**CELSIM(1)** **CELSIM(1)**

**NAME**

celsim – Whole genome shotgun data set simulator

**SYNOPSIS**

celsim [-s seed] [-d] [-F] [-u] [-n] [-b <batch size>] [-p] specfile

**DESCRIPTION**

*Celsim* reads from *specfile* the specification of a DNA sequence, a number of optional polymorphism specifications, and one or more sample specifications. The input specfile must have extension ".sim" or ".spec". If no extension is specified, celsim will look for <specfile>.sim. The net effect of *celsim* is to first build a base DNA sequence from a *frag* DNA specification, to then build several permutations of the base sequence according to a *poly* specification, and finally to sample the permuted sequences according to one or more *frag* sampling specifications. The resulting samplings are all written to a file whose name has the form *specfile.frg*. Normally comments detailing the construction are written to a file called *specfile.cms*. With the *–F* option set this comment file is not produced.

The *–s* option permits one to specify a positive integer with which to seed the random number generator so that one can consistently generates the same data set if desired. If celsim is invoked with *–s 0* the current time is used to seed the random number generator, resulting in a distinct data set with each invocation. Otherwise the default seed is used.

If the –b <batch size> option is specified, then the output is in a sequence of files whose names are in the form specfile_xxxx.suffix. Currently xxxx allows for 9998 batches and the file suffix can be .urc or .frg.

If the –u and –n options specify the probability distribution for the insert and fragment read lengths, as well as the fragment clear ranges. The –u option specifies a uniform distribution, and the –n option (default) specifies a normal distribution. See the frag document for more details.

If the –d option is specified, the generated dna is output to specfile.dna.

If the –p option is specified celsim will prompt before clobbering files.

Celsim requires that you have the AS_BIN environment variable to point to an AS installation directory, and that perl5 is in your PATH.

### 1. Specification File Syntax and Semantics:

A *celsim* input file consists of several sections, each of which is passed to a separate program to achieve the effect of the section. The input must consist of a DNA section, followed by zero or more polymorphism sections, followed by one or more sampling sections. Formally:

<Spec> ← <Seed section> <DNA section> <Polymorphism section> ∗ <Sampling section> + <BAC End section> + <UBAC section> + <FBAC section> + <LBAC

section>

Each section begins with a line that has a '.' in column 1 followed immediately by a keyword, e.g. '.dna'. The lines following such a header line up to the next header or end-of-input constitute the content of the section.

The Comment section consists of a single '.comment <comment string>' line. The comment string is attached to the output, and is intended to be used to pass CVS $ Id: $ information through the tool chain. More than one comment section can appear, the last section will override previous sections.

The Seed section begins with a '.seed' header line, followed by a seed. The command line –s option overrides the seed specified in the Seed section. All but the first line of the Seed section are ignored. Formally:

<Seed section> ← '.seed$_{\downarrow}$' <Seed>

The DNA section begins with a '.dna' header line, followed by a *frag* specification of either (1) a DNA sequence file, or (b) a stochastic grammar for generating a DNA sequence with the desired nucleotide composition and repeat structure. One is referred to the *frag* manual for a detailed explanation of the syntax and semantics of such a specification. In the *frag* document, the grammar non-terminal capturing the content of the DNA section is <DNA sequence>. Formally:

<DNA section> ← '.dna$_{\downarrow}$' <DNA sequence>

*Celsim* passes the DNA section proper to the *frag* program and captures the output as its base DNA sequence. Celsim extends the frag grammar by allowing annotation of an element of the <DNA sequence> with an '@'. This causes the fragments emitted by frag to be annotated if they derived from one of the annotated elements. This option is turned off if more than 1 polymorphism is generated..

A polymorphism section begins with a '.poly' header line where the keyword is followed by a list of one or more, white-space-separated real constants. The section following the keyword is passed to the *poly* program along with the base DNA sequence to produce a 'haplotype' version of the base sequence. One haplotype is created for each real constant in the header list. The real constant is associated with the given haplotype and it will be used to determine the extent to which the given haplotype is sampled. One may give several polymorphism sections with different mutating effects if desired. Conceptually, the result is a set of some number of haplotypes with associated real weights. One may also choose not to specify any polymorhpism sections in which case the DNA sequence is sampled directly by the sampling sections and constitutes 100% of the source of the DNA fragments.

*Note: in the current version of celsim does not have the poly features described in the follwing paragraph.*

One is referred to the *poly* manual for a detailed explanation of the syntax and semantics of a polymorphism specification. It is imperative to note that one should not give the name of the sequence to sample from in the section, *celsim* automatically inserts this before passing the section to *poly*. Formally then, a polymorphism section

has the grammar:

<Polymorphism section> ← '.poly' <real_constant> + '$_{\downarrow}$' <Mutation Operator> +

The sampling section is an extended version of that described in the frag manual. The syntax of a shotgun sampling specification is as follows:

```
<Shotgun~Sample> <- <Nat.frag> <Nat.minlen> <Nat.maxlen>
                   <Prob.forward>
                   <Nat.PrefixUnclearMin> <Nat.PrefixUnclearMax>
                   <Nat.SuffixUnclearMin> <Nat.SuffixUnclearMax>
                   <Prob.brate> <Prob.erate> <Prob.insert>
                   <Prob.delete>
```

The first four numbers pertain to the sampling of fragments from the dna strand. The number of fragments to be sampled, `Nat.frag`, the minimum length possible for any fragment, `Nat.minlen`, and the maximum length possible for any fragment (+/- 3 standard deviations if gaussian distribution is employed), `Nat.maxlen`, are all specified as positive integers, in that order. A fragment's length is chosen uniformly over the minimum to maximum length range (see above comments on gaussian distributions). Each fragment is chosen uniformly over all possible fragments of that length that could be sampled from the subject dna strand. The final parameter, Prob. forward is the probability (a real value between 0 and 1) with which the fragment is chosen from the subject strand versus its complement strand (in which the direction is reversed as well).

