BIM 105
Fall 2024

Lab 3: Sanger Sequencing

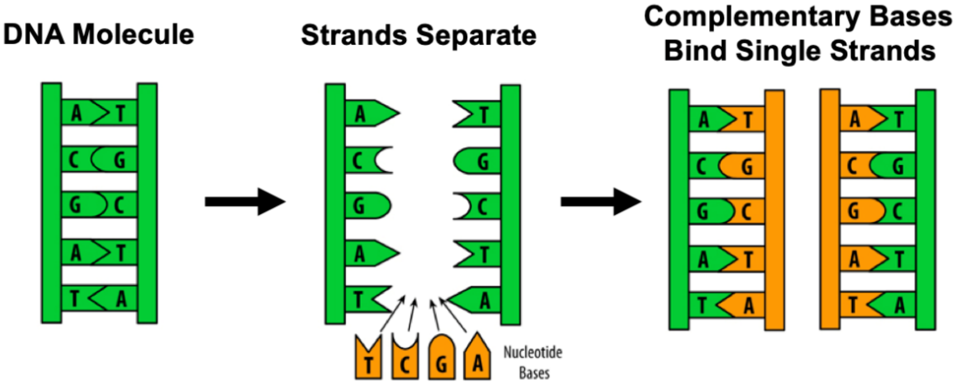
Instructions: Read the prompts and use MATLAB to answer the questions. Submit your solution as a single script that outputs all answers to the Command Window. A template script is available on Canvas.

Part 1. Background

The first practical method for determining the sequence of a DNA molecule was developed by Frederick Sanger in the 1970s. He and his colleagues won a Nobel prize in Chemistry in 1980 for their contributions (Sanger had also won a Nobel prize in 1958 for his work in determining the amino acid sequence of insulin – he's one of only four people to win two Nobel prizes). To this day, the Sanger sequencing method of DNA is still in use. Other methods, however, which are faster and cheaper, have since replaced much of Sanger sequencing's use. In this lab, we will examine Sanger sequencing and some of its stochastic issues. Our description of the method itself will be simplified, but the investigated issues are realistic.

To describe Sanger sequencing, we will first need to review some basic facts about DNA replication. DNA molecules normally contain two intertwined strands (the famous double-helix) built up from nucleotides A, T, C and G. The DNA contains two strings which are complementary: where one strand has nucleotide A, the other has nucleotide T, and vice versa; and where one strand has a C, the other strand has a G, and vice versa. For example, if one strand has the sequence AACTGTCCG, the other strand would have sequence TTGACAGGC.

At a crude level, DNA replication proceeds as follows:



(image from Visionlearning, Inc.[1])

A two-stranded DNA molecule replicates in a soup of individual A, T, C, and G nucleotides that will be used to build the new DNA molecule. During replication, the two strands of the DNA

[1] https://www.visionlearning.com/en/library/Biology/2/DNA-II/160

molecule separate, creating two separate single strands of DNA. A molecule called a *DNA polymerase* “reads” one of the strands, which is usually called the “template,” from one end to the other, and builds up a complementary strand for it. For example, if one strand is ACCTGTCCG, the polymerase reads A at the start (left end); and then grabs one of the T’s in the soup and attaches it across from the first A. At that point we have the two attached strings:

```
ACCTGTCCG
|
T
```

Next, the polymerase reads the second character, C, and grabs a G in the soup and attaches it to the C in the first DNA strand, and to the T in the second strand. At that point we have:

```
ACCTGTCCG
||
TG
```

Continuing this way, sequentially adding one more nucleotide at each step, the final result is a copy of the two-stranded DNA molecule we started with. Meanwhile, the same process is proceeding on the other strand from the original DNA molecule. The result is two copies of the original two-stranded DNA molecule.

