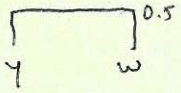
Tyler Bradley | ECES T-580 | HW 2 | 2/23/19

1) a)

X: TACCCGAT
Y: TAAACGAT
Z: AAAACGCG
W: AAAACGAT

Hamming Distance Matrix

	X	Y	Z	W
X	0	2	5	3
Y		0	3	1
Z			0	2
W				0



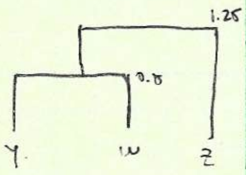
	X	Z	YW
X	0	5	2.5
Z		0	2.5
YW			0

$$d(yw,x) = \frac{d(y,x) + d(w,x)}{2} = \frac{2+3}{2}$$

$= 2.5$

$$d(yw,z) = \frac{d(y,z) + d(w,z)}{2} = \frac{3+2}{2}$$

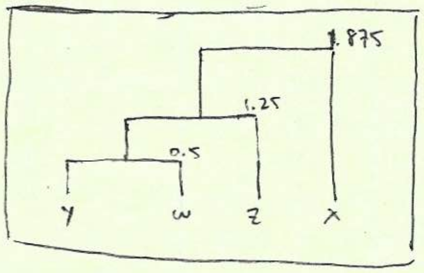
$= 2.5$



	X	ZYW
X	0	3.75
ZYW		0

$$d(zyw,x) = \frac{d(z,x) + d(yw,x)}{2} = \frac{5+2.5}{2}$$

$= 3.75$

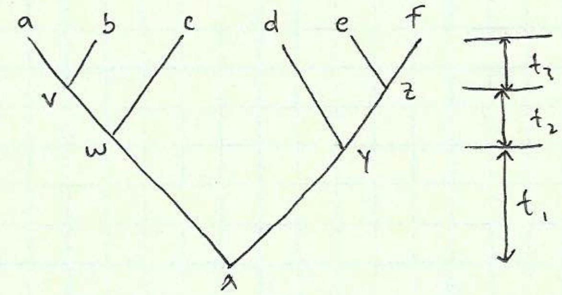


<u>This tree is not ultrametric</u>

* See part b) on bootstrapping in the attached matlab code

Tyler Bradley | ECES T580 Hw 2 | 2/23/19

② Show the expression for obtaining $Pr(a,b,c,d,e,f \mid T,M)$



$P_{xy}(t_1)$ ~ prob of going from x to y in time, t_1

$$\boxed{Pr(a,b,c,d,e,f \mid T,M) = \sum_x \sum_y \sum_z \sum_w \sum_v pr(x)\Big[P_{xy}(t_1)\cdot P_{yd}(t_2+t_3)\cdot P_{yz}(t_2)\cdot P_{ze}(t_3)\cdot P_{zf}(t_3)\cdot P_{xw}(t_1)\cdot P_{wc}(t_2+t_3)\cdot P_{wv}(t_2)\cdot P_{vb}(t_3)\cdot P_{va}(t_3)\Big]}$$

