Combinatorics of Sup35 - YFP Amyloids

Daniel Chen - Waterloo iGEM

June 2017

1 Introduction

An *amyloid* is just a stack of proteins for our purposes. The proteins, labelled say A,B,C,... will form block patterns as they stack: ABBACCBCBCA...

Looking at it this way, it becomes possible to analyze the properties of these amyloids using classical combinatorial enumeration methods.

2 Analysis of exclusively YFP Amyloids

Let us analyze the simplified case where the two halves of Yellow Fluorescent Protein (YFP) form an amyloid. Call the distinct halves A and B. There are no restrictions on the patterns in the amyloid - our regular expression will be $\{A, B\}^*$. To justify this, we are assuming that two proteins can be attracted to each other and held close through, say, London Dispersion Forces [we also have experimental evidence to show this happening for YFP].

We are interested in positions of an amyloid where A and B come together and fuse, producing yellow fluorescence.

Clearly, there are 2^n possible amyloids of length n.

We will start with the case where the halves of Split-YFP are well-mixed and of equal concentrations and are thus equally likely to appear as the next protein in an amyloid sequence.

2.1 Lighting Patterns

Now let us abstract away from A and B specifically and just analyze the lighting-patterns of amyloids of length n.

We first ask: How many distinct patterns of fluorescing locations could we possibly see from an amyloid of length n?

Suppose we had an amyloid starting with A. We can proceed to add proteins to the amyloid, but instead of labelling them by what protein they actually are, we can label them by their functionality. If the protein at the current position causes the previous to fluoresce, then we put a 1, in analogy to a "successful coin toss". Otherwise, we put a 0 for a "failure". Note that a protein can never cause the previous protein to fluoresce if the previous protein was already fluorescing. For example, ABA becomes 010.

For example, if we have ABBAAAAABA, then reading from left to right (because we have to think about how these proteins were added one at a time to analyze which ones fused), we get the functional abstraction 0101000010. Juxtaposition reveals:

$$ABBAAAAABA$$
$$0\ \ 10\ \ 10\ \ 00\ \ 01\ \ 0$$

We always start with a 0, since the first protein cannot cause the (non-existent) previous protein to fluoresce. Doing this, we no longer need think about whether we started with an A or a B.

As another example,

$$BBBBABABABABABABABAAAABAABABBBBABA$$
$$0\ \ 00\ \ 01\ \ 01\ \ 01\ \ 01\ \ 01\ \ 01\ \ 01\ \ 01\ \ 00\ \ 01\ \ 00\ \ 10\ \ 10\ \ 00\ \ 10\ \ 1$$

We notice that our unambiguous regular expression for these 01 block patterns is $(0^*01)^*0^*$. Note that if we were looking at an amyloid without knowing what the proteins actually were, we would still be able to identify its 01 block pattern by looking at the fluorescing locations. Indeed, each fluorescing location corresponds to a 1.

We can count the total number of distinct 01 block patterns of length n by using generating functions. Our function $G(x)$ is simply

$$G(x) := \sum_{i=0}^{\infty} a_n x^n = \frac{1}{1-\left(\frac{x^2}{1-x}\right)} = \frac{1-x}{1-x-x^2}$$

so that

$$a_n - a_{n-1} - a_{n-2} = \begin{cases} 1, & \text{if } n = 0 \\ -1, & \text{if } n = 1 \\ 0, & \text{if } n \geq 2 \end{cases}$$

This is a standard technique of reading off the recursive formula from the rational function representation of the generating function. To be explicit, this follows because:

$$G(x) := \sum_{i=0}^{\infty} a_i x^i = \frac{1-x}{1-x-x^2}$$
$$\implies \sum_{i=0}^{\infty} a_i x^i - \sum_{i=0}^{\infty} a_i x^{i+1} - \sum_{i=0}^{\infty} a_i x^{i+2} = 1-x$$
$$\implies \sum_{i=0}^{\infty} (a_i - a_{i-1} - a_{i-2}) x^i = 1-x, \text{where } a_i := 0 \text{ for } i < 0$$

so that we get what was claimed.

Anyways... we see that $a_0 = 1$, $a_1 = 0$, $a_2 = 1$, $a_3 = 1$, $a_4 = 2$, $a_5 = 3$, $a_6 = 5$ etc...

So $a_n = F_{n-2}$, where F_n is the nth Fibonacci number assuming $F_0 = F_1 = 1$.

2.2 Probability of k fluorescing locations in length n amyloid

Now lets analyze the probability of seeing k fluorescing pairs of proteins in an amyloid of length n.

2.2.1 Distinct block patterns with k 1s

First, let us count the number of block patterns of length n with k 1s. Our invariant states that no 1's may be adjacent. Furthermore, the first digit must be a zero. These are the only restrictions. So we can count the number of ways of putting k objects into $n-1$ boxes such that no two chosen boxes are adjacent. This number is:

$$\#(n,k) = \binom{n-k}{k}, k \leq \lfloor n/2 \rfloor$$

A proof can be found at [1].

