Harvard-MIT Division of Health Sciences and Technology
HST.508: Quantitative Genomics, Fall 2005
Instructors: Leonid Mirny, Robert Berwick, Alvin Kho, Isaac Kohane

Ken Roach
HST.508 PS#2

Problem 1

A) Starting with A, calculate the probabilities of A, G, C, and T for the Kimura two-parameter model by solving the linear system dP/dt = ΛP, where Λ is the transition matrix.

$$\Lambda = \begin{bmatrix} -\alpha-2\beta & \alpha & \beta & \beta \\ \alpha & -\alpha-2\beta & \beta & \beta \\ \beta & \beta & -\alpha-2\beta & \alpha \\ \beta & \beta & \alpha & -\alpha-2\beta \end{bmatrix} \qquad P = \begin{bmatrix} A \\ G \\ C \\ T \end{bmatrix} \qquad \frac{dP}{dt} = \Lambda P$$

Solving the differential equation gives:

$$P(t) = e^{\Lambda t} P_0 = U e^{Dt} U^{-1} P_0 \quad \text{where} \quad \Lambda = U D U^{-1} \quad \text{in diagonalized form}$$

Using Mathematica to find the eigenvalues and eigenvectors.

$$U = \begin{bmatrix} 1 & -1 & 0 & -1 \\ 1 & -1 & 0 & 1 \\ 1 & 1 & -1 & 0 \\ 1 & 1 & 1 & 0 \end{bmatrix} \qquad U^{-1} = \begin{bmatrix} 1/4 & 1/4 & 1/4 & 1/4 \\ -1/4 & -1/4 & 1/4 & 1/4 \\ 0 & 0 & -1/2 & 1/2 \\ -1/2 & 1/2 & 0 & 0 \end{bmatrix} \qquad P_0 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$D = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & -4\beta & 0 & 0 \\ 0 & 0 & -2(\alpha+\beta) & 0 \\ 0 & 0 & 0 & -2(\alpha+\beta) \end{bmatrix} \qquad e^{Dt} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & e^{-4\beta t} & 0 & 0 \\ 0 & 0 & e^{-2(\alpha+\beta)t} & 0 \\ 0 & 0 & 0 & e^{-2(\alpha+\beta)t} \end{bmatrix}$$

Putting these together gives:

$$P(t) = \begin{bmatrix} \frac{1}{4} + \frac{1}{4} e^{-4\beta t} + \frac{1}{2} e^{-2(\alpha+\beta)t} \\ \frac{1}{4} + \frac{1}{4} e^{-4\beta t} - \frac{1}{2} e^{-2(\alpha+\beta)t} \\ \frac{1}{4} - \frac{1}{4} e^{-4\beta t} \\ \frac{1}{4} - \frac{1}{4} e^{-4\beta t} \end{bmatrix}$$

B) Consider two homologous sequences of length L having P transitions and Q transversions. Calculate the number of substitutions K that have occurred between the two sequences.

First normalize K, P, and Q with respect to the length of the sequence to simplify later equations.

$$K' = \frac{K}{L} \qquad P' = \frac{P}{L} \qquad Q' = \frac{Q}{L}$$

The number of substitutions between two sequences is two times the product of the mutation rate and the time since divergence. We then need to find expressions relating ατ and βτ to P' and Q'.

$$K' = 2\mu\tau = 2(\alpha+2\beta)\tau$$

P' is the probability that two initially identical nucleotides will look like a transition after time τ.

$$P' = 2P(A)P(G) + 2P(C)P(T)$$

$$P' = 2\left(\frac{1}{4} + \frac{1}{4}e^{-4\beta\tau} + \frac{1}{2}e^{-2(\alpha+\beta)\tau}\right)\left(\frac{1}{4} + \frac{1}{4}e^{-4\beta\tau} - \frac{1}{2}e^{-2(\alpha+\beta)\tau}\right) + 2\left(\frac{1}{4} - \frac{1}{4}e^{-4\beta\tau}\right)^2$$

$$P' = \frac{1}{4}\left(1 + e^{-8\beta\tau} - 2e^{-4(\alpha+\beta)\tau}\right)$$

Q' is the probability that two initially identical nucleotides will look like a transversion after time τ.