The next four numbers specify the portion of the prefix and suffix of the fragment that will be outside the clear range. If all four values are zero, compatibility with the output of previous versions of celsim will be achieved. As with the fragment length, the min and max values specify the bounds for a uniform distribution or the +/- 3 standard deviations of a gaussian distribution.

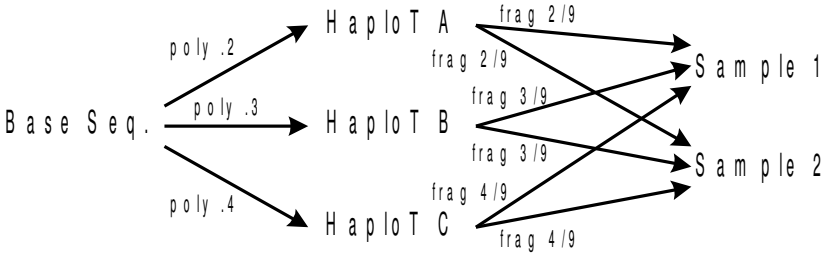
The last four numbers are concerned with the introduction of errors into fragments and all four are expressed as probabilities (real values between 0 and 1). An error rate ramp over the length of a fragment is specified by giving the error rate at the start of the fragment, `Prob.brate` and the error rate at the end of the fragment, `Prob.erate`. The rate at which errors are introduced at intermediate points along the fragment is determined by interpolating the end point rates, i.e., the probability of error at position *i* in the fragment is *brate + (i/fragment_length)×(erate - brate).* Errors are generated in a Poisson fashion according to the current error rate. Thus the number of errors in a given strand has a binomial distribution about the expected mean. The last two numbers specify the probabilities that an error is an insertion, `Prob.insert`, or deletion, `Prob.insert`, of a character. The probability that an error is a substitution is 1 minus the other two probabilities. In the case of a substitution error one of the other three nucleotides is chosen with equal probability, and if a character is inserted, each nucleotide is also chosen with equal probability.

If one has given a shotgun sampling specification, then one may further follow it with a four parameter specification of a dual-end sampling strategy. If so, then the output consist of a collection of reads, some of which are designated as being paired by virtue of having the same number followed by an f and r suffix, respectively (see later examples for further clarification). The specification is grammatically as follows

```
<Double_Pairs>  <-  <Prob.single>  <Nat.minins>  <Nat.maxins>
<Prob.prate>
```

The first number, `Prob.single`, is the probability with which fragments are not paired. For example, sampling 100 fragments and asserting that .1 will be unpaired results in an average of 10 unpaired reads in the data set. All other fragments are sampled in pairs from the end of a hypothetical insert. The next two positive integers, `Nat.minins` and `Nat.maxins`, specify the length range of the insert from which the end reads are obtained. For each pair, the insert length is chosen uniformly from this range. Finally, the last number, `Prob.prate`, is the probability that a given pairing is uncorrelated, i.e., that the paired reads actually come from arbitrary places within the target strand. Note that in such a case, they may, by chance, be in the right orientation and distance apart. This is important as one may later wonder why there are fewer false pairs reported by a piece of software than were actually requested/generated by *frag*.

For each sample, *celsim* calls *frag* to generate the requested number of reads. If more than one haplotype has been specified, then *celsim* asks *frag* to generate a number of fragments from that haplotype in proportion to the haplotype's weight to the total combined weight of all haplotypes. For example, if haplotypes A, B, and C, have weights .2, .3, and .4 respectively, then for a given sample, 2/9'ths of the fragments will be sampled from A, 1/3'rd from B, and 4/9'ths from C. In a picture:




Alternately, BAC end data can be simulated using a bacends section, with the same specification syntax as a .sample section.

<BAC end section> ← '.bacends $_{\downarrow}$' <Shotgun_Sample> <Double_Pairs> ?

Each sample is run through a post-processing step that converts the FASTA-formatted

fragments produced by *frag* into 3-code prototype records suitable for the Celera Assembler prototype. In addition, a 3-code distance record is output for each sample library. All messages are aggregated into the single file *specfile.data*. The clear ranges that are generated are determined by the sample section specification.

UBAC (Unfinished BAC) data is simulated using a .ubac section. The syntax of a UBAC sampling specification is as follows:

```
<UBAC~Sample> <- <Nat.shred> <Nat.coverage_depth>
                 <Nat.min_num_ubacs> <Nat.max_num_ubacs>
                 <Nat.min_ubac_length> <Nat.max_ubac_length>
                 <Prob.error> <Prob.insert> <Prob.delete>
                  <Nat.PrefixUnclearMin> <Nat.PrefixUnclearMax >
                 <Nat.SuffixUnclearMin> <Nat.SuffixUnclearMax >
```

The first value, Nat.shred, determines whether or not the bactigs, which comprise the UBAC, are shredded. If it is set to 0, the bactigs are output intact. If Nat.shred is set to 1, then the bactigs are shredded at a coverage depth of 3. The second value, Nat.coverage_depth, is used to simulate the depth at which a UBAC is sampled, which is used to determine the size of the bactigs in the UBACs. The third and fourth entries, Nat.min_num_ubacs and Nat.min_num_ubacs, are the minimum and maximum number of UBACs allowed. The fifth and six entries, Nat.min_ubac_length and Nat.max_ubac_length give the range of UBAC sizes. Prob.error gives the probability of a sequence error. Prob.insert and Prob.delete give the probabilities that the sequence error will be an insert or a delete, respectively. One minus their sum gives the probability that the error will be a substitution. In the case of a substitution error one of the other three nucleotides is chosen with equal probability, and if a character is inserted, each nucleotide is also chosen with equal probability. The meaning of the other four parameters is exactly the same as in a .sample section.