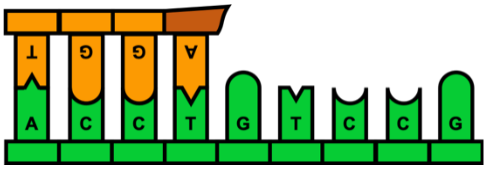
To simplify the description and modeling of Sanger sequencing, we start with a description of what we call a single experiment. We first separate the two strands of a DNA molecule. Suppose we now have isolated one of those strands; let’s call it S. The Sanger method makes four passes:

- One pass to learn some positions where nucleotide T is in S
- One pass to learn some positions where nucleotide A is in S
- One pass to learn some positions where nucleotide G is in S
- One pass to learn some positions where nucleotide C is in S

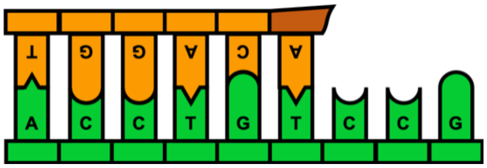
Let’s look at the first pass. In a test tube, a soup is created containing DNA polymerase, good copies of C, T and G, and a mix of good and “defective” copies of A. When the DNA strand S is added to the soup, the polymerase starts doing its job, starting at one end of the strand, making a new strand that is complementary to S. When it needs an A, it grabs one at random from the A’s in the soup; when it needs a G, it grabs one at random from the G’s in the soup; and so on. As long as good nucleotides are grabbed and used, the replication proceeds exactly as before. In particular, when S contains a T (so that an A is added to the growing strand) if a good A is added, the polymerase continues building up the new strand. But if a defective copy of A is used, the replication stops at that point.

We now separate the two strands and measure their lengths in order to determine where a defective nucleotide was incorporated (note that lengths of DNA fragments can be measured easily and precisely by lab techniques such as gel or capillary electrophoresis). There will be a short and a long string. Suppose the short string has length four. We can then deduce that there is a defective A at position four of the new strand, and hence there is a T at position four of the original DNA strand S.

Repeating the experiment many times allows us to learn many positions where a T occurs in the original molecule. Which positions are learned, however, depends on whether or not a defective nucleotide was incorporated at each specific T in the original strand. For example, if S is the sequence ACCTGTCCG, a defective A could be incorporated at the first T to give the following:



Or, if a good copy of A is incorporated at the first T, it's possible that a defective A is incorporated at the second T in S, which would give:



In other words, which T is found in an experiment is a *random variable*, and as such, we can apply probability theory to learn something about Sanger sequencing.

Part 2. The Role of Probability

The primary variable for consideration as an experimentalist is the *proportion of defective nucleotides* in the soup. For instance, in our example above, if *all* A's are defective, every experiment would result in a defective A being used at the first T of the original strand. Thus, no T's *after* the first T would be found. Conversely, if the soup of nucleotides is comprised of all "good" A's, the DNA strand would be replicated perfectly each time. No T's would be found.

The Proportion of Non-Defective Nucleotides: "p"

To further analyze how the ratio of good to defective A's affects the outcomes of the experiments, we introduce a parameter p, which denotes the proportion of A copies in the soup which are good. Recall that we assumed the number of nucleotides in the soup is so large that we don't worry about nucleotides being consumed during the sequencing reaction. This assumption is encapsulated into our model by keeping p fixed as experiments are being repeated throughout the Sanger reaction.

When the polymerase reaches a T in the original strand, it will sample a copy of A from the pool to continue the complementary strand. If the proportion of good copies of A in the pool is p, then the probability that the polymerase uses a *defective* copy of A is $1-p$. This also means the probability that an experiment ends on the first T in S is $1-p$.

Consider next the possibilities for outcomes at second T in S. The probability that the experiment ends on the *second* T in S would be $p(1-p)$, because in order to end on the second T, the experiment first needs to use a good nucleotide at the first T. This event occurs will probability p. Then the probability to use a defective nucleotide on the second T (independent of the first T) is $1-p$. The probability of an experiment ending on the second T is subsequently the product of these two probabilities.

