Module 17: Session 9: MORGAN computing session

Elizabeth Thompson
Department of Statistics, University of Washington.

1 Overview

MORGAN 2.6 was released to the web Feb 29, 2004. It can be found at www.stat.washington.edu/thompson/Genepi/MORGAN/Morgan.shtml. The version we are using here is MORGAN 2.6, with three small bug fixes. Additionally, for simplicity, I have compiled a version of **lm_markers** for a quantitative trait, and called it **lm_quant**. In the current MORGAN 2.6.1, to be released later this summer, the quantitative trait option in **lm_markers** is via a parameter statement, and does not require a separate program.

The structure of MORGAN is of numerous libraries of C-routines, grouped according to general purpose (e.g. **Sample, Nghds, Markers**) and other directories of main programs, also grouped (e.g. **PedComp, Genedrop, Autozyg**). The main set of programs we will use are actually in the **Autozyg** directory, but for purposes of the lab you do not need to know exactly where the programs are. We will focus on programs for the MCMC estimation of lod scores from multiple-marker data on an extended pedigree.

The primary documentation in MORGAN is via plain text README files within the directories of the package. In particular, each main program directory contains a README_userdoc file, which should be your primary resource if you become a MORGAN user. However, to get started, there is also a tutorial available in numerous formats – see the web page for details. The version available for release is still that for MORGAN 2.5, but we will have a pre-release version of the tutorial updated for 2.6 available in html format.

Each main-program directory contains a subdirectory Gold, or Gold1 and Gold2, containing test examples of use of each program. These are (mostly) different examples than the ones we will use to illustrate the use of the package, but you may like to look at them.

2 Set-up

How much of this is true will depend on how/where things are installed.

Morgan 2.6 (with bug fixes) will have been installed in a directory whose full path name we may need to know – I will here call it **Morganpath**. This means that all the executables will have been

created, and will be in their respective subdirectories of **Morganpath**.
In the **Morganpath** directory, I will also place a subdirectory **morgan-tut-html** containing a (hopefully) up-to-date version of the MORGAN 2.6 tutorial. It is recommended you open the file

Morganpath/morgan-tut-html/morgan-tut.html

in your browser.
In your own file space for computing for this module, you will have a directory **ncsu-examples**. This is where you will run the examples that have been set up for you, and look at parameter statements etc. It is analogous to the **examples** directory described in the tutorial, except some things have been modified for current purposes.
The following may or may not have been done for you. In the Makefile in **ncsu-examples** you need to set

MORGANDIR = **Morganpath**

as the first (non-comment) line of the Makefile (actually line 4 ?). This should be the ONLY change needed to the Makefile.
There are basically two ways to run MORGAN programs:
1. One is to run the specific examples which have been set up, using the **make** command. For example: "**make lm_quant.out**" will run the particular example of **lm_markers** for a quantitative trait that has been set up. This is the easiest way.
2. For the more ambitious who want to set up their own parameter files, programs can also be run using the executable names directly. To do this you must first **make linka** (this is not a typo). This **make** creates symbolic links to all the executable programs in the relevant subdirectories of **Morganpath**. Now, for example, you could say