$$Q' = 2P(A)P(C) + 2P(A)P(T) + 2P(G)P(C) + 2P(G)P(T)$$

$$Q' = 4\left[\left(\frac{1}{4} + \frac{1}{4}e^{-4\beta\tau} + \frac{1}{2}e^{-2(\alpha+\beta)\tau}\right) + \left(\frac{1}{4} + \frac{1}{4}e^{-4\beta\tau} - \frac{1}{2}e^{-2(\alpha+\beta)\tau}\right)\right]\left(\frac{1}{4} - \frac{1}{4}e^{-4\beta\tau}\right)$$

$$Q' = \frac{1}{2} - \frac{1}{2}e^{-8\beta\tau}$$

Solving the above equation for βτ gives:

$$\beta\tau = -\frac{1}{8}\ln(1-2Q')$$

Adding the two equations together to eliminate one exponential and solving gives:

$$2P'+Q' = 1-e^{-4(\alpha+\beta)\tau}$$

$$\alpha\tau+\beta\tau = -\frac{1}{4}\ln(1-2P'-Q')$$

We can now write K' in terms of P' and Q'.

$$K' = 2\alpha\tau + 4\beta\tau = -\frac{1}{2}\ln(1-2P'-Q') - \frac{1}{4}\ln(1-2Q')$$

$$K' = -\frac{1}{4}\ln\left[(1-2P'-Q')^2(1-2Q')\right]$$

As a sanity check, the expression limits to $K' = P' + Q'$ when P' and Q' are small. This is what one would expect if there is no more than one substitution per site.

Problem 2

Calculate the number of possible gapped alignments between two sequences of length L_1 and L_2.

Assuming that the relative ordering of unmatched symbols is unimportant, the number of possible alignments can be found by calculating the number of alignments with *k* matches and then summing over all possible values of *k*. The number of alignments with *k* matches is simply the number of ways *k* sites can be chosen from L_1 times the number of ways *k* sites can be chosen from L_2. A combinatorial identity provides a simplified version of the sum.

$$N(k) = \binom{L_1}{k}\binom{L_2}{k} \qquad N_{total} = \sum_{k=0}^{min(L_1,L_2)} \binom{L_1}{k}\binom{L_2}{k} = \binom{L_1+L_2}{L_1} = \binom{L_1+L_2}{L_2}$$

It should be noted that the above calculation only considers the matched sites and does not account for different orderings of the gapped sites. For example, all three alignments below would be considered identical and counted as a single alignment since the same sites are matched in each case. Whether or not this makes sense depends on the scoring algorithm. For some algorithms, the alignments below could have very different scores and treating them as the same would not make sense. However, for simple algorithms where each site is scored independently, the score would be the same regardless of the order in which the gapped sites are placed. Regardless of the chosen algorithm, it is hard to assign a physical significance to the order of the gapped sites unless the sequences are examined in some additional context like its function or evolutionary history.

```
AATTCG----CTAGAT    AATTCGCTAG----AT    AATTCG--CT--AGAT
||||||          ||    ||||||          ||    ||||||          ||
AGTTAGATTT----AT    AGTTAG----ATTTAT    AGTTAGAT--TT--AT
```

Using a similar technique as before, it is possible to calculate the number of alignments with the order of the gapped sites taken into account. An alignment of sequences length L_1 and L_2 with *k* matches will have length $L = L_1 + L_2 - k$. We can first imagine placing the L_1 symbols of the first sequence within these L spaces and leaving $L_2 - k$ gaps. This can be done in *L* choose L_1 ways. We can then pick which of the *k* symbols from the first sequence will be matched with symbols from the second sequence. This can be done in L_1 choose *k* different ways. Note that this fully specifies the placement of the symbols in the second sequence, since there are $L_2 - k$ gaps and *k* matches for a total of exactly L_2 available locations. The answer can also be thought of as a multinomial. The $L_1 + L_2 - k$ sites in the aligned sequence are divided into $L_1 - k$ unmatched symbols from the first sequence, $L_2 - k$ unmatched symbols from the second sequence, and *k* matched symbols. The multinomial is readily apparent when one expands the binomial coefficients. As before, the total number of alignments is simply the sum over all possible *k*. Unfortunately, there does not seem to be a convenient way to simplify the expression.

