

SNPs and Haplotypes - Lab session



Gabor T. Marth

Department of Biology, Boston College
marth@bc.edu

1. SNP discovery

Given the accurate sequence of a genomic clone (e.g. BAC) containing a chromosomal location of interest (e.g. a gene), and a collection of expressed sequence tag (EST) reads

- Create an accurate base-wise multiple alignment of these sequences.
- Identify and remove from further consideration ESTs that are likely to originate from a similar (duplicated) but disparate genomic location
- Analyze each redundantly represented nucleotide position for the presence of single-base variations.

Data preparation

Go to your home directory:

```
> cd
```

Copy the data set into your home directory:

```
> cp ~/marth_data/Discovery.tar.gz .
```

Uncompress the data

```
> tar zxvf Discovery.tar.gz
```

You should end up with a subdirectory named “Discovery” as a result.

Data preparation

Go to the “edit_dir” subdirectory:

```
> cd Discovery/edit_dir
```

There are several files there already, list them:

```
> ls -l
```

Take a look at these files, e.g.:

```
> less CLUSTER.anchor.fasta
```

```
> less CLUSTER.members.fasta
```

```
> less CLUSTER.members.fasta.qual
```

Anchored multiple alignment

We will align the EST cluster member sequences with an anchored technique implemented by POLYBAYES. The genomic clone sequence is used as a ‘template’ to guide the multiple alignment. First, look at the “wrapper” script provided that runs POLYBAYES with the appropriate command lines options:

```
> more runPolyBayesAlignment
```

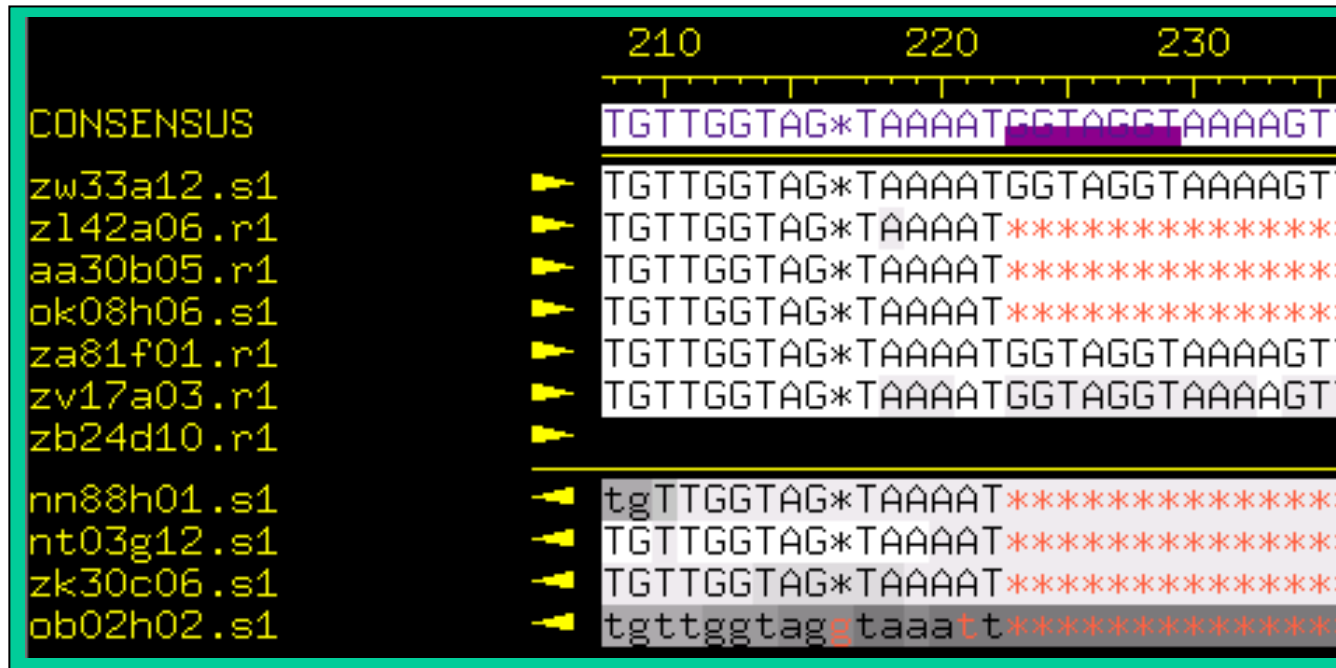
Produce the anchored alignment tool by typing:

```
> runPolyBayesAlignment
```

The anchored multiple alignment

Examine the resulting multiple alignment with CONSED:

```
> consed& (then click CLUSTER.polybayes.alignment.ace)
```



Observe the alternatively spliced EST forms in this alignment!

Identifying sequence paralogs

Use the POLYBAYES program to screen the multiple alignment for candidate SNP sites. This time, use the output ‘ace’ file of the previous step (command line is in the ‘runPolyBayesSNP’ script) and view the resulting marked-up alignment in consed:

```
> runPolyBayesNoFilter  
> consed& (then click CLUSTER.polybayes.paralog.ace)
```

Use the ‘Navigate’ utility in consed, and select ‘Navigate by tag type: polymorphism’ to find all candidate SNPs marked up by the SNP detection program. Examine all these locations (see next page).

Sequence paralogos



Are all these discrepancies true polymorphisms or could the EST 'nt03g12.s1' represent a similar region elsewhere in the genome?

Identifying sequence paralogs

Run POLYBAYES with the in-built paralog filtering algorithm turned on (wrapper script: runPolyBayesSNP) and view the result:

```
> runPolyBayesParalogFilter  
> consed& (then click CLUSTER.polybayes.paralogfilter.ace)
```

```
          940          950          960          970  
CONSENSUS TAAAACAATTACAAAATGAGTGTTGTTTGTAAAACAGTA  
z142a06.r1 taaaacgattacgaaaTGAGTgttgtttgta*ggcagtt  
aa30b05.r1 TAAAACAATTACAAAATGAGTGTTGTTTGTAAAACAGTA  
ok08h06.s1 GGGCaaagttcCCTATAGTGAGTCGTATTAAATTCGTAA  
nn88h01.s1 TAAAACAATTACAAAATGAGTGTTGTTTGTAAAACAGTA  
nt03g12.s1 TAA***AATTACAAA**AA*TGCTGATTGTAAAGG*GTA  
zk30c06.s1 TAAAACAATTACAAAATGAGTGTTGTTTGTAAAACAGTA  
ob02h02.s1 TAAAACAattacAAAATgagTgTTGTttgtAAAACAGTA  
nz30c09.s1 agcttattcccttTAGTGAGggTTAattatagctatggc  
zp07h12.r1 gctgtgcCGAATTCCTGCAGCCCCGgggatccAcTAGTTC  
ai79b06.s1 TAAAACAATTACAAAATGAGTGTTGTTTGTAAAACAGTA  
zv17a02.s1 TAAAACAATTACAAAATGAGTGTTGTTTGTAAAACAGTA
```

SNP detection

Finally, screen the alignment for candidate SNPs together with the paralog filtering algorithm (wrapper script: runPolyBayesSNP):

```
> runPolyBayesSNP  
> consed& (then click CLUSTER.polybayes.snp.ace)
```

Again, locate candidate SNPs with the ‘Navigate’ utility in consed. You should only find one, interestingly, a variation present among ESTs that represent an alternatively spliced transcript (see next page).