Homework 2

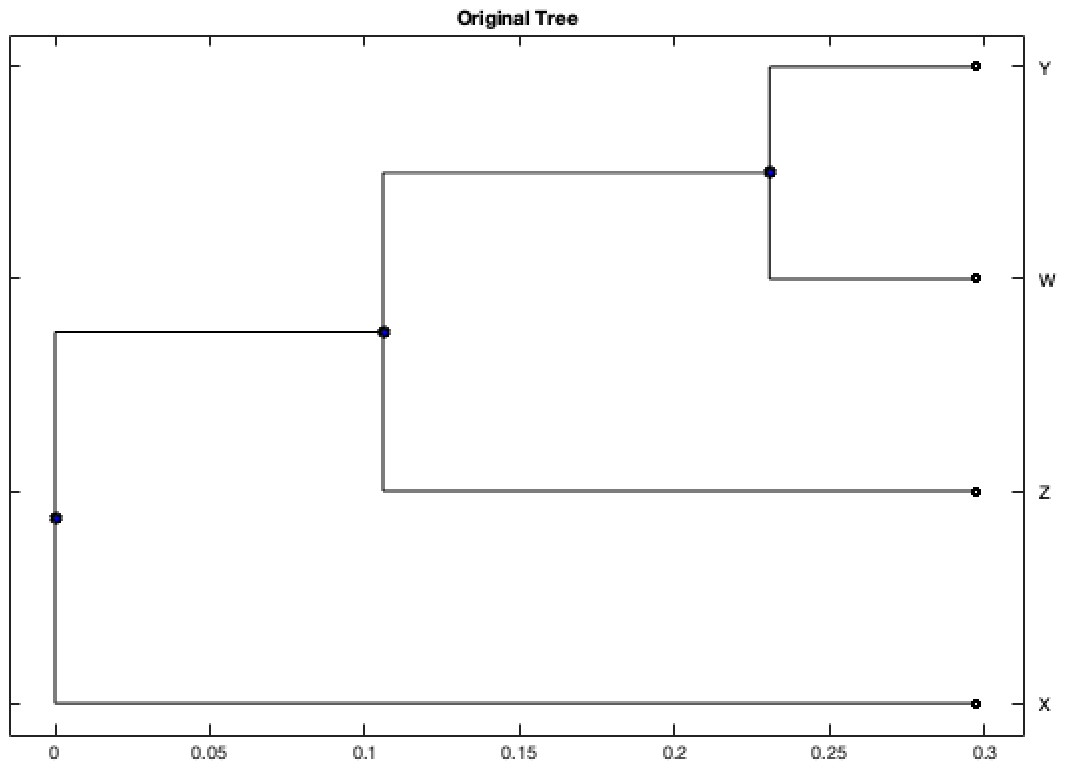
Table of Contents

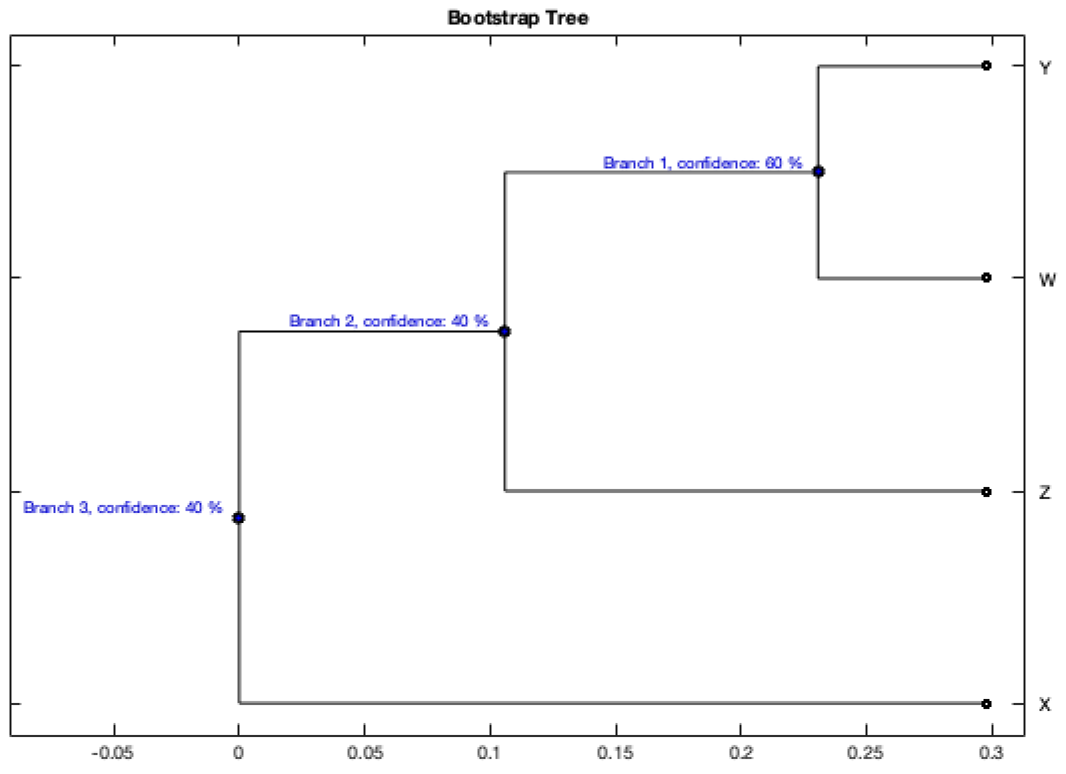
Tyler Bradley

Question 1 - Part B (Bootstrapping)

```
value1 = {'X', 'Y', 'Z', 'W'};
value2 = {'TACCCGAT', 'TAAACGAT', 'AAAACGCG', 'AAAACGAT'};

tree_seqs = struct('Header', value1, 'Sequence', value2);

% original tree
seq_dist = seqpdist(tree_seqs);
tree_orig = seqlinkage(seq_dist, 'average', tree_seqs);
plot(tree_orig)
title("Original Tree")

% Create empty vectors for output and data length
num_boot = 5;
seq_len = length(tree_seqs(1).Sequence);
num_seqs = length(tree_seqs);
boots = cell(num_boot, 1);
boots_dist = cell(num_boot, 1);
boots_trees = cell(num_boot, 1);

% Create the bootstrap trees
for i = 1:num_boot
    idx = randsample(seq_len, seq_len, 'true');
    for j = 1:num_seqs
        boot_seq(j).Header = tree_seqs(j).Header;
        boot_seq(j).Sequence = tree_seqs(j).Sequence(idx);
    end
    boots{i} = boot_seq;
    boot_dist = seqpdist(boot_seq);
    boots_dist{i} = boot_dist;
    boot_tree = seqlinkage(boot_dist, 'average', boot_seq);
    boots_trees{i} = boot_tree;
end

% Find the pointers and the leaves that each node is an ancestor of
% for the original tree
for i = 1:num_seqs-1
    node = i + num_seqs;
    sub_tree = subtree(tree_orig, node);
    orig_pointers{i} = getcanonical(sub_tree);
    orig_species{i} = sort(get(sub_tree, "LeafNames"));