One note about the bactigs: they are constructed by simulating the shot gunning of a sequence the length of the UBAC. Fragments that overlap by more than 40 base pairs are put into the same bactig.

FBAC (Finished BAC) data is simulated using a .ubac section. The syntax of an FBAC sampling specification is as follows:

```
<UBAC~Sample> <- <Nat.shred> <Nat.min_num_fbacs>
                 <Nat.max_num_fbacs> <Nat.min_fbac_length>
                 <Nat.max_fbac_length> <Prob.error>
                 <Prob.insert> <Prob.delete>
                  <Nat.PrefixUnclearMin> <Nat.PrefixUnclearMax >
                 <Nat.SuffixUnclearMin> <Nat.SuffixUnclearMax >
```

The first value, Nat.shred, determines whether or not the FBACs are shredded. If it is set to 0, the FBACs are output intact. If Nat.shred is set to 1, then the FBACs are shredded at a coverage depth of 3. The second and third entries, Nat.min_num_fbacs and Nat.min_num_fbacs, are the minimum and maximum

number of FBACs allowed. The fourth and fifth entries, Nat.min_fbac_length and Nat.max_fbac_length give the range of FBAC sizes. Prob.error gives the probability of a sequence error. Prob.insert and Prob.delete give the probabilities that the sequence error will be an insert or a delete, respectively. One minus their sum gives the probability that the error will be a substitution. In the case of a substitution error one of the other three nucleotides is chosen with equal probability, and if a character is inserted, each nucleotide is also chosen with equal probability. The meaning of the other four parameters is exactly the same as in a .sample section.

The LBAC (Lightly-shotgunned BAC) inputs are based largely on those of the sampling section. The syntax of an LBAC sampling specification is as follows:

```
<LBAC~Sample> <- <Nat.coverage_depth>
                 <Nat.minlen> <Nat.maxlen> <Prob.forward>
                 <Nat.PrefixUnclearMin> <Nat.PrefixUnclearMax>
                 <Nat.SuffixUnclearMin> <Nat.SuffixUnclearMax>
                 <Prob.brate> <Prob.erate> <Prob.insert>
                 <Prob.delete>
                 <Prob.single> <Nat.minins> <Nat.maxins>
                 <Prob.prate>
                 <Nat.min_num_lbacs> <Nat.max_num_lbacs>
                 <Nat.min_lbac_length> <Nat.max_lbac_length>
```

The first number, Nat.coverage_depth, specifies the coverage of the LBAC. As for a .sample section, the next three numbers pertain to the sampling of fragments from the dna strand. The number of fragments to be sampled, `Nat.frag`, the minimum length possible for any fragment, `Nat.minlen`, and the maximum length possible for any fragment (+/- 3 standard deviations if gaussian distribution is employed), `Nat.maxlen`, are all specified as positive integers, in that order. A fragment's length is chosen uniformly over the minimum to maximum length range (see above comments on gaussian distributions). Each fragment is chosen uniformly over all possible fragments of that length that could be sampled from the subject dna strand. The final parameter, Prob. forward is the probability (a real value between 0 and 1) with which the fragment is chosen from the subject strand versus its complement strand (in which the direction is reversed as well).

The next four numbers specify the portion of the prefix and suffix of the fragment that will be outside the clear range. If all four values are zero, compatibility with the output of previous versions of celsim will be achieved. As with the fragment length, the min and max values specify the bounds for a uniform distribution or the +/- 3 standard deviations of a gaussian distribution.

The next four numbers are concerned with the introduction of errors into fragments and all four are expressed as probabilities (real values between 0 and 1). An error rate ramp over the length of a fragment is specified by giving the

error rate at the start of the fragment, `Prob.brate` and the error rate at the end of the fragment, `Prob.erate`. The rate at which errors are introduced at intermediate points along the fragment is determined by interpolating the end point rates, i.e., the probability of error at position *i* in the fragment is *brate + (i/fragment_length)×(erate - brate)*. Errors are generated in a Poisson fashion according to the current error rate. Thus the number of errors in a given strand has a binomial distribution about the expected mean. The last two numbers specify the probabilities that an error is an insertion, `Prob.insert`, or deletion, `Prob.insert`, of a character. The probability that an error is a substitution is 1 minus the other two probabilities. In the case of a substitution error one of the other three nucleotides is chosen with equal probability, and if a character is inserted, each nucleotide is also chosen with equal probability.

The next number, `Prob.single`, is the probability with which fragments are not paired. For example, sampling 100 fragments and asserting that .1 will be unpaired results in an average of 10 unpaired reads in the data set. All other fragments are sampled in pairs from the end of a hypothetical insert. The next two positive integers, `Nat.minins` and `Nat.maxins`, specify the length range of the insert from which the end reads are obtained. For each pair, the insert length is chosen uniformly from this range. The next number, `Prob.prate`, is the probability that a given pairing is uncorrelated, i.e., that the paired reads actually come from arbitrary places within the target strand. Note that in such a case, they may, by chance, be in the right orientation and distance apart. This is important as one may later wonder why there are fewer false pairs reported by a piece of software than were actually requested/generated by *frag*.

