Evolutionary Computational Genomics Lab

Isaac Gluck '19, Junior Research Scholar for Professor Zhaxybayeva

Python Scripts; working towards a *Novel Algorithm to Detect Exchange of Genetic Material Among Microorganisms*.

Required Libraries

- Dendropy

Pseudocode

1. User I/O of M gene trees with bootstrap sample trees
    - Read in bootstrap sample gene trees from a file, return dendropy trees

```
def readTreeFile(filename):
  file = open_file(filename)
  trees = []
  for line in file:
    trees.add(line)
  return trees
```

1. Decompose the gene trees into all possible quartets.
    - Take a list of trees, return all unique combinations of 4 taxa as keys (using tuples) in a dictionary
    - **Dictionary Structure** { (a, b, c, d): [t1, b1, t2, b2, t3, b3] }
        - a-d are taxa names, t1-3 are the possible topologies and b1-3 will be their bootstrap support values.

```
def makeQuartetDictionary(trees):
  quartet_dictionary = {}

  for tree in trees:
    list_of_quartets = get_all_combinations_of_four(tree)
    for quartet in list_of_quartets:
      quartet_dictionary.add(quartet, get_topologies(quartet))

  return quartet_dictionary
```

1. (continuation of 2.) Calculate bootstrap support for the quartets in the gene tree
    - Takes a tree and a quartet dictionary, returns the dictionary with filled in support values

```
def getTreeQuartetSupport(tree, quartet_dictionary):
  for quartet in quartet_dictionary:
    for tree in trees:
      quartet_topology = get_quartet_topology(quartet, tree)
      if quartet_topology != None:
        quartet_dictionary[quartet][quartet_topology] += 1
  return quartet_dictionary
```

1. Summarize quartet support on all M gene trees. Calculate the IC values for all quartets.
    - Take in the quartet dictionary and a bootstrap cutoff value, normalize the bootstrap values (if above the cutoff), and calculate internode certainty, return the quartet dictionary with IC values as the last item of each list

```
def buildFullSupport(quartet_dictionary, bootstrap_cutoff):
  for quartet in quartet_dictionary:
    normalize(quartet) # { (a,b,c,d): [t1, P(t1), t2, P(t2), t3, P(t3), IC] }
    for topology in quartet:
      if topology > 0:
        ic += (quartet[topology] * log(quartet[topology], 3)) # P(ti) * log base 3
of P(ti)
  return quartet_dictionary
```

1. Calculate QC branch values for each branch on the reference tree. Map these onto the tree.
    - Takes a reference tree filename and a quartet_dictionary, returns the reference tree with support values mapped onto its branches

```
def buildSupportTree(referenceTreeFile, quartet_dictionary):
  reference_tree = readTreeFile(referenceTreeFile)

  branches = get_partitions(reference_tree)
  for branch in branches:
    # taxa choose 2
    left_combinations = get_combinations(branch['left'], 2)
    right_combinations = get_combinations(branch['right'], 2)

    total_support = 0

    for left_combination in left_combinations:
      for right_combination in right_combinations:
        quartet = left_combination + right_combination
        reference_topology = get_reference_topology(quartet)
        if quartet in quartet_dictionary:
          support_value = -1 # default
          if best_topology(quartet_dictionary[quartet]) == reference_topology:
            support_value = 1 # if the best topology agrees

          support_value *= quartet_dictionary[quartet][6] # multiply by the IC
value
```

```
            total_support += support_value
        add_tag(branch, total_support)
    return reference_tree
```