9 Introduction to directed evolution: single substitutions & error-prone PCR

Summary. Directed evolution involves creating random variations in the target protein followed by identification of the improved variants. This chapter explain how to calculate the numbers of possible variants (the numbers are astronomical) and considers three ways that one can find an improved variant with one amino acid substitution: 1) creating and testing every possible single substitution variant, 2) error prone PCR to create an incomplete set of random substitutions and 3) making substitutions only at the regions most likely to yield improvements. This chapter also explains degenerate codons, oversampling needed during screening and the incompleteness of error-prone PCR libraries.

Key learning goals

- The number of possible amino acid substitutions depends on the number of substitutions, the number of replacement amino acids, and number of amino acids in the chain. This chapter focuses on single substitutions.
- Saturation mutagenesis at a single codon creates variants where each of the nineteen other amino acids replace the wild type amino acid. Testing these variants identifies any with improved properties. Saturation mutagenesis in the substrate-binding site is a good approach to find substitutions that reshape the binding site to expand the substrate range or alter the selectivity.
- Degenerate codons can encode all twenty amino acids or a subset of amino acids. Screening a mixture of variants requires oversampling to ensure that at least one of each type of variant has been tested.
- Error-prone PCR is a good approach to make nucleotide substitutions over a large section of DNA. Error-prone PCR libraries are incomplete due to DNA polymerase bias and codon bias, so are best suited for problems like increasing protein stability where many substitutions can yield improvements. Random methods also favor locations outside the active site, so they are inefficient in finding substitutions within the substrate-binding regions.

9.1 Introduction

Directed evolution is the recursive generation of random variations and selection of improved variants, usually by screening. Directed evolution is a simple idea. Over forty years ago Eigen & Gardiner (1984) suggested a set of computer instructions for directed evolution.

```
10 PRODUCE A MUTANT SPECTRUM OF SELF-REPRODUCING TEMPLATES
20 SEPARATE AND CLONE INDIVIDUAL MUTANTS
30 AMPLIFY CLONES
40 EXPRESS CLONES
50 TEST FOR OPTIMAL PHENOTYPES
60 IDENTIFY OPTIMAL GENOTYPES
70 RETURN TO 10 WITH A SAMPLE OF OPTIMAL GENOTYPES
```

Figure 9.1. Suggested instructions for directed evolution.

In spite of this simplicity, it is hard to put in practice because the number of possible variants is so vast. One can never make all possible variants, so the challenge is to choose the subset of variants that is most likely to show the desired improvements. In this chapter, we consider the simplest goal of directed evolution – find one amino acid substitution that improves the protein. Sometimes a single substitution is enough. For example, Met to Ala substitution stabilized the laundry protease to bleach (Estell et al., 1985). In other cases, multiple substitutions are needed to create larger improvement or several different kinds of improvements. Strategies to find multiple substitutions will be considered in the next chapter.

9.2 Number of possible substitution variants

The most common change in a protein is substitution of one amino acid with another. An important question is how many substitutions are possible. The answer depends on 1) the number of substitutions (k), the number of replacement amino acids (n), and number of amino acids in the chain (L). It is useful to break the problem into two parts: how many replacements are possible and how many locations are possible. The number of possible protein variants with k substitutions, V_k, depends on the product of the number of location possibilities and the number of replacement possibilities, eq. 9.1.

$$V_k = \text{location possibilities} * \text{replacement possibilities} \quad (9.1)$$

Review: permutations and combinations

Permutations relate to the order of the elements, while combinations refer to selecting elements from a group. Situations where order matters are permutations, while situations where the order does not matter are combinations. A lock sequence such as 5-27-37 is a permutation because the numbers must be entered in that order. (Confusingly, these locks are called combination locks, while permutation locks would be more accurate.) The other thing to consider in the calculating the number of possibilities is whether items can repeat or not. In the lock sequence, the numbers can repeat. (Review of permutations and combinations: Kreyszig,

1972)