Candidate SNP site

	530	540	550	560
CONSENSUS	CAGTGTCTTACCCTGAAGAGAAAGTTGTAGGTTGGCTG			
zw33a12.s1	cagtgtct	*accctgaagagaa	*ggttgta	agttggc*g
z142a06.r1	*****	*****	*****	*****
aa30b05.r1	*****	*****	*****	*****
ok08h06.s1	*****	*****	*****	*****
za81f01.r1	tggacttccagctcaggggtctcccccgga			
zv17a03.r1	CAGTGTCTTACCCTGAAGAGAAAGTTGTAGGTTGGCTG			
zb24d10.r1	CAGTGTCTTACCCTGAAGAGAAAGTTGTAGGTTGGCTG			
nn88h01.s1	*****	*****	*****	*****
nt03g12.s1	*****	*****	*****	*****
zk30c06.s1	*****	*****	*****	*****
ob02h02.s1	*****	*****	*****	*****
nz30c09.s1	cAGTGTCTTACCCTGAAGAGAAAGTTGTAGGTTGGCTG			
zp07h12.r1	cagtg	*ctacccccgaagaa	aaaagttg	aaggttggctg

Mark-up tag with P(SNP)
values assigned by POLYBAYES

Comment:

P(1)=0.00011262338841526
P(2)=0.999869233918491
P(3)=1.81426348193338e-05
P(4)=5.82754169381083e-11
VAR=ag
P(ag)=0.999869207076724

2. Modeling ancestral processes - the Coalescent

In this exercise, we will run the Coalescent process to analyze the genealogy and mutation structure of DNA samples. We will do this under different models: low or high mutation rates, low or high effective population size, with or without recombination, under a stationary demographic history or under scenarios of changing effective size such as a genetic bottleneck. We will verify our simulation results with theoretical predictions.

Data preparation

Go to your home directory:

```
> cd
```

Make a new subdirectory:

```
> mkdir Coalescent
```

Go to this subdirectory:

```
> cd Coalescent
```

Running and viewing the Coalescent

Run the Coalescent simulation with a simple set of parameters:

```
> coalesce.pl --debug --I 1 --L 1000 --E 1 --N 10000 --r  
1E-8 --n 5 --writeGraph > ! graph.dot
```

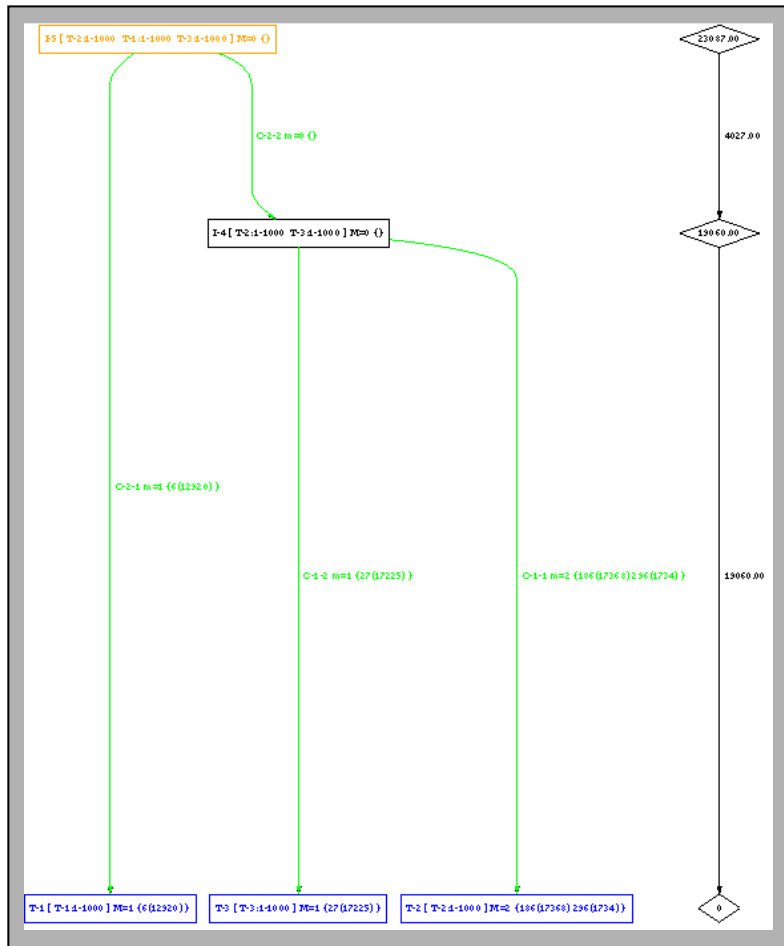
Use the **dot** drawing program to render the graphical details:

```
> dot -Tps graph.dot -o graph.ps
```

View the PostScript format graphical file with the **gv** program:

```
> gv&
```

The Coalescent genealogy



This is an example of a Coalescent genealogy. Remember, the Coalescent is a random process, your specific genealogy will look different from this picture!

The effect of the mutation rate

Run the Coalescent with low mutation rate (and then graph):

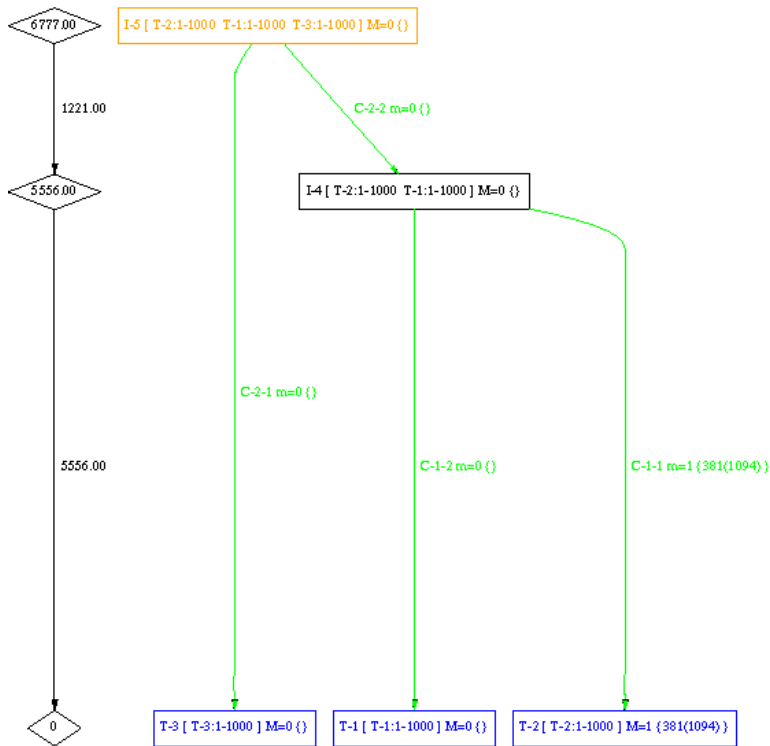
```
> coalesce.pl --debug --I 1 --L 1000 --E 1 --N 10000 --u  
2.0E-8 --r 0 --n 3 --writeGraph > ! graphM1.dot  
  
> dot -Tps graphM1.dot -o graphM1.ps
```