This formulation can then be generalized to express the probability of an experiment ending on the k-th T in S as:

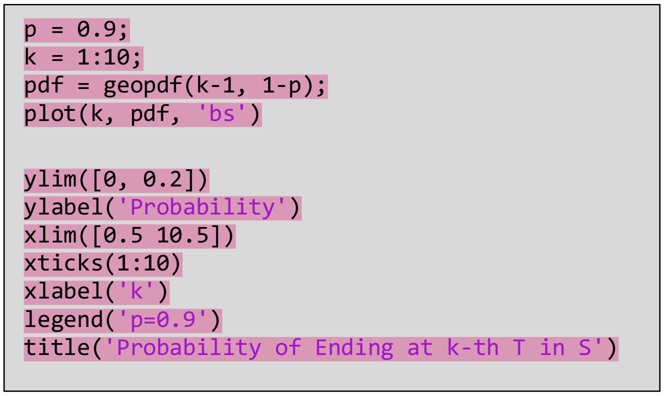
$$\mathrm{P}(\text{ending at } k\text{th T}) = \; p^{k-1}(1-p)$$

Now, compare this expression to the Geometric distribution covered in class (the number of failures before a success, given the probability of a success). The Geometric distribution is typically written as $\mathrm{P}(k) = (1-p)^{k-1}p$, where p is the probability of success. In our problem, however, what we've defined as a "success" is actually an experimental "failure" in that the incorporation of a defective nucleotide is what ends an experiment. Considering that $p_{\text{success}} = 1-p$, we can rewrite the Geometric distribution with

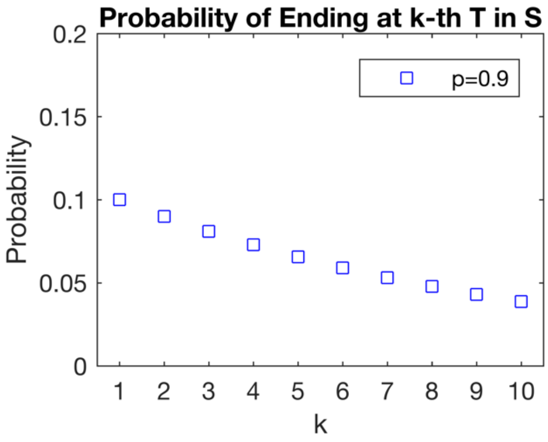
$$\mathrm{P}(k) = (1-p_{\text{success}})^{k-1}p_{\text{success}} = p^{k-1}(1-p),$$

which is the formulation written above in the context of our problem. For the rest of this lab, we will use p to denote only the proportion of non-defective nucleotides. The distribution of sequencing fragments is thus $\mathrm{P}(\text{ending at } k\text{th T}) = \mathrm{Geom}(1-p)$.

Let's use MATLAB to plot this distribution for several values of p over a range of values for k. To do this, we define an interval of values for k, then we use the method **geopdf** to compute probabilities. Note, as well, the value of p_{success} that we give to **geopdf** should be $1-p$ as MATLAB uses the same Geometric distribution notation from class. For instance, if p is set to 0.9, we could plot the distribution with the following code:



```
p = 0.9;
k = 1:10;
pdf = geopdf(k-1, 1-p);
plot(k, pdf, 'bs')

ylim([0, 0.2])
ylabel('Probability')
xlim([0.5 10.5])
xticks(1:10)
xlabel('k')
legend('p=0.9')
title('Probability of Ending at k-th T in S')
```



This shows us that the probability is highest at $k=1$ and drops as k gets larger. Note that when we call **geopdf** here, we use $k-1$ as **geopdf** interprets the input as the number of successes before failure (i.e., for $k=0$, there would be zero “good” nucleotides incorporated before the first defective one).

<u>Question 1</u>: Plot the probability distribution over values of k for $p = 0.5, 0.7$, and 0.95. What trends do you observe as p gets closer to 1?