Permutations with repetition. The lock sequence example is a permutation where repetition is possible. To determine the number of possible lock sequences, there are say, 39 numbers to choose for the first number, then 39 again for the second number and again 39 for the third. The total number of choices is 39*39*39 = 59,319. In general, for a permutation with repetition, there are n things to choose from and k choices each time so there are n^k possibilities. In protein engineering, the number of possible amino acid replacements at specified locations is a permutation with repetition. For two locations, x and y, an alanine at position x is different from an alanine at position y. Since order is important (substitution at x differs from substitution at y), it is a permutation. It is further a permutation with repetition since both x and y may contain an alanine.

$$\text{number of possibilities} = n^k$$

For example, number of possible double-substitution variants of a dipeptide is 19^2 because there are nineteen possible replacement amino acids and two positions that require a choice.

Permutations without repetition. When choices are used up upon being chosen, then repetition is not possible. For example, choosing an order for sixteen billiard balls is an example of a permutation without repetition. The there are 16 choices at first, then 15 remaining balls and so on. There are 16! different orders that these balls could be chosen (16! ~$21*10^{12}$). In general, there are n things to choose from the first time, n-1 the second time, etc. so there are n! choices in total. In some cases, one may choose fewer than the maximum possible. Then the choices are:

$$\text{number of possibilities} = \frac{\text{total number}}{\text{number not used}} = \frac{n!}{(n-k)!}$$ where k is the number choices made.

For example, one could choose only three billiard balls from the sixteen possible. In this case, k = 3 and n = 16, so the number of possibilities is 16!/13! = 16*15*14 = 3,360.

Combinations without repetition. Here the order does not matter, but choices cannot be repeated. For example, one could choose billiard balls again, but without regard to the order in which they were chosen. We already have the equation above for the case were order does matter, we just need to reduce it by the number of ways the objects could be in ordered. k objects can be ordered in k! (k for the first choice, k-1 for the second choice, etc.), So the number of combinations without repetition when choosing k objects from a total of n is:

$$\text{number of possibilities} = \frac{\text{total number}}{\text{order does not matter} * \text{number not used}} = \frac{n!}{k!(n-k)!} \qquad (9.2)$$

In protein engineering, the number of locations in a protein is a combination without repetition. It does not matter what order they are chosen, but once chosen, they cannot be chosen again. The choice x and y is the same as the choice y and x, but the choice x and x is

impossible.

We will not consider the fourth possibility, combinations with repetition since it does not occur in the context of protein engineering.

Replacement possibilities = n^k

The number of replacement possibilities at k locations with n replacements is n^k. In the general case, n is the number of objects to choose from (amino acids in this case) and k is the number of choices to be made (positions mutated in this case). For example, replacing the amino acids at positions x and y with new amino acids creates $19^2 = 361$ variants. If you include the original amino acids, then number of possibilities is $20^2 = 400$. This value includes the 361 two-substitution variants, 38 single-substitution variants and one wild-type enzyme. In two more examples, the number of nucleotide sequences of length 6 (k = 6) are possible from a set of 4 nucleotides (n= 4) is 4^6 or 4,096 and the number of possible amino acid sequences of length 300 (k = 300; n = 20) is 20^{300} or $\sim 2 \times 10^{390}$. These values include single substitution variants, double substitution variants, triple, etc. This simple calculation firmly establishes that you can never make all the variants of a protein. There are only $\sim 10^{70}$ particles in the universe, so 2×10^{390} is an impossibly large number. One will never be able to test all possibilities.

Problem 9.1. Consider a small protein of only 50 amino acids. How many variants are possible if all 20 amino acids are possible at every location? If the average molecular weight of these protein variants is 5,500 g/mol, what would be the weight, in grams, of a collection containing only one molecule of each variant? Compare this value to the weight of the earth, $\sim 6 \times 10^{27}$ g.

Location possibilities = L!/(k!(L-k)!)