2.2.2 Distinct amyloids with k 1s

Now, the number of distinct amyloids with k fluorescing pairs is:

$$\mathcal{A}(n,k) = 2^k \cdot \#(n-2,k-1) + 2^{k+1} \cdot \#(n-1,k)$$

To see this, we have two cases, where we fix $k = 5$ for the sake of example:

Block pattern ends with a 0

Consider the following block pattern. We write the number of possible proteins (A or B) at that location below the corresponding location of the block pattern.

0000100001010010000010000

2111121111212112111121111

Indeed, for each block of 0's, at the beginning of the block, any of either A or B may appear there, since it will not fuse with the previous protein. Then thereafter each protein is uniquely determined by this choice through to the 1 that appears.

Any block pattern of this case therefore has 2^{k+1} possible amyloids associated with it. There are $\#(n-1,k)$ possible patterns for this case.

Block pattern ends with a 1

Consider the following block pattern. We write the number of possible proteins (A or B) at that location below the corresponding location of the block pattern.

0000100001010010000000001

2111121111212112111111111

As we can see, putting a 1 at the end of the block pattern implies only 2^k possible amyloids with the particular pattern. There are $\#(n-2,k-1)$ patterns with this property.

Hence the probability of seeing k fluorescing pairs in an amyloid of length n is

$$P(n,k) = \frac{\mathcal{A}(n,k)}{2^n}$$

2.3 Probability of at least m fluorescing locations

The probability of seeing at least m fluorescing pairs on an amyloid of length n is simply the sum:

$$P_{\geq}(n,m) = \frac{1}{2^n} \sum_{k=m}^{\lfloor n/2 \rfloor} \mathcal{A}(n,k)$$

since the events $k = m, \ldots, \lfloor n/2 \rfloor$ are independent.

2.4 Expected number of fluorescing locations

This is simply the sum:

$$E(n) = \frac{1}{2^n} \sum_{k=0}^{\lfloor n/2 \rfloor} k \cdot \mathcal{A}(n,k)$$

by definition of expected value.

2.5 Expected percentage of proteins involved in fluorescence

This is the number:

$$R(n) = \frac{2E(n)}{n}$$

Note that it is conjectured that

$$R(n) = 2/3$$

due to a computer simulation using the following script in Python:

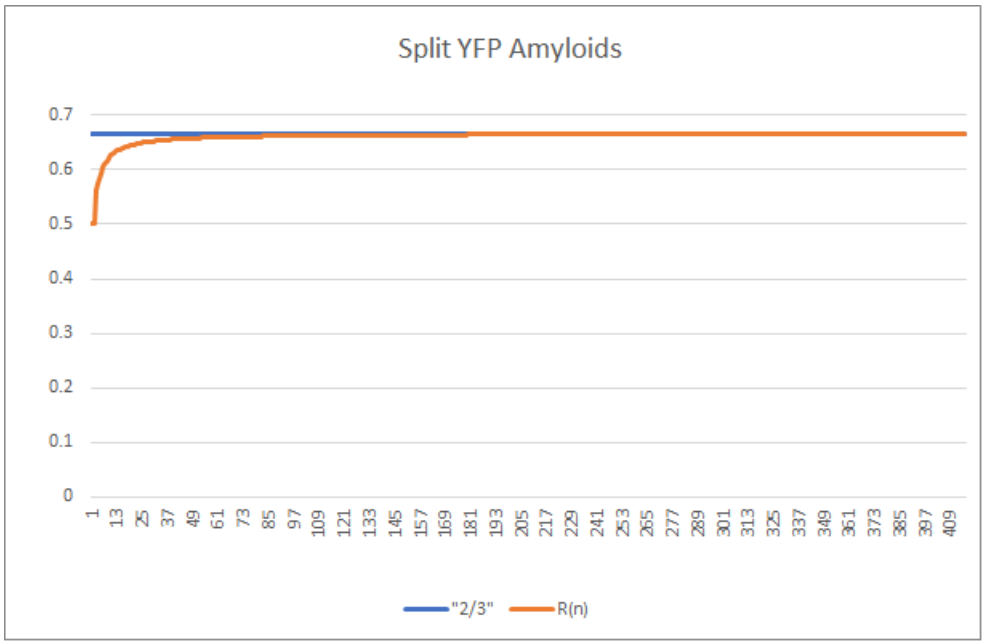
```
#!/usr/bin/python

import math
import matplotlib.pyplot as plt
#The binomial coefficient nCk...
def binom(n, k):
        if k > n or k < 0:
                return 0
        a = math.factorial(n)
        b = math.factorial(k)
        c = math.factorial(n - k)
        return a / (b * c)

#This is the #(n,k) function I defined in my writeup...
def num(n, k):
        return binom(n - k, k)

#This is the A(n,k) function I defined in my writeup...
def A(n, k):
        return pow(2, k) * num(n - 2,k - 1) + pow(2, k + 1) * num(n - 1, k)

#The expected number of fluorescing locations in a length n
#amyloid, according to formula I derived, is:
def expected_value(length):
        result = 0.0
        for i in range(0, int(length / 2)):
                result += i * A(length, i)
```

```
        result /= pow(2, length)
        return result



if __name__ == '__main__':
        for i in range(4, 1000):
                r = 2 * expected_value(i) / i
                print "{}: {}".format(i, r)
```

which produces the following sample of the output:

```
2: 0.5
3: 0.5
4: 0.5625
5: 0.575
6: 0.59375
7: 0.602678571429
8: 0.611328125
9: 0.6171875
10: 0.622265625
11: 0.626242897727
12: 0.629638671875
13: 0.632474459135
14: 0.634922572545
15: 0.637036132813
16: 0.638889312744
17: 0.640522676356
18: 0.641975402832
19: 0.643274809185
20: 0.644444465637
21: 0.645502635411
22: 0.646464651281
23: 0.647342992866
24: 0.648148149252
25: 0.648888888359
...
417: 0.665600852651
```

As a graph, the first 400 values appears as:



where the blue line is 2/3 and the orange is our data.