The last four numbers specify the number and size of the LBACs. Nat.min_num_lbacs gives the mininum number of LBACS, and Nat.max_num_lbacs gives the maximum. The bounds on the lengths of the LBACs are given by Nat.min_lbac_length and Nat.max_lbac_length.

As an example one might specify:

```
.dna
# This is derived from Granger^s synthetic human (98Dec18).
A = 141                       ; # half of ALU
@ B = A A m(.25)              ; # whole ALU
@ C = 7000                    ; # LINE
D = 2-8                       ; # short tandem repeat unit
@ E = D m(0.01,0.02) n(80)    ; # short tandem repeat
F = 200 - 400                 ; # medium tandem repeat unit
@ G = F m(0.00,0.07) n(10)    ; # medium tandem repeat
# H ~ 5000                    ; # long tandem satellite unit (rDNA)
# @ I = H m(0.001,0.003) n(5,35) ; # long tandem satellite  "16s RNA
units"
@ J = 2000           ; # genes
#@ K = 50000                ; # translocations (1M,10M,100M)
# Z should be 3500000000 for human sized problem
# Z should be  125000000 for fruit fly sized problem
```

```
Z = 35000
# 10% of human genome is ALUs with about 15% variation
 B o(.5) m(0.05,0.3)  n(0.1)   f(0.05,0.05,0.02,0.20,0.95)
 B o(.5) m(0.05,0.3)  n(0.001) f(0.00,0.00,1.00,0.20,0.70)
#  5% of human genome is LINEs with suffix fractures
 C o(.5) m(0.05,0.15) n(0.047) f(0.00,1.00,0.00,0.05,0.20)
 C o(.5) m(0.05,0.15) n(0.003) f(0.00,1.00,0.00,0.20,1.00)
 C o(.5) m(0.05,0.15) n(0.0001)
# We have used 15.1% of the genome so far in ALUs and LINEs.
#
#
# We will use !(floating number) to represent the fraction of
# base pairs involved with this structure.
# tandem repeats
  E o(.5) # Each one different (   630bp)  1%
  G o(.5) # Each one different (  4200bp)  1%
# I o(.5) # Each one different (100000bp)  0.7-5.0%
# genes
  J o(.5) m(0.01,0.05) n(2) # Two copies per gene(  6000bp)  0.6-2.8%
;
# 30% no mate, 0.7*(80% 2k separated mate, 20% 10k separated mate)
# 30% no mate, 56% 2k separated mate, 14% 10k separated mate
# Let G be the genome length and B the BAC length: 15*G = B*n/2
# Bac ends are a bit shorter, higher error rate, higher chimer rate,
# low single rate. The desired length range is mean 150k stddev 33k
# (2*10**(-4))*G  BAC END Fragments

# .sample spec
# number of fragments
# min frag length, max frag length, F/R odds
# min prefix unclear range, max prefix unclear range, min suffix
unclear range, max suffix unclear range (massage spec)
# Error ramp Min, Error ramp Max, Ins odds, Del odds
# Single Odds, min insert length, max insert length, False mate rate

.sample
560
300 800 0.5
0 0 0 0
.005 .020 .33 .33
.3 1200 2349 .01

.bacends
10
300 500 0.5
0 0 0 0
0.0025 0.050 .33 .33
.01 50000 250000 .20

.lbac
5
300 700 0.5
0 0 0 0
0.0025 0.050 .33 .33
1.0 50000 250000 .20
1 1 10000 10000

.ubac
1 3
20 20 3000 10000 0.001 .33 .33
```

```
0 0 0 0

.fbac
0
5 7 3000 10000 0.001 .33 .33
0 0 0 0
```