The number of location possibilities is given by equation 9.3, which is the same as 9.2, except n has been replaced by L to indicate sequence length.

$$\text{location possibilities} = \frac{L!}{k!(L-k)!} = \frac{\text{total number of permutations}}{\text{order not important} \times \text{number of permutations not used}} \tag{9.3}$$

For example, to choose two locations in a 300 amino acid protein, there are 300 places to put the first substitution, but only 299 places to put the second substitution. The total number of permutations is 300 × 299 or 89,700. Since order is not important, we divide by 2! or 2.

$$(300!/(2!(300-2)!)) = (300!/(2! \times 298!)) = (300 \times 299)/2 = 44{,}850$$

Hint: Large factorials are too large for many calculators. For example, $300! \sim 3 \times 10^{614}$. Simplify the ratio L!/(L-k)! To L*(L-1)*(L-2)*...(L-k-1) as shown above to avoid calculating large factorials.

Number of possible variants

Combining the expression for the number of replacements possible with the number of locations possible yields an expression for the number of possible protein variants with k substitutions, V_k:

$$V_k = \frac{L!}{k!(L-k)!}19^k \tag{9.4}$$

The Python script 9.1 in the supporting information calculates values of V_k. For example, for one substitution anywhere in a 300-aa protein: V_1 = 300 *19 = 5,700. For two substitutions anywhere in a 300-aa protein: V_2 = 44,850 * 361 = 16,190,850

9.3 Exhaustive search of all single amino acid substitutions

The straightforward way to find a single amino acid substitution that improves a protein is to make all possible single amino acid substitution variants and test them. For example, DeSantis and coworkers (2003) made and screened all possible single-amino-acid-substitution variants of a nitrilase to find one with increased enantioselectivity for the synthesis of a cholesterol-lowering drug, atorvastatin (Lipitor®), Figure 9.1. The starting nitrilase showed an enantioselectivity of only ~15. Their experiments identified the Ala190His variant, which showed an enantioselectivity of >100.

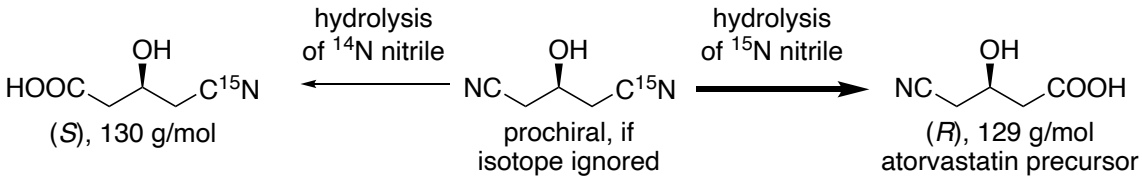

Figure 9.2. A nitrilase can catalyze the hydrolysis of either nitrile in the dinitrile shown in the center. Hydrolysis of the nitrile on the left yields the (*S*)-enantiomer, while hydrolysis of the nitrile on the right yields the (*R*)-enantiomer. To find variants with higher enantioselectivity for the (*R*)-enantiomer, DeSantis and coworkers (2003) used the isotopically labelled dinitrile shown. The two hydrolysis products differed in mass, so mass spectrometry revealed the ratio of the (*R*)- and (*S*)-products.

The advantage of such systematic screening is completeness. Testing every possibility ensures finding the best one. Other methods described below make incomplete sets of single-substitution variants, so the best possible variant might not be in the library. For example, libraries of single substitution variants created by error-prone PCR are incomplete and would likely not include the Ala190His variant. Converting a codon for alanine into a codon for histidine requires two nucleotide substitutions, which makes this substitution much less likely than those requiring one nucleotide substitution, see section 9.4.3 below. Thus, systematic screening is most suitable when one expects the beneficial variants to be rare and it is important not to miss these variants.

The disadvantage of this systematic, exhaustive approach is the large amount of work. The nitrilase above contained 330 amino acids, so there are 330 × 19 = 6270 possible single amino

acid substitution variants. The researchers made and tested all of them.