```

```
end

% Find the pointers and the leaves that each node is an ancestor of
% for each of the boostrap trees
for j = 1:num_boot
    for i = 1:num_seqs-1
      node = i + num_seqs;
      sub_tree = subtree(boots_trees{j}, node);
      boot_pointers{i,j} = getcanonical(sub_tree);
      boot_species{i, j} = sort(get(sub_tree, "LeafNames"));
    end

end

% Finding the count of bootstrap trees that match the original tree
% for both the decesdents of a given node and the species inside of it
match_count = repelem(0, num_seqs-1);
for i = 1:num_seqs-1
    for j = 1:num_boot
       if isequal(orig_pointers{i},boot_pointers{i,j})
            if isequal(orig_species{i},boot_species{i,j})
                match_count(i) = match_count(i) + 1;
            end
        end
    end
end

% Calculate the confidence for each of the nodes
node_conf = match_count/num_boot;

% extract the major structure of the original tree
[ptrs,dist,names] = get(tree_orig,'POINTERS','DISTANCES','NODENAMES');

% Add the confidence for each branch to the tree names
for i = 1:num_seqs -1  % for every branch
    branch_ptr = i + num_seqs;
    names{branch_ptr} = [names{branch_ptr} ', confidence: '
 num2str(100*node_conf(i)) ' %'];
end

% create a new phylogenetic tree and plot it with branch confidences
conf_tree = phytree(ptrs,dist,names);
plot(conf_tree, "BranchLabels", true)
title("Bootstrap Tree")
```