$$N(k) = \binom{L_1+L_2-k}{L_1}\binom{L_1}{k} = \frac{(L_1+L_2-k)!}{(L_1-k)!(L_2-k)!k!} \qquad N_{total} = \sum_{k=0}^{min(L_1,L_2)} \frac{(L_1+L_2-k)!}{(L_1-k)!(L_2-k)!k!}$$

<u>Problem 3</u>

Perform the MacDonald-Kreitman test for five genes using the Seattle SNPs data for polymorphisms and BLAT alignments against the chimp genome for fixed mutations.

Six genes were chosen from the Seattle SNPs website based on having interesting function and a fairly large number of SNPs in the coding region. The synonymous and non-synonymous sites listed in the SNP data files were counted using a small Matlab script. The coding sequence for each gene was then compared to the chimp genome using BLAT. The resulting alignment was run through a Matlab script provided by Carlos, generating a list of mismatched codons. Another script was written to determine whether the codons were synonymous or non-synonymous and tally the results for each gene. The final counts were arranged in the 2x2 contingency tables shown below and p-values were calculated using the two-tailed Fisher's exact test and the G-test with William's correction. The Matlab scripts will be included at the end of the problem set.

ABOBG

	Fixed Mutation	*Polymorphism*
Non-Synonymous	5	13
Synonymous	7	8
Fisher p-value:	0.3005	
G statistic:	1.2026	
G-test p-value:	0.2728	

IFGR1

	Fixed Mutation	*Polymorphism*
Non-Synonymous	4	3
Synonymous	0	2
Fisher p-value:	0.4636	
G statistic:	2.2078	
G-test p-value:	0.1373	

COMP3

	Fixed Mutation	*Polymorphism*
Non-Synonymous	10	6
Synonymous	19	13
Fisher p-value:	1.0000	
G statistic:	0.0421	
G-test p-value:	0.8375	

PLMGN

	Fixed Mutation	*Polymorphism*
Non-Synonymous	7	7
Synonymous	6	10
Fisher p-value:	0.7131	
G statistic:	0.4525	
G-test p-value:	0.5012	

ICAM1

	Fixed Mutation	*Polymorphism*
Non-Synonymous	18	6
Synonymous	7	2
Fisher p-value:	1.000	
G statistic:	0.0255	
G-test p-value:	0.8731	

VLDLR

	Fixed Mutation	*Polymorphism*
Non-Synonymous	3	6
Synonymous	6	4
Fisher p-value:	0.3698	
G statistic:	1.2686	
G-test p-value:	0.2600	

None of the genes showed statistically significant evidence of positive selection. Two of the genes fit the null hypothesis almost perfectly and had p-values of 1.000 using Fisher's exact test. The p-values were quite high for the other genes as well. One potential issue may have been the relatively small number of SNPs and fixed mutations used in the analysis. With more mutations to analyze, the tests would have had more statistical power. Unfortunately, most of the BLAT alignments were incomplete, so not all of the fixed mutations could be found. With the relatively small number of observations in the above tables, Fisher's exact test is generally preferred over the G-test, but both were performed for the sake of experience. The difference between the two tests is clear from the p-values.

Problem 4

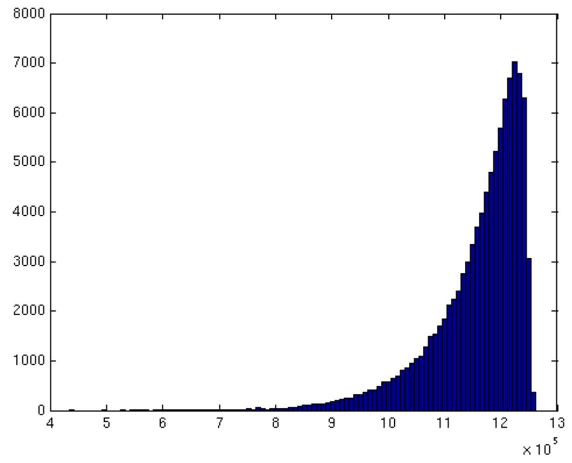
From the list of available pathways, I chose to examine branched-chain amino acid and histidine biosynthesis. I began by looking up each pathway in the KEGG database and selecting E. coli as the target organism. From there I was able to find a list of the involved genes and their positions on the chromosome, as well as a table of orthologous proteins in related organisms. Several Matlab scripts were written to help solve various parts of the problem. The code for these scripts will be attached at the end of the problem set.