3 Introducing Sup-35 into Amyloids

We now want to consider the facts above but with the introduction of Sup-35 (which we label with an S) into our amyloids. Our amyloids now come from the regular expression $\{A, B, S\}^*$. We would like to first analyze the case where Sup-35 and the halves of Split-YFP come in equal concentrations, and are thus equally likely to occur as the next protein in an amyloid sequence.

3.1 Number of distinct lighting patterns

The introduction of Sup-35 does not change this. When we abstract away the details we are still looking at block patterns of the form $(0^*01)^*0^*$. Hence we have F_{n-1} distinct lighting patterns for an amyloid of length n.

3.2 Probability of k fluorescing locations in length n amyloid

The number of amyloids of length n with k fluorescing locations is:

$$\mathcal{A}(n,k) = \sum_{q=0}^{n-2k} b_q \sum_{\substack{\sum_{i=1}^{k} \phi_i = n-q \\ \phi_i \geq 2}} \prod_{i=1}^{k} a_{\phi_i}$$

where $a_0 = a_1 = 0, a_2 = 2$, $a_i = 2a_{i-1} + a_{i-2}, i > 2$ and $b_0 = 1, b_1 = 3$, $b_i = 2b_{i-1} + b_{i-2}, i > 1$.

Explicitly,

$$b_n = -\frac{1}{2+2\sqrt{2}} \cdot \left(\frac{1}{-1-\sqrt{2}}\right)^n - \frac{1}{2-2\sqrt{2}} \cdot \left(\frac{1}{-1+\sqrt{2}}\right)^n, n \geq 0$$

and

$$a_n = \begin{cases} 0, & \text{if } n = 0 \\ \frac{3+2\sqrt{2}}{\sqrt{2}+2}\left(\frac{1}{-1-\sqrt{2}}\right)^n - \frac{3-2\sqrt{2}}{\sqrt{2}-2}\left(\frac{1}{-1+\sqrt{2}}\right)^n, & \text{if } n \geq 1 \end{cases}$$

And so the probability of k fluorescing locations in an amyloid of length n is $1/3^n\mathcal{A}(n,k)$.

The idea of this formula is precisely as follows. q is the length of the tail of zeroes at the end of the amyloid, functionally speaking. b_q represents the number of A-B-S block patterns of length q of the form 000...0. There are then k functional blocks of the form $(0*01)$ preceding the tail, and ϕ_i represents the length of the ith block. a_{ϕ_i} represents the number of A-B-S block patterns of length ϕ_i of the functional form 0000...01. With this in mind, the reader can confirm the formula claimed.

Let us show all of this...

Firstly, b_i represents the number of A-B-S block patterns represented by a 0-1 block pattern of the form 00000....0 where there are precisely i 0's. To obtain the recursive formula, we note that the A-B-S block patterns of this type are represented by the A-B-S regular expression

$$S^*\{AA^*SS^*, BB^*SS^*\}^*\{AA^*, BB^*, \phi\}$$

where ϕ is the empty set,

and so the generating function is

$$G(x) = \sum_{i=0}^{\infty} b_i x^i = \frac{1}{1-S} \cdot \frac{1}{1 - \left(\frac{A}{1-A} \cdot \frac{S}{1-S} + \frac{B}{1-B} \cdot \frac{S}{1-S}\right)} \cdot \left(\frac{A}{1-A} + \frac{B}{1-B} + 1\right)$$

.

Setting $A = B = S = x$ where x is an indeterminate, we have that b_i represents as stated above. Then doing some algebra the reader can confirm that

$$G(x) = \frac{1+x}{1-2x-x^2}$$

and so

$$b_n - 2b_{n-1} - b_{n-2} = \begin{cases} 1, & \text{if } n = 0 \\ 1, & \text{if } n = 1 \\ 0, & \text{if } n \geq 2 \end{cases}$$

where the technique to read off this recursion formula has already been discussed. Then we get what we claimed about the b_i's earlier.

To get the closed form formula for the value of b_i, do partial fractions on the generating function:

$$\begin{aligned} \sum_{i=0}^{\infty} b_i x^i = G(x) = \frac{1+x}{1-2x-x^2} &= \frac{(1/2)}{(-1-\sqrt{2})-x} + \frac{(-1/2)}{(-1+\sqrt{2})-x} \\ &= -\frac{1}{2+2\sqrt{2}} \cdot \frac{1}{1-(1/(-1-\sqrt{2}))x} - \frac{1}{2-2\sqrt{2}} \cdot \frac{1}{1-(1/(-1+\sqrt{2}))x} \\ &= -\frac{1}{2+2\sqrt{2}} \sum_{i=0}^{\infty} \left(\frac{1}{-1-\sqrt{2}}\right)^i x^i - \frac{1}{2-2\sqrt{2}} \sum_{i=0}^{\infty} \left(\frac{1}{-1+\sqrt{2}}\right)^i x^i \end{aligned}$$

Therefore we explicitly get

$$b_n = -\frac{1}{2+2\sqrt{2}} \cdot \left(\frac{1}{-1-\sqrt{2}}\right)^n - \frac{1}{2-2\sqrt{2}} \cdot \left(\frac{1}{-1+\sqrt{2}}\right)^n, n \geq 0$$

as claimed.