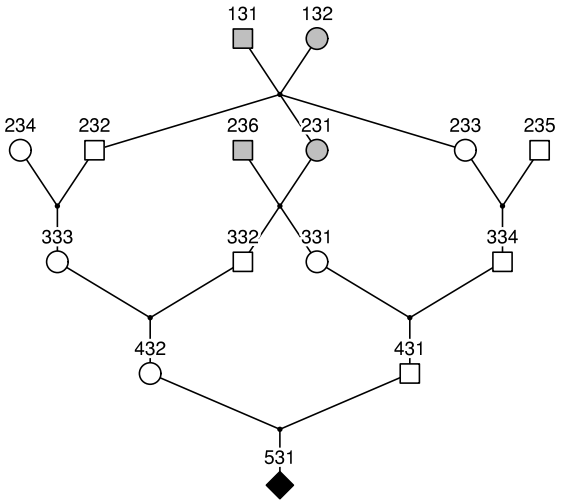
lm_quant my-par-file > my-out-file

to run the **lm_quant** program, using your parameter file **my-par-file** and producing output **my-out-file**.
(This may not work on Solaris: there were problems in testing.)
Note that since most of these programs use Monte Carlo, we need random numbers, and hence seeds for the random number generator. For the Gold standards, and for test examples I will provide output for, we use default seeds. However, it will be more interesting for you to have varying seeds. In MORGAN this is typically done by reading a starting seed from a *seed file*. There are several in **ncsu-examples**, but I will mostly use **seedfil**. At the end of a run, a seed is output to (usually) the same *seed file*, and this will become the seed for the next program. Since you will all start with the same **seedfil**, if you all run the same programs in the same order you will get identical results. But if you do things in different orders, then your seeds will be different and you will get different results.
Finally, note that MCMC programs can take CPU time. For the Gold standard examples, runs are very short, as these examples are simply to validate installation. For real analyses of real data, sometimes runs are quite long! The examples in **ncsu-examples** are a compromise. The runs should be long enough to give reasonable results, but don't expect too much.

3 Basics of Morgan

The examples in this section are primarily to introduce a variety of MORGAN parameter statements, which will then be used in our main example of MCMC lod-score estimation. For more details of parameter statements, see the tutorial and/or the README documentation. Note that parameters statements are to a very large extent free format. Each statement must be begun

on a new line and starts with a particular keyword – such as **input, select, ...**. A statement may occupy any number of lines; only the first 4 characters of keywords are significant; any amount of "white space" may occur between items etc. The statements in these examples have been "tidied" to make them easier to read, but items do **not** have to be alligned etc.



In our examples we will use the standard small 15-member pedigree used in the book and other documentation. This gets a bit boring, but it has the advantage of being
(a) small, so we can do exact and Monte Carlo computation,
(b) will give results in reasonable time
(c) has enough loops to be interesting
(d) having pictures of it in the notes
(e) is familiar to you from the lectures, and
(f) has been very well studied by many people as computational example.

Figure 1: JV pedigree. The pedigree structure derives from a real study of a rare recessive trait by Goddard et al. (1996: Am J Hum Genet **58:** 1286-1302).

3.1 Pedigree specification: Pedchk example

We start with pedigree specification: as an example we will run the **pedchk1** example from the tutorial examples. The program **pedcheck** checks validity of pedigrees (duplication of identifiers, missing parent individuals, gender, etc.). Additionally, if requested (in fact, by default?), it will reorder individuals into chronological order (i.e. parents before offspring) by component. The test pedigree for this example consists of two copies of the JV pedigree: the second is not chronologically ordered.
Here is the parameter file **pedchk_par** for this example:

```
input pedigree file 'pedchk_ped'
input pedigree size 30
input pedigree record  gender absent
input pedigree record  observed present
assign gender
output pedigree chronological
output pedigree file 'pedchk_oped1'
```

Here are a few lines of the specified input pedigree file **pedchk_ped**:

```
input pedigree record names 3 integers 1
***************
132    0     0    0
131    0     0    0
236    0     0    0
231   131  132    0
232   131  132    0
....
```

Note that the dividing line of (at least) 4 asterisks is significant. All else is free format. Parameter statements can be here (above the asterisks) or in the parameter file. Since we have told it there are 30 individuals, if there are more in the file it will read only the first 30.
You may run the example by saying **make pedchk1.out**. Alternatively, if you have linked the executables you may say **pedcheck pedchk_par > pedchk_out1**. Note that in either case your standard output is in the file **pedchk_out1**. If using **make** the standard error output is also in the output file: otherwise it will come to the screen. In either case, the output pedigree file is as specified in the parameter file: **pedchk_oped1**. Here are the first few lines

```
input pedigree size 30
input pedigree record names 3 integers 2
input pedigree record father mother
input pedigree record gender present
input pedigree record observed present
********
132   0   0  2  0
131   0   0  1  0
236   0   0  1  0
231 131 132  2  0
232 131 132  1  0
234   0   0  2  0
...
```

Note it puts out the parameter statements relative to the file, so the file can be used directly for other MORGAN programs. Note the second component has been ordered chronologically. Note parent individuals now have gender assigned: non-parents remain of unknown gender.

