UPGMA—**U**nweighted **P**air-**G**roup **M**ethod using arithmetic **A**verages

Principle: the two sequences i and j with the shortest distance d_{ij} must have have been the last to diverge; their branches the same length, at *half* their distance

$$d_{XY} = \frac{1}{N_X N_Y} \sum_{i \in X, j \in Y} d_{ij}$$

UPGMA—Unweighted Pair-Group Method using arithmetic Averages

Principle: the two sequences i and j with the shortest distance d_{ij} must have have been the last to diverge; their branches the same length, at *half* their distance

$$d_{XY} = \frac{1}{N_X N_Y} \sum_{i \in X, j \in Y} d_{ij}$$

X and Y are clusters containing 1 or more sequences; N_X and N_Y are their sizes

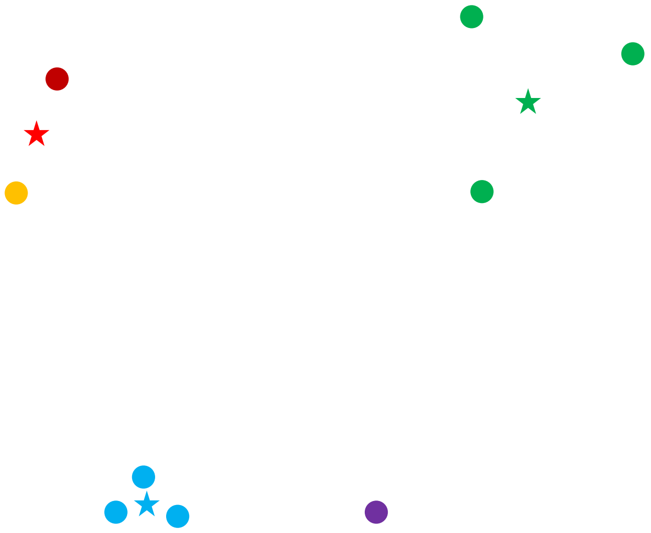
Sum the distances between all pairs of sequences, one from X and one from Y; calculate their arithmetic average

UPGMA—Unweighted Pair-Group Method using arithmetic Averages

Distances between clusters can be determined based on the distances of the two which are merged (computationally more efficient)

$$d_{ZW} = \frac{N_X d_{XW} + N_Y d_{YW}}{N_X + N_Y}$$

Merge X and Y into new cluster Z,
based on X and Y's distances to others (W)