A) Do genes of a given pathway tend to be close to each other on the chromosome?
Yes. In the branched amino acid biosynthesis pathway, the involved genes were found in three tight clusters. Within these clusters, the genes were almost directly adjacent to each other. In the histidine biosynthesis pathway, all of the genes were found together in a single cluster. The mean distance between all possible pairs of genes in each pathway was calculated using a small Matlab script. For branched amino acid biosynthesis, the mean distance was found to be 451395 nucleotides, which turns out to be about half the expected distance between two arbitrary genes on the chromosome. For histidine biosynthesis, the mean distance was found to be 1793 nucleotides, reflecting the fact that all of the genes were directly adjacent to each other.

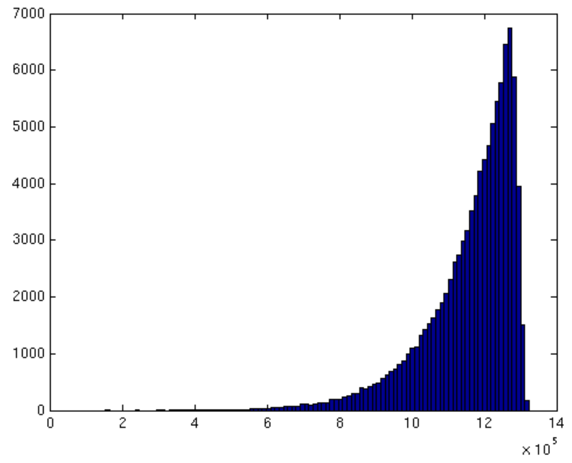
It is quite straightforward to derive a statistical distribution for the distance between two arbitrary genes. The simplest model is to assume that the genes are small compared to the size of the genome and independently placed on the chromosome with a uniform probability distribution. Since bacterial chromosomes are circular, the distance between two arbitrary genes will be uniformly distributed between 0 and $L/2$ where L is the length of the genome. The expectation is of course $L/4$. A small correction for nonzero gene size can be made by replacing L with $(L - 2L_g)$, where L_g is the average gene length. The problem becomes much more complicated when one tries to derive a statistical distribution for the mean pairwise distance of a set of genes. The expectation will still be $L/4$, but the variance calculations will have to take into account the fact that the pairwise distances are far from independent. The easiest way to get the distribution is to perform a Monte-Carlo simulation on the complete genome. For this purpose, a complete list of E coli genes was downloaded from the URL "ftp://ftp.ncbi.nlm.nih.gov/genbank/genomes/Bacteria/Escherichia_coli_K12/U00096.ptt".

As an initial sanity check for the simple uniform model described earlier, a Matlab script was written to compute the mean distance between all pairs of genes in the E. coli genome. Using this script, the mean pairwise distance was found to be 1159025 nucleotides, which is quite close to the predicted value of $E[d] = (L - 2L_g)/4$ = (4639676 - 2*959)/4 = 1159440 nucleotides. Another script was written to carry out the Monte-Carlo simulation. The script repeatedly chooses a set of N random genes from the gene list and calculates the mean pairwise distance, adding the result to a growing list. The data can then be used to plot histograms and calculate statistical parameters for significance tests. The mean pairwise distance computed for each pathway was evaluated for statistical significance in a one-tailed test with the null hypothesis that the genes were randomly chosen from the complete genome. Critical values for $p < 0.01$ were calculated using Monte-Carlo simulations with 100,000 samples. As can be seen in the histograms, the distributions are quite skewed, so assuming normality is not a good idea. When run with only two genes, the simulation generates a uniform distribution as predicted by the simple model. The results of the tests are shown in the table below. Both pathways had means well below the critical values, so the null hypothesis can be firmly rejected in both cases.