Then run it with high mutation rate:

```
> coalesce.pl --debug --I 1 --L 1000 --E 1 --N 10000 --u  
8.0E-8 --r 0 --n 3 --writeGraph > ! graphM2.dot  
  
> dot -Tps graphM2.dot -o graphM2.ps
```

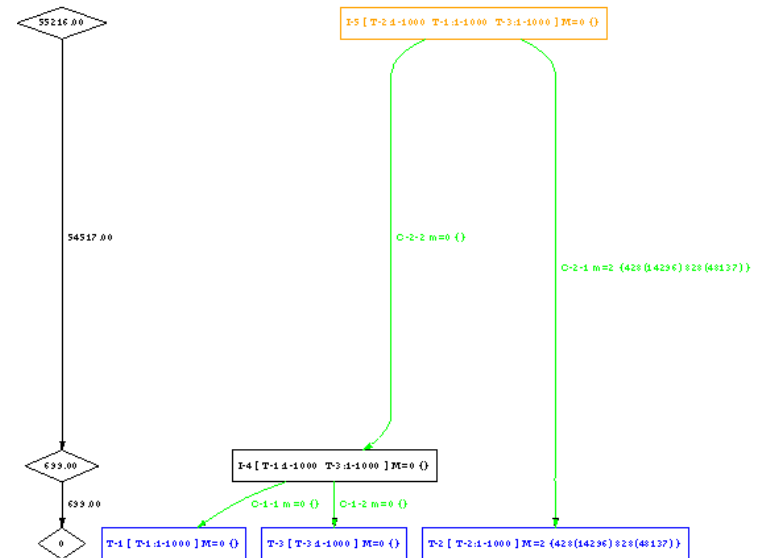
You will see something like these:

The effect of the mutation rate



low mutation rate

high mutation rate



Effective population size

Run the Coalescent with small effective population size:

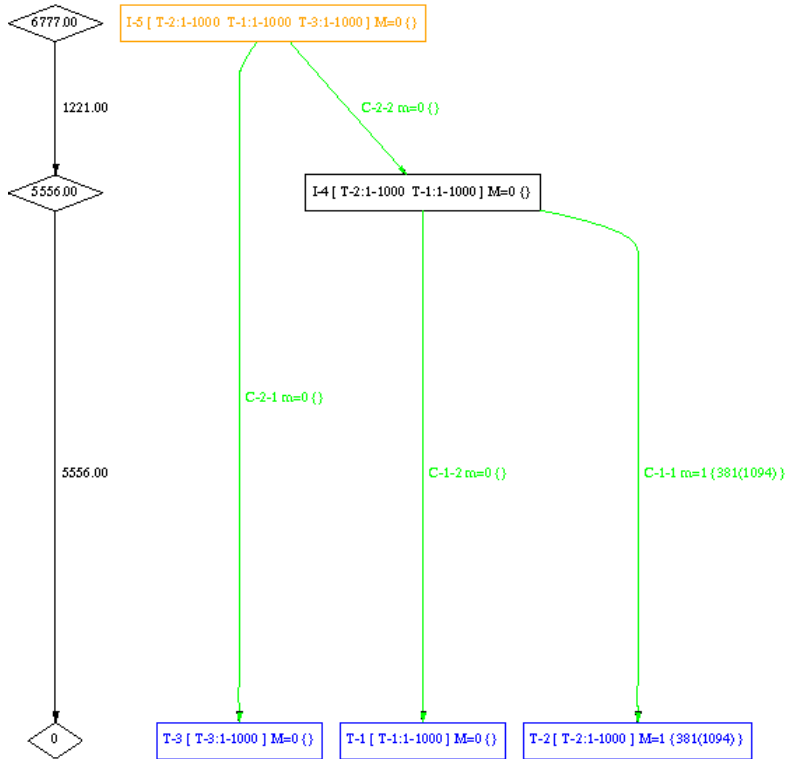
```
> coalesce.pl --debug --I 1 --L 1000 --E 1 --N 5000 --u  
2.0E-8 --r 0 --n 3 --writeGraph > ! graphS1.dot  
  
> dot -Tps graphS1.dot -o graphS1.ps
```

Then run it with large effective size:

```
> coalesce.pl --debug --I 1 --L 1000 --E 1 --N 50000 --u  
2.0E-8 --r 0 --n 3 --writeGraph > ! graphS2.dot  
  
> dot -Tps graphS2.dot -o graphS2.ps
```

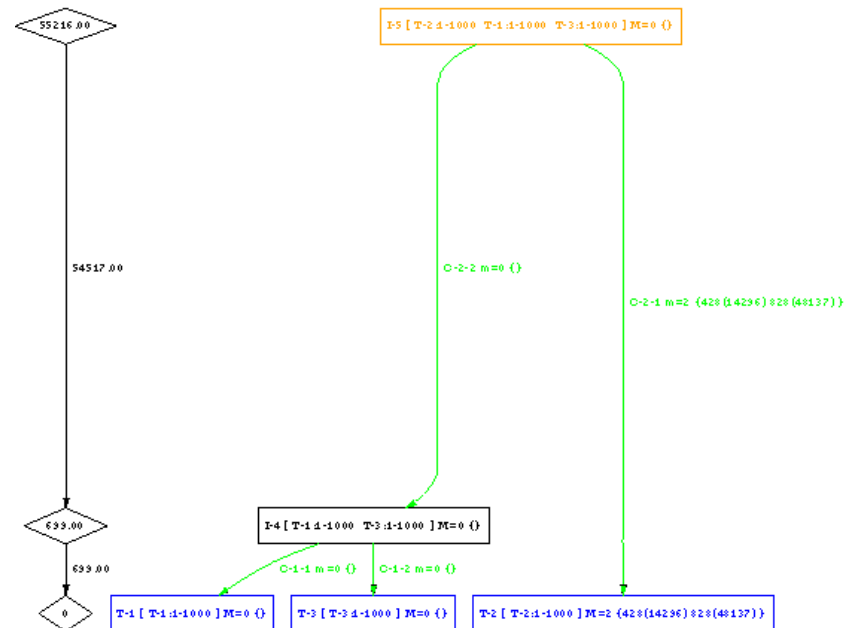
Compare the graphs:

