Family Based Association Tests Using the fbat package

Weiliang Qiu
e-mail: stwxq@channing.harvard.edu
Ross Lazarus
e-mail: ross.lazarus@channing.harvard.edu
Gregory Warnes
e-mail: warnes@bst.rochester.edu
Nitin Jain
e-mail: nitin.jain@pfizer.com

October 3, 2007

Contents

1 Introduction 2
2 Pedigree data file format 2
3 Data quality control 3
4 Examples 3
A Notation 10
B Genotype coding methods 11
C Trait coding methods 12
1 Introduction

The R package fbat can be used to test the following null hypotheses for each marker based on family pedigrees:

\[ H_{01} : \text{the marker has no association and no linkage with the trait}; \]
\[ H_{02} : \text{the marker has no association with the trait in the presence of linkage}. \]

We assume that

- the families are nuclear families
- there are no missing genotypes and phenotypes for children
- markers are bi-allelic.

A more general software FBAT is available as a stand-alone executable with documentation and example files from http://www.biostat.harvard.edu/~fbat/fbat.htm. While this R package has some important limitations as present, these will be addressed in further versions.

2 Pedigree data file format

All fields are separated by whitespace (e.g. one or more spaces).

**First line**: names of all markers in the sequence of the genotype data. For example,

marker1, marker2, ..., marker_m.

**Remaining lines**: The remaining lines contain only non-negative integers and have the same format:

| family | pid | father | mother | sex | affection | marker1,1 | marker1,2 | ... | marker_m,1 | marker_m,2 |

where

- **family**: family id
- **pid**: patient id
- **father**: father id.

Use 0 (zero) for founders or marry-ins (parents not specified) in a pedigree. A **founder** in a pedigree is an individual who is not a child of any individuals in the pedigree.
**mother**: mother id.

Use 0 (zero) for founders or marry-ins (parents not specified) in a pedigree.

A **founder** in a pedigree is an individual who is not a child of any individuals in the pedigree.

**sex**: 1 – male; 2 – female;

**affection**: affection status (i.e., trait)

2 – affected; 1 – unaffected; 0 – unknown

**marker**<sub>i,j</sub>: allele <i>j</i> of marker <i>i</i>, <i>j</i> = 1, 2; <i>i</i> = 1, 2, . . . , <i>m</i>.

non-missing Alleles are represented by positive integers. Missing alleles are represented by zero (0).

### 3 Data quality control

The R package *fbat* also provides some basic QC functions.

The function *missGFreq* checks the completeness of genotypes. This function outputs counts of missing genotypes per marker and per subject.

The function *pedHardyWeinberg* checks the assumption of the Hardy-Weinberg equilibrium for markers.

The function *checkMendelian* checks the following possible Mendelian-related errors:

1. father id = subject id;
2. mother id = subject id;
3. could not determine if an individual is a parent or a child in a family;
4. inconsistent parental sex in a family;
5. parental genotypes are not compatible with childrens’ genotypes in a family;
6. all childrens’ genotypes are missing in a family;
7. inconsistent sib genotypes in a family.

### 4 Examples

To call the functions in the R package *fbat*, we first need to load it into R:

To read the pedigree file *CAMP.ped* into R, we use the function *readGenes* in the R package *GeneticsBase*:

```r
gSet<-readGenes(gfile="CAMP.ped", gformat="fbat")
```
The function `readGenes.ped` returns back an object of the R class `geneSet`.