Pathway Name	***Number of Genes***	***Mean Pairwise Distance***	***Monte-Carlo Mean***	***Monte-Carlo St Dev***	***Critical Value (p = 0.01)***
Branched	12	451,395	1,158,862	82,442	871,973
Histidine	8	1,793	1,158,452	127,337	721,888



Histogram for 12 Genes



Histogram for 8 Genes

B) Do the genes cluster in other related genomes?
Yes, for both pathways. The genes tend to form clusters in almost all of the bacterial genomes shown in the ortholog tables. The histidine pathway forms a single cluster in most of the genomes, but is quite scattered in a few. However, these tend to be from the more distantly related organisms. The branched amino acid pathway forms several clusters in most of the genomes. A few genomes are more scattered than E. coli, but many follow the exact same pattern.

C) Are the same genes close to each other? Is gene order preserved?
The same genes tend to be close to each other, particularly in closely related organisms. In many of these organisms, gene order is also preserved. This seems to be true for both pathways. For several organisms in the branched amino acid pathway, gene order was exactly preserved except for insertion or movement of a transcriptional regulator.

D) Is there a pattern of phylogenetic co-occurrence in the genes?
There does seem to be a pattern of phylogenetic co-occurrence in the genes. The organisms tend to have either all or none of the enzymes needed in a particular segment of the pathway. This makes sense since losing a single enzyme would make the segment nonfunctional. For branched amino acid biosynthesis, the four enzymes specific to leucine production are missing in certain organisms. With few exceptions, they are either all present or all absent. Histidine biosynthesis follows a linear set of reactions, so similar patterns are not visible. A few organisms seem to be missing one or two random enzymes in an otherwise complete pathway, but perhaps the genes have simply not been identified yet in these organisms.

E) Do any non-enzymes tend to cluster with the enzymes of the pathway? If so, why?
Most of the non-enzymes that cluster with the enzymes of the pathway seem to be transcriptional regulators of the enzymes. This makes sense since the regulators are useless if the organism lacks the enzymes. Placing them in the same operon also allows for positive feedback and rapid deployment of the enzymes when needed. A few enzymes used in other metabolic pathways also seem to cluster with those used in the pathway of interest. For example, the enzymes acetyl-CoA synthetase and threonine dehydratase cluster with branched amino acid synthesis enzymes despite not being in the pathway. This may reflect higher level integration of metabolic processes within the cell that are not obvious when viewing metabolism as a collection of independent pathways.

F) Do any transporters cluster with the enzymes? Do they show phylogenetic co-occurrence?
The amino acid transporters for histidine or branched chain amino acids do not seem to cluster with the metabolic enzymes. The individual components of the transporter are usually found together, but not necessarily close to the enzymes. There was no obvious evidence of phylogenetic co-occurrence. One would expect that organisms with no synthesis capability would have a greater need for transporters, but they seem to be present in almost all of the organisms.

