User Manual for GIGI-Check v1.00

Author:
Charles Y K Cheung [cykc@uw.edu]
Department of Biostatistics
University of Washington

Last Modified on 4/24/2014

Contents

Introduction

GIGI-Check is a C++ program to detect Mendelian consistent genotyping errors of dense markers in pedigree data. It detects genotyping errors by using Inheritance Vectors (IVs), which are inferred by using sparse framework genotypes available on a subset of relatives in the pedigree. Thus, our error detection approach consists of two steps. The first step is to infer IVs at the positions of framework markers using gl_auto, a MCMC-based program from the MORGAN package. We assume that these markers are free of genotyping errors. The second step is to detect errors in dense genotypes by GIGI-Check using the IVs and pedigree structure file from MORGAN.

In this documentation, we use the following terminology. Framework markers are a relatively sparse set of markers used to infer IVs on a chromosome of interest. Dense markers are markers with missing genotypes on some subjects that we want to evaluate for possible genotyping errors. For example, these dense markers may be genotypes obtained from sequence data or from a dense SNP panel, and may be typed on fewer and even different subjects in the pedigree. See the publication describing GIGI-Check, below, for more information.

GIGI-Check is developed under the linux environment.

Citing GIGI-Check

Cheung, CYK., Thompson, E.A., Wijsman, E.M. (2014) Detection of Mendelian Consistent Genotyping Errors in Pedigrees. *Genetic Epidemiology* 38(4):291-299.

Software URL

http://faculty.washington.edu/wijsman/software.shtml

Files in GIGI-Check software distribution

GIGI-Check software code and its dependency files - the Mersenne random number generator

- example folder

Inferring IVs using gl_auto

The first step is to use **framework markers** to infer IVs. For this purpose, we use gl_auto, a program in the MORGAN package that is freely available at http://www.stat.washington.edu/thompson/Genepi/MORGAN/Morgan.shtml. To infer IVs in gl_auto, we need to supply the required files in MORGAN format:

(1) Pedigree file
(2) Marker file: this is a composite file that contains the map positions of framework markers (in centiMorgans assuming the Haldane map function), allele frequencies of framework markers ,

and genotype data of framework markers
(3) Parameter file used to run gl_auto

Refer to the documentation of MORGAN for guidance on setting up these files and on running gl_auto.

Example files used to infer IVs using gl_auto are included under the "example/gl_auto_example" directory.

Overview
We need to obtain 2 files from running gl_auto:
(a) Framework IVs file: GIGI-Check uses this file, which contains IVs at framework positions
(b) Pedigree-meiosis file: this file contains the information about the structure of the pedigree on which we want to detect genotyping errors. This file is different from the pedigree file used in gl_auto. In addition to the pedigree structure, this file also contains information that GIGI-Check needs to determine how the Inheritance Vectors (Meiosis indicators) are organized (i.e. the i^th line of the meisosis indicator belongs to which subject in the pedigree and whether this meiosis indicator is this person's maternal or paternal chromosome). We need to create this file from the console output of gl_auto.

Obtaining the files

(a) Framework IVs file
In gl_auto's parameter file, instruct gl_auto to display the output as Meiosis Indicators instead of Founder Genome Labels. Please make sure to use the option

output meiosis indicators

instead of "output founder genome labels" in the control/parameter file of gl_auto.

(b) Pedigree-Meiosis file
We need to create the pedigree-meiosis file (b) from the console output of gl_auto. It is very easy to make this file.
When gl_auto runs, the program prints a huge amount of output to the console. This console output actually contains the content of the pedigree-meiosis file that we need to extract.

1. In order to extract this content, we first need to direct the console output to a file by using the ">" directive so we can subsequently extract the content from this file.
   ie. **./gl_auto gl_auto_parameter_file > glauto_console_output.txt**

2. Then, we extract the pedigree-meiosis content from the console output to a new file. To simplify the creation of this file, use the Perl script "extractPedMeiosis.pl"
   - **Usage: perl extractPedMeiosis.pl glauto_console_output.txt FILENAME_PED_MEIO**
     - We need to have Perl installed in linux.
     - We assume that glauto_console_output.txt is in the same directory as

extractPedMeiosis.pl

- Alternatively, this file can also be easily created by the user. See Appendix B for the creation of this file manually and also for an example of how this file looks like.