Likewise, a_i represents the number of A-B-S block patterns represented by a 0-1 block pattern of the form 000...01 where the pattern is of length i.

To obtain the recursive formula, we note that the A-B-S block patterns of this type are represented by the A-B-S regular expression

$$S^*\{AA^*SS^*, BB^*SS^*\}^*\{AA^*B, BB^*A\}$$

and so the generating function is

$$G(x) = \sum_{i=0}^{\infty} b_i x^i = \frac{1}{1-S} \cdot \frac{1}{1-\left(\frac{A}{1-A} \cdot \frac{S}{1-S} + \frac{B}{1-B} \cdot \frac{S}{1-S}\right)} \cdot \left(\frac{A^2}{1-A} + \frac{B^2}{1-B}\right)$$

.

Setting $A = B = S = x$ where x is an indeterminate, we have that a_i represents as stated above. Then doing some algebra the reader can confirm that

$$G(x) = \frac{2x^2}{1-2x-x^2}$$

and so

$$a_n - 2a_{n-1} - a_{n-2} = \begin{cases} 0, & \text{if } n = 0 \\ 0, & \text{if } n = 1 \\ 2, & \text{if } n = 2 \\ 0, & \text{if } n \geq 2 \end{cases}$$

where the technique to get read off this recursion formula has already been discussed. Then we get what we claimed about the a_i's earlier.

To get the closed form formula for the value of a_i, do partial fractions on the generating function:

$$\begin{aligned} \sum_{i=0}^{\infty} a_i x^i = G(x) = \frac{2x^2}{1-2x-x^2} &= \frac{4x-2}{(x-(-1-\sqrt{2}))(x-(-1+\sqrt{2}))} - 2 \\ &= \frac{(3+2\sqrt{2})/\sqrt{2}}{x-(-1-\sqrt{2})} + \frac{(-3+2\sqrt{2})/\sqrt{2}}{x-(-1+\sqrt{2})} - 2 \\ &= \frac{3+2\sqrt{2}}{\sqrt{2}+2} \cdot \frac{1}{1-(1/(-1-\sqrt{2}))x} + \frac{-3+2\sqrt{2}}{\sqrt{2}-2} \cdot \frac{1}{1-(1/(-1+\sqrt{2}))x} - 2 \\ &= \frac{3+2\sqrt{2}}{\sqrt{2}+2} \sum_{i=0}^{\infty} \left(\frac{1}{-1-\sqrt{2}}\right)^i x^i - \frac{3-2\sqrt{2}}{\sqrt{2}-2} \sum_{i=0}^{\infty} \left(\frac{1}{-1+\sqrt{2}}\right)^i x^i - 2 \end{aligned}$$

Therefore, we explicitly get

$$a_n = \begin{cases} 0, & \text{if } n = 0 \\ \frac{3+2\sqrt{2}}{\sqrt{2}+2}\left(\frac{1}{-1-\sqrt{2}}\right)^n - \frac{3-2\sqrt{2}}{\sqrt{2}-2}\left(\frac{1}{-1+\sqrt{2}}\right)^n, & \text{if } n \geq 1 \end{cases}$$

Now, rewrite the claimed formula for $\mathcal{A}(n,k)$ as

$$\mathcal{A}(n,k) = \sum_{\sum_{i=0}^{k}\phi_i = n} \prod_{i=0}^{k} a_{\phi_i} + \sum_{q=1}^{n} b_q \sum_{\sum_{i=0}^{k}\phi_i = n-q} \prod_{i=0}^{k} a_{\phi_i}$$

which is clearly equal to what was claimed.

Then the first term is all A-B-S patterns ending with a 1 and the second term is all A-B-S patterns ending with a 0. This is a similar case analysis to what we did for the Split-YFP-exclusive case.

Thus we have shown the formula holds.

3.3 Probability of at least m fluorescing locations

The probability of seeing at least m fluorescing pairs on an amyloid of length n is simply the sum:

$$P_{\geq}(n,m) = \frac{1}{3^n} \sum_{k=m}^{\lfloor n/2 \rfloor} \mathcal{A}(n,k)$$

since the events $k = m, \ldots, \lfloor n/2 \rfloor$ are independent.

3.4 Expected number of fluorescing locations

This is simply the sum:

$$E(n) = \frac{1}{3^n} \sum_{k=0}^{\lfloor n/2 \rfloor} k \cdot \mathcal{A}(n,k)$$

by definition of expected value.