Before we apply family based association tests, it would be good practice to check Hardy-Weinberg equilibrium for each marker based on parental data. We can use the function `pedHardyWeinberg` to do this.

```r
> data(CAMP)
Reading 8 markers and 2011 subjects from `CAMP.ped' ... 
generating 'geneSet' object...

Successfully read the pedigree file `CAMP.ped'.

Number of Markers: 8
Number of Subjects: 2011
Number of Families: 651

Reading 12 vars from `CAMPZ.phe' ... Done.

Number of Phenotype Variables: 12
Number of Observations : 2011

> ch <- pedHardyWeinberg(CAMP)

converting geneSet object to numerical matrix...
HWE test...

<table>
<thead>
<tr>
<th>nInfoInd</th>
<th>nGenotype</th>
<th>nHET</th>
<th>nHOM</th>
<th>nAllele</th>
<th>nMissing</th>
<th>chi2</th>
<th>df</th>
<th>p-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>m709</td>
<td>1271</td>
<td>3</td>
<td>6</td>
<td>1265</td>
<td>2</td>
<td>32</td>
<td>1</td>
<td>0.933</td>
</tr>
<tr>
<td>m654</td>
<td>1265</td>
<td>3</td>
<td>544</td>
<td>721</td>
<td>2</td>
<td>38</td>
<td>1</td>
<td>0.017</td>
</tr>
<tr>
<td>m47</td>
<td>1250</td>
<td>3</td>
<td>585</td>
<td>665</td>
<td>2</td>
<td>53</td>
<td>1</td>
<td>0.985</td>
</tr>
<tr>
<td>p46</td>
<td>1251</td>
<td>3</td>
<td>582</td>
<td>669</td>
<td>2</td>
<td>52</td>
<td>1</td>
<td>0.280</td>
</tr>
<tr>
<td>p79</td>
<td>1244</td>
<td>3</td>
<td>572</td>
<td>672</td>
<td>2</td>
<td>59</td>
<td>1</td>
<td>0.405</td>
</tr>
<tr>
<td>p252</td>
<td>1168</td>
<td>3</td>
<td>377</td>
<td>791</td>
<td>2</td>
<td>135</td>
<td>1</td>
<td>0.030</td>
</tr>
<tr>
<td>p491</td>
<td>1268</td>
<td>3</td>
<td>31</td>
<td>1237</td>
<td>2</td>
<td>35</td>
<td>1</td>
<td>0.659</td>
</tr>
<tr>
<td>p523</td>
<td>1269</td>
<td>3</td>
<td>377</td>
<td>892</td>
<td>2</td>
<td>34</td>
<td>1</td>
<td>0.241</td>
</tr>
</tbody>
</table>

The column `nInfoInd` means the number of informative individuals, i.e. individuals whose genotypes contain no missing alleles for the specified marker; the column `nGenotype` means number of possible genotypes; the column `nHET` means number of heterozygous genotypes; the column `nHOM` means number of homozygous genotypes; the column `nAllele` means number of alleles; the column `nMissing` means number of missing alleles; the column `chi2` means chi square test statistic; the column means `df` means degree of freedom of the chi square test statistic under the null hypothesis that Hardy-Weinberg condition holds; and the column `p-value` means p-value of the test.
To view the statistics for individual markers, we can use the function `viewHW`. For example,

```r
> viewHW(ch, "p79")
```

```
number of possible genotypes for marker p79 >>
[1] 3
genotype frequency >>
p79.1 p79.2 freq
[1,] 1 1 486
[2,] 1 2 572
[3,] 2 2 186
allele frequency >>
1 2
0.621 0.379
```

```
nInfoInd nGenotype nHET nHOM nAllele nMissing chi2 df
1244.000 3.000 572.000 672.000 2.000 59.000 0.693 1.000
p-value
0.405
```

To check Mendelian-related errors, we can use the function `checkMendelian`. For example,

```r
> tmp <- checkMendelian(CAMP, quiet = TRUE)
> cat("For each marker, how many families contains mendelian errors?\n")
```

```
For each marker, how many families contains mendelian errors?
> print(tmp$nMerrMarker)
m709 m654 m47 p46 p79 p252 p491 p523
20 159 160 155 160 140 26 122
```

```r
> cat("For each family, how many markers contains mendelian errors?\n")
```

```
For each family, how many markers contains mendelian errors?
> cat("tmp$nMerrFamily[1:10]>>\n")
tmp$nMerrFamily[1:10]>>
```

