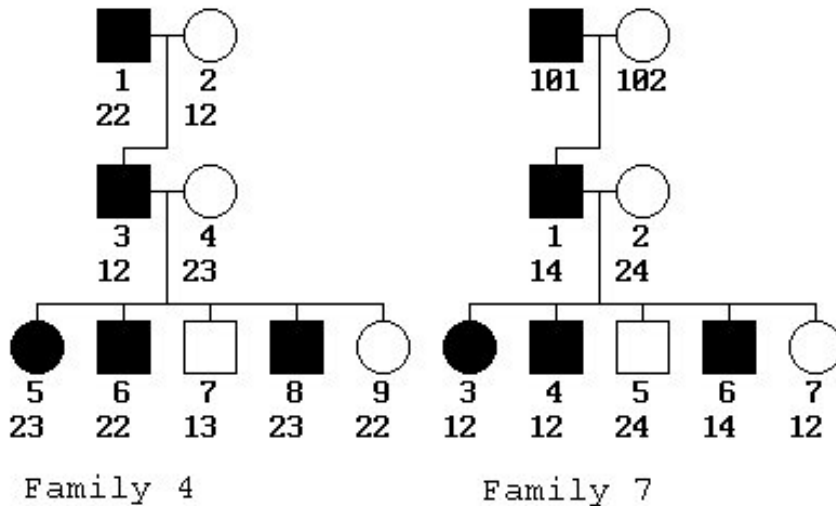


Homework #5

Due Tuesday Feb. 19 at the beginning of class. Assignments turned in more than 5 minutes after the beginning of class will be penalized 10 points, with an additional 10 points every 24 hours thereafter. You may discuss the homework assignment with other students, but do not share your work. **All Python programs should be run before being turned in. Even experienced programmers can seldom write a program perfectly on the first try.**



- (10 points) For pedigree 4, how many non-recombinant and recombinant offspring are there? *5 non-recombinant (3, 5-8) and 1 recombinant (9).*
- (10 points) For pedigree 7, if grandfather (individual 101) had genotype 13, how many non-recombinant and recombinant offspring are there? *4 non-recombinant (3, 4, 5, 6) and 1 recombinant (7). We cannot score individual 1 because we lack his grandmother.*
- (10 points) For pedigree 7, if grandfather (individual 101) had genotype 44, how many non-recombinant and recombinant offspring are there? *4 recombinant (3, 4, 5, 6) and 1 non-recombinant (7). We still can't score individual 1.*
- (10 points) Brief answer: Why do pedigree researchers try to collect grandparents? *The more close relatives are collected, the more of the individuals in the pedigree will be scorable. In this case, having both grandparents might allow us to score individual 1 as well as being sure whether we have 1 recombinant or 4.*
- (20 points) Write a Python program to search a file of DNA sequences for a recognition site which follows this rule:
 - First two bases must be A, T
 - Next base either C or G
 - The next 2 or 3 positions can be any base
 - Next base either C or G
 - Last two bases must be T, A

You may use the `re` module if you wish but are not required to do so.

```
import sys
import re
dnafile = open(sys.argv[1], "r")
```

```

recog1 = re.compile(r"AT[CG][ACGT]{2,3}[CG]TA",re.IGNORECASE)

filecontents = dnafile.read()

search1 = recog1.search(filecontents)
allmatches = recog1.findall(filecontents)

# It's not necessary to be as thorough as this, as I only asked for
# one match, but this solution demonstrates a couple of different
# approaches

print "the file",sys.argv[1],
if search1 != None :
    print "contained a match starting at position",search1.start(),
    print "which read as:",search1.group()
    if len(allmatches) != 1 :
        print " the following additional matches were also found:",
        print allmatches[1:]
else :
    print "contained no matches for the recognition site."

```

6. (20 points) Write a Python program to compute the lod score for a specified number of recombinant and non-recombinant offspring and a specified θ . Read the recombinant and non-recombinant counts and θ value from the command line. Hint:

$$Lod = \log_{10} \frac{(1 - \theta)^{NR} \times \theta^R}{0.5^{(NR+R)}}$$

```

import sys
import math
NR = float(sys.argv[1])
R = float(sys.argv[2])
theta = float(sys.argv[3])
term1 = ((1.0-theta)**NR * theta**R) / 0.5**(NR+R)
if term1 > 0 :
    print "Lod score for",theta,"is",math.log10(term1)
else :
    print "Illegal input"
# there are better ways to do error checking here; this one is
# minimal

```

7. (20 points) Transform your program into a function. Import it into a new program which computes the lod score for all values of θ from 0 to 0.5 in increments of 0.05. (There should be 11 values computed.) Print a table of these lod scores, with labels. Also print a line indicating the maximum lod encountered and the corresponding value of θ .

```

# in file lod.py
import math
def lodscore(NR, R, theta) :
    term1 = ((1.0-theta)**NR * theta**R) / 0.5**(NR+R)
    if term1 > 0 :
        return math.log10(term1)
    else :
        print "Illegal input"
        return None

```

```

# in main program file
import sys
import lod

R = float(sys.argv[1])
NR = float(sys.argv[2])
theta = 0.05

print "For",R,"recombinants and",NR,"non-recombinants, the LOD scores",
print "are:\n"
printstrings = ["Theta","Lodscore"]
print "%20s %20s" %(printstrings[0],printstrings[1])
print (" "*10),("-"*35)
printstrings = ["0.00","undefined"]
print "%20s %20s" %(printstrings[0],printstrings[1])
lods = []
thetas = {}
while (theta < 0.51) :
    lods.append(lod.lodscore(R,NR,theta))
    thetas[lods[-1]] = theta
    print "%20s %20s" %(str(theta),str(lods[-1]))
    theta += 0.05

maxlod = max(lods)
print "\nThe maximum LOD was",maxlod,"for a theta of",thetas[maxlod]
namx = lods.count(maxlod)
if namx != 1 :
    print "\t\tand",namx-1,"other theta values (see table)"

```