### 2. Sample Execution:

We conclude with an example of the output of the *celsim* program when run on the specification immediately above. *Celsim* was run with neither option flag set, so that it generated a random seed (reported in the first comment) and it generated a *.cms*-file. In order to shorten the listing, segments were deleted and such deletions are denoted by ellipses, '. . .'. Below is a listing of the first distance message and first three fragment messages output to the *.data*-file in the 3-code prototype format:

```
{BAT
bna:celsim batch 0
crt:953318935
acc:0
com:
(No comment)
.
}
{ADT
{ADL
who:Celsim 1.2 (gaussian)  2000/03/06
ctm:953318935
vsn: $Id: CelsimManual.rtf,v 1.1.1.1 2004/04/14 13:45:53 catmandew Exp $
com:
Genome Length is 35000
 ********Celsim Input ********
# A member of the "a" family of DNA specification files are intended
for
# end-to-end run checks.
#
#
# The first of the family a001.sim is a full sized
# synthetic human without the natural 0.2% polymorphism.
# Later members of the family are reduced in size,
# but not by a straight forward size scaling.
# Sometimes amount of important features were maintained as while
# reducing the total problem size.
# The a002.sim is the 1/10th human.
# The a003.sim is the 1/100th human intended for week-end runs.
# The a004.sim is the 1/1000th intended for nightly runs.
# The a005.sim is the 1/10000th human intended for sandbox testing.
# The a006.sim is the 1/100000th human intended for sandbox testing.
#
.comment $Id: CelsimManual.rtf,v 1.1.1.1 2004/04/14 13:45:53 catmandew
Exp $
.seed
17131
.dna
# This is derived from Granger^s synthetic human (98Dec18).
A = 141                       ; # half of ALU
```

```
@ B = A A m(.25)             ; # whole ALU
@ C = 7000                   ; # LINE
…
.fbac
1
3 3 10000 10000 0.001 .33 .33
0 0 0 0
 ********End Celsim Input ********
.
}
.
}
{DST
act:A
acc:1
mea:1774.000000
std:191.333328
}
{FRG
act:A
acc:2
typ:R
src:
3f
[7347,6897]
J.0.1 (2) [873,1323] [450,0]
.
etm:0
seq:
cccagattccttccctctgctgcggaccacagtagttttcatgtgcagttacacccactgaaggccattc
tcacattaagcatgcaccttatccaggagatgtgcgcacggttgcgaagattctgcggcgccgggggtta
ataccattagacatgaatcaagcggtgtaaagcgtgtcgaagctcagaataaagatgtttaatgccagac
cacgtgtggagtagtgttgtgtattactaacgcgagtaactagacaggcatcgagtgcttccggtagaga
ggagttattataaagtagatagatagtcatgaaggagagcattattctggcctatactgttttaattgcg
tacggtccaacacgtaccttctggagccggtgtgaccgtaaacgattaagtgtgataatgggagccgctt
tcacaatgccccctgcatccgcgaagac
.
qlt:
JJJJJJFJJSTJPJOJTJJDJ8JSJJJEJIJ9JJJIEJJ@9OJ6JJJJJJIJJJ86JJJJJJJJJPJJFJ
JJSJJQJJJJJJJJJJJJQJJJSJJJJJJJJJJSJNJCBJJJFJJJJJJJJJJJJJJJJJJJJJJJJJJ
;JJJJMJJ>JJJJJJ>JJJJJJJJJJJJJJJ96JJJJ7JJJOJJJJJ6JJJJJJJJJJJE:JJ>JJJJJJ