Effective population size



small effective size

large effective size



Demographic history – changing effective size

Expansion:

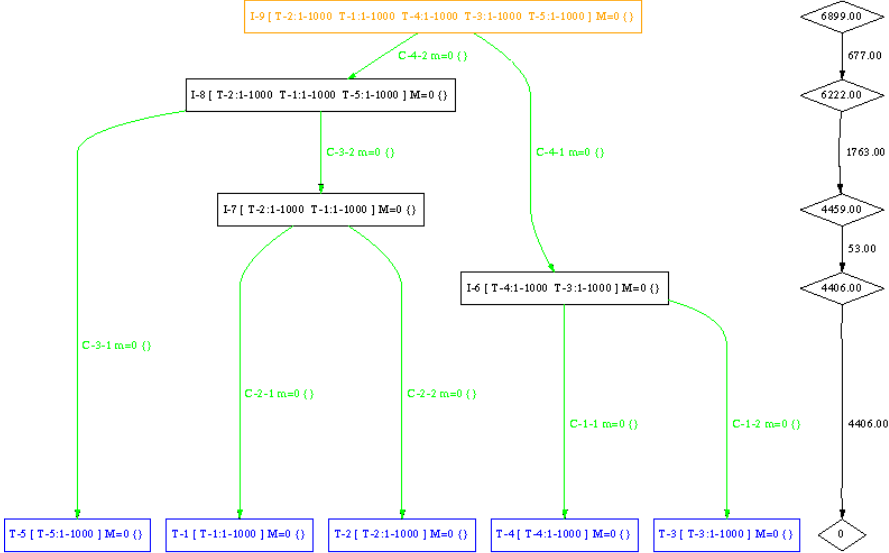
```
> coalesce.pl --debug --I 1 --L 1000 --E 2 --N 10000 --T  
4000 --N 500 --u 2.0E-8 --r 0 --n 5 --writeGraph > !  
graphC1.dot  
  
> dot -Tps graphC1.dot -o graphC1.ps
```

Collapse:

```
> coalesce.pl --debug --I 1 --L 1000 --E 2 --N 500 --T 2000  
--N 10000 --u 2.0E-8 --r 0 --n 5 --writeGraph > !  
graphC2.dot  
  
> dot -Tps graphC2.dot -o graphC2.ps
```

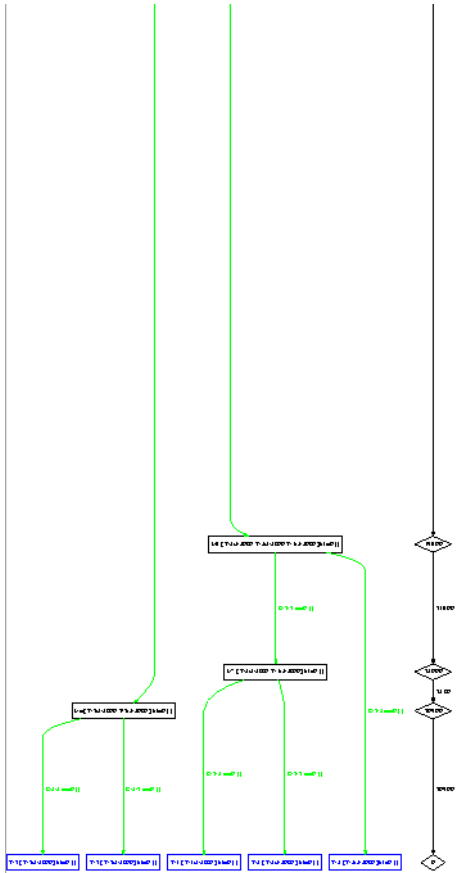
Compare the graphs:

Demographic history



expansion

collapse



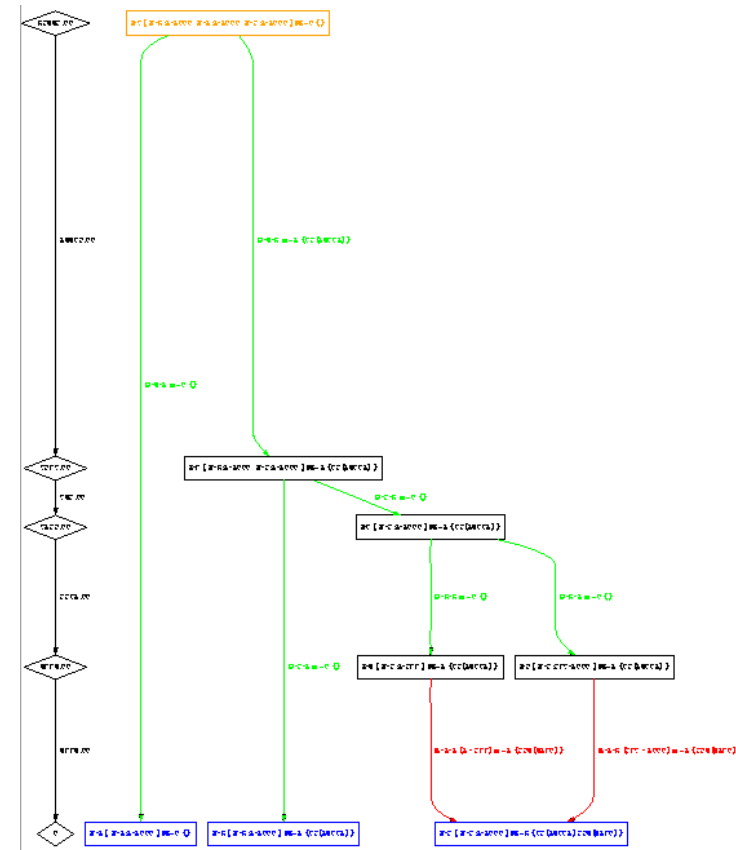
Recombination

Run the Coalescent with a non-zero recombination rate:

```
> coalesce.pl --debug --L 1000
--E 1 --N 10000 --u 2.0E-8 --r
1.0E-8 --n 3 --writeGraph > !
graph.dot

> dot -Tps graphR.dot -o
graphR.ps
```

Note the fact that the genealogy is no longer a tree!