```r
> print(tmp$nMerrFamily[1:10])
family1 family2 family3 family4 family5 family6 family7 family8
0 4 4 2 2 2 0 4
family9 family10
4 1
```
> cat("For each family, how many times\n")

For each family, how many times

> cat("'father id = subject id' or 'mother id = subejct id'?\n")

'father id = subject id' or 'mother id = subejct id'? 

> cat("tmp$nErrFamilySample[1:10]\n")

tmp$nErrFamilySample[1:10]>

> print(tmp$nErrFamilySample[1:10])

 family1 family2 family3 family4 family5 family6 family7 family8
    0    0    0    0    0    0    0    0
family9 family10
    0    0

To count the number of missing genotypes for a marker or for a subject, we can use the function missGFreq. For example,

> res <- missGFreq(CAMP, founderOnly = FALSE, quiet = TRUE)
> cat("The number of missing genotypes for markers\n")

The number of missing genotypes for markers

> print(res$nMissMarkers)

   00 0* +0
m709 55 0 0
m654 60 0 0
m47  89 0 0
p46  68 0 0
p79  90 0 0
p252 188 0 0
p491 57 0 0
p523 53 0 0

> cat("The number of missing genotypes for the first 10 subjects\n")

The number of missing genotypes for the first 10 subjects

> print(res$nMissSubjects[1:10, ])

6
To get the family based association test statistics, we use the function `fbat`:

```r
> res <- fbat(CAMP)
```

The usage of the function `fbat` is

`fbat(geneSetObject, model="a", traitMethod=3, traitOffset=0, quiet=TRUE)`

The function argument `model` specifies the genotype codings. By default, we use the additive model (`model="a"`). Other available models include dominant (`model="d"`), recessive (`model="r"`), and genotype (`model="g"`) models.

The function argument `traitMethod` indicates the trait coding method. If `traitMethod` is equal to 1, then the trait is represented by `trait-offset` where `trait` is the sixth column (i.e., affection status) of the pedigree matrix and the value of `offset` is provided by the argument `traitOffset`. If the argument `traitMethod` takes value other than 1, then the trait is set to be 1 if the sixth column of the pedigree matrix takes value 2 and the trait is set to be 0 if the sixth column of the pedigree matrix takes value 1.

The function `fbat` returns a list. To summarize the values, degrees of freedom, and p-values of the test statistics for the markers, we can use the function `summaryPvalue`:

```r
> summaryPvalue(res)
```

```
****************************
<table>
<thead>
<tr>
<th>chisq</th>
<th>rank</th>
<th>pvalue</th>
</tr>
</thead>
<tbody>
<tr>
<td>m709</td>
<td>1.28</td>
<td>2.568393e-01</td>
</tr>
<tr>
<td>m654</td>
<td>9.57</td>
<td>1.977491e-03</td>
</tr>
<tr>
<td>m47</td>
<td>16.10</td>
<td>5.988938e-05</td>
</tr>
<tr>
<td>p46</td>
<td>10.03</td>
<td>1.540692e-03</td>
</tr>
<tr>
<td>p79</td>
<td>10.27</td>
<td>1.353705e-03</td>
</tr>
<tr>
<td>p252</td>
<td>20.00</td>
<td>7.744216e-06</td>
</tr>
<tr>
<td>p491</td>
<td>12.45</td>
<td>4.183778e-04</td>
</tr>
<tr>
<td>p523</td>
<td>26.09</td>
<td>3.245492e-07</td>
</tr>
</tbody>
</table>
****************************
```
To adjust multiple comparisons, we can use the function `p.adjust` in the R package `base` to adjust the \( p \)-values. For example,