3.2 Map specification: Genedrop example

To introduce the specification of marker maps and trait information we will use the **genedrop** program. This is a straightforward simulation program to generate data at linked markers and on quantitative traits with major gene and (in general) additive polygenic effects.
The example uses the pedigree file **good_ped** which is two copies of JV, now reordered correctly. In fact, there should be no difference between your generated **pedchk_oped1** and the provided **good_ped** file. Note in this example in the Makefile, **make genedrop out** specifies the command

genedrop genedrop_par ped good_ped oped genedrop_oped > genedrop_out

That is, input and output pedigree files are specified in the command line, not in the parameter file.
Here is the parameter file **genedrop_par** (reordered slightly):

```
input seed file 'mymarkseeds'
output marker seeds only
output seed file 'mymarkseeds'

simulate chrom 5 markers 5 traits 1
simulate chrom 7 markers 3

map chrom 5 marker dist  25.54 25.54 25.54 25.54
map chrom 7 marker recom frac 0.1 0.2

set chrom 5 markers 1 2 3 4 5  freqs 0.2 0.2 0.4 0.2
set chrom 7 markers 1 2 3      freqs 0.2 0.2 0.4 0.2

map chrom 5 trait 1 marker 2 recom frac 0.1127
set trait 1 freqs 0.95 0.05

set trait 1 geno means -10. 0.0 10.0
set trait 1 residual variance 50.
set trait 1 additive variance 0.0

output pedigree chronological
output pedigree record founder gene labels
output pedigree record trait latent variables
```

Note the specification of input and output seedfiles: similar for all programs using Monte Carlo. Note the map statements, specified for each chromosome used, via recombination frequencies or genetic distances. (See below for an example of gender-specific maps.) See the specification of trait loci relative to the marker map. See the specification of marker and trait allele frequencies, and of other items relating to a quantitative trait, potentially with a polygenic component (although in this example the additive variance is 0).
If you wish you may run this example either via **make genedrop.out** or by

genedrop genedrop_par ped good_ped oped genedrop_oped > genedrop_out

3.3 MCMC specification: lm_auto example

Finally we will look at a small MCMC example: this one is not lod_scores, but the program **lm_auto** which estimates gene identity by descent pattern probabilities, given marker data. We include this example mainly for the specification of MCMC variables.
Here is the parameter file (slightly rearranged):

```
input pedigree file 'ped15'
input seed file 'seed15'

map gender F  markers            recomb fract .2 .2 .2 .2
map gender M  markers            recomb fract .1 .3 .1 .3
map gender F  trait 1  marker 2  recomb fract .1127
```

```
map gender M  trait 1  marker 2  recomb fract .055

set markers 1 2 3 4 5  freqs  .2 .2 .4 .2
set trait   1          freqs  .95 .05

set marker data 5
  333   1 3  1 3  1 3  1 3  1 3
  331   3 4  3 4  3 4  3 4  3 4
  334   2 3  2 3  2 3  2 3  2 3
  431   3 4  3 4  3 4  3 4  3 4
  531   3 3  3 3  3 3  3 3  3 3
select all markers  traits 1

set L-sampler probability 0.2
set MC iterations 2000

set proband gametes  531 1  531 0  331 0  333 1
set window patterns  0 4
set locus window   3
```

Pedigree **ped15** is one copy of JV pedigree – correctly ordered.
Note the gender-specific maps: note also that the trait location is in interval from marker 2 to 3, but at the mid-point on the female map, but close to marker two on the larger male map interval. (This might have been a mistake originally, but retained for demonstration purposes: the prograns will check that male and female meioses have same locus order, but this is the only constraint.)
Note the specification of marker data: if preferred this information can be given in a separate marker data file. Here the data are very simple and highly artificial: we have identical data at each of the five loci. Note that only individuals who have data at any markers need be included. A missing marker genotype is denoted "0 0"; individuals with no data (all genotypes "0 0") can be included if more convenient (for example for using the same files with non-MORGAN programs). Here we **select all markers** – that is all 5 will be used. In the lod-score example below we will choose to select a subset of the markers.
Note also the MCMC specification: here the simplest possible: a request for 2000 MCMC scans and a proportion 20% of L-sampler (80% M-sampler). See the output for the used defaults on starting configurations and burn-in. From the output:

```
 Sequential imputation to be used for setup, using 20 samples
 Sampling is to be by scan, with scores estimated every scan
 L-sampler probability set to 0.200
 The number of MC burn-in iterations is 200
 Number of MC iterations requested is 2000
 with progress checked every 2000 iterations
```

Finally, the parameter specifies which haplotypes are to be scored for gene identity by descent, and how it is to be scored: see the tutorial if you are interested in this.