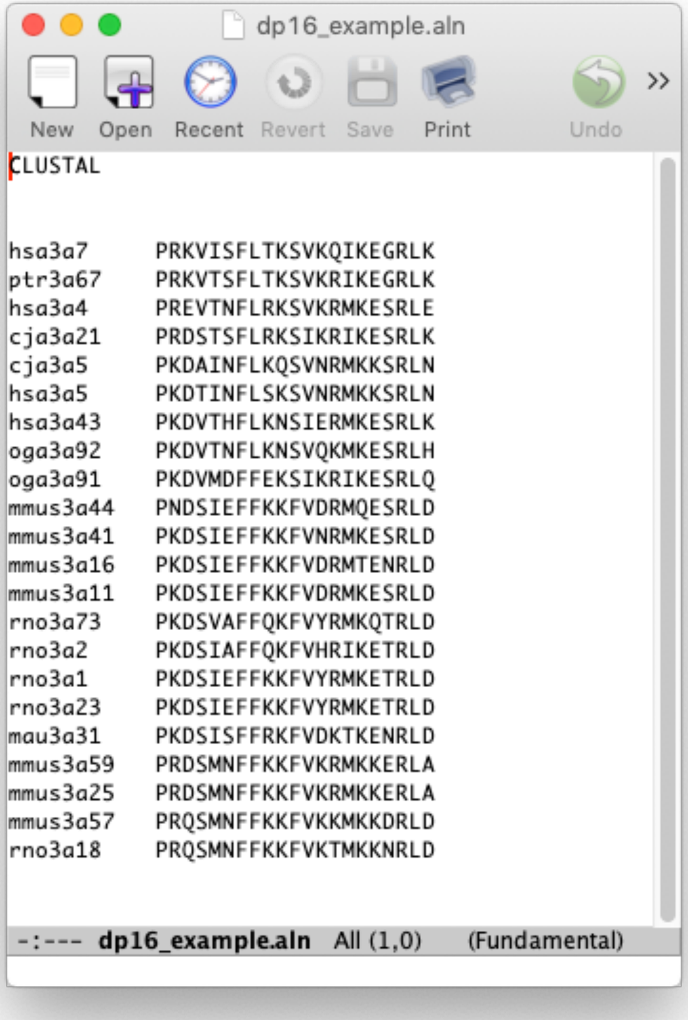


From phylo.py (also in guide.py)

```
""" ----------------------------------------------------------------------
   Methods for generating a single tree by clustering, here UPGMA Zvelebil and Baum p. 278
   ----------------------------------------------------------------------"""

def runUPGMA(aln, measure, absoluteDistances=False):
    """ Generate an ultra-metric, bifurcating, rooted tree from an alignment based on pairwise distances.
        Use specified distance metric (see sequence.calcDistances).
        If absoluteDistances is True, the tree will be assigned the total distance from provided species.
        Otherwise, the relative addition at each path will be assigned."""
    D = {}
    N = {}  # The number of sequences in each node
    M = aln.calcDistances(measure)  # determine all pairwise distances
    nodes = [PhyloNode(label=seq.name) for seq in aln.seqs]  # construct all leaf nodes
    """ For each node-pair, assign the distance between them. """
    for i in range(len(nodes)):
        nodes[i].sequence = aln.seqs[i]
        nodes[i].dist = 0.0
        N[nodes[i]] = 1  # each cluster contains a single sequence
        for j in range(0, i):
            D[frozenset([nodes[i], nodes[j]])] = M[i, j]
```

```python
""" Treat each node as a cluster, until there is only one cluster left, find the *closest*
    pair of clusters, and merge that pair into a new cluster (to replace the two that merged).
    In each case, the new cluster is represented by the (phylo)node that is formed. """
while len(N) > 1:  # N will contain all "live" clusters, to be reduced to a single below
    closest_pair = (None, None)  # The two nodes that are closest to one another according to supplied metric
    closest_dist = None  # The distance between them
    for pair in D:  # check all pairs which should be merged
        dist = D[pair]
        if closest_dist == None or dist < closest_dist:
            closest_dist = dist
            closest_pair = list(pair)
    # So we know the closest, now we need to merge...
    x = closest_pair[0]  # See Zvelebil and Baum p. 278 for notation
    y = closest_pair[1]
    z = PhyloNode()  # create a new node for the cluster z
    z.dist = D.pop(frozenset([x, y])) / 2.0  # assign the absolute distance, change to relative distance later
    Nx = N.pop(x)  # find number of sequences in x, remove the cluster from list N
    Ny = N.pop(y)  # find number of sequences in y, remove the cluster from list N
    dz = {}  # new distances to cluster z
    for w in N:  # for each node w ...
        # we will merge x and y into a new cluster z, so need to consider w (which is not x or y)
        dxw = D.pop(frozenset([x, w]))  # retrieve and remove distance from D: x to w
        dyw = D.pop(frozenset([y, w]))  # retrieve and remove distance from D: y to w
        dz[w] = (Nx * dxw + Ny * dyw) / (Nx + Ny)  # distance: z to w
    N[z] = Nx + Ny  # total number of sequences in new cluster, insert new cluster in list N
    for w in dz:  # we have to run through the nodes again, now not including the removed x and y
        D[frozenset([z, w])] = dz[w]  # for each "other" cluster, update distance per EQ8.16 (Z&B p. 278)
    x.parent = z
    y.parent = z
    z.children = [x, y]
    nodes.append(z)
```