Installing GIGI-Check

Simply unzip the files, navigate to the code directory, and type

make

If make does not work, go to the GIGI-Check.cpp's directory and install the program by

g++ GIGI-Check.cpp -o GIGI-Check

Running GIGI-Check

GIGI-Check accepts a parameter file. To run GIGI-Check, type

./GIGI-Check <parameter file> <options>

To run the example file, go to the main GIGI-Check program directory, and type

```
./GIGI-Check example/param.txt
```

Options

[The flags are case-sensitive.]

Flag	**Purpose**
-seed=NUM	to change the default seed. NUM should be a number between 1 and 999999
-prog=NUM	to print the progress of imputation at every NUM markers
-outD=	to specify the absolute directory path of where the output files will be created. If this flag is missing, the output files will be saved to the user's current directory. e.g. /home/charles/output [no character '/' at the end of the directory name]
-long	to read dense genotype marker file as the "long" format (rows are markers and columns are genotypes of individuals)
-A1	to only perform error detection using the faster Method A1
-threshold=NUM	this threshold determines the error detection threshold for which markers to be flagged by A1, as summarized in the output file "summary_markers_flagged.txt" this number must be between 0 and 1 (default is 0.05)

An example of running GIGI-Check with additional options.

```
./GIGI-Check example/param.txt  -outD=/home/charles/GIGI-Check_output
```

To run GIGI-Check with error detection threshold of 0.1 and with A1 only,

```
./GIGI-Check example/param.txt  -threshold=0.1 -A1
```

Making the Parameter File

The parameter file tells GIGI-Check where to look for the required files and where to save the output files. GIGI-Check needs the pedigree meiosis file (that user prepares from the output of gl_auto), dense marker file, IVs file (from gl_auto), map of the sparse marker file, map of the dense marker file, and allele frequencies of the dense marker file.

An **example** of the parameter file is found in the example directory under
example/param.txt

In GIGI-Check, the parameter file is organized as follows:
line 1 - filename of the pedigree meiosis file
line 2 - framework IVs file
line 3 - number of sampled realizations in the IV file
line 4 - filename of the map positions of the framework markers file (cM based on Haldane map function)
line 5 - filename of the map positions of the dense markers file (cM based on Haldane map function)
line 6 - filename of the dense marker genotype file
line 7 - filename of the allele frequencies of the dense markers file (optional)
line 8 - the assumed allelic error rate (e.g. 0.01 for 1%) (optional???!!!)

Notes:
I suggest using the absolute paths of the filenames instead of relative paths. A relative path is relative to the directory containing the executable program. (The parameter file in the example folder is created using a relative path.)

*line 1: user creates this file from the gl_auto's console output. GIGI-Check can only process 1 pedigree at a time.
*line 2: when you run gl_auto, you should instruct gl_auto to print Meiosis Indicators instead of Founder Genome Labels
*line 3: – this corresponds to the number of samples that the user actually prints to the meisois indicator file.

File Formats

Examples of these files are provided in the example directory [refer to the param.txt for the filename of these files]

1. pedigree meiosis file [line 1]. The creation of this file was discussed in the **Inferring IVs using gl_auto** section

The pedigree meiosis specifies the pedigree structure and the index of meioses that GIGI-Check needs to use to read in the corresponding Inheritance Vectors. The content of this file is generated by gl_auto.

2. framework IVs file [line 2]

The Inheritance Vectors file describes the descent pattern of chromosomes at the positions of the framework markers. It is the output file that **gl_auto** generates.

