Introduction and Purpose

The purpose of this project was to develop pedigree graphs for use in pedigree- linkage analysis projects. More speficially, this project sought to:

- find a source of pedigree and quantitative trait loci (QTL) data that is suitable (i.e.: large enough) for building arbitrary pedigrees
- use the aforementioned data to build bayesian network representations of arbitrary pedigrees
  - for this project, segregation networks were used (TODO)
- convert arbitrary bayesian network pedigrees into a standardized format (UAI)
- extract meaningful subsets of variables from the aforementioned bayesian network pedigrees for use in queries

This paper assumes general familiary with "segregation networks" (Bayesian network representations of pedigrees). For more information, see TODO.

Generating Pedigree/QTL Data with QMSim

Because of an ostensible scarcity of reliable pedigree/QTL data, I used a pedigree/QTL data simulator (QMSim) to generate all the data necessary for this project.

"QTL and Marker Simulator", or QMSim, is lightweight software that generates pedigree, quantitative trait loci, and genetic marker data for simulated livestock populations. QMSim was developed at the University of Guelph. To read more about QMSim, see Sargolzaei's and Schenkel's paper "QMSim: A Large Scale Genome Simulator for Live Stock."

QMSim is a fairly complex piece of software with myriad uses, input parameters, and output parameters, and so can be somewhat daunting to use. Because of the small scope of this project, however, there is a fairly small subset of usage parameters that are useful for this project. Below, I will describe how to use QMSim in the context of this project, which parameters should be considered in the context of this project, and the effects each parameters has on networks built from data generated by QMSim.

How to Use

Although QMSim comes with a large user manual that contains an exhaustive list of paramter descriptions, many of the use cases and parameters described in the user manual are not useful in the context of this project. The full user manual can be found at: http://www.aps.uoguelph.ca/~msargol/qmsim/QMSim_documentation.pdf.

Setup

Note: this guide assumes a linux based operating system.

There are three version of QMSim, each for one of three major operating systems (Linux, OS X, Windows). QMSim can be downloaded at: http://www.aps.uoguelph.ca/~msargol/qmsim/. After downloading QMSim, unzip the downloaded directory. The precompiled QMSim executable in the unziped directory is now ready for use from the command line.

Command Line Execution

To execute from a command line shell:

```
./QMSim [path/to/prm/file]
```

Parameters and Parameter Files

Parameter Files

QMSim takes as input parameter files which have the file extention ".prm". There are several mandatory parameters that all QMSim parameter files must contain, but to avoid repeating what is already present in the QMSim instruction manual, I'll only discuss the parameters that were of interest to me in generating data for pedigrees. There are several valid `prm` files in the `QMSim/prm` directory that came with this report, most of which were provided as samples with QMSim and modified as necessary. When generating new pedigrees, I recommend using any of these files as templates and modifiying them as necessary.

Parameters

Historical Population Parameters

Historical population parameters affect the simulation of the populations that precede the population for which data is generated by QMSim. In other words, if QMSim generates data for generations 0-10, historical population parameters affect the simulation process for all generations < 0. It's especially important to pay attention to population data when generating small pedigrees: if the historical population isn't sufficiently large, QMSim will throw QTL variance errors.

Historical population parameters are specified between `hp` tags:

```
begin_hp;
    ...
end_hp;
```

Parameter: `hg_size`

The `hg_size` is an optional parameter that affects the number of individuals in each historical generation, as well as the number of historical generations. To use:

```
hg_size = v1 [v2] ... v3 [v4];
```

Where `v1` is the number of individuals in generation `v2` (which should always) start at 0, and `v3` is the number of individuals in generation `v4`. Arbitrary size/generation values can be placed after between the values for the first and last generations.

Although these parameters don't have a direct effect on networks generated from this data, they are nonetheless very important for correctly generating diverse populations with sufficiently diverse QTLs. The QMSim manual states that legal values for the populations size of the historical generations (`v1`) can be as small as 2, and that the number of historical generations (`v4`) can be as small as 1, these values, in my experience, must be significantly higher to produce populations with desirable genetic diversity and with sufficiently balanced gender distributions. Anecdotally, population sizes should be > 10 and the number of generations should be > 20. Of course, these number all correlate, so some experimentation may be required depending on the situation. For all of my experiments, I used 200 generations with population sizes of 500. In general, more and larger historical generations produces more genetically diverse reported populations (depending on a few other paramters; see genome parameters section).

Population Parameters

Population parameters affect the simulation of the populations that QMSim actually reports on.

Population parameters are specified between `pop` tags:

```
begin_pop = "str";
    ...
end_pop;
```

where `str` is a string name for the population.

Parameter: `male/female`

The `male` and `female` founder parameters control various aspects of the male and female founder populations, and are primarly used, along with `ls` and `ng`, to control the size of the generated pedigree. Although there are several ways in which QMSim allows its users to configure the male/female founder populations, the most "interesting" use for these parameters in the context of this project is controlling the male/female ratio in the founding generation, which in turn affects the sizes of subsequent generations, and therefore the number of variables in a segregation network build from generated data.