You may run the example using **make lm_auto.out** or equivalently

lm_auto par15a > lm_auto_out

The output file is **lm_auto_out**. See the tutorial for more information.

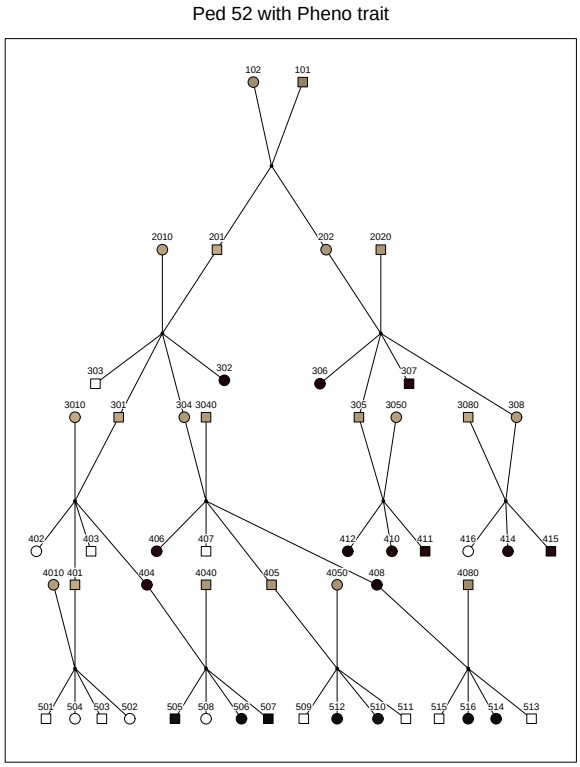
4 MCMC Lod score estimation

The three lod-score programs are **lm_lods, lm_markers, and lm_bayes**. See the tutorial for a bit of information about these programs. **lm_lods** is the oldest, and is based on lielihood-ratio estimation. **lm_markers** is our implementation of the Lange-Sobel estimator, where sampling is conditional only on marker data (hence the name). **lm_bayes** is the pseudo-Bayes estimator of George and Thompson (2003).

Additionally, there is a version of **lm_markers** (here called **lm_quant**) which does lod scores for a quantitative trait. In MORGAN 2.6.1 lm_quant is an option within lm_markers, set by a parameter statement

set trait data quantitative

but in 2.6 there is a hard-wired flag QUANT. The same quantitative trait option will be added, eventually, for other programs based on the LM-sampler.



Pedigree drawing by **Pedfiddler**,
written by J.C.Loredo-Osti.
See www.stat.washington.edu/Genepi/
pangaea.shtml for details.

Grey: unobserved
For the phenotypic trait:
White: unaffected
Black: affected

Figure 2: Ped52-simulation pedigrees for lod scores

The pedigree used here is Ped52, a constructed 5-generation 52-member pedigree with 12 founders and 80 meioses (Figure 2). Data are simulated, so the actual gene descents are known. One diallelic

locus corresponds to a quantitative trait, for which additive genetic effects and residual effects are also simulated. 33 individuals are assumed observed at 10 equally spaced (10 cM) 4-allele (freqs 0.1, 0.2, 0.3, 0.4) marker loci. The trait locus is mid-way between markers 5 and 6. The same 33 individuals are assumed observed for the trait. To facilitate reasonable results in limited computing time, in the analyses here only 4 markers are used, through the parameter statement

select markers 1 4 6 10 traits 1

In these analyses, three traits are constructed. The genotypic trait, is the standard coded version of the trait locus genotype. A quantitative trait is formed by adding the residual effects (variance 25.0) to the genotypic mean values (90, 100, 110) at the trait locus. Here the additive polygenic effects are ignored. A dichotomous trait with incomplete penetrance is formed by approximate thresholding of the quantitative trait, including the additive effects. Broadly, individuals with values of the full quantitative trait over 100.0 are affected (phenotype 2). Most other observed individuals are phenotype 1. (However, the trait was modified slightly to segregate in each sibship in which the trait alleles segregate.)