9.2.1 Degenerate codons encode multiple amino acids

Site-directed mutagenesis is the replacement of one amino acid with another at a particular location. This replacement involves synthesis of new DNA strands, typically with a DNA polymerase to extend mutagenic DNA primers. Details of one site-directed mutagenesis method were in Chapter 1. These mutagenic primers are made by chemical synthesis, usually by a DNA synthesis company. These primers encode the replacement amino acid instead of the original amino acid, Figure 9.2.

```
wild type sequence
5'...GATTGCAGCTGCTGTTTTCCACAATTCAGTATTGCCAGACACCGAGC...

primer for site-directed mutagenesis His103Val
  5' GATTGCAGCTGCTGTTTTCGTGAATTCAGTATTGCCAGAC 3'

primer mixture for site-saturation mutagenesis His103Xxx
            5' GCTGTTTTCNNKAATTCAGTATTGCCAGAC 3'
```

Figure 9.2. Example of primers for site-directed and site-saturation mutagenesis at the underlined codon. The wild type codon is CAC, which encodes histidine. In the primer for site-directed mutagenesis, this codon is replaced by GTC, which encodes valine. The mixture of primers for site-saturation mutagenesis contains thirty-two different primers as represented here by the degenerate codon NNK.

Site-saturation mutagenesis makes all nineteen possible amino-acid substitutions at a particular location. One approach is nineteen site-directed mutagenesis experiments at each amino acid position. This approach requires, at each location, nineteen primer pairs, one encoding each of the replacement amino acids, and nineteen separate site-directed mutagenesis procedures. This solution is tedious and rarely used.

More commonly, researchers use mixtures of primers that encode multiple amino acids. These primers contain degenerate codons, which are mixtures of codons prepared by DNA synthesis, Table 9.1. For example, the NNK codon represents a mixture of 32 different codon. N represents any of the four nucleotides (A, T, G, C), while K represents the two nucleotides containing a ketone group (G or T). Site-directed mutagenesis using degenerate codons creates a mixture of multiple protein variants. Transfer of the mixture of DNA into bacteria yields a mixture of bacteria, each one producing one variation at the targeted location. This approach saves time by requiring only one site-directed mutagenesis experiment, but increases the amount of screening since it yields a mixture of variants.

Table 9.1. One-letter ambiguity codes indicate mixtures of several nucleotides.

one-letter code	nucleotides
N (any)	A, T, G, C
B (not A)	T, G, C
D (not C)	A, T, G
H (not G)	A, T, C
V (not T)	A, G, C
K (keto)	G, T
M (amino)	A, C
R (purine)	A, G
S (strong)	G, C
W (weak)	A, T
Y (pyrimidine)	C, T

An example nucleotide sequence containing a degenerate codon is 5'-NNKATG, which contains the degenerate codon NNK. Chemists make this nucleotide mixture by adding mixtures of nucleotide precursors at the appropriate step. Chemical DNA synthesis starts at the 3'-end. For this example, the chemist adds the phosphoramidite precursor for G, then, the precursor for T, then the precursor for A. At the next step, the chemist adds an equimolar mixture of G and T precursors and in the last two steps the chemist adds an equimolar mixture of A, T, G and C. The result is a mixture of 32 DNA molecules which differ in nucleotide sequence at the NNK locations. Besides the twenty codons encoding the twenty amino acids, it also contains eleven redundant codons and one stop codon, Table S9.1 in the supporting information. Twelve amino acids are encoded once: Asn, Asp, Cys, Gln, Glu, His, Ile, Lys, Met, Phe, Trp, Tyr; five amino acids are encoded twice: Ala, Gly, Pro, Thr, Val and three amino acids are encoded three times: Arg, Leu, Ser.

An alternative to the NNK degenerate codon is the NNS degenerate codon, which also encodes all the amino acids, but changes sixteen of the thirty-two codons. K represents G or T, while S represents G or C. The sixteen codons that contain G in the last position are identical in both choices of degenerate codon. The remaining sixteen codons are NNT for the NNK degenerate codon, but are NNC for the NNS degenerate codon. The NNT codons encode the same amino acids as the NNC codons, but using NNS in place of NNK increases the GC content. Higher GC content increases the melting temperature of the primers and, depending on the rest of sequence, may alter the secondary structure of the primers.

The extra codons in the NNK or NNS degenerate codons beyond the twenty codons needed to encode the twenty amino acids increase the amount of screening needed, see section 9.2. An alternative to this extra screening is to replace the NNK or NNS mutagenic primer with a mixture of four primers: two with degenerate codons and two with specific codons, Table 9.3.