```r
> pvals <- res$statPvalue[, 3]
> p.adjust.M <- p.adjust.methods
> p.adj <- sapply(p.adjust.M, function(meth) p.adjust(pvals, meth))
> noquote(apply(p.adj, 2, format.pval, digits = 3))
```

<table>
<thead>
<tr>
<th></th>
<th>holm</th>
<th>hochberg</th>
<th>hommel</th>
<th>bonferroni</th>
<th>BH</th>
<th>BY</th>
<th>fdr</th>
<th>none</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.256839</td>
<td>0.256839</td>
<td>0.256839</td>
<td>1.0000000</td>
<td>0.256839</td>
<td>0.698052</td>
<td>0.256839</td>
<td>0.256839</td>
</tr>
<tr>
<td>2</td>
<td>0.005415</td>
<td>0.003955</td>
<td>0.003955</td>
<td>0.015820</td>
<td>0.002260</td>
<td>0.006142</td>
<td>0.002260</td>
<td>0.001977</td>
</tr>
<tr>
<td>3</td>
<td>0.000359</td>
<td>0.000359</td>
<td>0.000359</td>
<td>0.000479</td>
<td>0.000160</td>
<td>0.000434</td>
<td>0.000160</td>
<td>5.99e-05</td>
</tr>
<tr>
<td>4</td>
<td>0.005415</td>
<td>0.003955</td>
<td>0.003081</td>
<td>0.012326</td>
<td>0.002054</td>
<td>0.005583</td>
<td>0.002054</td>
<td>0.001541</td>
</tr>
<tr>
<td>5</td>
<td>0.005415</td>
<td>0.003955</td>
<td>0.002966</td>
<td>0.010830</td>
<td>0.002054</td>
<td>0.005583</td>
<td>0.002054</td>
<td>0.001354</td>
</tr>
<tr>
<td>6</td>
<td>5.42e-05</td>
<td>5.42e-05</td>
<td>5.42e-05</td>
<td>6.20e-05</td>
<td>3.10e-05</td>
<td>8.42e-05</td>
<td>3.10e-05</td>
<td>7.74e-06</td>
</tr>
<tr>
<td>7</td>
<td>0.0002092</td>
<td>0.0002092</td>
<td>0.0002092</td>
<td>0.0003347</td>
<td>0.000837</td>
<td>0.002274</td>
<td>0.000837</td>
<td>0.000418</td>
</tr>
<tr>
<td>8</td>
<td>2.60e-06</td>
<td>2.60e-06</td>
<td>2.60e-06</td>
<td>2.60e-06</td>
<td>2.60e-06</td>
<td>7.06e-06</td>
<td>2.60e-06</td>
<td>3.25e-07</td>
</tr>
</tbody>
</table>

To view summary statistics of individual marker, we can use the function `viewstat`. For example,

```r
> viewstat(res, "p79")
```

******************************************************************************************
651 pedigree 2011 persons
360 informative families at marker p79
The alleles of marker p79 >>
[1] 1 2
Score for marker p79 >>
[1] 474 298
Expected score for marker p79 >>
[1] 437.5 334.5
Covariance matrix of the score for marker p79 >>

```
[,1] [,2]
[1,] 129.75 -129.75
[2,] -129.75 129.75
```
Moore-Penrose generalized inverse of covariance matrix

```
[,1] [,2]
[1,] 0.0001926782 -0.0001926782
[2,] -0.0001926782 0.0001926782
```
test statistics for marker p79 >>
```
chisq  rank  pvalue
10.267822736 1.0000000000 0.001353705
```
******************************************************************************************
Note that if the covariance matrix of the S score vector is singular, the Moore-Penrose generalized inverse is used.

Sometimes the user might want to know if a genotype a homozygous or heterozygous. The function `pedFlagHomo` can provide those information. For example,

```r
> res.f <- pedFlagHomo(CAMP)

converting geneSet object to numerical matrix...
flag homozygotes and heterozygotes...
dim(flagHomoMat)= 1303 8
length(ped[,2])= 1303
numHomo -- number of homozygous genotypes
numHetero -- number of homozygous genotypes
numMiss1 -- number of genotypes containing one missing allele
numMiss2 -- number of genotypes containing two missing alleles

counts>>>

numHomo numHetero numMiss1 numMiss2
m709 1265 6 0 32
m654 721 544 0 38
m47 665 585 0 53
p46 669 582 0 52
p79 672 572 0 59
p252 791 377 0 135
p491 1237 31 0 35
p523 892 377 0 34
```

The function `pedGFreq` gets genotype frequencies and percentages. For example,