Matlab Scripts for Problem 3

```
%%----------------------------------------------------------------------------------
function [counts, p_fisher, gstat, p_gtest] = mktest(gene);
% Fills in the contingency table using BLAT results and Seattle SNPs data,
% then calculates p-values using Fisher's exact test and the G-test.

[sm, nm] = count_mutations(strcat(gene,'_blat.txt'));
[sp, np] = count_polymorphisms(strcat(gene,'_csnps.txt'));

counts = [nm, np; sm, sp];
p_fisher = fisher_exact(counts);
[gstat, p_gtest] = gtest(counts);


%%----------------------------------------------------------------------------------
function [synon, nonsynon] = count_mutations(filename);
% Count the number of synonymous and nonsynonymous mutations in a BLAT file

[mismatchPositions, initialCodons, finalCodons] = parse_BLAT_alignment(filename);

synon = 0;
nonsynon = 0;
for n = 1:length(mismatchPositions);
    if translate_codon(initialCodons(n,1:3))==translate_codon(finalCodons(n,1:3));
       synon = synon + 1;
    else;
       nonsynon = nonsynon + 1;
    end;
end;


%%----------------------------------------------------------------------------------
function [synon, nonsynon] = count_polymorphisms(filename);
% Counts the number of synonymous and nonsynonymous mutations listed in a
% Seattle SNPs data file

synon = 0;
nonsynon = 0;
fid = fopen(filename);
while 1;
   currentline = fgetl(fid);
   if ~ischar(currentline), break, end;
   if ~isempty(strfind(currentline, 'SYNON')),
      synon = synon + 1;
   end;
   if ~isempty(strfind(currentline, 'NON-SYN')),
      nonsynon = nonsynon + 1;
   end;
end;
```

```
%%-----------------------------------------------------------------------------------
function amino_acid = translate_codon(codon)
% Returns the one letter amino acid symbol for the provided codon.
% The symbol ! is returned for a stop codon and - is returned for an
% unrecognized sequence.

genetic_code =
['FSYC';'FSYC';'LS!!';'LS!W';'LPHR';'LPHR';'LPQR';'LPQR';'ITNS';'ITNS';'ITKR'; ...
 'MTKR';'VADG';'VADG';'VAEG';'VAEG'];

p = [0, 0, 0];
codon = upper(codon);
for n = 1:3
    if codon(n) == 'U'
       codon(n) = 'T';
    end;
    k = find('TCAG' == codon(n));
    if isempty(k)
       amino_acid = '-';
       return;
    else
       p(n) = k;
    end;
end;
amino_acid = genetic_code(4*(p(1)-1)+p(3),p(2));


%%-----------------------------------------------------------------------------------
function [gadjusted, pvalue] = gtest(observed);
% Calculate the p-value for a 2x2 contingency table using the G-test.

rowsum = sum(observed,2);
colsum = sum(observed,1);
total = sum(rowsum);

expected = [rowsum(1)*colsum(1)/total,rowsum(1)*colsum(2)/total; ...
            rowsum(2)*colsum(1)/total,rowsum(2)*colsum(2)/total];
gmatrix = observed.*log(observed./expected);
for j=1:2, for k=1:2, if isnan(gmatrix(j,k)), gmatrix(j,k)=0; end; end; end;
ginitial = 2*sum(sum(gmatrix));
williams = 1 + (total/rowsum(1)+total/rowsum(2)-1)* ...
               (total/colsum(1)+total/colsum(2)-1)/(6*total);
gadjusted = ginitial / williams;

pvalue = 1 - chi2cdf(gadjusted,1);
```