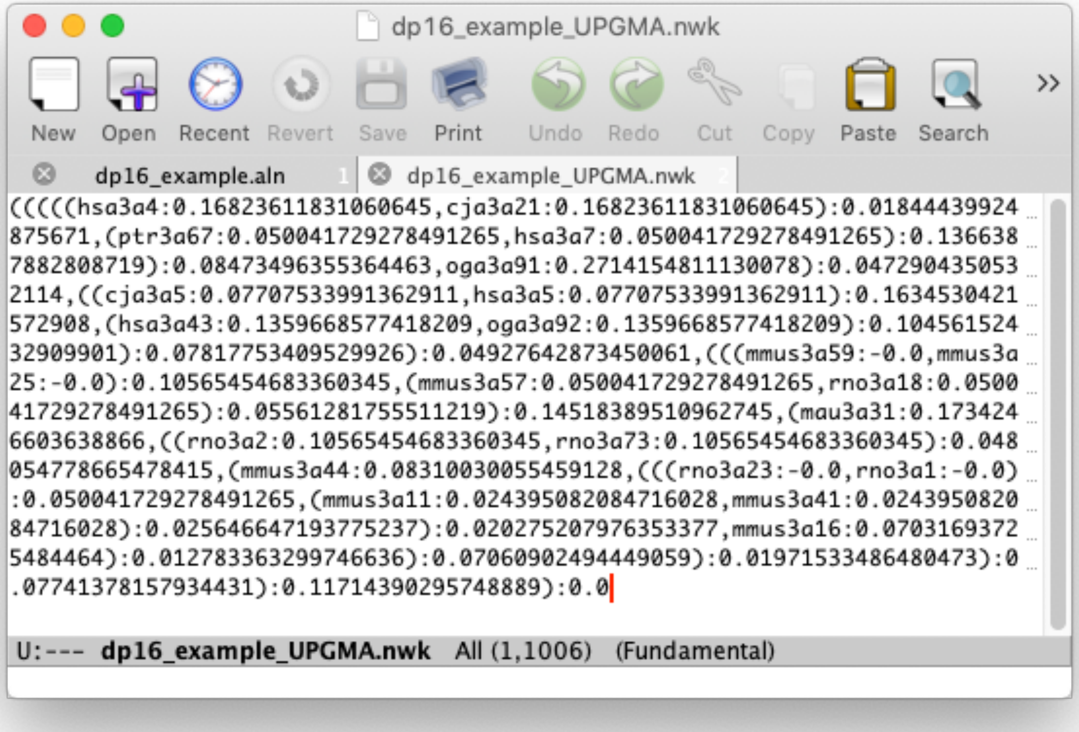


```
CLUSTAL


hsa3a7      PRKVISFLTKSVKQIKEGRLK
ptr3a67     PRKVTSFLTKSVKRIKEGRLK
hsa3a4      PREVTNFLRKSVKRMKESRLE
cja3a21     PRDSTSFLRKSIKRIKESRLK
cja3a5      PKDAINFLKQSVNRMKKSRLN
hsa3a5      PKDTINFLSKSVNRMKKSRLN
hsa3a43     PKDVTHFLKNSIERMKESRLK
oga3a92     PKDVTNFLKNSVQKMKESRLH
oga3a91     PKDVMDFFEKSIKRIKESRLQ
mmus3a44    PNDSIEFFKKFVDRMQESRLD
mmus3a41    PKDSIEFFKKFVNRMKESRLD
mmus3a16    PKDSIEFFKKFVDRMTENRLD
mmus3a11    PKDSIEFFKKFVDRMKESRLD
rno3a73     PKDSVAFFQKFVYRMKQTRLD
rno3a2      PKDSIAFFQKFVHRIKETRLD
rno3a1      PKDSIEFFKKFVYRMKETRLD
rno3a23     PKDSIEFFKKFVYRMKETRLD
mau3a31     PKDSISFFRKFVDKTKENRLD
mmus3a59    PRDSMNFFKKFVKRMKKERLA
mmus3a25    PRDSMNFFKKFVKRMKKERLA
mmus3a57    PRQSMNFFKKFVKKMKKDRLD
rno3a18     PRQSMNFFKKFVKTMKKNRLD
```