```r
> res <- pedGFreq(CAMP)

converting geneSet object to numerical matrix...
counting genotype frequencies...
genotype counts>>>

1/1 1/2 2/2
m709 1265 6 0
m654 537 544 184
m47 175 585 490
p46 209 582 460
p79 486 572 186
p252 70 377 721
p491 1237 31 0
p523 840 377 52
```

The function `pedAFreq` gets allele frequencies and percentages. For example,
> res <- pedAFreq(CAMP)

converting geneSet object to numerical matrix...
count allele frequencies...
allele frequencies and percentages

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>m709</td>
<td>2536</td>
<td>6</td>
<td>0.998</td>
<td>0.002</td>
</tr>
<tr>
<td>m654</td>
<td>1618</td>
<td>912</td>
<td>0.640</td>
<td>0.360</td>
</tr>
<tr>
<td>m47</td>
<td>935</td>
<td>1565</td>
<td>0.374</td>
<td>0.626</td>
</tr>
<tr>
<td>p46</td>
<td>1000</td>
<td>1502</td>
<td>0.400</td>
<td>0.600</td>
</tr>
<tr>
<td>p79</td>
<td>1544</td>
<td>944</td>
<td>0.621</td>
<td>0.379</td>
</tr>
<tr>
<td>p252</td>
<td>517</td>
<td>1819</td>
<td>0.221</td>
<td>0.779</td>
</tr>
<tr>
<td>p491</td>
<td>2505</td>
<td>31</td>
<td>0.988</td>
<td>0.012</td>
</tr>
<tr>
<td>p523</td>
<td>2057</td>
<td>481</td>
<td>0.810</td>
<td>0.190</td>
</tr>
</tbody>
</table>

The functions `fbat`, `pedHardyWeinberg`, `pedFlagHomo`, `pedGFreq`, and `pedAFreq` have default forms (`fbat.default`, `pedHardyWeinberg.default`, `pedFlagHomo.default`, `pedGFreq.default`, and `pedAFreq.default`) that use a pedigree matrix as input.

**Appendix**

### A Notation

For a given marker,

- $Y_{ij}$ — Observed trait of the $j$-th offspring in family $i$.
- $T_{ij}$ — A function of $Y_{ij}$.
  \[
  T_{ij} = T(Y_{ij}).
  \]
  For example
  \[
  T_{ij} = T(Y_{ij}) = Y_{ij} - \mu_{ij},
  \]
  where $\mu_{ij}$ is an offset.
- $g_{ij}$ — Genotype of the $j$-th offspring in family $i$;
- $X_{ij}$ — A function of $g_{ij}$.
  \[
  X_{ij} = X(g_{ij}).
  \]
- $S$ score:
  \[
  S = \sum_{ij} T_{ij}X_{ij} = \sum_{ij} T(Y_{ij})X(g_{ij}).
  \]
• test statistic:

\[ U = S - \text{E}[S|H_0, \mathcal{C}], \]

where \( \mathcal{C} \) is a condition set. When parental genotypes are complete, the condition set \( \mathcal{C} = \mathcal{T} \cup \mathcal{G} \), where \( \mathcal{T} \) is the observed traits in all family members and \( \mathcal{G} \) is the parental genotypes. When parental genotypes are incomplete, the condition set \( \mathcal{C} = \mathcal{T} \cup \mathcal{G}^* \cup \mathcal{G}_{\text{offspring}} \), \( \mathcal{G}^* \) is the partially observed parental genotypes and \( \mathcal{G}_{\text{offspring}} \) is the set of offspring genotypes (i.e., the offspring genotype configuration).

• \( V \) – variance or covariance matrix of \( U \) under the null hypothesis \( H_0 \). I.e.,

\[ V = \text{Cov}(U|H_0, \mathcal{C}) = \text{Cov}(S|H_0, \mathcal{C}). \]

• For the univariate case,

\[ Z = \frac{U}{\sqrt{V}} \bigg|_{H_0, \mathcal{C}} \sim \text{N}(0, 1). \]