3.5 Expected percentage of proteins involved in fluorescence

This is the number:

$$R(n) = \frac{2E(n)}{n}$$

Note that it is conjectured that

$$R(n) = 1/3$$

by simulation via the following Python script:

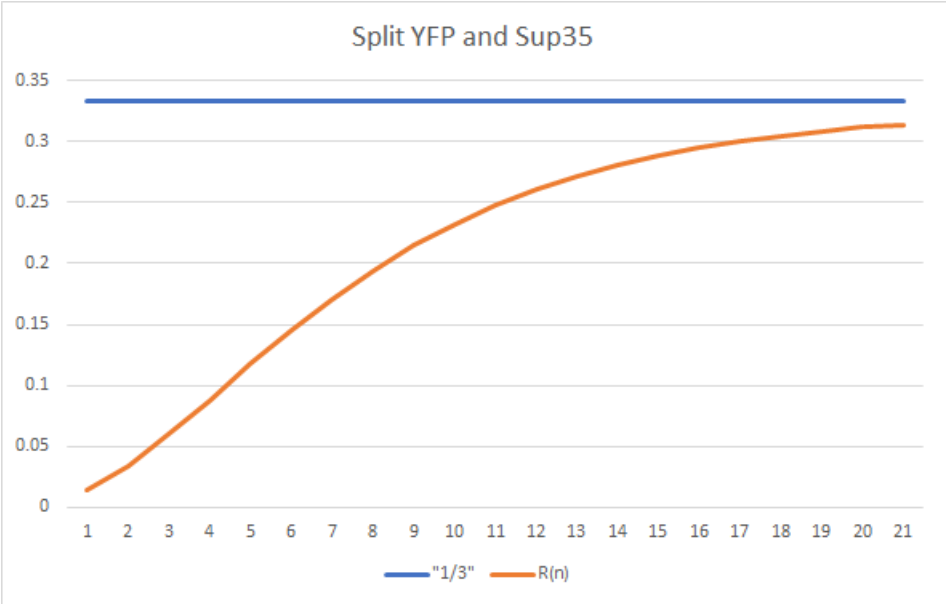
```
#!/usr/bin/python
import itertools
import math

#The binomial coefficient nCk...
def binom(n, k):
    if k > n or k < 0:
        return 0
    a = math.factorial(n)
    b = math.factorial(k)
    c = math.factorial(n - k)
    return a / (b * c)

def a(n):
    if n == 0:
        return 0
    x = math.sqrt(2)
    result = ((3 + 2 * x) / (x + 2) * 1.0) * pow((1.0 / (-1.0 - x)), n) +
    ((2 * x - 3) / (x - 2) * 1.0) * pow((1.0 / (x - 1)), n)
    return result

def b(n):
    x = math.sqrt(2)
    result = - (1.0 / (2 * x + 2)) * pow((1.0 / (-1 - x)), n) -
    (1.0 / (2 - 2 * x)) * pow((1.0 / (x - 1)), n)
    return result

def product(lst):
    val = 1.0
    for item in lst:
        val *= item
    return val

#This is the A(n,k) function I defined in my writeup...
def A(n, k):
    result = 0.0
    for q in xrange(0, n + 1 - 2 * k):
        all_lists =
        itertools.product(range(2, n - q - 2 * (k - 1)), repeat=k)
        lists_of_sum_n = [lst for lst in all_lists if sum(lst) == n - q]
        innerSum = 0.0
        for lst in lists_of_sum_n:
            a_lst = [a(phi_i) for phi_i in lst]
            innerSum += product(a_lst)
        result += b(q) * innerSum
```

```
    return result

#The expected number of fluorescing locations in a length n
#amyloid, according to formula I derived, is:
def expected_value(length):
    result = 0.0
    for i in xrange(0, length / 2 + 1):
        result += i * A(length, i)
    result /= pow(3, length)
    return result

if __name__ == '__main__':
    for i in xrange(2, 1000):
        r = 2 * expected_value(i) / i
        print "{}: {}".format(i, r)
```

which returns the following output:

```
6: 0.0146319158665
7: 0.0334443791234
8: 0.0600518213687
9: 0.0880624566038
10: 0.117326288337
11: 0.145103937818
12: 0.171026824552
13: 0.194258484153
14: 0.214795454455
15: 0.232595648807
16: 0.24787920548
17: 0.26086528489
18: 0.271832547986
19: 0.281041609429
20: 0.288746154913
21: 0.295172255476
22: 0.300521909795
23: 0.304969639296
24: 0.308665548904
25: 0.311736795713
26: 0.314290575502
```

and the following graph:



4 Using FRET in amyloids of YFP and CFP

We now analyze the case where full YFP (as a *dimer*) and CFP are well-mixed in a container under ambient lighting that causes CFP to fluoresce. When CFP is bonded to YFP, the fluorescence from CFP will cause YFP to fluoresce (FRET).

We are interested in measuring the *total* yellow fluorescence emanating from amyloids of this kind. That is, the YFPs may fluoresce at a fixed level (call it Level 1) when it is next to one and only one CFP, but it may also fluoresce twice as much (Level 2) if it is next to *two* CFPs as in a YFP being sandwiched between two CFPs. We want to then sum over the total yellow fluorescence of an amyloid rather than just the raw number of YFPs that are caused to fluoresce.