For example, a founder population of 10 males and 10 females produced a segregation network with 840 variables. A founder population consisting of 2 males and 18 females produced a segregation network with 1480 variables, albeit with more inbreeding. I did find, also, that a larger male/female ration increases the induced width of the corresponding segregation graph. For example, using a $\frac{1}{5}$ male/female ($\frac{5}{25}$) generated a graph with an induced width of ≈ 85 (calculated using an average of 5 different networks), whereas using a ration of $\frac{1}{1}$ ($\frac{15}{15}$) resulted in networks with induced widths of ≈ 110 (also calculated using an average of 5 different networks). To use:

```
begin_founder
    male  [n = int, pop = "str"];
    female[n = int, pop = "str"];
end_founder
```

where `int` is an integer in the range $[1 - 50000]$ and `str` is the string name of the population for which data is being generated. The male/female ration can be at most 1.

Parameter: `ng`

This parameter controls the number of generations in the population. To use:

```
ng = int;
```

where `int` is an int in the range $[0 - 2000]$. `ng`, as well as `ls` and `male/female`, is used to control the size of the generated pedigree.

Based on my observations, `ng` has no measurable effect on network complexity outside of affecting changes in network size. A network created using 20 male founder and 20 female founders, with an `ng` of 10, contained 840 variables , had a pseudotree depth of 80, and an induced width of 54. Doubling the number of generations to 20 almost doubles the number of variables and the pseudotree depth (to 1640 and 144, respectively), but only increased the width to 63. Doubling `ng` once more to 40 about doubled the number of nodes and tree depth once again (to 3240 and 277, respectively), but again caused a modes increase in tree width to from 63 to 72.

Parameter: `ls`

This parameters controls the maximum number of offspring produced by each female. `ls`, along with `ng` and `male/female` are used to control the size of the generated pedigree.

Population Output Parameters

Population output paramters allow users to specify the data that QMSim outputs.

Population output parameters are specified between `popoutput` tags, and are also nested between `pop` tags:

```
begin_pop;
    ...
    begin_popoutput;
        ...
    end_popoutput;
end_pop;
```

Parameter: `data`

Specifying the `data` output parameter causes QMSim to output family/pedigree data for the specified generations. To use:

```
data \gen0 ... \gen_n;
```

where `\gen_n` refers to an arbitrary generation. If no generation is specified, data for all generations will be reported. See the 'Output' section for a more detailed description of the format of files generated by the `data` tag.

Note: the files generated by `data` are necessary to build segregations networks.

Parameter: `allele_freq`

Specifying the `allele_freq` output parameters causes QMSim to output allele frequency statistics. To use:

```
allele_freq \gen0 ... \gen_n
```

where `\gen_n` refers to an arbitrary generation. If no generation is specified, data for all generations will be reported. See the 'Output' section for a more detailed description of the format of files generated by the `allele_freq` tag.

Note: the files generated by `allele_freq` are necessary to build segregations networks.

Parameter: `genotype`

Specifying the `genotype` output parameters causes QMSim to output allele assignments for individuals in the pedigree. To use:

```
genotype \gen0 ... \gen_n
```

where `\gen_n` refers to an arbitrary generation. If no generation is specified, data for all generations will be reported. See the 'Output' section for a more detailed description of the format of files generated by the `genotype` tag.

Note: the files generated by the `genotype` are necessary to build segregations networks.

Genome Parameters

Genome parameters control the way QMSim simulates genotype/QTL/marker data for the generated pedigrees. Genome parameters are specified between `genome` tags. Additionally, all genome parameters relevant to this project are specified between `chr` (chromosome) tags:

```
begin_genome;
    begin_chr = int;
        ...
    end_chr;
end_genome;
```

where `int` is a integer that specifies the number of chromosomes for which to generate data.

QMSim produces both genetic marker and QTL data, but this project uses only QTL data.

Parameter: `nqloci`

This parameter affects the number of quantitative trait loci (QTL) on each chromosome. To use:

```
nqloci = int;
```

where `int` is a integer in the range $[0 - 50,000]$. If `nqloci` is 10, and the number of chromosomes being simulated is 20, then QMSim will generate data for $10 \cdot 20 = 200$ QTL.

`nqloci` affects the complexity of segregation network by allowing users to incorporate more alleles into a network. For examples, creating a pedigree using `male = female = 10`, `ls = 10`, and `ng = 10` produces a network with 840 variables if only one QTL is used to build the network. Adding another QTL to the network increases its size of the network to 1680, and adding another QTL to the network increases its size to 2520. It is easy to see that linearly increasing the number of QTLs incoporated into a network results in a linear increase in that network's size (measured in number of variables). It is not quite so easy to describe the increase in network complexity that occurs as a result of the incorporation of more QTLs: the increase in network complexity caused by adding more QTL is largely dependent on the number of allelic types that occur at that QTL and the underlying distribution of those types. See TODO for more information.

