Forces in Molecular Evolution
There are many forces involved in molecular evolution.

1. Mutation can happen when mistakes are made in DNA replication, just because the DNA replication machinery isn’t 100% perfect.
2. Exposure to radiation, such as UV radiation,
3. Certain chemicals can induce mutations,
4. Viruses can induce mutations of the DNA (as well as insert genetic material).
5. Recombination is a genetic exchange between chromosomes or regions within a chromosome. During meiosis, genes and genomic DNA are shuffled between parent chromosomes.
6. Genetic drift is a stochastic process of changing allele frequencies over time due to random sampling of organisms.
7. Natural selection is the process where differences in phenotype can affect survival and reproduction rates of different individuals.
Because genetic drift is a stochastic process, it can be modeled as a “rate”. The rate of nucleotide substitutions $r$ can be expressed as

$$r = \frac{\mu}{2T}$$
In this expression, $\mu$ is the substitutions per site across the genome, and $T$ is the time of divergence of the two (extant) species to their common ancestor. The factor of two can be understood by the fact that it takes a total of $2T$ to go from $x_1$ to $x_2$, stopping at $x_a$ along the way. The mutations that occur over time separating $x_1$ and $x_2$ can be viewed as distributed over a time $2T$.

**Figure**: For two extant species $x_1$ and $x_2$ diverged for a time $T$ from a common ancestor $x_a$, the mutation rate can be expressed as a time $2T$ separating $x_1$ and $x_2$. 
In most organisms, the rate is observed to be about $10^{-9}$ to $10^{-8}$ mutations per generation. Some viruses have higher mutation rates $10^{-6}$ mutations per generation. The generation times of different species can also affect the nucleotide substitution rate $r$. Organisms with shorter generation times have more opportunities for meiosis per unit time.

$$r = \frac{\mu}{2T}$$
Multiple Sequence Alignment
A multiple sequence alignment is an alignment of more than 2 sequences. It turns out that this makes the problem of alignment much more complicated, and much more computationally expensive. Dynamic programming algorithm such as Smith-Waterman can be extended to higher dimensions, but at a significant computing cost. Therefore, numerous methods have been developed to make this task faster.
Dynamic Programming Approaches

Despite the computational cost of MSA by Dynamic Programming, there have been approaches to compute multiple sequence alignments using these approaches.

1. MSA
2. MULTALIN
Progressive alignments begin by performing pairwise alignments, by aligning each pair of sequences. Then it combines each pair and integrates them into a multiple sequence alignment. The different methods differ in their strategy to combine them into an overall multiple sequence alignment. Most of these methods are “greedy”, in that they combine the most similar pairs first, and proceed by fitting the less similar pairs into the MSA. The following programs use progressive alignment:

1. T-Coffee
2. ClustalW
3. PSAlign
Iterative Alignment

Iterative Alignment is another approach that improves upon the progressive alignment because it starts with a progressive alignment and then iterates to incrementally improve the alignment with each iteration. The following programs use iterative alignment:

1. CHAOS/DIALIGN
2. MUSCLE
Specialized multiple sequence alignment approaches have been developed for aligning complete genomes, to overcome the challenges associated with aligning such long sequences. The following programs have been developed for aligning full genomes:

1. MLAGAN (using LAGAN)
2. MULTIZ (using BLASTZ)
3. MUSCLE
4. MUMmer
>D. melanogaster
--------------------------------------------------
---------------------------------MSEARNLFTTFGILAII
FFLYLIYA--------------------------VL----------------

>D. sechellia
--------------------------------------------------
---------------------------------MSEARNLFTTFGILAII
FFLYLIYAPAAKSESIMNEAKSLFTTFILAFLLFLLYAFYEAADF

>D. pseudoobscura
MSEAKNLMTTFGILAFLLCFLYLIYASNNSKRWPFTFCGEAEFRSENSESQ
LLRAFSYERLEQCPNKYPKQPTTTTTTKPIKMNEARSLFTTFILAFLL
FFLYAFYEA------------------------AF----------------

>D. busckii
--------------------------------------------------
---------------------------------MNEAKSLVTTFILAFLL
FFLYAFYEA------------------------AF----------------
CLUSTAL W (1.83) multiple sequence alignment

