Quiz Section Week 6
May 3, 2016

Exam Review
Dictionaries
What is a p-value?

Given a particular null hypothesis/null distribution and an observed value of the test statistic, it’s the probability that you observe that value or greater.
Why do we use Bonferroni correction to do multiple hypothesis testing?

• Multiple testing increases compared to the single test
  • The probability of having at least one false discovery (observing at least one p-value under your threshold)
  • The expected number of false discoveries

• Bonferroni correction adjust the p-value threshold, not the p-value itself to original threshold/(# of tests)

• By decreasing the threshold, Bonferroni decreases
  • The expected number of false discoveries
  • The probability of having at least one false discovery

Back to about the level of the single test
Linear gap vs affine gap

• Linear gap penalties penalize each gap independently regardless of where they fall

• Affine gap penalties have a penalty for starting a gap and a smaller penalty for each successive gap. This is to be more consistent with the fact that multiple successive gaps are dependent on the first because
  • They might have been caused by the same single event
  • Having the initial gap increases the likelihood of increasing the gap because of slippage during DNA replication or other mechanisms
Hierarchical clustering

• UPGMA implies average linkage as the distance between two clusters
  • The average of the distances between A->D, A->C, B->D, and B->C

• How to draw branch lengths using UPGMA
  • UPGMA creates a rooted tree, where the distance between the root and all the leaves are the same. This usually creates discrepancies with the distance matrix
Data structures

1. Gene sequence

2. Distances between several sequences

3. Names of genes in yeast genome sorted alphabetically

4. A network
Dictionaries are like lists that are indexed by arbitrary values

# Initialize a dictionary
aminoAcids = {'A': 'Alanine', 'R': 'Arginine'}

# Values and keys can be anything!
# Access a value given its key
print aminoAcids['R']

Arginine

# Put a key and value pair into the dictionary
aminoAcids['H'] = 'Histidine'
print aminoAcids

{'A': 'Alanine', 'H': 'Histidine', 'R': 'Arginine'}
Dictionaries are like lists that are indexed by arbitrary values

```python
aminoAcids = {'A':'Alanine', 'R':'Arginine'}
print aminoAcids['L']
KeyError: 'L'
print 'L' in aminoAcids  # Returns True or False
False
aminoAcids['L'] = 'Lysine'
print 'L' in aminoAcids
True
print aminoAcids['L']
Lysine
```
Using a dictionary to store a “matrix”

# You can copy and paste this

distances = {}
distances[('A','B')] = 1
distances[('A','C')] = 2
distances[('A','D')] = 4
distances[('B','A')] = 1
# etc.
Using dictionaries

# Iterate through the keys in a for loop
for key in aminoAcids:
    print key, aminoAcids[key]
A Alanine
H Histidine
R Arginine

# Or just get a list of the keys or the values
keys = aminoAcids.keys()
print keys
['A', 'H', 'R']
values = aminoAcids.values()
print values
['Alanine', 'Histidine', 'Arginine']
How would you inspect the distances between A and all the other points?

# You can copy and paste this

distances = {}

distances[('A','B')] = 1

distances[('A','C')] = 2

distances[('A','D')] = 4

distances[('B','A')] = 1

# etc.
Using a dictionary of dictionaries to store matrices by row

# You can copy and paste this

distances = {}
distances['A'] = {'B':1, 'C':2, 'D':4}
distances['B'] = {'A':1, 'C':2, 'D':5}
distances['C'] = {'A':2, 'B':2, 'D':5}
distances['D'] = {'A':4, 'B':5, 'C':5}
Exercise: print the pair of points that have the minimum distance

# You can copy and paste this

```
minimum_distance = float('inf')
closest_pair = ()  # An empty tuple
```
Exercise: print the pair of points that have the minimum distance

# You can copy and paste this

minimum_distance = float('inf')
closest_pair = () # An empty tuple

for x in distances:
    for y in distances[x]:
        if distances[x][y] < minimum_distance:
            minimum_distance = distances[x][y]
            closest_pair = (x,y)
print closest_pair, minimum_distance
Exercise: Count letter frequencies

# Returns a dictionary where the each key is a letter and the value is the number of times the amino acid appears
# Hint: You can adaptively add things to the dictionary (or not) by checking whether it exists in the dictionary already
# if ‘A’ in dict

peptide = ‘ILIKEHIKING’
def count_letters( s ):
    # Your code here

print count_letters( peptide )

Exercise: Count letter frequencies

def count_letters(s):
    letters = {}
    for c in s:
        if not c in letters:
            letters[c] = 1
        else:
            letters[c] += 1
    return letters