Nathan Newbury **HW 8** **Writeup**

Equation for estimating pairs

We need to estimate the stationary probabilities and the rate parameters (gtr_rates). In order to calculate the rate parameters, we need to find an estimate of the rate matrix R. For a pair of sequences we first calculate our probability matrix P by counting the number of transition instances.

```
    for i in range(len(s)):
        if(r[i] == s[i]):
            P[PROB_KEYS.index(r[i])][PROB_KEYS.index(r[i])] += 2
            pis[PROB_KEYS.index(r[i])] += 2
        else:
            try:
                ind = Orderedratekeys.index(r[i] + s[i])
            except:
                ind = Orderedratekeys.index(s[i] + r[i])
            #AC and CA
            if ind == 0:
                P[1][0] +=1
                P[0][1] +=1
```

In my algorithm, if the sequence letters were the same I add two to my count at that index in my probability matrix (for both directions). In the case where both letters are different, I add 1 to my count in two indexes of my probability matrix (for both orders of letters). In this way, P is symmetrical and P[n][l] = P[l][n]. I then normalize my P matrix so the rows and columns sum to 1.

```
row_sum = P.sum(axis=1)
    P = P/row_sum[:,np.newaxis]
```

We can find R by taking the inverse of $P = e^{Rd}$. So $R = log(P)/d$.

```
    R = sp.linalg.logm(P)/d + np.exp(-15)
```

To find our stationary probabilities we count the total appearance of letters A, C, G,T and normalize. With the R matrix entries and the stationary probabilities we can easily solve for our gtr rates ($\theta_1\theta_2\theta_3\theta_4\theta_5\theta_6$). (These are different symbols than class see the diagram below).

REV	A	C	G	T
A	$-\Sigma$	$\theta_1\pi_C$	$\theta_2\pi_G$	$\theta_3\pi_T$
C	$\theta_1\pi_A$	$-\Sigma$	$\theta_4\pi_G$	$\theta_5\pi_T$
G	$\theta_2\pi_A$	$\theta_4\pi_C$	$-\Sigma$	$\theta_6\pi_T$
T	$\theta_3\pi_A$	$\theta_5\pi_C$	$\theta_6\pi_G$	$-\Sigma$

```
    gtr_rates['CT'] = R[3][1]/pis[1]
    gtr_rates['AT'] = R[3][0]/pis[0]
    gtr_rates['GT'] = R[3][2]/pis[2]
    gtr_rates['AC'] = R[1][0]/pis[0]
    gtr_rates['CG'] = R[2][1]/pis[1]
    gtr_rates['AG'] = R[2][0]/pis[0]
```

For example, θ_1 = R[C,A]/π_a = gtr_rates['AC'].

Algorithm for estimating from multiple sequences

To estimate the parameters for a tree we iterate through each pair of leaves. For each pair, I call gtr_params_pair which estimates the parameters for that pair.

```
    for node1 in tree.traverse_postorder():
        if node1.is_leaf():
            for node2 in tree.traverse_postorder():
                if node2.is_leaf() and node1 != node2:
                    d = tree.distance_between(node1,node2)
                    temp_probs,temp_rates =
gtr_params_pair(seqs[node1.get_label()],seqs[node2.get_label()],d)
```

I then take the weighted average of these parameters to find the total estimate of my rate parameters (gtr_rates) and stationary probabilities(π). I use 1 / variance(stationary probabilities) for each pair of nodes as my weight. This way for a smaller variance(all letters appear equally) I find a larger weight. I don't weight my stationary probability estimates with this because that would bias it(these are close anyways).

```
weight = 1/(np.var(list(temp_probs.values())) + .00001)
                    for key, value in temp_probs.items():
                        gtr_probs[key] += value

                    for key, value in temp_rates.items():
                        gtr_rates[key] += weight*value
```

Finally, I normalize my rate parameters and my stationary probabilities.

```
# normalize
    norm = gtr_rates['GT']
    for key, value in gtr_rates.items():
        gtr_rates[key] = value/norm

    #normalize
    probsum = sum(list(gtr_probs.values()))
    for key, value in gtr_probs.items():
        gtr_probs [key] = value/probsum
```