Solution:

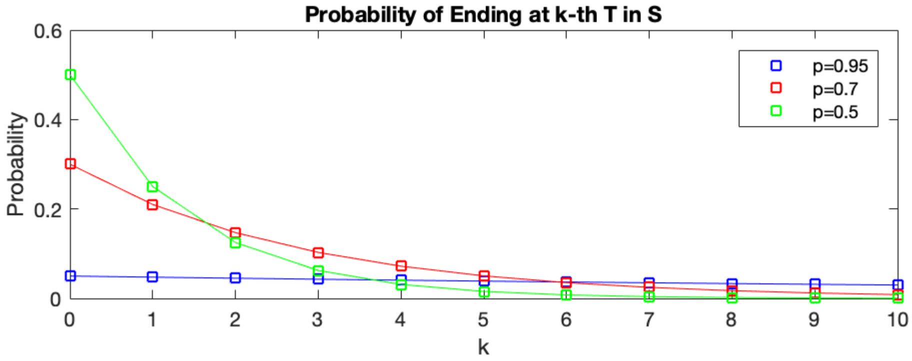
```
k = 0:10;
pdf9 = geopdf(k, 1-0.95);
pdf7 = geopdf(k, 1-0.7);
pdf5 = geopdf(k, 1-0.5);
plot(k, pdf9, '-bs')
hold on
plot(k, pdf7, '-rs')
plot(k, pdf5, '-gs')

ylim([0, 0.6])
ylabel('Probability')
xlabel('k')
legend('p=0.95', 'p=0.7', 'p=0.5')
title('Probability of Ending at k-th T in S')
```

figure;

This command creates a new figure window for plotting. Each time figure is called, a new empty figure window is generated, where subsequent plots will be displayed.

'-bs' specifies the plot style: a solid line (-), blue color (b), and square markers (s).

If p is small, then most experiments will stop early on the fragment. Conversely, if p is large, experiments will tend to a longer duration until a defective nucleotide is included. (Harder to include a defective nucleotide)



The probabilities diminish as *k* gets large.

<u>Question 2</u>: Which p from Question 1 gives the highest probability of an experiment ending at the 10th T in S?

Solution:
P(k=10 | p=.95) = 0.0299
P(k=10 | p=0.7) = 0.009
P(k=10 | p=0.5) = 0.0005
p=0.95 gives the highest probability.

<u>Question 3</u>: What trend in the distributions do you observe as k gets large? How might this affect the sequencing of very large DNA fragments?

Solution: The distribution's probabilities diminish at large k. It's unlikely to detect nucleotides that are far away from the priming site.

Part 3. Optimizing p

From Part 2, we've seen that the proportion of "good" nucleotides in the sequencing soup is an important parameter affecting the outcome of the experiment. If p is small, then most experiments will stop early on the fragment. Conversely, if p is large, experiments will tend to a longer duration until a defective nucleotide is included.

Suppose that you have a string of DNA, and you want to estimate how many experiments it should take to "detect" the k-th T. The expected number of experiments is just the inverse of the probability of a certain outcome; in other words:

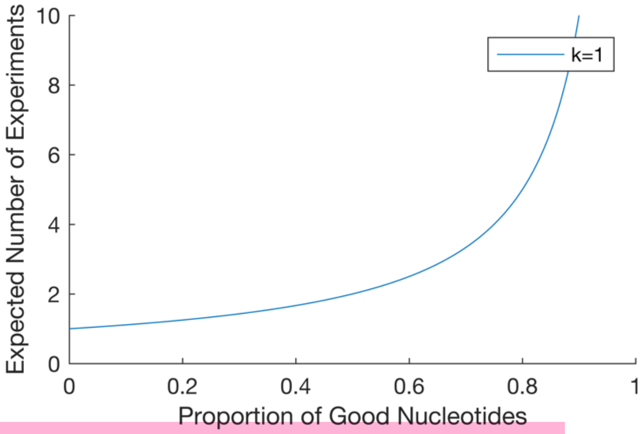
$$\text{Expected Number of Experiments to Detect } k'\text{th T} = \mathrm{E}\left[N_{\text{experiments}}\right] = \frac{1}{p^{k-1}(1-p)}$$

