Quiz Section Week 3
April 12, 2016

Functions
Reading from and writing to files
Complexity
Hypothesis testing: how interesting is my data?

- **Test statistic**: a number that describes how “interesting” your data is
  - The alignment score

- **Null hypothesis**: define what uninteresting means
  - These two sequences are not related

- **Null distribution**:
  - Decide how the test statistic is distributed if your data is uninteresting

Empirical null distribution

Parametric null distribution

A p-value (shaded green area) is the probability of an observed (or more extreme) result arising by chance
Which null distribution do we use?

Test statistic
- Alignment score for any two random sequences
- Maximum alignment score between one sequence and many random sequences

Null distribution
- Normal
- Extreme Value
Bonferroni controls family-wise error rate (probability of at least 1 false discovery)

• Uncorrected p-value threshold $\alpha : 0.01$
  • If $p$-value $< \alpha$, reject null hypothesis ($H_0$)
  • $P(\text{reject} \mid H_0) = \alpha = 0.01$
  • $P(\text{not reject} \mid H_0) = 1 - \alpha = 0.99$

• $N = 50$ hypothesis tests
  • Probability of no rejection in 50: $P(\text{not reject} \mid H_0)^N = (1 - 0.01)^{50} = 0.605$
  • Probability of at least one rejection $= 1 - P(\text{not reject} \mid H_0)^N = 0.395$

• Bonferroni changes threshold $\alpha_{\text{Bonferroni}} = \alpha / N$
  • What’s the probability of at least one rejection using $\alpha_{\text{Bonferroni}}$ (false discovery)?
Probability of false discovery vs # of hypothesis tests

![Graph showing the probability of false discovery vs the number of hypothesis tests. The graph plots the probability on the y-axis and N on the x-axis. There are three lines: threshold 0.05, Bonferroni threshold 0.05/N, and a line at 0.05.](image-url)
def sum_list(numbers):  # Define a function with one input
    total_sum = 0
    for number in numbers:
        total_sum = total_sum + number
    return total_sum  # Return replaces the function call
                       # with total_sum

x = range(0,5)
print sum_list(x)
10
Exercise: Write a function to check if a string is a palindrome

def is_palindrome(s):
    # Return True if it is
    # Return False if it isn’t
s = ‘tacocat’
# Print ‘Not palindrome!’ if not
# Do nothing if it is

for i in range(0,len(s)):
    reverse_index = -1-i
    if s[i] == s[reverse_index]:
        continue
    else:
        print ‘Not palindrome!’
        break
Exercise: Write a function to check if a string is a palindrome

```python
def is_palindrome(s):
    for i in range(0, len(s)):
        reverse_index = -1 - i
        if s[i] == s[reverse_index]:
            continue
        else:
            return False
    return True

is_palindrome('tacocat')
True
is_palindrome('tacodog')
False
```
Reading data from a file line-by-line

Text from file ‘example.txt’:

hello
world!

Code to read in data

# open() returns an object that reads files
fin = open( ‘example.txt’, ‘r’) # ‘r’ stands for ‘read’
for line in fin: # In a for loop, fin acts like a list of strings
    print line
fin.close() # Lets the computer know it can free up resources used to read the file

hello

World!
Outputting a file

fout = open('output.txt','w')  # 'w' stands for 'write'
fout.write('Hello! How')
fout.write(' are you?
I’m fine.')  # '\n' starts a new line
fout.close()

Outputted file 'output.txt'
Hello! How are you?
I’m fine.
Example: Reading a file, analyzing it, and outputting results

`inputfile.txt:`
tacocat
tacodog
12321
3.14159

`outputfile.txt:`
tacocat yes
tacodog no
12321 yes
3.14159 no
Example: Reading a file, analyzing it, and outputting results

inputfile.txt:
tacocat
tacodog
12321
3.14159

outputfile.txt:
tacocat yes
tacodog no
12321 yes
3.14159 no

```
fin = open('inputfile.txt','r')
fout = open('outputfile.txt','w')
for line in fin:
    word = line.rstrip()  # gets rid of \n at the end of the line
    answer = is_palindrome( word )
    fout.write( word + ' ' )
    fout.write( str(answer) + '
' )
fin.close()
fout.close()
```
Using the command line to tell the program what files to read and write

```
python analyze_palindrome.py inputfile.txt outputfile.txt
```

```python
# analyze_palindrome.py
fin = open(sys.argv[1], 'r')
fout = open(sys.argv[2], 'w')
for line in fin:
    word = line.rstrip()  # gets rid of \n at the end of the line
    answer = is_palindrome(word)
    fout.write(word + ' ')
    fout.write(str(answer) + '
')
fin.close()
fout.close()
```
“Big O” notation for complexity: global alignment

• Two sequences of length $M$ and $N$
• $3(M+1)(N+1) + M + N = O(MN)$
  • 3 is the number of possible sums to consider: match or either gap for each element in the matrix
  • $M+1$ and $N+1$ are the two dimensions of the matrix
  • $3MN + 4M + 4N + 3$
  • $MN$ grows fastest as $M$ and $N$ go to infinity
What is the time complexity in $O()$ to compute the sum of a list?

```python
# x is a list of length N
sum = 0
for v in x:
    sum = sum + v
print 'The sum is:', sum
# O(N)
```
UPGMA Algorithm complexity

\[ O(?) = O(N \times N^2) = O(N^3) \]

1. Generate a table of pairwise sequence distances and assign each sequence to a list of \( N \) tree nodes.
2. Look through current list of nodes (initially these are all leaf nodes) for the pair with the smallest distance.
3. Merge the closest pair, remove the pair of nodes from the list and add the merged node to the list.
4. Repeat until only one node left in list - it is the root.
How to represent a 2D matrix in python?

- List of lists!
- Each row is a different list

```
matrix = [ [0, -4, -8, -12], ['?', ' ', ' '], ...
```

print matrix[0][1]

-8
What is the time complexity in $O()$ to compute the sum of a matrix?

```python
# x is a list of N lists
# each list has N elements
sum = 0
for row in x:  # Do this N times
    for v in row:  # N times again
        sum = sum + v
print 'The sum is:', sum
# The answer is $O(N^2)$
```