• For the multivariate case,

\[ \chi^2 = U'V^{-1}U \bigg|_{H_0, \mathcal{C}} \sim \chi^2_r, \]

where \( r = \text{rank}(V) \).

B Genotype coding methods

Denote \( K \) as the number of all possible different alleles for the locus and \( X \) as the vector of genotype coding.

\textbf{GEN} \( X \) is a vector with length equal to the number of genotypes that are possible given the parental genotypes in the sample, a maximum of \( K(K+1)/2 \) genotypes, and with elements equal to 1 or 0 to indicate which of the possible genotypes is equal to the genotype \( g \).

\textbf{GDOM} codes the \( j \)th element of the vector \( X \) as \( x_j = 1 \) if genotype \( g \) has one or two alleles of type \( j \), otherwise \( x_j = 0 \). \( X \) is a vector of length \( K \).

\textbf{GREC} codes the \( j \)th element of the vector \( X \) as \( x_j = 1 \) if genotype \( g \) has two alleles of type \( j \), otherwise \( x_j = 0 \). \( X \) is a vector of length \( K \).

\textbf{GTDT} scores the number of alleles of a particular type by coding \( x_j \) equal to the number of alleles of type \( j \) in the genotype \( g \) (i.e., \( x_j = 0, 1, \) or 2 if \( g \) has 0, 1 or 2 alleles of type \( j \)). \( X \) is a vector of length \( K \).
2-allele case

Example of different marker codings for a marker with $K = 2$ alleles, see Schaid (1996)

<table>
<thead>
<tr>
<th>genotype</th>
<th>$X(g)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$g$</td>
<td>GEN</td>
</tr>
<tr>
<td></td>
<td>$(A, a)$</td>
</tr>
<tr>
<td>$AA$</td>
<td>(0,0,0)</td>
</tr>
<tr>
<td>$Aa$</td>
<td>(1,0,0)</td>
</tr>
<tr>
<td>$aa$</td>
<td>(0,1,0)</td>
</tr>
</tbody>
</table>

3-allele case

Example of different marker codings for a marker with $K = 3$ alleles, see Schaid (1996) (This table is Table 4 of Horvath et al.’s report for FBAT software)

<table>
<thead>
<tr>
<th>genotype</th>
<th>$X(g)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$g$</td>
<td>GEN</td>
</tr>
<tr>
<td></td>
<td>$(A, B, C)$</td>
</tr>
<tr>
<td>$AA$</td>
<td>(0,0,0,0)</td>
</tr>
<tr>
<td>$AB$</td>
<td>(1,0,0,0)</td>
</tr>
<tr>
<td>$AC$</td>
<td>(0,1,0,0)</td>
</tr>
<tr>
<td>$BB$</td>
<td>(0,0,1,0)</td>
</tr>
<tr>
<td>$BC$</td>
<td>(0,0,0,1)</td>
</tr>
<tr>
<td>$CC$</td>
<td>(0,0,0,0)</td>
</tr>
</tbody>
</table>

C Trait coding methods

Denote $Y_{ij}$ as the trait of the $j$-th child of the $i$-th nuclear family. $Y_{ij}$ can be dichotomous, measured (i.e., continuous?), time-to-onset (i.e., censored?)

The trait coding methods ($T_{ij} = T(Y_{ij})$) are listed below:

- $T_{ij} = 1$ if the $j$th child is affected; $T_{ij} = 0$ otherwise.
- $T_{ij} = Y_{ij} - \mu_{ij}$, where $\mu_{ij}$ is an offset.
- $T_{ij} = Y_{ij} - \mu_{ij}(x'\beta)$, where $E(Y_{ij}|x) = \mu_{ij}(x'\beta)$, and $x$ are design matrix of covariates, $\beta$ are unknown parameters.