Genome-wide SNP distributions

Coalescent simulations can be used to predict the curve shape of SNP distributions in the genome under different model conditions and model parameters. Marker density:

```
> coalesce.pl --r 0 --n 2 --E 1 --N 10000 --L 10000 --M 25  
--I 5000 --siteMrca --infiniteSites --debug --  
writeCensoredDensity --censoredDensityOut density-sim.txt
```

For this situation we can calculate the distribution directly from theoretical results, implemented in computer code:

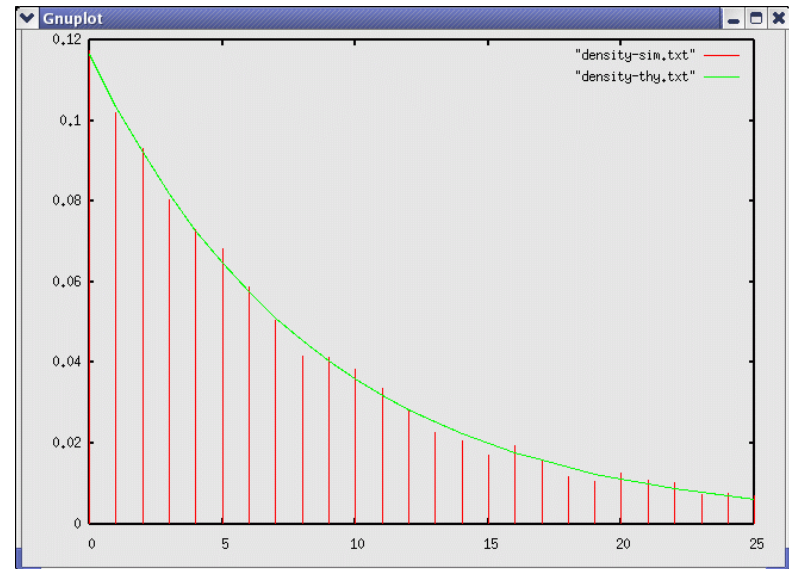
```
> theory.pl --E 1 --N1 10000 --u 2.0E-8 --L 10000 --N 25 --  
M 25 > ! density-thy.txt
```

Simulating marker density

Plot the distributions with **gnuplot**:

```
> gnuplot  
gnuplot> plot "density-sim.txt" with impulses  
gnuplot> replot "density-thy.txt" with lines
```

This is what you will see:



Simulating the allele frequency spectrum

Use simulations to approximate the AFS:

```
> coalesce.pl --r 0 --E 1 --N 10000 --n 21 --L 4000 --M  
4000 --I 1000 --siteMrca --infiniteSites --debug --  
writeFreqSpectrum --freqSpectrumOut afs-sim.txt
```

The allele frequency spectrum can be calculated using theoretical formulae:

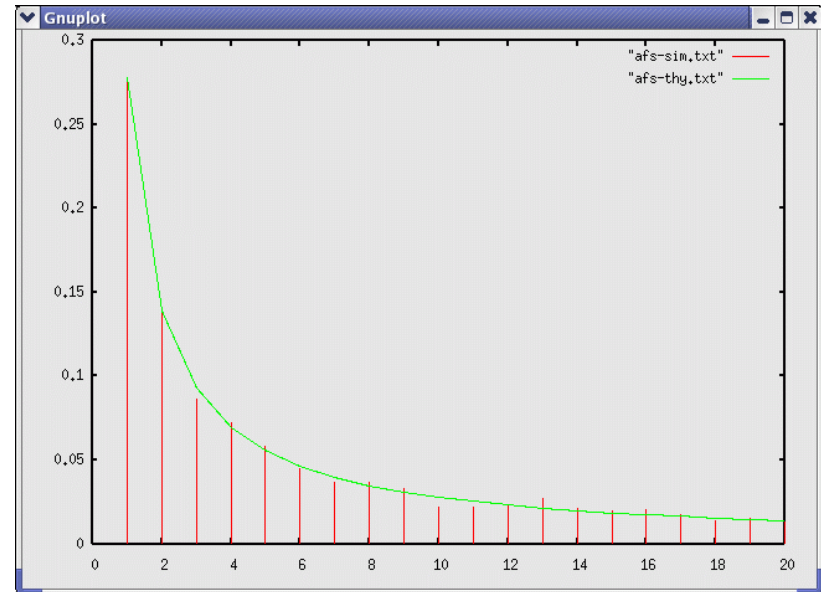
```
> spectrumModel.pl --E 1 --N 10000 --n 21 >! afs-thy.txt
```

Simulating the allele frequency spectrum

Plot the distributions with **gnuplot**:

```
> gnuplot  
gnuplot> plot "afs-sim.txt" with impulses  
gnuplot> replot "afs-thy.txt" with lines
```

Here is the theoretical curve and the approximation by multi-replicate simulations:



3. Model fitting using the AFS

In this exercise we use experimentally collected allele frequency data to infer model parameters for the demographic history of human populations, as discussed in the lecture.

Data preparation

Go to your home directory and make a new subdirectory:

```
> cd  
> mkdir AFS  
> cd AFS
```

Copy the data and README files here:

```
> cp ~/marth_data/alleleCountsShort.txt .  
> cp ~/marth_data/README-ALLELECOUNTFILE .
```

Multi-population allele count data

Here is what the allele count file looks like (explanation of the individual fields are in the README file):

```
TSC0000026 G A 1 1 244 69 175 G 69 1 1 80 12 68 G
12 1 1 82 33 49 G 33 1 1 82 24 58 G 24
TSC0000143 G T 0 0 80 12 68 G 12 1 1 80 12 68 G
12 O 0 0 0 - O 0 0 0 0 - O 0
TSC0000218 G A 1 1 246 214 32 A 32 1 1 84 76 8 A
8 1 1 80 75 5 A 5 1 1 82 63 19 A 19
TSC0000314 G A 1 1 238 45 193 G 45 1 1 82 34 48 G
34 1 1 82 7 75 G 7 1 1 74 4 70 G 4
TSC0000323 G A 1 1 216 72 144 G 72 1 1 74 7 67 G
7 1 1 74 16 58 G 16 1 1 68 49 19 A 19
TSC0000343 G A 0 0 84 80 4 A 4 1 1 84 80 4 A
4 O 0 0 0 - O 0 0 0 0 - O 0
TSC0000364 G A 0 0 82 52 30 A 30 1 1 82 52 30 A
30 O 0 0 0 0 0 - O 0 0 0 0 - O 0
TSC0000383 G T 1 0 228 9 219 G 9 1 1 74 6 68 G
6 1 1 78 3 75 G 3 1 0 76 0 76 G 0
TSC0000416 G T 0 0 166 89 77 T 77 1 1 84 58 26 T
26 1 1 82 31 51 G 31 0 0 0 0 - O 0
TSC0000445 G T 1 1 240 209 31 T 31 1 1 84 79 5 T
5 1 1 76 75 1 T 1 1 1 80 55 25 T 25
TSC0000454 C T 0 0 164 18 146 C 18 1 1 82 13 69 C
13 1 1 82 5 77 C 5 0 0 0 0 - O 0
TSC0000481 G A 1 1 210 36 174 G 36 1 1 80 18 62 G
18 1 1 78 17 61 G 17 1 1 52 1 51 G 1
TSC0000525 G T 1 1 248 128 120 T 120 1 1 84 42 42 G
42 1 1 82 42 40 T 40 1 1 82 44 38 T 38
TSC0000526 G T 0 0 84 76 8 T 8 1 1 84 76 8 T
8 O 0 0 0 0 - O 0 0 0 0 - O 0
TSC0000584 G T 1 0 238 32 206 G 32 1 1 82 2 80 G
2 1 1 74 30 44 G 30 1 0 82 0 82 G 0
TSC0000588 G A 0 0 158 140 18 A 18 1 1 76 73 3 A
3 1 1 82 67 15 A 15 0 0 0 0 - O 0
TSC0000598 G A 1 1 248 101 147 G 101 1 1 84 44 40 A
40 1 1 82 12 70 G 12 1 1 82 45 37 A 37
TSC0000625 C T 1 1 238 162 76 T 76 1 1 82 53 29 T
29 1 1 82 65 17 T 17 1 1 74 44 30 T 30
TSC0000689 G T 0 0 80 28 52 G 28 1 1 80 28 52 G
```