For example, if you're interested in how many experiments it will take to detect the first T given that $p = 0.5$, we could plug in $k = 1$ and $p = 0.5$ to get:

$$\mathrm{E}\left[N_{\text{experiments}}\right](k = 1, p = 0.5) = 2$$

This should make some intuitive sense. If half of the nucleotides are defective, it should take on average two experiments before an experiment terminates at the first T. Instead of assuming a specific value of *p*, however, let's use MATLAB to plot how the expected number of experiments to detect the first T (*k*=1) in the DNA is effected by the parameter *p*. Slightly altering the code from Part 2 produces the desired result:

```
p = 0:0.001:1;
k = 1;

n_exp_1 = 1./geopdf(k-1, 1-p);

plot(p, n_exp_1)
xlim([0 1])
xlabel('Proportion of Good Nucleotides')
ylabel('Expected Number of Experiments')
ylim([0 10])
legend('k=1')
```

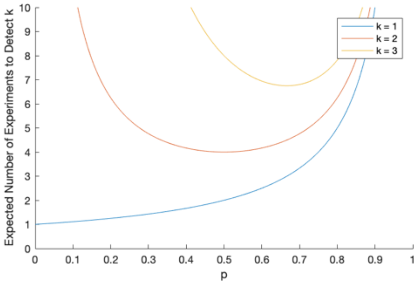


From this plot, we can see that when $p = 0$, the expected number of experiments reaches a minimum value of 1. This should also make some intuitive sense – when all nucleotides are defective, every experiment will end at the first T!

Question 4: Plot the expected number of experiments to detect the k-th T in S over a range of p for $k = 1$, $k = 2$, and $k = 3$ (each value of k should produce a distinct curve).

Solution:

```
p = 0:0.001:1;
n_exp_1 = 1./geopdf(0, 1-p);
n_exp_2 = 1./geopdf(1, 1-p);
n_exp_3 = 1./geopdf(2, 1-p);
f = figure;
hold on
plot(p, n_exp_1)
plot(p, n_exp_2)
plot(p, n_exp_3)
xlim([0 1])
ylim([0 10])
```



Question 5: For each of the above values of k, describe the trends you observe in how the expected number of experiments changes over values in p.

Solution:

The expected number of experiments follows a curve where if *p* is too low or too high, the expected number of experiments increases. For each *k*, there seems to be a *p* which minimizes the number of experiments.

find a p so that we can do least experiment

Consider that we have a DNA strand with k T's, and we want to optimize p such that the number of experiments to detect the last T is minimized. In other words, we want to determine what value of p minimizes our equation for the expected number of experiments. Normally, we'd take the derivative of the function and find where it is equal to zero in order to find possible minima. We could do this by hand, but MATLAB can also be used to do the derivation for us through the use of symbolic variables. The **diff** and **solve** commands can be combined to give us a formula for p that minimizes the number of experiments to detect the k-th T:

k and p as symbolic variables. Symbolic variables in MATLAB are used in symbolic expressions for mathematical functions, allowing for algebraic manipulations.

```
syms k p
solve(diff(1/(p^(k-1)*(1-p)))==0, p)
```

which outputs the following formula for the optimal p:

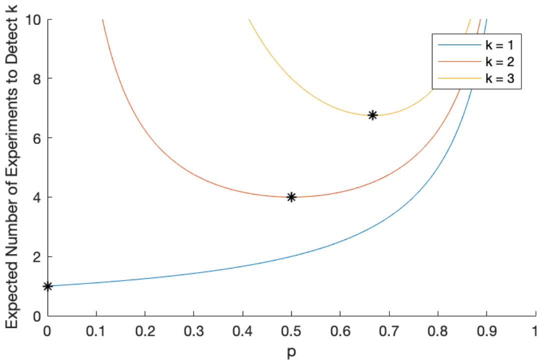
```
ans = (k - 1)/k
```

means at p=(k-1)k, we can minimize expected number of experiment

Question 6: Use the formula we found to calculate the optimal p for $k = 1$, $k = 2$, and $k = 3$. For each optimal p, calculate the expected number of experiments to detect the k-th T in S. Plot these values against the curves you produced in Question 5. How do these points relate to the curves you produced in Question 5?