Label YFP as Y and CFP as C. Then our block patterns are $\{Y, C\}^*$ and as with the split-YFP exclusive case, extracting away the details we get patterns on the form $(0^*01^*1)^*0^*$ where we write a 1 whenever the full protein at that location causes the previous to fluoresce OR is caused to fluoresce by the previous. We note that, unlike previous cases, once we decide on the first protein in a 0-1 block pattern, the entire protein's Y-C block pattern is uniquely determined. Therefore, for each 0-1 block pattern there are 2 Y-C block patterns.

It is now remarked that there is a one-to-one correspondence between 1's as appearing in 0-1 block patterns as described above and the total yellow fluorescence emanating from an amyloid of the kind. To be specific, the number of 1's is equal to the total yellow fluorescence (if we take Level 1 to be 1 and Level 2 to be 2 as numerical values for which we sum up). For example...

$$\{C, Y\}^*\text{: } CYYYCYCYYCC \tag{1}$$

$$(0^*01^*1)^*0^*\text{: } 0 \;\; 10 \;\; 01 \;\; 11 \;\; 10 \;\; 10 \tag{2}$$

$$\text{Yellow fluorescence: } 0 \;\; 10 \;\; 01 \;\; 20 \;\; 11 \;\; 00 \tag{3}$$

and we note the second and third rows sum to the same number, as claimed.

4.1 Probability of k total yellow fluorescence in length n amyloid

The number of amyloids of length n with k total yellow fluorescence is:

$$\mathcal{A}(n,k) = 2 * \binom{n-1}{k}, k \in [0, n-1]$$

and thus the probability of observing k total yellow fluorescence in a length n amyloid is

$$P(n,k) = \frac{2 * \binom{n-1}{k}}{2^n} = \frac{\binom{n-1}{k}}{2^{n-1}}, k \in [0, n-1]$$

.

These are all easily seen from the remarks above.

4.2 Expected total yellow fluorescence

This is simply the sum:

$$E(n) = \sum_{k=0}^{n-1} k \cdot P(n,k)$$

by definition of expected value.

Note that, by [2]:

$$E(n) := \frac{1}{2^{n-1}} \cdot \sum_{k=0}^{n-1} k \cdot \binom{n-1}{k} = \frac{1}{2^{n-1}} \cdot (n-1)2^{n-2} = \frac{n-1}{2}$$

So that for very large n, we expect to see about as much yellow unit fluorescence as half the length of the amyloid.

Note that the quantity:

$$R(n) := E(n)/n$$

goes to $1/2$ as n goes to infinity.

5 Using FRET in amyloids of YFP, CFP, and Sup35

As with section 3, we will now analyze the case of introducing Sup35 to the system in Section 4. That is, we have Sup35, YFP, and CFP well mixed and of equal concentrations in a container under ambient lighting which causes CFP to fluoresce. The cyan fluorescence of a CFP will cause an adjacent YFP to fluoresce yellow due to FRET, and this fluorescence can be stacked so that a YFP sandwiched between two CFPs will fluoresce twice as much as in the case where a YFP is next to only 1 CFP. The presence of Sup35 will block the YFPs and CFPs from interacting under FRET.

Our block pattern becomes $\{C, Y, S\}^*$ where C represents CFP, Y represents YFP, and S represents Sup35. The functional pattern is still $(0^*01^*1)^*0^*$ as before.

5.1 Probability of k total yellow fluorescence in length n amyloid

We remark that the number of amyloids of length n with k total yellow unit fluorescence is:

$$\mathcal{A}(n,k) = \sum_{q=0}^{n-k} b_q \sum_{i=1}^{k} \sum_{\substack{\sum_{j=1}^{i}\phi_j = n-q \\ \phi_j \in [2,n-q]\cap\mathbb{Z}}} \sum_{\substack{\sum_{j=1}^{i}\psi_j = k \\ \psi_j \in [1,\phi_j]\cap\mathbb{Z}}} \prod_{j=1}^{i} a_{\phi_j,\psi_j}$$

where

$$\begin{aligned} b_0 &= 1, \\ b_1 &= 3, \\ b_i &= 2b_{i-1} + b_{i-2} \text{ for } i > 1, \end{aligned}$$

and

$$\begin{aligned} a_{n,y} &= 0, \text{ for } n < y+1 \\ a_{y+1,y} &= 2 \\ a_{y+2,y} &= 2 \\ a_{n,y} &= 2a_{n-1,y} + a_{n-2,y}, \text{ for } n > y+2 \end{aligned}$$

Explicitly,

$$b_n = -\frac{1}{2+2\sqrt{2}} \cdot \left(\frac{1}{-1-\sqrt{2}}\right)^n - \frac{1}{2-2\sqrt{2}} \cdot \left(\frac{1}{-1+\sqrt{2}}\right)^n, n \geq 0$$

which is the same b_n in section 3, and

$$a_{n,y} = \begin{cases} 0, \text{if } n \leq y \\ -\frac{4+3\sqrt{2}}{2+\sqrt{2}} \left(\frac{1}{-1-\sqrt{2}}\right)^{n-y} + \frac{4-3\sqrt{2}}{2-\sqrt{2}} \left(\frac{1}{-1+\sqrt{2}}\right)^{n-y}, \text{if } n > y \end{cases}$$

Now let us show all of this...