Original Tree
Y
W
Z
X
0
0.05
0.1
0.15
0.2
0.25
0.3

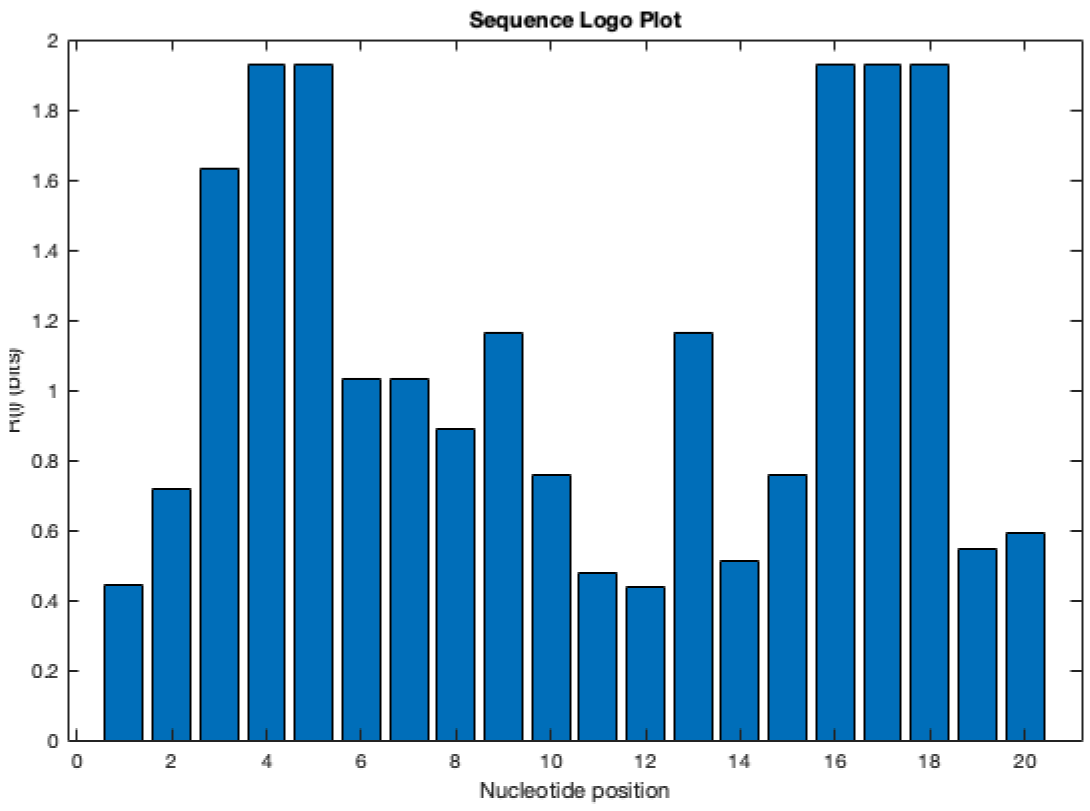

Bootstrap Tree
Branch 1, confidence: 60 %
Branch 2, confidence: 40 %
Branch 3, confidence: 40 %
Y
W
Z
X
-0.05
0
0.05
0.1
0.15
0.2
0.25
0.3

Question 3

Go to a database of E. Coli binding sites: http://arep.med.harvard.edu/ecoli_matrices/. Click on the lexA link. Then, the alignment link contains the list of binding sites. Unfortunately between each sequence, there are notes about the sequence (you will need to strip these when importing the sequence into Matlab, hint: fastaread).