Parameter: `qpos`

This parameter affects the positions of QTLs on their respective chromosomes. This parameter has several potential values. See the QMSim manual for a full list of possible values. To use:

```
qpos = even;
```

Causes QTLs to be spaced evenly along their respective chromosomes.

```
qpos = rnd;
```

Causes QTLs to be spaced randomly along their respective chromosomes.

```
qpos = pd float_1 float_2 ... float_n
```

Causes QTLs to be positioned at predefined locations specified by the float values `float_1` to `float_n`, where all values v are $0 \leq v \leq 100$. If the `pd` option is used, the number of float position values specified must be the same as the value of `nqloci`.

`qpos` affects the complexity of networks in a manner that is difficult to measure: QTL positions are used to build factor tables for segregation graph cliques that consists of multiple segregation nodes. See TODO for more information.

Parameter: `nqa`

This parameter affects the number of possible allele assignments for each QTL in the first generation. More concretely, this parameter affects the cardinalities of founder nodes (and factor tables for single non-segregation node cliques) in segregation network. This parameter has several potential values. See the QMSim manual for a full list of possible values. To use:

```
nqa = all int;
```

Causes all founder QTLs to have `int` possible allelic types.

```
nqa = rnd int_1 int_2 ... int_n;
```

Causes all founder QTLs `int_*` possible allelic types.

```
nqa = pd int_1 int_2 ... int_n;
```

Causes founder QTL at index n to have `int_n` allelic types. If the `pd` option is used, the number of int values specified must be the same as the value of `nqloci`.

`nqloci` affects the complexity of segregation networks in a manner that is difficult to measure, but, intuitively, more allelic types means more complex networks.

Output

QMSim produces various output files, only some of which are necessary for creating pedigrees and segregation networks.

Data Files

Data files are produced by QMSim when the `data` output tag specified (see the Populaton Output Parameter section). These files contain all information necessary for building pedigrees. These files are made up of rows that have the following format:

```
Progeny ID    Sire ID    Dam ID    Sex   Generation ID
```

Progeny, sire, dam , and generations IDs are all integers. Sex is one of the two strings 'M'/'F'.

These files contain more information, but the columns listed above are the only ones necessary for building pedigrees.

Data files are named using the following convention:

```
[population_name]_data_[int].txt
```

Allele Frequency Files

Allele frequency files are produced by QMSim when the `allele_freq` output tag specified (see the Populaton Output Parameter section). These files contain the empircal distributions of allele assignments at each QTL. These files are used to build the factors tables for cliques made up of single founder nodes in segregations networks. Allele frequency files are made up of rows that have the following format:

```
QTL ID    Generation ID    Chromosome ID    Variance    Allele_n:Frequency_n ...
```

If there are n possible allelic types for QTL m, then there will be n `allele:frequency` pairs for in the row describing QTL m.

Allele frequency files are named using the following convention:

```
[population_name]_freq_qtl_[int].txt
```

Note: QMSim also produces marker frequency files, which are name using the following convention:

```
[population_name]_freq_mrk_[int].txt
```

Genotype Files

Genotype files are produced by QMSim when the `genotype` output tag specified (see the Populaton Output Parameter section). These files contain allele assignments for individuals in the data files. These files are used to build evidence files for segregation networks in UAI format. Genotype frequency files are made up of rows that have the following format:

```
Progeny ID    Paternal Genotype 0    Maternal Genotype 0    ... P Genotype n ...
```

If there are n QTLs and m chromosomes being simulated, then the row describing individual i will have $2nm$ entries describing allele assignments.

Genotype files are named using the following convention:

```
[population_name]_qtl_[int].txt
```

QTL Position Files

QTL position files are produced by QMSim by default. These files describe the positions of individual QTLs on their respective chromosomes. These files are used to build factor tables for segregation graph cliques that consists of multiple segregation nodes (see TODO for more information). QTL position files are made up of rows that have the following format:

```
QTL ID    Chromosome ID    Position
```

Position values are floats that are in the range $[0 - 100]$

These files contain more information, but the columns listed above are the only ones necessary for building pedigrees.

Genotype files are named using the following convention:

```
lm_qtl_[int].txt
```

Examples

TODO

Creating Bayesian Network Representations of Pedigrees

In the context of this project, the purpose of generating pedigree and QTL data is to create graphical representation of pedigrees and to convert those graphs into standardized representations that can be used as input to various graph algorithms.

Below, I will describe how the data described above is used to create graphical representations of pedigrees, how to use the Python code (provided with this report in the `uai` directory) that converts raw data into pedigree/segregation networks, how to use the Python code that converts segregation graphs into UAI files, and how to use the Python code that extracts query variables from segregation networks.

Pedigrees

Note: this report assumes a basic familiary with pedigrees in the context of pedigree-linkage analysis. For more information, see TODO.

The first step in creating a segregation graph is creating a representation of a pedigree that is more useful than the row/column representation created by QMSim.

`pedigree.py`

Segregation Networks

`qtl.py`

Converting Segregation Networks to UAI

`uai.py`

Extracting Query Variables

`query.py`