3. An integer that indicates the number of sampled realizations in the IV file [line 3]. This is the number of samples obtained from the output of gl_auto, which is usually not the same as the number of MCMC realizations run in gl_auto because gl_auto has the option to output every few (e.g. 30 or 50) MCMC realizations to avoid printing highly correlated MCMC samples.

4. map positions of framework markers [line 4]

The marker map positions of the framework markers file is a text-file which contains the map distance in centi-Morgans (cM) based on the Haldane map function. Markers must be ordered in ascending order and consistent with the order used in gl_auto. Each line contains the position of a marker.

```
position of Marker1
position of Marker2
position of Marker3
...
position of MarkerN
```

eg

```
1.0
2.0
3.0
4.0
...
```

5. map positions of dense markers [line 5]
Similar to the marker map for framework panel of markers, the marker map of dense markers is a text-file which contains the map distance in centi-Morgans (cM) based on the Haldane map function. Markers must be ordered in ascending order. Each line contains the position of a marker.

eg
0.5
0.7
0.9
1.1
1.15
...

6. dense marker genotype file [line 6]

The dense markers are the markers that we want to detect Mendelian consistent genotyping errors. The dense marker file contains the genotypes of observed individuals.

(a) This file is space-delimited
(b) The markers should be sorted by ascending map positions.
(c) Alleles are labeled numerically starting from 1,2,... in ascending order. A missing allele is indicated with a 0. *If the original genotypes are alpha-numeric, users first need to convert the genotypes to an indexed numerical format.*

Users have two options in the format of the genotype file:
(1) a *long* format (rows are markers and columns individuals) or
(2) a *wide* format (rows are individuals and columns are markers). The long format should be used for files with many markers (e.g. > 10,000).

Note: The default (assumed) genotype file format in GIGI-Check is the traditional *wide* format typically used in linkage analysis programs. To let GIGI-Check know that your file is in the *long* format, you use the "-long" flag.
e.g. ./GIGI-Check example/param_longFormat.txt **-long**

Option 1. The *long* format
Each row of the file contains the genotypes for a *marker*. Consistent with the BEAGLE's genotype data file format, this file requires a header line.

The header contains the following information:
id person1 person1 person2 person2 person3 person3 …

Here the id is the name of the marker. The pair of columns for each individual represents the first allele and second allele of the specified individual. Unlike BEAGLE's genotype file format, GIGI-Check's long file format does not have the "I" column because GIGI-Check assumes that every row contains a marker.

An example of this file looks like:
id 101 101 102 102 103 103
rs0001 1 1 1 2 1 1
rs0002 1 2 0 0 1 2
rs0003 2 2 2 2 1 2

Note: if the *long* format is used, the outputs will also be generated in the *long* format.

Option 2. The default *"wide"* format
Each row of the file contains the genotypes for an *individual*. This file does not contain a header line. The first column specifies the name of an individual. The subject_ID can be an alphanumeric string. In this "wide" format, the name of the marker is not retained.

Subject_ID allele1_marker1 allele2_marker1 allele1_marker2 allele2_marker2 allele1_marker3 allele2_marker3 ...
An example of this file looks like:

```
101 1 1 1 2 2 2
102 1 2 0 0 2 2
103 1 1 1 2 1 2
```

7. allele frequencies of the dense marker file [line 7]

Each row contains the allele frequencies of the dense markers. The first column is the allele frequency of allele 1, the second column is the allele frequency of allele 2, etc... Each row must sum to 1. The allele frequencies file is space-delimited.

allele frequencies of marker 1
allele frequencies of marker 2
allele frequencies of marker 3
...

eg.

```
0.4 0.6
0.2 0.3 0.5
0.8 0.2
...
```

Output files

GIGI-Check displays the results of error detection in two files: “results_GIGI-Check.txt” and “summary_markers_flagged.txt”.