JJQJJJJJJQJFJJBJIJJ9IJJJJJPJOJJJJDDIJJJ=OJP>JJJJJJJEJJJJJJJJJJJJJJ<@J
GJJJTJKJJJGJJJ5JJJJJEJJ7JJAJJJJJJLJJIJJJJJJJJJJJF@JJJJKJJGJHHJJJ78J85J
JJJJLJJJJDJJJJ4JJJJJJJJ:JKJQJJJ6JJJJJJJJDJQJAJJ7JFJT;JMJPJJJJJJ<JJJJLJ
JJPJJJJJJJJJ>JJJJJJJJJJ=JJJJ
.
clr:0,448
}
{FRG
act:A
acc:3
typ:R
src:
3r
[5515,6135]
B.0.3 (14) [16,281] [0,265]
J.0.1 (2) [0,111] [509,620]
.
etm:0
seq:
```

```
atcgagtctcaactccttgagctggaattatcgtccacaacgctagagatgcaccgcggtaacctgtcta
cgtagtcacaccacgcgggccgatgaggtacttgacgaccggcacctgtcaccttcttaatatcgtattg
agagttaatacgcgctttgcgcctgcatcttattcgccgcagagcagcactacaccccgctctaacgtgg
atctacattcaggccggtcgcttgtaataatttgatgcccgactagtccgggtctatccctgtgctggta
acggttgatcccgcaagtaactcgtctcaaagatcaatattacacagagatgccagctcgtttgggctaa
tggacagttacaaaagagagaggtgtcagggtgtggccacgcgctttagatatggctacccgcttcacat
taacctgatcaataggcattagatgcgggtacagtcacaagccgattctaacaatttataccacaatcag
cgcagacacccgttcattttaacgcctagatcgcttctggtcgctcaaccataatccggtcatgaagtgg
gggaacaagatctgtaagaaatccggagtctggctcgacccatcggcctggagatct
.
qlt:
JFJJJJ@JJ>JJGPJ>JJCJJJJJJAJJJJHJ7JSJMJJJJJJJJJJ<J<JJJJJJJJ95JJJJJJJJJC
JJBJJJ6OJJJJJ?JJJJJJ>JJJ;JJJJ8JJJJJJJJJJQJJJJJJTJJJJJBJDJJEJJRAJ@JJ9JJ
JJJB=JJJJJJJJ=JJJJJJJJJJJJARJJJJJ8JDJJJJKJ5PJJJJ;JJ@JJ?JQJJTJJJJJGJJ=J
JJJJ<J<J=JJJJJRJJ5JJJJJ=<;JDJJJJJJ<MJGJJJJJJJJRJJ5J>EJ<JJJJJJJ8QJJJJJJF
JJJJJBJ>J<JJJIQJ?JJJJJJJNJMJJJJJ:JJJJJPJJJNCJ6JJJJEJ:<JJJJJJIJJJJIJJJJ
J<<J?JBJJJJJJ7JJJJJ8JJCJJJ?@JPJJJJJOJPICJJIJHTJLJAJ?JJJ?FJJJJJJJJJJGJJ
>JJJJJJJJJFJJJJGJJJJJHHJJQLJJJJJJAJJJJJ7JJ@JDJJJGJJJJJJJJJPJJ7JJJJJG<JJ
<5JJJJJN?JJJJJJ6TJJJJJJJJNJJJJJJJJJJJJJL>JJJ@JJJTJDLJE;JJJGLJGJJJFJ86J
JMJJJ?JJCJJJJJ6RJSJJJJJJJDJJRJPJJJN4JJ<TJJJ?KJJRCJJJ=OJHJ
.
clr:0,617
}
{LKG
act:A
typ:M
fg1:2
fg2:3
etm:0
dst:1
ori:I
}
{FRG
act:A
acc:4
typ:R
src:
5
[6718,6290]
J.0.1 (2) [266,694] [428,0]
.
etm:0
seq:
tatgctaaattgggaagggcgcgtgggaatagacgtgcaaccgaccgtagaggtgatggctcggcttgga
gtctgggagacgccaccgtgcgatcgcccacaaacgtttgattgagtatactagcagttacgacgtctta
ttaactcaacgatctgagttcgagaacccgcgggagttcaatgtaatccttgatttattagctctgacct
cgtataccgcgcaacactcgcgcctcagaaattaatagcaatgtacggacagtctccacacggaacaatg
actacagtttcacaccttatacagcttctgctaacaccggtgaacatgcacatgcttgatacggagcttt
agccgccttgcgttacagttcgtgaacttgcgaggcaccccaaaagtagaggctcgcgtgccgaaaattc
cgggg
.
qlt:
JEJJJJGJAJJFNJEJJJJJ@J6JJJSJJJJJJJJJJJJJJJJ:JJ<=J7JJJJJJIAJJJKE<JJJMN
IJTJJBOAJJBJJJJJJ5JJJJJJJJ8JJJ8JJ@JJJJJJJJJJJJJJJ@<C;JJN5JJJJJJJJDJJJJ
JJ5JJJFJFMJJJJJJJJJJGJJJJJDJJJJJJJMJJ<CJJJJJJJJJJJJJJJJJJBJJ@JJDTJJJJJ;
GJJJJJJJJJJJJGJEJJJJJJHJJJJJJ>JJJ:JL<JJJJJJJ5JMJJJ=JJJJAJJJ4JCK:JF>DJJ
JJRJ>TJIJJJJJJJJJJJJJJJ>TJJJ5:J=JJJFJ8EJJJFJJHJGFJJJJJJLJJJJ=JJJDJJJJJ
JJ@7JTJJJ@JJJJ=JJJJJJ7J9JJJJ@JPJJJCJ:?JIJ4?JJJJJJT4<J@AJ>9JJIJJJHJNJJJ
JJJJ;
.
```

```
clr:0,425
}
...
```