Processing observed AF data

Parse and process the allele counts for the European samples with the following (longish) command line, piping several programs together:

```
> cat alleleCountsShort.txt | spectrumProcessAlleleCounts.pl  
--pop 1 --debug --allSuccess | spectrumReduce.pl --multiN --m  
41 --folded | curveNormalize.pl >  
data-eu-multiN-afs-reduced-m41-norm.txt
```

Copy the data and README files here:

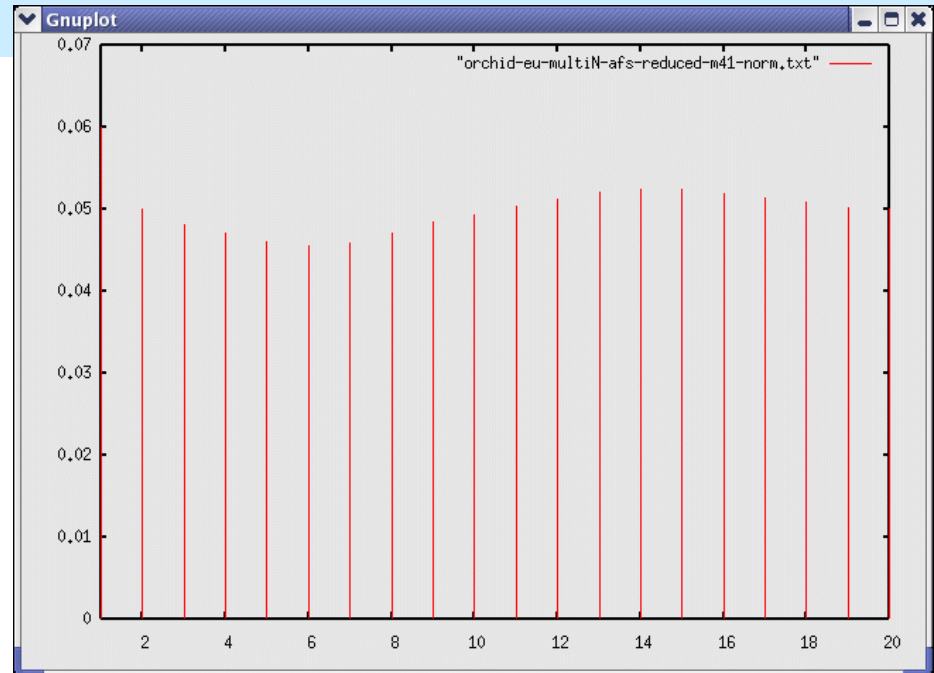
```
> cp ~/marth_data/alleleCountsShort.txt .  
> cp ~/marth_data/README-ALLELECOUNTFILE .
```

Observed AF data

Plot the distribution with **gnuplot**:

```
> gnuplot
```

```
gnuplot> plot [1:20][0:1200]"data-eu-multiN-afs-reduced-  
m41.txt" with impulses
```



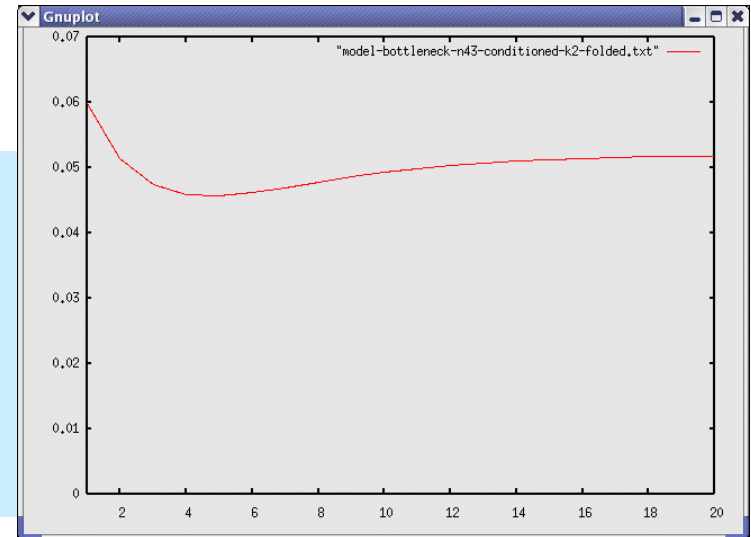
Generating model-predicted AFS

Generate the corresponding model-predicted spectrum:

```
> spectrumModel.pl --E 3 --N 20000 --T 3000 --N 2000 --T 500  
--N 10000 --n 43 | spectrumFold.pl --n 43 |  
spectrumCondition.pl --n 41 --k 2 --folded >! model-  
bottleneck-n43-conditioned-k2-folded.txt
```

Plot:

```
> gnuplot  
gnuplot> plot [1:20][0:0.07]"model-  
bottleneck-n43-conditioned-k2-  
folded.txt" with lines
```



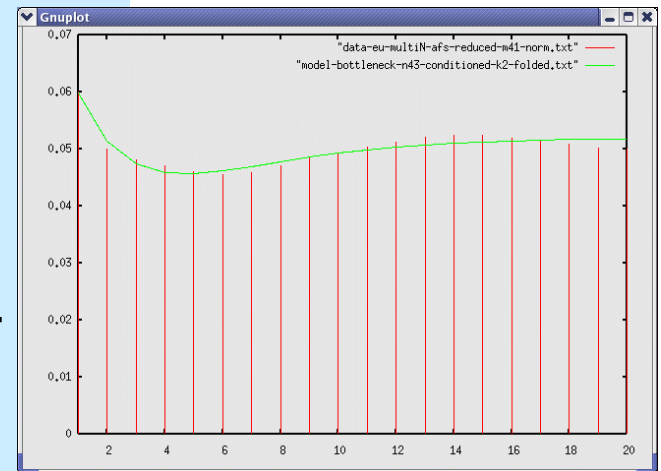
Quantifying model fit

Compare the curves visually:

```
> gnuplot
```

```
gnuplot> plot [1:20][0:0.07]"data-eu-multiN-afs-reduced-m41-norm.txt" with impulses
```

```
gnuplot> replot "model-bottleneck-n43-conditioned-k2-folded.txt" with lines
```



