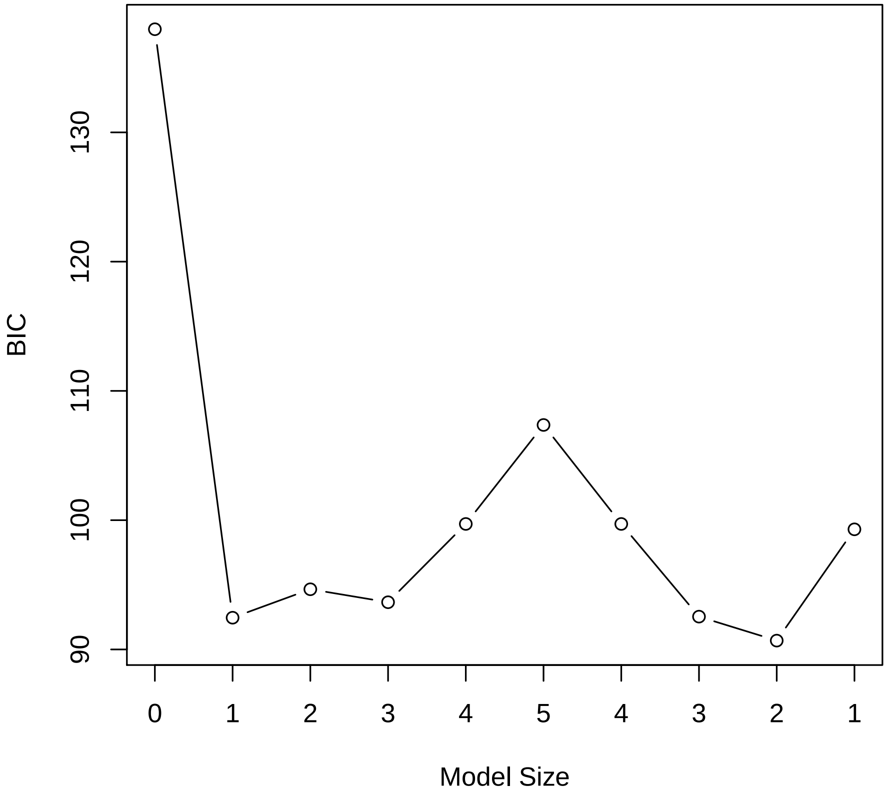

Figure 2: BIC Plot

4.5 Permutation Test

If the best model size is zero, there will appear to be no genetic association in the region of interest. There is no need to perform a permutation test. For final models with at least 1 SNP, we permute case-control labels 1000 times regardless of the genotypic data, carry out model searching for each permuted dataset, and compute the nominal p-value using a Wald test. Finally the experiment-wise p-value is computed by comparing the observed p-value to its null distribution (Besag and Clifford, 1991).

Here is an example that tests the SHARE algorithm result from the cross-validation method.

```
> extdataDir <- system.file("extdata", package = "SHARE")
> if ("unphasedPermuPValue.RData" %in% dir(extdataDir)) {
+     load(file.path(extdataDir, "unphasedPermuPValue.RData"))
+ } else {
+     unphasedPermuPValue <- shareTest(outObj = unphasedKerem[["Cross-Val"]],
+         haploObj = unphasedHaplo, status = "CF", nperm = 1000)
+ }
> unphasedPermuPValue
```

```
[1] 0.002997003
```

5 Phased Genotype Sequences

For the data of phased DNA sequences, we first need to determine the set of unique haplotypes:

```
> hapSeq <- apply(keremRandSeq, 1, function(x) {
+     paste(x, sep = "", collapse = "-")
+ })
> uniHap <- unique(hapSeq)
> nHap <- length(uniHap)
> hapNum <- unlist(sapply(1:nHap, function(x) {
+     paste(paste(rep("0", ceiling(log10(nHap)) - nchar(as.character(x))),
+         collapse = ""), x, sep = "", collapse = "")
+ }))
> names(uniHap) <- paste("hap_", hapNum, sep = "")
> hapPool <- strsplit(uniHap, "-")
> hapPool <- data.frame(t(data.frame(lapply(hapPool, function(x) {
+     as.numeric(x)
+ }))))
> colnames(hapPool) <- colnames(keremRandSeq)
```

There are 115 unique haplotypes determined, and they are stored in the *data.frame* object `hapPool`.

For each unique haplotype, we also need to calculate its frequency among the observed haplotype sample:

```
> hapCount <- sapply(uniHap, function(x) {
+     sum(x == hapSeq)
+ })
> hapFreq <- hapCount/sum(hapCount)
```

Then we need to determine the index of every haplotype for each subject:

```
> noNameUniHap <- uniHap
> names(noNameUniHap) <- NULL
> hapIndex <- sapply(hapSeq, function(x) {
+     which(x == noNameUniHap)
+ })
> hap1Index <- hapIndex[c(1:length(hapIndex)%%2) == 1]
> hap2Index <- hapIndex[c(1:length(hapIndex)%%2) == 0]
```

The object `hap1Index` indicates the first haplotype for each subject, and `hap2Index` indicates the second haplotype.

We can then store all information together in a *haplo* object:

```
> phasedHaplo <- new("haplo", haploSeq = hapPool,
+     haploFreq = hapFreq, hap1 = hap1Index, hap2 = hap2Index,
+     poolHapPair = as.character(1:length(hap1Index)),
+     nPosHapPair = table(1:length(hap1Index)),
+     post = rep(1, length(hap1Index)), pheno = data.frame(CF = keremRandStatus))
```

The rest of the commands are the same as those for unphased data in Section 4.3.

6 Session Information

The version number of R and packages loaded for generating the vignette were:

```
R version 2.10.0 (2009-10-26)
x86_64-unknown-linux-gnu

locale:
 [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
 [3] LC_TIME=en_US.UTF-8        LC_COLLATE=en_US.UTF-8
 [5] LC_MONETARY=C              LC_MESSAGES=en_US.UTF-8
 [7] LC_PAPER=en_US.UTF-8       LC_NAME=C
 [9] LC_ADDRESS=C               LC_TELEPHONE=C
[11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C

attached base packages:
[1] tools     stats     graphics  grDevices utils     datasets  methods
[8] base

other attached packages:
[1] SHARE_1.0.4       MASS_7.2-46       haplo.stats_1.3.8
```

References

J. Besag and P. Clifford. Sequential monte carlo p-values. *Biometrika*, 78(2):301–304, June 1 1991.

S. R. Browning. Multilocus association mapping using variable-length markov chains. *American Journal of Human Genetics*, 78(6):903–913, Jun 2006.

J. Y. Dai, M. LeBlanc, N. L. Smith, B. M. Psaty, and C. Kooperberg. SHARE: an adaptive algorithm to select the most informative set of SNPs for genetic association. *Biostatistics*, In Press, 2009.

M. Stephens and P. Donnelly. A comparison of bayesian methods for haplotype reconstruction from population genotype data. *American Journal of Human Genetics*, 73 (5):1162–1169, Nov 2003.

M. Stephens, N. J. Smith, and P. Donnelly. A new statistical method for haplotype reconstruction from population data. *American Journal of Human Genetics*, 68(4): 978–989, Apr 2001.