As before in Section 3, the argument is to let b_q represent the tail of zeros of length q (which corresponds to the tail 0^* in our functional regular expression), i the number of blocks of the form (0^*01^*1) (as per our functional regular expression), ϕ_j is the length of the jth block of the form (0^*01^*1) , ψ_j is the number of 1's in the jth block of the form (0^*01^*1), and $a_{x,y}$ is the total number of $C-Y-S$ block patterns corresponding to a functional block of the form (0^*01^*1) of length x and with y 1's. Given these, the reader can see why the formula must hold true. It is a similar construction to the formula of $\mathcal{A}(n,k)$ in Section 3.

Now we must justify the formulas themselves...

For b_i, note that it represents the number of C-Y-S block patterns represented by a 0-1 block pattern of the form 000...0 where there are precisely i 0's. To obtain the recursive formula, we note that the C-Y-S block patterns of this type are represented by the C-Y-S regular expression

$$S^*\{CC^*SS^*, YY^*SS^*\}^*\{CC^*, YY^*, \phi\}$$

where ϕ is the empty set. Notice that this is the same as in Section 3 where b_i represented the number of A-B-S block patterns represented by a 0-1 block pattern of the form 000...0 where there are precisely i 0's – the regular expression is equivalent by replacing C with A and Y with B. Thus, this is why we get the same formula for b_i here as we did in Section 3 and do not need to show any new details.

As for $a_{n,y}$, consider that it represents the number of C-Y-S patterns corresponding to a 0-1 functional pattern of length n and with y 1's, and looks like 00000...00011111...111. We fix $y = j$. Then for each $n \in \mathbb{N}$, a regular expression is as follows:

$$S^*\{CC^*SS^*, YY^*SS^*\}^*\{C(YCY...), Y(CYC...)\}$$

where for the last term the patterns C(YCY...) or Y(CYC...) are of length $j + 1$.

Our generating function is thus:

$$G(x) = \sum_{i=0}^{\infty} a_{i,j}x^i = \frac{1}{1-S} \cdot \frac{1}{1 - \left(\frac{C}{1-C} \cdot \frac{S}{1-S} + \frac{Y}{1-Y} \cdot \frac{S}{1-S}\right)} \cdot 2x^{j+1}$$

.

Setting $A = B = S = x$ where x is an indeterminate, doing some algebra the reader can confirm that

$$G(x) = \frac{2x^{j+1}(1-x)}{1-2x-x^2}$$

and so

$$a_{n,j} - 2a_{n-1,j} - a_{n-2,j} = \begin{cases} 0, & \text{if } n < j+1 \\ 2, & \text{if } n = j+1 \\ -2, & \text{if } n = j+2 \\ 0, & \text{if } n > j+2 \end{cases}$$

where the technique to get read off this recursion formula has already been discussed. Then we get what we claimed about the $a_{n,y}$'s earlier.

To get the explicit formula, note that we cannot do partial fractions on $G(x)$ directly, however we can notice that the values of $a_{n,y}$ is simply a translation in the positive direction along $\mathbb{R}$ of the values $a_{n,0}$ in its first index by y – the second index. That is, given that the recursive formula is:

$$\begin{aligned} a_{n,y} &= 0, \text{ for } n < y+1 \\ a_{y+1,y} &= 2 \\ a_{y+2,y} &= 2 \\ a_{n,y} &= 2a_{n-1,y} + a_{n-2,y}, \text{ for } n > y+2, \end{aligned}$$

one should notice that y determines when the sequence formally starts to have non-zero values and thus becomes non-trivial. With that in regard, $a_{n,y}$ is essentially a class of sequences in terms of y which all have essentially the same sequence of values for all $n \in \mathbb{N}$, modulo the zeroes at the beginning. Hence the notion of the second index y being a "translation" factor. The translation factor rule is formally

$$a_{n,j} = a_{n+i,j-i}$$

With this in mind, consider that we have the generating function in terms of j which is the second index. Setting $j = 0$ gives a generating function for $a_{n,0}$ which we can do partial fractions on:

$$\begin{aligned} G(x) = \frac{2x(1-x)}{1-2x-x^2} &= 2 + \frac{6x-2}{1-2x-x^2} = 2 + \frac{(4+3\sqrt{2})/\sqrt{2}}{(-1-\sqrt{2})-x} - \frac{(4-3\sqrt{2})/\sqrt{2}}{(-1+\sqrt{2})-x} \\ &= 2 - \frac{4+3\sqrt{2}}{2+\sqrt{2}} \cdot \frac{1}{1-\frac{1}{-1-\sqrt{2}}x} + \frac{4-3\sqrt{2}}{2-\sqrt{2}} \cdot \frac{1}{1-\frac{1}{-1+\sqrt{2}}x} \\ &= 2 - \frac{4+3\sqrt{2}}{2+\sqrt{2}} \sum_{i=0}^{\infty} \left(\frac{1}{-1-\sqrt{2}}\right)^i x^i + \frac{4-3\sqrt{2}}{2-\sqrt{2}} \sum_{i=0}^{\infty} \left(\frac{1}{-1+\sqrt{2}}\right)^i x^i \end{aligned}$$

Hence we get that

$$a_{n,0} = \begin{cases} 0, \text{if } n \leq 0 \\ -\frac{4+3\sqrt{2}}{2+\sqrt{2}} \left(\frac{1}{-1-\sqrt{2}}\right)^n + \frac{4-3\sqrt{2}}{2-\sqrt{2}} \left(\frac{1}{-1+\sqrt{2}}\right)^n, \text{if } n > 0 \end{cases}$$

Hence by the translation rule,

$$a_{n,y} = \begin{cases} 0, \text{if } n \leq y \\ -\frac{4+3\sqrt{2}}{2+\sqrt{2}} \left(\frac{1}{-1-\sqrt{2}}\right)^{n-y} + \frac{4-3\sqrt{2}}{2-\sqrt{2}} \left(\frac{1}{-1+\sqrt{2}}\right)^{n-y}, \text{if } n > y \end{cases}$$

which is what was required.