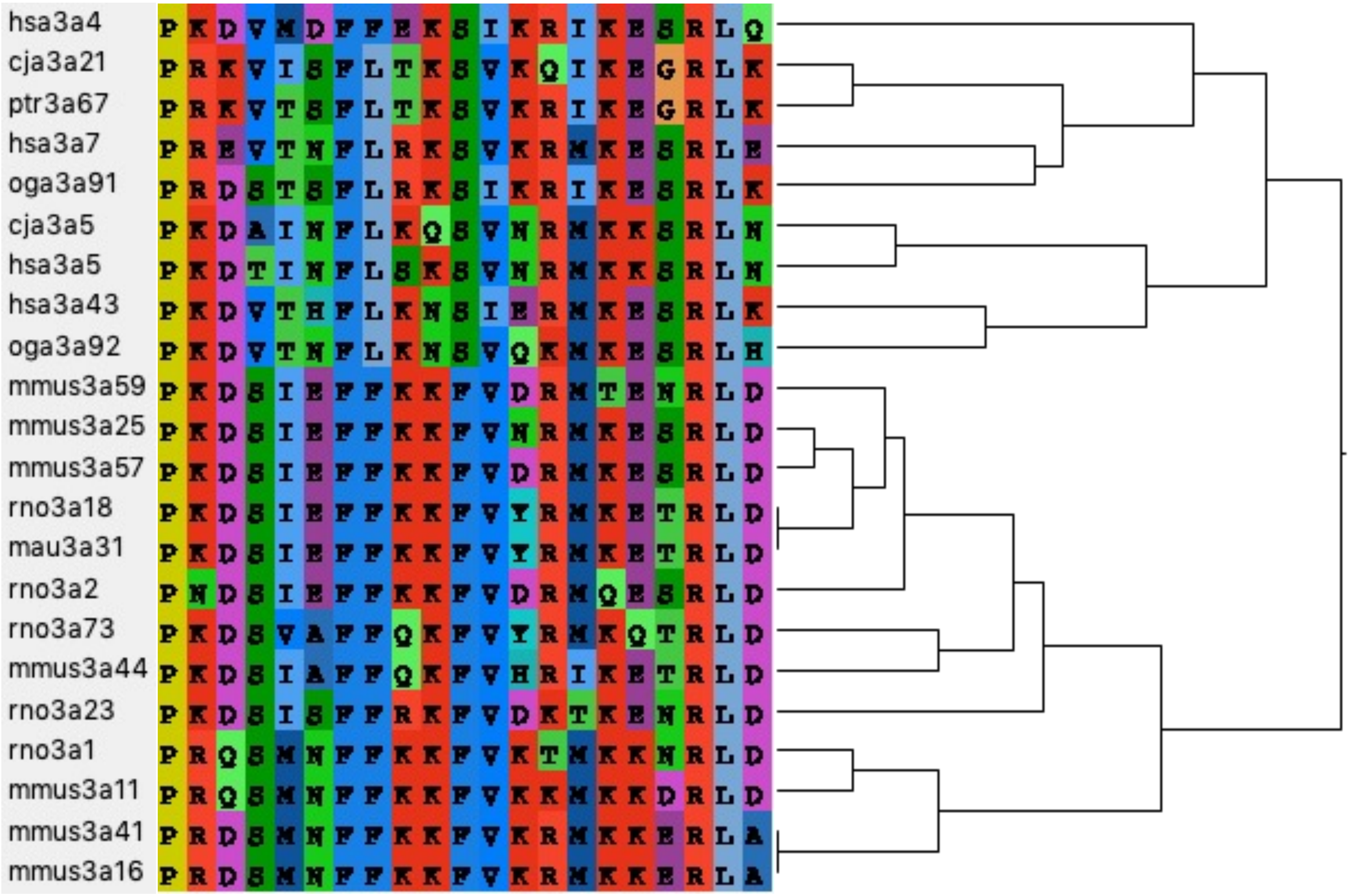
```
-:---  dp16_example.aln   All (1,0)      (Fundamental)
```

```
aln = sequence.readClustalFile('dp16_example.aln',
sequence.Protein_Alphabet)
tree = phylo.runUPGMA(aln, "poisson")
phylo.writeNewickFile('dp16_example_UPGMA.nwk', tree)
```



```
(((((hsa3a4:0.16823611831060645,cja3a21:0.16823611831060645):0.01844439924
875671,(ptr3a67:0.050041729278491265,hsa3a7:0.050041729278491265):0.136638
7882808719):0.08473496355364463,oga3a91:0.2714154811130078):0.047290435053
2114,((cja3a5:0.07707533991362911,hsa3a5:0.07707533991362911):0.1634530421
572908,(hsa3a43:0.1359668577418209,oga3a92:0.1359668577418209):0.104561524
32909901):0.07817753409529926):0.04927642873450061,(((mmus3a59:-0.0,mmus3a
25:-0.0):0.10565454683360345,(mmus3a57:0.050041729278491265,rno3a18:0.0500
41729278491265):0.05561281755511219):0.14518389510962745,(mau3a31:0.173424
6603638866,((rno3a2:0.10565454683360345,rno3a73:0.10565454683360345):0.048
054778665478415,(mmus3a44:0.08310030055459128,(((rno3a23:-0.0,rno3a1:-0.0)
:0.050041729278491265,(mmus3a11:0.024395082084716028,mmus3a41:0.0243950820
84716028):0.025646647193775237):0.020275207976353377,mmus3a16:0.0703169372
5484464):0.012783363299746636):0.07060902494449059):0.01971533486480473):0