Matlab Scripts for Problem 4

```
%%-------------------------------------------------------------------------------------
function mean_dist = mean_pairwise_distance(file);
% Calculate the mean distance between pairs of genes listed in the file

fid = fopen(file);

% Get length of genome from first line
cline = fgetl(fid);
str = regexp(cline, '(\d+)\.\.(\d+)', 'tokens', 'once');
genome_length = str2num(char(str{2})) + 1;

% Skip next two lines
fgetl(fid); fgetl(fid);

% Fill in a Nx2 array of start and end positions
genepos = [];
i = 1;
while 1;
   cline = fgetl(fid);
   if ~ischar(cline), break, end;
   str = regexp(cline, '(\d+)\.\.(\d+)', 'tokens', 'once');
   if length(str)~=2, continue, end;
   genepos(i,1) = str2num(char(str{1}));
   genepos(i,2) = str2num(char(str{2}));
   i = i + 1;
end;

% Calculate the sum of all pairwise distances
N = length(genepos);
total = 0;
for i = 1:(N-1);
   for j = (i+1):N;
      dist = compute_distance(genepos(i,:), genepos(j,:), genome_length);
      total = total + dist;
   end;
end;

% Divide by the number of pairwise comparisons
mean_dist = total/(0.5*N*(N-1));
```

```
%%---------------------------------------------------------------------------------
function [exp_dist, genome_length, mean_gene_length] = expected_distance(file);
% Calculate the average gene length and the expected distance between two
% randomly chosen genes in a circular genome using a simple uniform model

fid = fopen(file);

% Get length of genome from first line
cline = fgetl(fid);
str = regexp(cline, '(\d+)\.\.(\d+)', 'tokens', 'once');
genome_length = str2num(char(str{2})) + 1;

% Skip next two lines
fgetl(fid); fgetl(fid);

% Fill Nx2 array of start and end positions
genepos = [];
i = 1;
while 1;
   cline = fgetl(fid);
   if ~ischar(cline), break, end;
   str = regexp(cline, '(\d+)\.\.(\d+)', 'tokens', 'once');
   if length(str)~=2, continue, end;
   genepos(i,1) = str2num(char(str{1}));
   genepos(i,2) = str2num(char(str{2}));
   i = i + 1;
end;

% Calculate the sum of all gene lengths
N = length(genepos);
total = 0;
for i = 1:N;
   gene_length = abs(genepos(i,1)-genepos(i,2)) + 1;
   total = total + gene_length;
end;

% Divide by the number of genes
mean_gene_length = total/N;

% Calculate expected distance between two randomly chosen genes
exp_dist = (genome_length - 2 * mean_gene_length) / 4;
```

```
%%----------------------------------------------------------------------------------
function [mean_dist, stdev_dist, crit_val] = ...
   distance_distribution(file, groupsize, samples, threshold);
% Run a Monte-Carlo simulation to estimate statistical parameters of the mean
% pairwise distance for a specific number of arbitrarily chosen genes

fid = fopen(file);

% Get length of genome from first line
cline = fgetl(fid);
str = regexp(cline, '(\d+)\.\.(\d+)', 'tokens', 'once');
genome_length = str2num(char(str{2})) + 1;

% Skip next two lines
fgetl(fid); fgetl(fid);

% Fill in a Nx2 array of start and end positions
genepos = [];
i = 1;
while 1;
   cline = fgetl(fid);
   if ~ischar(cline), break, end;
   str = regexp(cline, '(\d+)\.\.(\d+)', 'tokens', 'once');
   if length(str)~=2, continue, end;
   genepos(i,1) = str2num(char(str{1}));
   genepos(i,2) = str2num(char(str{2}));
   i = i + 1;
end;

% Repeat Monte-Carlo simulation requested number of times
samplelist = [];
for n = 1:samples;
   samplelist(n) = group_mean_distance(genepos, genome_length, groupsize);
end;

% Compute mean and standard deviation of distribution
mean_dist = mean(samplelist);
stdev_dist = std(samplelist);

% Compute the critical value for significance at level 'threshold'
threshold_count = 0;
bin_number = 1;
[hist_counts, hist_bins] = hist(samplelist,100);
while (threshold_count/samples) < threshold;
   threshold_count = threshold_count + hist_counts(bin_number);
   bin_number = bin_number + 1;
end;
crit_val = hist_bins(bin_number - 1);

% Plot a histogram of the resulting distribution
hist(samplelist,100);
```

```
%%---------------------------------------------------------------------------------
function mean_dist = group_mean_distance(genepos, genome_length, groupsize)
% Helper function for calculating mean pairwise distance of a randomly selected
% group of genes.  Run repeatedly to generate distribution.

% Generate random list of genes with length groupsize and no repeats
genelist = [];
maxindex = length(genepos);
while length(genelist) < groupsize;
   newgene = ceil(maxindex*rand);
   if ~ismember(newgene,genelist);
      genelist(length(genelist)+1) = newgene;
   end;
end;

% Calculate the mean pairwise distance for this group of genes
total = 0;
for i = 1:(groupsize-1);
   for j = (i+1):groupsize;
      dist =
compute_distance(genepos(genelist(i),:),genepos(genelist(j),:),genome_length);
      total = total + dist;
   end;
end;
mean_dist = total / (0.5*groupsize*(groupsize-1));


%%---------------------------------------------------------------------------------
function dist = compute_distance(gene1, gene2, genome_length);
% Compute the minimum distance between two genes on a circular chromosome

dist = genome_length;
for m=1:2, for n=1:2;
   temp = abs(gene1(m)-gene2(n));
   dist = min([dist, temp, (genome_length - temp - 2)]);
end; end;
if overlapping(gene1, gene2);
   dist = 0;
end;

% Helper function to determine if genes overlap each other
function result = overlapping(a, b);
if (a(1) >= b(1) && a(1) <= b(2)) || ...
   (a(2) >= b(1) && a(1) <= b(2)) || ...
   (b(1) >= a(1) && b(1) <= a(2)) || ...
   (b(2) >= a(1) && b(2) <= a(2));
   result = true;
else;
   result = false;
end;
```