Quantify fit:

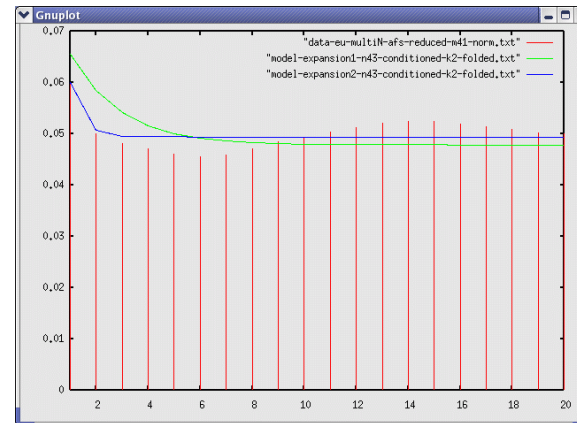
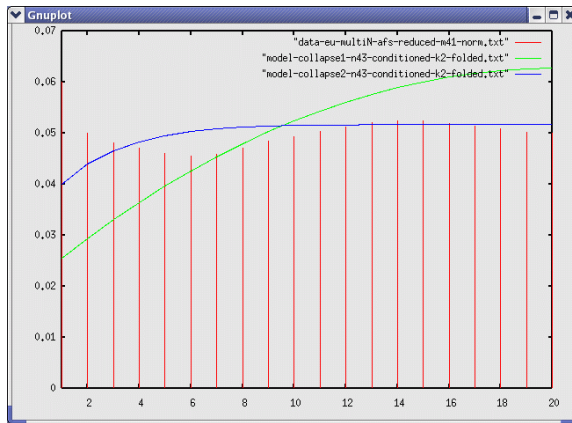
```
> curveFit.pl --dataFile data-eu-multiN-afs-reduced-m41.txt --modelFile model-bottleneck-n43-conditioned-k2-folded.txt --m 1 --M 20
```

Explore parameter space

Try other demographic histories (expansion, collapse):

```
> spectrumModel.pl --E 2 --N 5000 --T 1000 --N 10000 --n 43  
| spectrumFold.pl --n 43 | spectrumCondition.pl --n 41 --k  
2 --folded >! model-collapse-n43-conditioned-k2-folded.txt
```

```
> spectrumModel.pl --E 2 --N 140000 --T 2000 --N 10000 --n  
43 | spectrumFold.pl --n 43 | spectrumCondition.pl --n 41 -  
-k 2 --folded >! model-expansion2-n43-conditioned-k2-  
folded.txt
```



Observed data from another population

Parse and process the allele counts for the African samples:

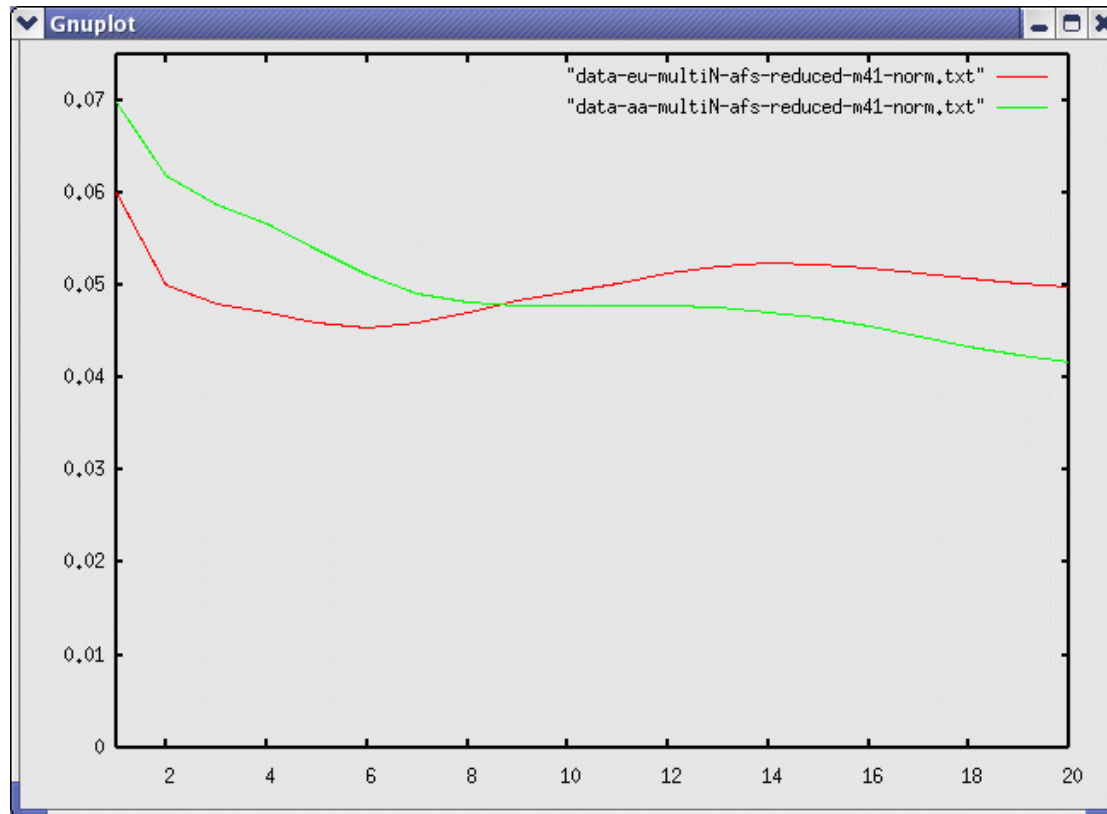
```
> cat alleleCountsShort.txt | spectrumProcessAlleleCounts.pl
--pop 2 --debug --allSuccess | spectrumReduce.pl --multiN --m
41 --folded | curveNormalize.pl >!
data-aa-multiN-afs-reduced-m41-norm.txt
```

Compare with **gnuplot**:

```
> gnuplot
gnuplot> plot [1:20][0:0.075]"data-eu-multiN-afs-reduced-
m41-norm.txt" with lines
gnuplot> replot "data-aa-multiN-afs-reduced-m41-norm.txt"
with lines
```

Observed data from another population

They do look different, don't they?