Solution:

```
p = 0:0.001:1;
k = 1:3;
p_opt = (k-1)./k;
n_opt = [1./geopdf(0, 1-p_opt(1)),...
         1./geopdf(1, 1-p_opt(2)),...
         1./geopdf(2, 1-p_opt(3))];
n_exp_1 = 1./geopdf(0, 1-p);
n_exp_2 = 1./geopdf(1, 1-p);
n_exp_3 = 1./geopdf(2, 1-p);
figure(3)
hold on
plot(p, n_exp_1)
plot(p, n_exp_2)
plot(p, n_exp_3)
plot(p_opt, n_opt, 'k*')
xlim([0 1])
ylim([0 10])
xlabel('p')
ylabel('Expected Number of Experiments to Detect k')
legend('k = 1', 'k = 2', 'k = 3')
```



./ - This is the element-wise division operator. It divides each element of the numerator by the corresponding element of the denominator. If you use / without the dot in MATLAB, it implies matrix division, which is different and requires compatible matrix dimensions for it to work.

The points correspond to the minima of the curves from Q5.

Part 4. Simulations

Let’s perform some simulations of Sanger sequencing experiments to (1) check our understanding and (2) consider some practical limits of the experiment.

To simulate a single Sanger experiment, all we need to do is call **`geornd(1-p)`** which returns a geometric random variable describing the number of good nucleotides incorporated before a bad nucleotide was used. Recall that MATLAB syntax assumes that the input value is the probability of *failure*, so we use the argument $1-p$ when we call the function if p is the proportion of “good” nucleotides. To determine which k the experiment stops on, we add 1 to the output of **`geornd`**, and we also need to threshold the outcomes to the length of the DNA (i.e., the outcome k cannot be greater than N_k if there are N_k T’s in the DNA). Putting this together, to simulate 100 Sanger experiments with $p = 0.5$ on a DNA with 30 T’s, we would use:

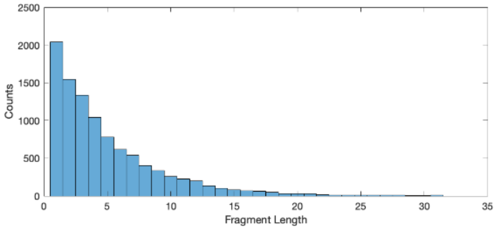
```
p = 0.5;    % Proportion of good nucleotides
Nk = 30;    % Length of the DNA
N_experiments = 100;
outcomes = NaN(N_experiments, 1);% Initialize outcomes array, 10x1 NaN
for i = 1:N_experiments
    outcomes(i) = min([geornd(1-p)+1, Nk+1]);% terminate position, max length
                                             (didn’t terminate in this DNA)
end
display(outcomes)
```

Outcomes in which $k > N_k$ are all assigned the value of $N_k + 1$ to indicate that the experiment read through the entire DNA sequence without using a defective nucleotide. The other outcomes tell us at which k the experiment stopped.

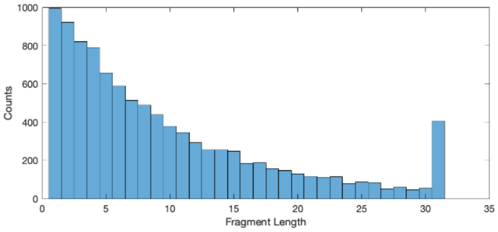
Question 7: Simulate 10,000 Sanger experiments for a DNA strand with 30 T's with two values of p; $p = 0.8$ and $p = 0.9$. Plot the distribution of fragment lengths for each case using the function **`histogram`**. Which value of p do you think works better in this experiment?