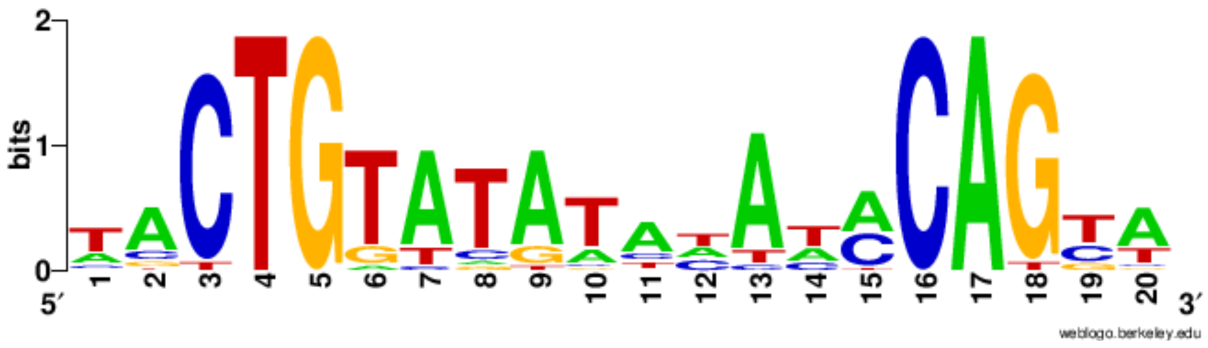
```
lexA = fastaread("hw2-files/lexA.fasta");

% Find R_sequence(l) for this alignment
len_seq = length(lexA(1).Sequence);
num_seqs = length(lexA);

% create empty vectors for output values
prob_A = repelem(0, len_seq);
prob_C = repelem(0, len_seq);
prob_G = repelem(0, len_seq);
prob_T = repelem(0, len_seq);
H = repelem(0, len_seq);
b_table = zeros(len_seq, 4);

for i = 1:len_seq
  count_A = 0;
  count_C = 0;
  count_G = 0;
  count_T = 0;

  for j = 1:num_seqs
      if lexA(j).Sequence(i) == "a"
          count_A = count_A + 1;
      elseif lexA(j).Sequence(i) == "c"
          count_C = count_C + 1;
      elseif lexA(j).Sequence(i) == "g"
          count_G = count_G + 1;
      elseif lexA(j).Sequence(i) == "t"
          count_T = count_T + 1;
      end
  end

  prob_A(i) = count_A/num_seqs + 0.0000001;
  prob_C(i) = count_C/num_seqs + 0.0000001;
  prob_G(i) = count_G/num_seqs + 0.0000001;
  prob_T(i) = count_T/num_seqs + 0.0000001;
  H(i) = -1*(prob_A(i)*log2(prob_A(i)) + prob_C(i)*log2(prob_C(i))
 + ...
             prob_G(i)*log2(prob_G(i)) + prob_T(i)*log2(prob_T(i)));

  R_seq(i) = 2 - H(i);

  b_table(i, 1) = prob_A(i)*R_seq(i);
  b_table(i, 2) = prob_C(i)*R_seq(i);
  b_table(i, 3) = prob_G(i)*R_seq(i);
  b_table(i, 4) = prob_T(i)*R_seq(i);
```

```
end

%R_seq = 2 - H;

bar(R_seq)
ylabel("R(l) (bits)")
title("Sequence Logo Plot")
xlabel("Nucleotide position")

rownames = {'1', '2', '3', '4', '5', '6', '7', '8', '9', '10', ...
    '11', '12', '13', '14', '15', '16', '17', '18', '19', '20'};
array2table(b_table, "VariableNames",
 {'A', 'C', 'G', 'T'}, "RowNames", rownames)

% The plot created above is fairly similar to the one created using
 the
% online tool. Bases 3, 4, 5 and 16, 17, 18 are the highest groupings
% of R values in both of the plots. The e(n) value in the online tool
% (set to zero here) likely corrects for potential bias that may be
% introduced when only a small amount of data is used to determine the
% entropy in a sequence alignment.

% See online version on next page


ans =

  20×4 table

                 A              C              G              T
             __________    __________    __________    __________


    1           0.11588       0.04635      0.023175       0.23175
    2            0.4913      0.075585      0.075585      0.037792
    3        1.6328e-07         1.461    1.6328e-07      0.085939
    4        1.9261e-07    1.9261e-07    1.9261e-07        1.8247
    5        1.9261e-07    1.9261e-07        1.8247    1.9261e-07
    6           0.05428    1.0313e-07       0.16284       0.75992
    7           0.75992       0.05428    1.0313e-07       0.16284
    8          0.046649      0.093298      0.046649       0.65309
    9           0.91997    1.1653e-07       0.12266      0.061331
    10          0.11965      0.039883      0.039883       0.51848
    11          0.27731       0.07563       0.02521       0.07563
    12          0.11507       0.13809    4.3727e-08        0.1611
    13          0.91997      0.061331    1.1653e-07       0.12266
    14          0.13402       0.10722    5.0927e-08       0.24124
    15          0.39911       0.27938    7.5832e-08      0.039911
    16       1.9261e-07        1.8247    1.9261e-07    1.9261e-07
    17           1.8247    1.9261e-07    1.9261e-07    1.9261e-07
    18       1.9261e-07    1.9261e-07        1.8247    1.9261e-07
    19       5.4375e-08       0.17171      0.085855       0.25757
    20           0.3413      0.031027      0.031027       0.15513
```



Published with MATLAB® R2018b

Homework 2 – Question 3
Sequence Logo generated from http://weblogo.berkeley.edu


bits
5′
3′
weblogo.berkeley.edu