The *.cms*-file produced by *celsim* is basically the concatenation of the comments produced by the various invocations of *frag* and *poly*. Thus one is referred to the documentation on these programs for the details of what is reported. Suffice it here to observe that these comments give one a complete description of the repeat structure of the base sequence and the exact locations at which polymorphisms were induced to create the haplotypes.

```
# Celsim V1.2  $Id: CelsimManual.rtf,v 1.1.1.1 2004/04/14
13:45:53 catmandew Exp $
# DNA SEQUENCE GENERATION
# Frag $Revision: 1.1.1.1 $:(gaussian)
#
#
# Seed = 17131
#
# Element: A
#    Length = [141,141], ACGT odds = 0.25/0.25/0.25/0.25
# Element: B
#    Concatenation of:
#      A: O'odds = 1.00, Mut's = 0.00-0.00, 1-1 Rep's, 0-0
Gen's, 0% Fractures
#      A: O'odds = 1.00, Mut's = 0.25-0.25, 1-1 Rep's, 0-0
Gen's, 0% Fractures
# Element: C
#    Length = [7000,7000], ACGT odds = 0.25/0.25/0.25/0.25
# Element: D
#    Length = [2,8], ACGT odds = 0.25/0.25/0.25/0.25
# Element: E
#    Concatenation of:
#      D: O'odds = 1.00, Mut's = 0.01-0.02, 80-80 Rep's, 0-
0 Gen's, 0% Fractures
# Element: F
#    Length = [200,400], ACGT odds = 0.25/0.25/0.25/0.25
# Element: G
#    Concatenation of:
#      F: O'odds = 1.00, Mut's = 0.00-0.07, 10-10 Rep's, 0-
0 Gen's, 0% Fractures
# Element: J
#    Length = [2000,2000], ACGT odds = 0.25/0.25/0.25/0.25
# Element: Z
#    Length = [35000,35000], ACGT odds = 0.25/0.25/0.25/0.25
#    Containing:
#      B: O'odds = 0.50, Mut's = 0.05-0.30, 10-10 % of
Basis, 0-0 Gen's, Fract's = (0.05 %p, 0.05 %s, 0.02 %r,
length = 0.2-0.95
#      B: O'odds = 0.50, Mut's = 0.05-0.30, 0.1-0.1 % of
Basis, 0-0 Gen's, Fract's = (0.00 %p, 0.00 %s, 1.00 %r,
length = 0.2-0.7
#      C: O'odds = 0.50, Mut's = 0.05-0.15, 4.7-4.7 % of
Basis, 0-0 Gen's, Fract's = (0.00 %p, 1.00 %s, 0.00 %r,
length = 0.05-0.2
#      C: O'odds = 0.50, Mut's = 0.05-0.15, 0.3-0.3 % of
Basis, 0-0 Gen's, Fract's = (0.00 %p, 1.00 %s, 0.00 %r,
length = 0.2-1
```

```
#      C: O'odds = 0.50, Mut's = 0.05-0.15, 0.01-0.01 % of
Basis, 0-0 Gen's, 0% Fractures
#      E: O'odds = 0.50, Mut's = 0.00-0.00, 1-1 Rep's, 0-0
Gen's, 0% Fractures
#      G: O'odds = 0.50, Mut's = 0.00-0.00, 1-1 Rep's, 0-0
Gen's, 0% Fractures
#      J: O'odds = 0.50, Mut's = 0.01-0.05, 2-2 Rep's, 0-0
Gen's, 0% Fractures
#
#  Z.0 (35000)
#  > B.0f at 831-1113, mutated 0.12.
#    = A.0f at 831-972, mutated 0.00.
#    = A.0f at 972-1113, mutated 0.25.
#  > E.0r at 2225-2622, mutated 0.00.
#    = D.0r at 2225-2230, mutated 0.02.
#    = D.0r at 2230-2235, mutated 0.01.
#    = D.0r at 2235-2240, mutated 0.01.
#    = D.0r at 2240-2245, mutated 0.02.
#    = D.0r at 2245-2250, mutated 0.02.
#    = D.0r at 2250-2255, mutated 0.02.
#    = D.0r at 2255-2261, mutated 0.01.
#    = D.0r at 2261-2266, mutated 0.01.
#    = D.0r at 2266-2271, mutated 0.02.
#    = D.0r at 2271-2276, mutated 0.01.
#    = D.0r at 2276-2281, mutated 0.02.
#    = D.0r at 2281-2286, mutated 0.01.
#    = D.0r at 2286-2291, mutated 0.02.
#    = D.0r at 2291-2296, mutated 0.01.
#    = D.0r at 2296-2301, mutated 0.02.
#    = D.0r at 2301-2306, mutated 0.02.
#    = D.0r at 2306-2311, mutated 0.01.
#    = D.0r at 2311-2315, mutated 0.02.
#    = D.0r at 2315-2320, mutated 0.02.
#    = D.0r at 2320-2325, mutated 0.01.
#    = D.0r at 2325-2330, mutated 0.02.
#    = D.0r at 2330-2335, mutated 0.01.
#    = D.0r at 2335-2340, mutated 0.01.
#    = D.0r at 2340-2345, mutated 0.02.
#    = D.0r at 2345-2350, mutated 0.02.
#    = D.0r at 2350-2354, mutated 0.02.
#    = D.0r at 2354-2359, mutated 0.02.
#    = D.0r at 2359-2364, mutated 0.02.
#    = D.0r at 2364-2369, mutated 0.01.
#    = D.0r at 2369-2374, mutated 0.01.
#    = D.0r at 2374-2379, mutated 0.01.
#    = D.0r at 2379-2384, mutated 0.01.
#    = D.0r at 2384-2389, mutated 0.01.
#    = D.0r at 2389-2393, mutated 0.02.
#    = D.0r at 2393-2398, mutated 0.01.
#    = D.0r at 2398-2403, mutated 0.01.
#    = D.0r at 2403-2408, mutated 0.01.
#    = D.0r at 2408-2413, mutated 0.02.
#    = D.0r at 2413-2418, mutated 0.02.
#    = D.0r at 2418-2423, mutated 0.01.
#    = D.0r at 2423-2428, mutated 0.02.
#    = D.0r at 2428-2433, mutated 0.02.
#    = D.0r at 2433-2438, mutated 0.02.
#    = D.0r at 2438-2443, mutated 0.02.
#    = D.0r at 2443-2448, mutated 0.01.
#    = D.0r at 2448-2453, mutated 0.02.
```

```
#    = D.0r at 2453-2458, mutated 0.01.
#    = D.0r at 2458-2463, mutated 0.01.
#    = D.0r at 2463-2468, mutated 0.01.
#    = D.0r at 2468-2473, mutated 0.01.
#    = D.0r at 2473-2478, mutated 0.01.
#    = D.0r at 2478-2483, mutated 0.01.
#    = D.0r at 2483-2488, mutated 0.02.
#    = D.0r at 2488-2493, mutated 0.01.
#    = D.0r at 2493-2498, mutated 0.02.
#    = D.0r at 2498-2503, mutated 0.02.
#    = D.0r at 2503-2508, mutated 0.01.
#    = D.0r at 2508-2513, mutated 0.02.
#    = D.0r at 2513-2518, mutated 0.01.
#    = D.0r at 2518-2523, mutated 0.02.
#    = D.0r at 2523-2528, mutated 0.01.
#    = D.0r at 2528-2533, mutated 0.01.
#    = D.0r at 2533-2538, mutated 0.01.
#    = D.0r at 2538-2543, mutated 0.01.
#    = D.0r at 2543-2547, mutated 0.01.
#    = D.0r at 2547-2552, mutated 0.01.
#    = D.0r at 2552-2557, mutated 0.01.
#    = D.0r at 2557-2562, mutated 0.01.
#    = D.0r at 2562-2567, mutated 0.02.
#    = D.0r at 2567-2572, mutated 0.01.
#    = D.0r at 2572-2577, mutated 0.01.
#    = D.0r at 2577-2582, mutated 0.01.
#    = D.0r at 2582-2587, mutated 0.01.
#    = D.0r at 2587-2592, mutated 0.01.
#    = D.0r at 2592-2597, mutated 0.02.
#    = D.0r at 2597-2602, mutated 0.01.
#    = D.0r at 2602-2607, mutated 0.01.
#    = D.0r at 2607-2612, mutated 0.01.
#    = D.0r at 2612-2617, mutated 0.02.
#    = D.0r at 2617-2622, mutated 0.02.
#  > B.0r at 4305-4365, mutated 0.07, random fracture
[175,236].