Solution:

```
p = 0.8;
Nk = 30;
N_experiments = 10000;
outcomes = NaN(N_experiments, 1);
for i = 1:N_experiments
    outcomes(i) = min([geornd(1-p)+1, Nk+1]);
end
histogram(outcomes)
```



```
p = 0.9;
Nk = 30;
N_experiments = 10000;
outcomes = NaN(N_experiments, 1);
for i = 1:N_experiments
    outcomes(i) = min([geornd(1-p)+1, Nk+1]);
end
histogram(outcomes)
```



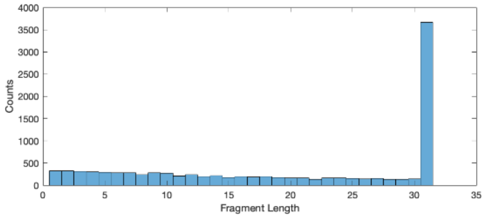
p=0.9 provides a greater coverage of fragments over the transcript, and a greater number of counts for larger *k*'s.

Question 8: Repeat the simulation from Question 7, but this time use the optimal p from the formula derived in Part 3. How does the distribution of fragments change compared to using $p = 0.8$ or 0.9?

optimal p=(k-1)k that can minimize expected number of experiment

Solution:

```
Nk = 30;
p = (Nk-1)/Nk;
N_experiments = 10000;
outcomes = NaN(N_experiments, 1);
for i = 1:N_experiments
    outcomes(i) = min([geornd(1-p)+1, Nk+1]);
end
histogram(outcomes)
xlabel('Fragment Length')
ylabel('Counts')
```



The distribution of fragments over the DNA is much more uniform. We see as well that most experiments produce fragments that read through the entire DNA.

Indicate that the experiment read through the entire DNA sequence without using a defective nucleotide

cannot find

We've seen that fragments of any k could be produced from experiments using some p. However, not all fragment lengths will be produced at equal rates in a pool of experiments. In practice, there is a limitation on the detection of fragments that result from Sanger experiments. Only fragments that exceed some critical frequency, D, can be detected after the experiments are carried out.

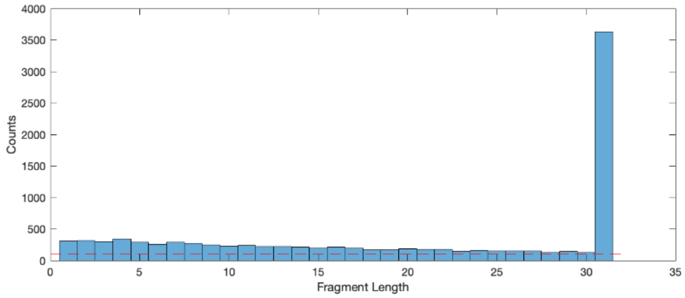
For instance, if the detection limit is $D = 0.01$, only fragment lengths which comprise 1% or more of the fragment pool could be detected. If there are 10,000 fragments from a pool of experiments, only locations which produced 100 or more fragments in the experiment could be detected. If a specific location only produced 1 (or 5, or 50) fragment, it would not be detectable in practice despite the sequencing "event" having occurred.

Question 9: Assume $D = 0.01$. For your distribution in Question 8, can all fragment lengths in the experiment be detected? What challenge emerges as the length of the DNA gets long (longer than ~100)?

Solution:

(additional coding is not necessary, although can be helpful to plot D against the histogram to compare which lengths have enough counts)

```
Nk = 30;
p = (Nk-1)/Nk;
N_experiments = 10000;
outcomes = NaN(N_experiments, 1);
for i = 1:N_experiments
    outcomes(i) = min([geornd(1-p)+1, Nk+1]);
end
histogram(outcomes)
hold on
plot([0, 32], [100 100], '--r')
xlabel('Fragment Length')
ylabel('Counts')
```



As a DNA gets long (and the optimal *p* gets closer to 1), the relative abundance of fragments for each *k* drops to smaller and smaller proportions. This depletion may drop below the detection limit if the DNA is sufficiently large.