.07741378157934431):0.11714390295748889):0.0
```

```
U:---  dp16_example_UPGMA.nwk   All (1,1006)   (Fundamental)
```



```
hsa3a4    P K D V M D F F E K S I K R I K E S R L Q
cja3a21   P R K V I S F L T K S V K Q I K E G R L K
ptr3a67   P R K V T S F L T K S V K R I K E G R L K
hsa3a7    P R E V T N F L R K S V K R M K E S R L E
oga3a91   P R D S T S F L R K S I K R I K E S R L K
cja3a5    P K D A I N F L K Q S V N R M K K S R L N
hsa3a5    P K D T I N F L S K S V N R M K K S R L N
hsa3a43   P K D V T H F L K N S I E R M K E S R L K
oga3a92   P K D V T N F L K N S V Q K M K E S R L H
mmus3a59  P K D S I E F F K K F V D R M T E N R L D
mmus3a25  P K D S I E F F K K F V N R M K E S R L D
mmus3a57  P K D S I E F F K K F V D R M K E S R L D
rno3a18   P K D S I E F F K K F V Y R M K E T R L D
mau3a31   P K D S I E F F K K F V Y R M K E T R L D
rno3a2    P N D S I E F F K K F V D R M Q E S R L D
rno3a73   P K D S V A F F Q K F V Y R M K Q T R L D
mmus3a44  P K D S I A F F Q K F V H R I K E T R L D
rno3a23   P K D S I S F F R K F V D K T K E N R L D
rno3a1    P R Q S M N F F K K F V K T M K K N R L D
mmus3a11  P R Q S M N F F K K F V K K M K K D R L D
mmus3a41  P R D S M N F F K K F V K R M K K E R L A
mmus3a16  P R D S M N F F K K F V K R M K K E R L A
```