D. melanogaster
D. sechellia
D. pseudoobscura
D. busckii

D. melanogaster
D. sechellia
D. pseudoobscura
D. busckii

*:**:.*.*** ***:***:** :*

D. melanogaster
D. sechellia
D. pseudoobscura
D. busckii

*:**:.*.*** ***:***:** **

D. melanogaster
D. sechellia
D. pseudoobscura
D. busckii

*:**:.*.*** ***:***:** **

D. melanogaster
D. sechellia
D. pseudoobscura
D. busckii

*:**:.*.*** ***:***:** **

D. melanogaster
D. sechellia
D. pseudoobscura
D. busckii

*:**:.*.*** ***:***:** **

D. melanogaster
D. sechellia
D. pseudoobscura
D. busckii
MAF File

## maf version=1 scoring=roast.v3.3
a score=49441.000000
s hg38.chr22 10514742 28 + 50818468 acagaatggattatttggaacagaataga
s panTro4.chrUn_GL393523 96163 28 + 405060 agacaatggattagtggaacagaagaga
i panTro4.chrUn_GL393523 C 0 C 0
s ponAbe2.chrUn 66608224 28 - 72422247 aaagaatggattagtggaacagaataga
i ponAbe2.chrUn C 0 C 0
s nomLeu3.chr6 67506008 28 - 121039945 acagaatagattagtgggaacagaataga
i nomLeu3.chr6 C 0 C 0
s rheMac3.chr7 24251349 14 + 170124641 ----------------tggaacagaataga
i rheMac3.chr7 C 0 C 0
s macFas5.chr7 24018429 14 + 171882078 ----------------tggaacagaataga
i macFas5.chr7 C 0 C 0
s chlSab2.chr26 21952261 14 - 58131712 ----------------tggaacagaataga
i chlSab2.chr26 C 0 C 0
s calJac3.chr10 24187336 28 + 132174527 acagaataagaccagtggatcagaataga
i calJac3.chr10 C 0 C 0
s saiBol1.JH378136 10582894 28 - 21366645 acataatagactagtggatcagaataga
i saiBol1.JH378136 C 0 C 0
s eptFus1.JH977629 13032669 12 + 23049436 -----------------gaacaaagcaga
i eptFus1.JH977629 C 0 C 0
e odoRosDiv1.KB229735 169922 2861 + 556676 I
e felCat8.chrB3 91175386 3552 - 148068395 I
e otoGar3.GL873530 132194 0 + 36342412 C
e speTri2.JH393281 9424515 97 + 41493964 I
e myoLuc2.GL429790 1333875 0 - 11218282 C
e myoDav1.KB110799 1333834 0 + 1195772 C
e pteAle1.KB031042 11269154 1770 - 35143243 I
e musFur1.GL896926 13230044 2877 + 15480060 I
e canFam3.chr30 13413941 3281 + 40214260 I
e cerSim1.JH767728 28819459 183 + 61284144 I
e equCab2.chr1 43185635 316 - 185838109 I
e orcOrc1.KB316861 20719861 245 - 22150888 I
e camFer1.KB017752 865624 507 + 1978457 I
The s lines contain 5 fields after the s at the beginning of the line.

1. First, the source of the column usually consists of a genome assembly version, and chromosome name separated by a dot “.”.

2. Next, is the start position of the sequence in that assembly/chromosome.

3. This is followed by the size of the sequence from the species, which may of course vary from species to species.

4. The next field is a strand, with “+” or “-”, indicating what strand from the species’ chromosome the sequence was taken from.

5. The next field is the size of the source, which is typically the length of the chromosome in basepairs from which the sequence was extracted.

6. Lastly, the sequence itself is included in the alignment block.
Representing Phylogenetic Trees
A phylogenetic tree is often a “binary tree” where each branch point goes from one to two branches. The junction points where the branching takes place are called “internal nodes”. One way of representing a tree is with nested parentheses corresponding to branching. Consider the following example

$$(((A,B),(C,D));$$
```python
>>> from Bio import Phylo
>>> tree = Phylo.read("tree.txt","newick")
>>> Phylo.draw_ascii(tree)

\n
```

```
```
```
```
```
```

A

B

C

D

```
This particular tree has all the characters at the same level, and does not include any distance or “branch length” information. Using real biological sequences, we can compute the distances along each branch to get a more informative tree. For example, we can download 18s rRNA sequences from NCBI Nucleotide. Using clustalw, we can compute a multiple sequence alignment, and produce a phylogenetic tree. In this case, the command

$ clustalw -infile=18s_rRNA.fasta -type=DNA -outfile=18s_rRNA.aln
This particular tree has all the characters at the same level, and does not include any distance or “branch length” information. Using real biological sequences, we can compute the distances along each branch to get a more informative tree. For example, we can download 18s rRNA sequences from NCBI Nucleotide. Using clustalw, we can compute a multiple sequence alignment, and produce a phylogenetic tree. In this case, the command

$ clustalw -infile=18s_rRNA.fasta -type=DNA -outfile=18s_rRNA.aln

This command will produce the output file 18s_rRNA.dnd, which is a tree in newick tree format. The file contains the following information

(fly:0.16718,(mouse:0.00452,human:0.00351):0.05186,chicken:0.24654);
The command to draw a tree image is simply `Phylo.draw`, which will allow the user to save the image.

```python
>>> from Bio import Phylo
>>> tree = Phylo.read('18s_rRNA.dnd','newick')
>>> Phylo.draw(tree)
```
Figure: A phylogenetic tree computed with Clustalw for 18s rRNA sequences for *Drosophila melanogaster*, *Homo sapiens*, *Mus musculus*, and *Gallus gallus*.
Phylogenetic trees are often computed as binary trees with branch lengths that optimally match the pair-wise distances between species. In order to compute a phylogenetic tree, we need a way of defining this distance. One strategy developed by Feng and Doolittle is to compute a distance from the alignment scores computed from pair-wise alignments. This distance is defined as

$$D_{ij} = -\ln S_{\text{eff}}(i, j)$$

so that pairs of sequences $(i, j)$ that have high scores will have a small distance between them. The effective score $S_{\text{eff}}(i, j)$ is defined as

$$S_{\text{eff}}(i, j) = \frac{S_{\text{real}}(ij) - S_{\text{rand}}(ij)}{S_{\text{iden}}(ij) - S_{\text{rand}}(ij)} \times 100$$
\[ S_{\text{eff}}(i,j) = \frac{S_{\text{real}}(ij) - S_{\text{rand}}(ij)}{S_{\text{iden}}(ij) - S_{\text{rand}}(ij)} \times 100 \]

Where in this expression, \( S_{\text{real}}(ij) \) is the observed pairwise similarity between sequences from species \( i \) and \( j \). The value \( S_{\text{iden}}(ij) \) is the average of the two scores when you align species \( i \) and \( j \) to themselves, which represents the score corresponding to aligning “identical” sequences, the maximum possible score one could get. \( S_{\text{rand}}(ij) \) is the average pairwise similarity between randomized, or shuffled, versions of the sequences from species \( i \) and \( j \). After this normalization, the score \( S_{\text{eff}}(i,j) \) ranges from 0 to 100.