Refitting for another population

Generate model-predicted spectrum:

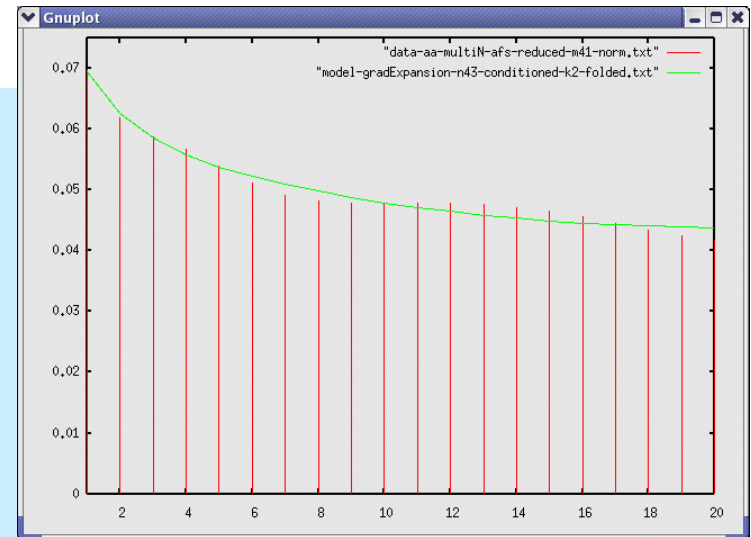
```
> spectrumModel.pl --E 3 --N 26000 --T 2400 --N 16000 --T  
15000 --N 10000 --u 2E-8 --n 43 | spectrumFold.pl --n 43 |  
spectrumCondition.pl --n 41 --k 2 --folded > ! model-  
gradExpansion-n43-conditioned-k2-folded.txt
```

Compare model and data:

```
> gnuplot
```

```
gnuplot> plot [1:20][0:0.075]"data-  
aa-multiN-afs-reduced-m41-norm.txt"  
with impulses
```

```
gnuplot> replot "model-  
gradExpansion-n43-conditioned-k2-  
folded.txt" with lines
```



4. Haplotype block analysis

In this exercise we will analyze haplotype blocks, regions of reduced haplotype diversity in the polymorphism structure of population samples, as discussed in the lecture.

Data preparation

Go to your home directory and make a new subdirectory:

```
> cd  
> mkdir Haplotype  
> cd Haplotype
```

Copy the data in the present directory:

```
> cp ~/marth_data/haps.txt .  
> cp ~/marth_data/hapsA.txt .  
> cp ~/marth_data/hapsB.txt .
```

Generating haplotypes with simulation

Use Coalescent simulations to generate a set of haplotypes:

```
> subdivision.pl --EA 2 --NA 2000 --TA 2000 --NA 10000 --EB 2
--NB 20000 --TB 3000 --NB 10000 --EM 2 --mAB 0 --mBA 0 --TM
2000 --mAB 0.001 --mBA 0.001 --nA 23 --nB 23 --L 1000 --M
10000 --I 2 --siteMrca 1 --debug --writeSnpDescriptor --debug
>! snps.txt
```

Look at the file:

```
> less snps.txt
```

```
1 1 18 5000 23 000000000000000000000000 5000 23 11010100010010001111100 0
2 1 817 5000 23 000000000000000000000000 5000 23 11010100010010001111100 0
3 1 1132 5000 23 000000000000000000000000 5000 23 000000000000000010000000 0
4 1 1175 5000 23 000000000000000000000000 5000 23 000010000000000010000000 0
5 1 2902 5000 23 000000000000000000000000 5000 23 000010000000000010000000 0
6 1 3428 5000 23 000000000000000000000000 5000 23 000001000000000010000000 0
7 1 4083 5000 23 000000000000000000000000 5000 23 110101101111111101010111 0
8 1 4388 5000 23 000000000000000000000000 5000 23 000010000000000010000000 0
```

Haplotypes

Parse out haplotypes:

```
> cat snps.txt | snp2Hap.pl >! haplotypes.txt
```

```
> cat snps.txt | snp2Hap.pl --minFreqA 0.1 --minFreqB 0.1 >!  
haplotypes10.txt
```

```
> less haplotypes.txt
```

```
> less haplotypes10.txt
```

```
A-16 00000000  
A-17 00000000  
A-18 00000000  
A-19 00000000  
A-20 00000000  
A-21 00000000  
A-22 00000000  
A-23 00000000  
B-1 11000010  
B-2 11000010  
B-3 00000000  
B-4 11000010  
B-5 00011001  
B-6 11000110  
B-7 00000010  
B-8 00000000  
B-9 00000010  
B-10 11000010  
B-11 00000010
```

Haplotype block and htSNP extraction

Use dynamic programming to extract haplotype blocks:

```
> cat haplotypes10.txt | extractHapBlocks.pl --fMin 0.2 --fTot 0.8 --debug >! hapBlocks.txt
```

```
block1: left-1 right-2 length-2 htSnps-2 haplotypes-[00 (145), 11 (42), 10 (13)]
block2: left-3 right-4 length-2 htSnps-2 haplotypes-[10 (111), 00 (45), 01 (44)]
block3: left-5 right-7 length-3 htSnps-3 haplotypes-[011 (83), 000 (78), 100 (30), 001 (9)]
block4: left-8 right-9 length-2 htSnps-2 haplotypes-[00 (126), 01 (41), 11 (33)]
block5: left-10 right-11 length-2 htSnps-2 haplotypes-[00 (91), 11 (83), 10 (23), 01 (3)]
block6: left-12 right-15 length-4 htSnps-2 haplotypes-[0000 (162), 1111 (34), 0111 (4)]
block7: left-16 right-25 length-10 htSnps-2 haplotypes-[0010011011 (107), 1101100100 (66), 1101100000 (27)]
block8: left-26 right-30 length-5 htSnps-3 haplotypes-[00001 (86), 11010 (53), 00101 (50), 10101 (7), 00000 (3), 11011 (1)]
block9: left-31 right-31 length-1 htSnps-1 haplotypes-[0 (124), 1 (76)]
block10: left-32 right-35 length-4 htSnps-3 haplotypes-[0000 (101), 0111 (44), 1110 (40), 0100 (15)]
block11: left-36 right-36 length-1 htSnps-1 haplotypes-[1 (106), 0 (94)]
blocks-11 htSnps-23
extractHapBlocks.pl completed
```

Block definition and block structure

Use the pre-canned data to extract haplotype blocks:

```
> cat haps.txt | extractHapBlocks.pl --fMin 0.2 --fTot 0.8 --  
debug >! blocks20-80.txt
```

Rerun with more or less stringent frequency requirements:

```
> cat haps.txt | extractHapBlocks.pl --fMin 0.2 --fTot 0.9 --  
debug >! blocks20-90.txt
```

```
> cat haps.txt | extractHapBlocks.pl --fMin 0.15 --fTot 0.8 -  
-debug >! blocks15-80.txt
```

Compare:

```
> less blocks20-80.txt
```

```
> less blocks20-90.txt
```

```
> less blocks15-80.txt
```

Populations and block structure

Use the population specific data to extract haplotype blocks:

```
> cat hapsA.txt | extractHapBlocks.pl --fMin 0.2 --fTot 0.8  
>! blocksA.txt  
  
> cat hapsB.txt | extractHapBlocks.pl --fMin 0.2 --fTot 0.8  
>! blocksB.txt
```

Compare:

```
> less blocksA.txt  
> less blocksB.txt
```

See any differences?