The NDT degenerate primer encodes twelve amino acids: N, S, I, H, R, L, Y. C, F, D, G, V (N = A, T, C, or G and D = A, T, or G). The VMA degenerate primer encodes six amino acids: E, A, Q, P, K, T (V = A, C, or G and M = A or C). Finally, the ATG codon encodes M and the TGG codon encodes W. Adding the primers in a 12: 6: 1: 1 ratio encodes all twenty amino acids, with no stop codons, no rare codons, and no synonymous codons (Tang et al., 2012).

Table 9.2. A 20-codon mixture of two degenerate codons (NDT encodes twelve amino acids; VMA encodes six amino acids) and two specific codons (ATG and TGG) efficiently randomizes the encoded amino acid. A 12:6:1:1 mixture of primers containing these four codons encodes all 20 amino acids, no stop codons and includes no redundant codons and no rare codon for yeast or *E. coli*.

codon	nucleotide mixture	encoded amino acid
NDT	AAT	Asn
	AGT	Ser
	ATT	Ile
	TAT	Tyr
	TGT	Cys
	TTT	Phe
	GAT	Asp
	GGT	Gly
	GTT	Val
	CAT	His
	CGT	Arg
	CTT	Leu
VMA	AAA	Lys
	ACA	Thr
	GAA	Glu
	GCA	Ala
	CAA	Gln
	CCA	Pro
ATG	ATG	Met
TGG	TGG	Trp

The disadvantage of this approach is the requirement of four mutagenic primers for each location randomized instead of one mutagenic primer for the NNK or NNS approach. For randomization of a single site, the NNK/NNS approach is simpler, but for randomization of multiple sites the time savings of less screening tips the balance toward the four mutagenic primer approach.

9.2.2 Oversampling to find all variants in a mixture

Site-saturation mutagenesis using degenerate codons yields a mixture of bacteria, each producing a different protein. Spreading the mixture on an agar plate and allowing them to grow yields individual colonies where each colony is derived from a single bacterium. The bacterial colonies are picked and each tested separately. Randomly picking colonies results in picking some duplicate variants, so one needs to test more colonies than the number of possibilities to have a high probability of testing each possible variant. This section explain how much oversampling is required. The conclusion will be that for saturation mutagenesis of a single position using an NNK codon, sampling 94 colonies ensures that the rarest codon in the mixture (an amino acid encoded only once) has a 95% probability of being included in the sample. To ensure that at least one of each codon was tested requires sampling 200 colonies for a 95% probability.

Including the rarest variant, three-fold oversampling. To derive a general equation, consider the probability to pick a blue ball from an equal and infinite mixture of red, white & blue balls. The probability is 1/3, which is the frequency of the blue balls in the mixture. Lets call this frequency F_{blue}. Similarly, the probability to NOT pick a blue ball from this mixture is $1 - F_{blue}$ or $1 - 1/3$ or $2/3$.

If we pick a second ball, the probability to pick a second blue ball is 1/3, the same as for the first ball since it is an infinite mixture. The probability for both balls to be blue is $(1/3)^2$ or 1/9. Similarly, the probability for neither ball to be blue after two picks is $(2/3)^2$ or $(1 - 1/3)^2 = 0.436$. In the general case, the probability NOT to pick a blue ball after T picks would be $(1 - F_{blue})^T$. Therefore, the probability of having a least one blue ball is $1 - (1 - F_{blue})^T$. The probability of picking a least one blue ball after two picks from an equal mixture of red, white & blue is $1 - (1 - 1/3)^2 = 0.564$. In the general case, the probability of picking object i from an infinite mixture, P_i, is $1 - (1 - F_i)^T$ where F_i is the frequency of object i in the mixture and T is the number of times an object is selected from the mixture.