There are 7 combinations of program and genotypic/phenotypic trait than can be run, using the command **make xxxx** where xxxx is as specified in the first column of the table below.

Make command	program	parameter files		output file
lm_lods_g.out	lm_lods	ped52.par	ped52_ge.par	lm_lods_geno.out2
lm_mrks_g.out	lm_markers	ped52.par	ped52_ge.par	lm_mrks_geno.out2
lm_bayes_g.out	lm_bayes	ped52.par	ped52_ge.par	lm_bayes_geno.out2
lm_lods_p.out	lm_lods	ped52.par	ped52_ph.par	lm_lods_phen.out2
lm_mrks_p.out	lm_markers	ped52.par	ped52_ph.par	lm_mrks_phen.out2
lm_bayes_p.out	lm_bayes	ped52.par	ped52_ph.par	lm_bayes_phen.out2
lm_quant.out	lm_markers with QUANT=1	ped52.par	ped52_qu.par	lm_quant.out2

The main parameter file is **ped52.par**, The additional small parameter files simply specify the type of the trait, and which item it is in the pedigree data file. For example, for the phenotypic trait we have:

```
input pedigree record trait 1 integer 8
set trait data phenotypic
```

telling us the trait is phenotypic and is the 8 th. integer column in the pedigree file. For a phenotypic trait (now called discrete trait in MORGAN 2.6.1) penetrances are currently specified on the end of the pedigree file.

Here is the main parameter file:

```
input pedigree file 'ped52.ped'

#For default seeds comment out the following lines
input seed file         'seedfil'
```

```
output seed file        'seedfil'

input marker data file 'ped52.markers'
select markers 1 4 6 10 trait 1

map trait 1 all interval proportions 0.3 0.7
map trait 1 external recomb fracts   0.05 0.15 0.3 0.4 0.45

set trait 1  freqs 0.5 0.5

# Monte Carlo setup and requests

use locus-by-locus sampling for setup
sample by scan
set L-sampler probability 0.2
set MC iterations 3000
set burn-in iterations 150
check progress MC iterations 1000
```

Important new statements are the **map trait** statements, which specify the locations at which lod scores are to be estimated. The **external** locations are outside the range of the markers, and the **interval proportions** specify proportions within each marker interval. (These need not be the same for each interval, although here they are.) Note these refer to intervals in the full set of markers in the data set. If we select a subset of markers, the locations are **not changed**. This is so that comparisons between estimates using different marker subsets can be readily made.
In this example, the marker information is in a separate marker data file **ped52.markers**. The file begins

```
map markers dist 10 10 10 10 10 10 10 10 10
set markers 1 2 3 4 5 6 7 8 9 10    freqs 0.4 0.3 0.2 0.1

set marker data 10

302 1 1 1 2 1 3 1 2 2 4 3 4 1 2 1 1 1 2 4 4
303 1 1 1 2 1 1 1 2 2 4 3 4 2 3 2 3 2 4 1 4
....
```

Note that this file is just a collection of parameter statements relating to the markers. They could just be included in the main parameter file, but since the same marker data may be used with many different analyses it is usually more convenient to put them in a separate file.
The pedigree file is **ped52.ped**. Here are the first few, and last few, lines:

```
input pedigree size 52
input pedigree record names 3 integers 9 reals 1
#  simulated pedigree with 52 members
#  col 4 gender; cols 5 observed/unobs status
#  col 6,7 FGL at trait; cols 8,9 genotypes at trait
```

```
#  col 10, genotypic trait  1,3,4, with zeros for unobs
# col 11, pheno: > 100 affected (2);  < 100 unaff (1);
# col 12, a dummy integer, to avoid a now-fixed bug
# col 13: quant trait with no additive variance  (999.5 = unobserved)
#  see ped52.all dat for full version:
*****************************************
101 0      0 1 0 1 2 1 2    0 0 0 999.5
102 0      0 2 0 3 4 1 1    0 0 0 999.5
201 101  102 1 0 2 4 1 2    0 0 0 999.5
202 101  102 2 0 2 3 1 2    0 0 0 999.5
....
....
515 4080 408 1 1 24 11 1 1  1 1 0   91.055
516 4080 408 2 1 24 6 1 2   3 2 0   98.456
****
0.05 0.6 0.95
```