1. “results_GIGI-Check.txt”

The first line is the header. Results from GIGI-Check on each dense marker is displayed in a separate line.

e.g.

```
A1_pctConsistent A2_probNoErr ML_person ML_person_prob allele1 allele2
0.866667 0.888544 . . . .
0.9 0.954627 . . . .
0.766667 0.784216 . . . .
0.933333 0.927275 . . . .
1 0.988929 . . . .
1 0.989176 . . . .
0.4 0.439488 . . . .
0 0 514  0.823529 2 2
0.566667 0.561086 . . ….
```

Main summaries: (refer to the GIGI-Check paper)

A1_pctConsistent	- percent consistency [Method A1]
A2_probNoErr	- the posterior probability that there is no error in this marker [Method A2]

Summaries associated with Method A2

ML_person	- the most likely person with genotyping error
ML_person_prob	- the posterior probability that this person has a genotyping error
allele1 allele2	- the most likely true genotype of ML_person

Note: If ML_person_prob < A2_probNoErr, ML_person, ML_person_prob allele 1, and allele 2 would not be printed because it is more likely that the marker does not contain a genotyping error.

B. “summary_markers_flagged.txt”

This file contains a summary of which markers are flagged.

e.g.

```
Markers flagged by percent consistency (A1) because they are below the error detection threshold of
0.05. Markers are indexed by 1, ..., N
8 27 63 67 74 86 90 91 132
```

Frequently Asked Questions (FAQs)

1. **What threshold should we use to flag for error?** This decision is a tradeoff between sensitivity and specificity. From computer simulation, we see that in almost all cases a threshold very close to 0 can often detect most of the errors that are detectable by this class of approach. However, we often see that sensitivity may increase markedly if we relax the detection threshold slightly from 0, although after that increase, sensitivity does not continue to increase at that rate if we continue to relax the threshold. For convenient purpose, we simply set the default threshold to 0.05 (5%), and user can flexibly adjust. Please refer to the paper.

2. **How do I choose framework markers?** Because the framework markers are assumed to be sparse, we want to choose framework markers that are informative about which chromosomes are being transmitted at the framework loci. First, framework markers typed on a large number of subjects tend to be most informative. Second, framework markers that are multi-allelic tend to be more informative than di-allelic markers if they are available. Third, if framework markers are SNPs, markers with high minor allele frequencies in the sample tend to be more informative. Fourth, framework markers should be moderately but not too dense (eg. not denser than 1 marker per 0.3 cM because of concern about MCMC mixing and violation of the assumption of Linkage Equilibrium) If the framework markers are multiallelic, this spacing should be greater, e.g., not denser than 1 marker per 2 cM for good MCMC mixing.

3. **What do I have to do to be careful about the genetic map that I am using?** The map positions are in map distances based on the Haldane map function, instead of Kosambi map function or sequence positions. If this map is based on the Kosambi map function, the user will need to convert the map positions to Haldane map function using an appropriate conversion method.

   Also, since recombination fractions are relative to each other, we strongly encourage the user to generate both the framework map positions and the dense marker positions at the same time.

Appendix A

Creating the Pedigree-Meiosis file manually from the console output of gl_auto.

Using a text editor, we open the console_output.txt and fetch the line that begins with
"name   name.pa   name.ma Compnt pat.meio mat.meio"
We copy this line and table below. We paste this table to another file and save it.
The file includes the header line and looks like this:

```
 name      name.pa      name.ma Compnt pat.meio mat.meio
        2100_6           0            0     1     0     0
       2100_21           0            0     1     0     0
       2100_25           0            0     1     0     0
       2100_29           0            0     1     0     0
       2100_31           0            0     1     0     0
       2100_39           0            0     1     0     0
…
      2100_907           0            0     1     0     0
      2100_908      2100_901      2100_902     1     2     1
      2100_909           0            0     1     0     0
      2100_910      2100_901      2100_902     1     4     3
      2100_911      2100_901      2100_902     1     6     5
      2100_915      2100_907      2100_908     1     8     7
```
...(until the end of table)

License

Acknowledgement

Supported by funding from the National Institutes of Health, grants R37GM046255, P01HL030086, and P50AG05136.

I would also like to thank Elizabeth Marchani Blue for providing feedback.