A simulation of the quantity

$$R(n) := E(n)/n$$

is done using the following Python script:

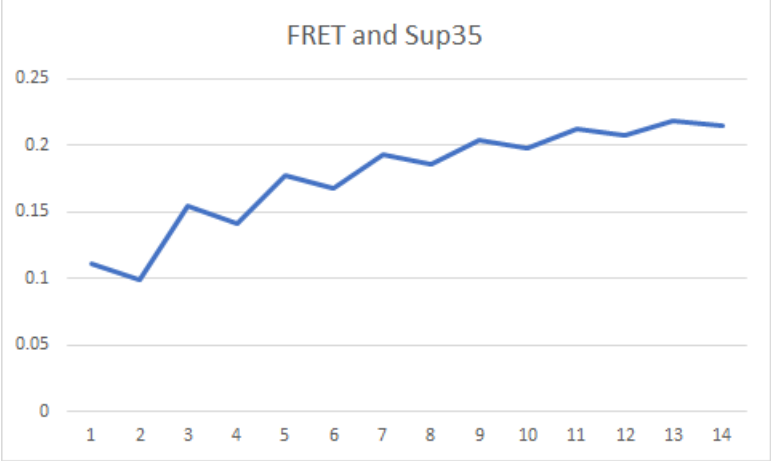
```
#!/usr/bin/python
import itertools
import math

#The binomial coefficient nCk...
def binom(n, k):
    if k > n or k < 0:
        return 0
    a = math.factorial(n)
    b = math.factorial(k)
    c = math.factorial(n - k)
    return a / (b * c)

def a(n,y):
    if n <= y:
        return 0
    x = math.sqrt(2)
    result = - ((4 + 3 * x) / (x + 2) * 1.0) * pow((1.0/(-1.0-x)),n-y) -
```

```
    ((4 - 3 * x) / (2 - x) * 1.0) * pow((1.0 / (x - 1)), n - y)
    return result

def b(n):
    x = math.sqrt(2)
    result = - (1.0 / (2 * x + 2)) * pow((1.0 / (-1 - x)), n) -
    (1.0 / (2 - 2 * x)) * pow((1.0 / (x - 1)), n)
    return result

def product(lst):
    val = 1.0
    for item in lst:
        val *= item
    return val

#This is the A(n,k) function I defined in my writeup...
def A(n, k):
    result = 0.0
    for q in xrange(0, n - k + 1):
        innerSum = 0.0
        for i in xrange(1, k + 1):
            all_lists = itertools.product(xrange(1, n - q + 1), repeat=i)
            lists_of_psi = [lst for lst in all_lists if sum(lst) == k]
            all_lists = itertools.product(xrange(2, n - q + 1), repeat=i)
            lists_of_sum_n=[lst for lst in all_lists if sum(lst)==n - q]
            for lst in lists_of_sum_n:
                for lst2 in lists_of_psi:
                    fin = zip(lst,lst2)
                    a_lst = [a(x,y) for x,y in fin]
                    innerSum += product(a_lst)
        result += b(q) * innerSum
    return result

#The expected number of fluorescing locations in a length n
#amyloid, according to formula I derived, is:
def expected_value(length):
    result = 0.0
    for i in xrange(0, length / 2 + 1):
        result += i * A(length, i)
    result /= pow(3.0, length)
    return result

if __name__ == '__main__':
    for i in xrange(1, 1000):
        r = expected_value(i) / i
        print "{}: {}".format(i, r)
```

which yields the following results up to $n = 15$ before becoming computationally inviable:

```
2: 0.111111111111
3: 0.0987654320988
4: 0.154320987654
5: 0.141563786008
6: 0.177411979881
7: 0.168005748253
8: 0.192805974699
9: 0.185721462966
10: 0.203698623177
11: 0.198284219013
12: 0.211710236884
13: 0.207558366105
14: 0.217786065517
15: 0.214599388418
```

which when graphed appears as the following:



The data provided is qualitatively inconclusive to give an estimate of the steady-state value of $R(n)$.

References

[1] miniparser (https://math.stackexchange.com/users/99402/miniparser). Proof for number of ways to select k non-consecutive elements from n consecutive terms. Mathematics Stack Exchange. URL:https://math.stackexchange.com/q/1347532 (version: 2015-07-02).

[2] user63181. Sum of $k\binom{n}{k}$ is $n2^{n-1}$. Mathematics Stack Exchange. URL:https://math.stackexchange.com/q/683740 (version: 2014-02-20).