#    = A.0r at 4305-4366, mutated 0.25, random fracture
[34,95].
#  > B.0f at 5499-5780, mutated 0.24.
#    = A.0f at 5499-5640, mutated 0.00.
#    = A.0f at 5640-5781, mutated 0.25.
#  > J.0f at 6024-8024, mutated 0.04.
#  > C.0r at 8827-10195, mutated 0.15, prefix fracture
[5631,7000].
#  > G.0f at 12080-14263, mutated 0.00.
#    = F.0f at 12080-12298, mutated 0.05.
#    = F.0f at 12298-12517, mutated 0.01.
#    = F.0f at 12517-12736, mutated 0.06.
#    = F.0f at 12736-12954, mutated 0.04.
#    = F.0f at 12954-13172, mutated 0.01.
#    = F.0f at 13172-13391, mutated 0.02.
#    = F.0f at 13391-13609, mutated 0.05.
#    = F.0f at 13609-13826, mutated 0.01.
#    = F.0f at 13826-14044, mutated 0.07.
#    = F.0f at 14044-14263, mutated 0.05.
#  > B.0r at 14339-14621, mutated 0.08.
#    = A.0r at 14339-14480, mutated 0.25.
#    = A.0r at 14480-14621, mutated 0.00.
#  > B.0r at 15060-15342, mutated 0.18.
#    = A.0r at 15060-15201, mutated 0.25.
```

```
#    = A.0r at 15201-15342, mutated 0.00.
#  > B.0f at 15955-16236, mutated 0.20.
#    = A.0f at 15955-16096, mutated 0.00.
#    = A.0f at 16096-16237, mutated 0.25.
#  > B.0r at 16657-16938, mutated 0.22.
#    = A.0r at 16657-16798, mutated 0.25.
#    = A.0r at 16798-16939, mutated 0.00.
#  > B.0f at 17447-17730, mutated 0.23.
#    = A.0f at 17447-17588, mutated 0.00.
#    = A.0f at 17588-17729, mutated 0.25.
#  > B.0r at 18909-19190, mutated 0.08.
#    = A.0r at 18909-19050, mutated 0.25.
#    = A.0r at 19050-19191, mutated 0.00.
#  > B.0f at 20894-21175, mutated 0.11.
#    = A.0f at 20894-21035, mutated 0.00.
#    = A.0f at 21035-21176, mutated 0.25.
#  > B.0f at 21247-21375, mutated 0.21.
#    = A.0f at 21247-21375, mutated 0.00, suffix fracture
[0,128].
#  > B.0r at 22749-23031, mutated 0.14.
#    = A.0r at 22749-22890, mutated 0.25.
#    = A.0r at 22890-23031, mutated 0.00.
#  > B.0r at 23195-23433, mutated 0.22, prefix fracture
[44,282].
#    = A.0r at 23195-23336, mutated 0.25.
#    = A.0r at 23336-23433, mutated 0.00, prefix fracture
[44,141].
#  > C.0f at 23816-24915, mutated 0.11, prefix fracture
[5900,7000].
#  > J.0f at 25315-27314, mutated 0.01.
#  > C.0r at 27522-33044, mutated 0.10, prefix fracture
[1478,7000].
#  > B.0r at 34399-34681, mutated 0.08.
#    = A.0r at 34399-34540, mutated 0.25.
#    = A.0r at 34540-34681, mutated 0.00.
#
#
# NO POLYMORPHISMS APPLIED, USED BASE SEQUENCE
#
#
# FRAGMENT LIBRARY 1
#
# Fragments   Seed   Poly Source 1
# ---------   ----   -----------
#       560  17133   a006.poly.1.17131
# ---------
#       560
#
#    Length Range = [300,800], F/R odds = 0.50/0.50
#
# Clear Range Characteristics:
#    Prefix Unclear [0,0]
#    Suffix Unclear [0,0]
# Edit Characteristics:
#    Error Ramp = 0.01->0.02, Ins/Del/Sub Odds =
0.33/0.33/0.34
#
# Dual-End Inserts:
#    Single Odds = 0.30
#    Insert Range = [1200,2349]
```

```
#    Pairing Error Rate = 0.01
#
#
# FRAGMENT LIBRARY 2
#
# Fragments   Seed   Poly Source 1
# ---------   ----   -----------
#       140  17134   a006.poly.1.17131
# ---------
#       140
#
#    Length Range = [300,800], F/R odds = 0.50/0.50
#
# Clear Range Characteristics:
#    Prefix Unclear [0,0]
#    Suffix Unclear [0,0]
# Edit Characteristics:
#    Error Ramp = 0.01->0.02, Ins/Del/Sub Odds =
0.33/0.33/0.34
#
# Dual-End Inserts:
#    Single Odds = 0.30
#    Insert Range = [6050,12800]
#    Pairing Error Rate = 0.01
#
#
# FRAGMENT LIBRARY 3
#
# Fragments   Seed   Poly Source 1
# ---------   ----   -----------
#        10  17135   a006.poly.1.17131
# ---------
#        10
#
#    Length Range = [300,500], F/R odds = 0.50/0.50
#
# Clear Range Characteristics:
#    Prefix Unclear [0,0]
#    Suffix Unclear [0,0]
# Edit Characteristics:
#    Error Ramp = 0.00->0.05, Ins/Del/Sub Odds =
0.33/0.33/0.34
#
# Dual-End Inserts:
#    Single Odds = 0.01
#    Insert Range = [50000,250000]
#    Pairing Error Rate = 0.20
#

#
```

**AUTHORS**

Gene Myers:

Created October 12, ‘98

Revised SAK 1/5/99 Got rid of -b option, and added frag < option in the specification grammar.

Revised CMM 1/6/99 Added the new -b option for

incremental processing by batch. Also, set the creation times in the FRG messages to a fixed time to facilitate regression testing.

Revised SAK 1/20/99 Added the new .comment section, and nuked the obsolete –u option.

Revised SAK 2/10/99 Added the new –u and –n options

Revised SAK 3/15/99 Added the new -p

Revised SAK 4/23/99 BAC End guides and clear ranges