The coded trait genotype is column 10, with code 1,3,4 for genotypes 11, 12, 22, respectively. A dichotomous phenotype is column 11, and the final column of reals is the quantitative trait (with a real with integer part 999 denoting unobserved). As a **temporary** measure, penetrances for a dichotomous trait (Pr(Affected | genotype g), for g =11,12, 22), are placed at the end of the file after any non-empty separating line. For simplicity, we use our standard "****" separator.
Hopefully, we will have time to run several of the 7 lod score runs that have been set up. In case we do not, typical outputs from these runs have been given in the ***.test** files, and are plotted in the following figures.

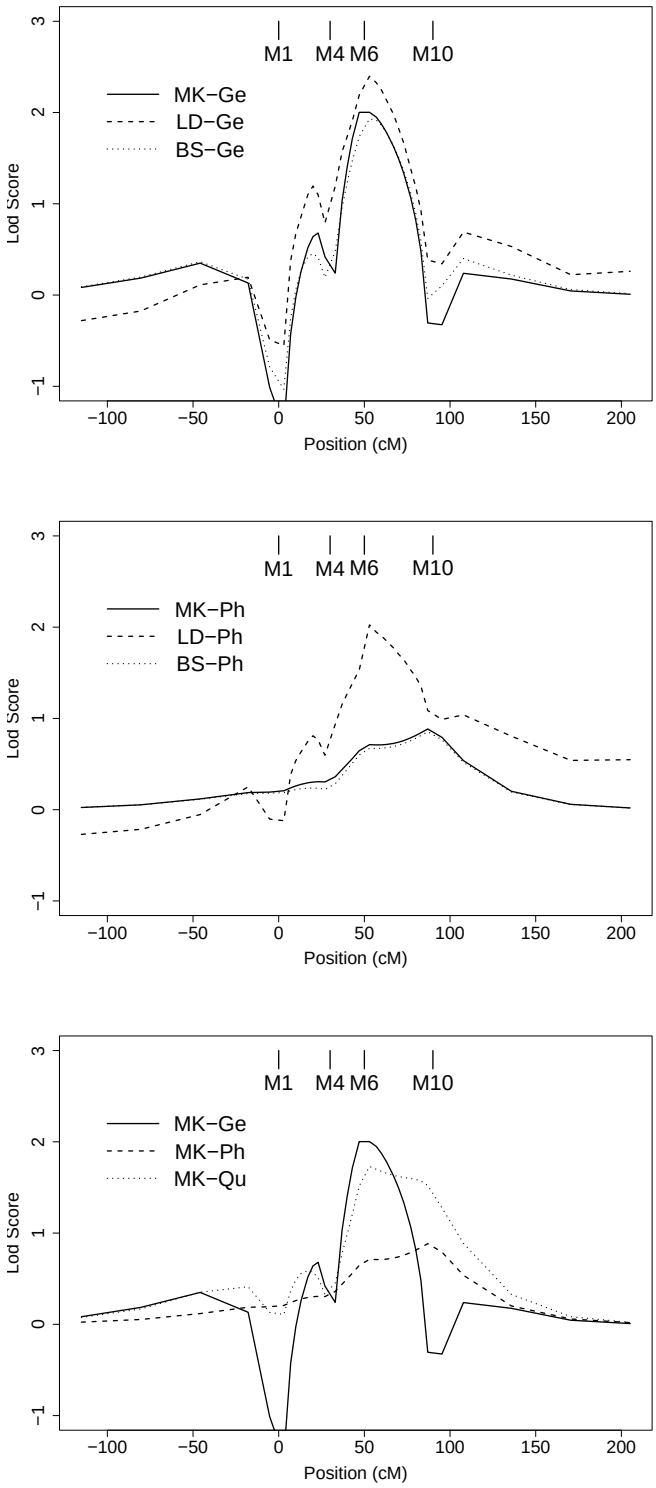


Figure 3: Comparison of MCMC lod-score programs on Ped52 traits. (a) 3 programs on Geno trait; (b) 3 programs on Pheno trait; (c) **lm_markers** program on all 3 traits.