In the context of protein engineering, the probability P_i, that a particular variant, i, is among T tested transformants (colonies) is given by:

$$P_i = 1 - (1 - F_i)^{T_{rarest}} \tag{9.5}$$

where F_i is the frequency at which sequence i is present in the library. This frequency can be

replaced by 1/N where N is the number of replacement codons. These equations assume the variants form independently according to a Poisson distribution.

$$P_i = 1 - \left[1 - \left(1/N\right)\right]^{T_{rarest}} \tag{9.6}$$

A common situation is a site-saturation mutagenesis created using an NNK degenerate codon. The number of different codons, N, is 32, so the frequency F_i is 0.03125. While some amino acids are encoded by two or even three codons in this mixture and their frequency is higher, one is normally concerned about the rarest members, those amino acids encoded by a single codon.

Equation 9.6 can be rearranged to yield the number of transformants that needed to be tested.

$$T_{rarest} = \frac{\ln\left(1 - P_i\right)}{\ln\left(1 - \frac{1}{N}\right)} \tag{9.7}$$

For a 95% probability of including a particular variant in a saturation mutagenesis experiment using an NNK codon, one needs to test 94 colonies, equation 9.8.

$$T_{rarest} = \frac{\ln\left(1 - 0.95\right)}{\ln\left(1 - \frac{1}{32}\right)} = 94 \tag{9.8}$$

Example 9.2. How many colonies must be screened to have a 95% probability of including a particular variant when three sites are simultaneously varied using site-saturation mutagenesis with a NNK codons. How would this change if a 20-codon mixture replaces the NNK codons?

For NNK, N = 32*32*32 or 32,768; equation 9.7 yields T = 98,163, while for the 20-codon mixture, N = 20*20*20 = 8,000; equation 9.7 yields T = 23,964, which is about four-fold fewer colonies. This example shows the increasing advantage of the 20-codon mixture (Table 9.2) when randomizing multiple sites. As the number of sites increases, the advantage of the smaller set of codons increases.

Researchers typically screen 3·N transformants with the expectation that they have included the rarest member of the library. This guideline comes from equation 9.6 above with a probability of 0.95.

$$0.95 = 1 - \left[1 - \left(1/N\right)\right]^{T_{rarest}};\ 0.05 = \left[1 - \left(1/N\right)\right]^{T_{rarest}};\ \ln(0.05) = T_{rarest}\ln\left[1 - \left(1/N\right)\right]$$

Using the approximation that ln(1-x) = -x when x is small, yields:

$$\ln(0.05) = T_{rarest}(-1/N);\ T_{rarest} = -N \cdot \ln(0.05)$$
$$T_{rarest} = 3 \cdot N \tag{9.9}$$

Thus, three-fold oversampling, or testing three times the number of expected variants, gives a 95% probability that any individual variant has been tested. This three-fold oversampling does not mean that the library has a 95% chance of being 100% complete. To ensure with a 95% probability that ALL variants have been tested requires even more oversampling.

Including all the variants. What is the probability of picking at least one of each object? If you pick 9 balls, the probability to have at least one blue ball is $1-(2/3)^9$ or $1-(1-1/3)^9 = 0.976$. Similarly the probability to have a least one red ball among the nine is 0.976 and the same for the white ball. The probability of having at least one of each color is $(0.976)^3$ or 0.93. In the general case, the probability of having all possibilities, $P_i = \left[1-\left(1-\left(\frac{1}{N}\right)\right)^{T_{all}}\right]^N$ where N is the number of objects in the mixture. Rearranging this equation yields eq. 9.10.

$$T_{all} = \frac{\ln\left(1-\left(P_i\right)^{\frac{1}{N}}\right)}{\ln\left(1-\frac{1}{N}\right)} \tag{9.10}$$

Problem 9.2. Show that screening 203 colonies gives a 95% probability to screen all variants of a single site saturation site made with an NNK degenerate codon.

While equation 9.7 could be simplified to the three-fold oversampling rule in equation 9.9, no similar simplification is possible for equation 9.10. For typical cases, including all variants requires six- to ten-fold oversampling. Python Script 9.2 in the supporting information calculates T_{rarest} and T_{all} using equations 9.7 and 9.10 for different numbers of variants and probabilities.

9.2.3 Library quality: completeness and uniqueness

For this discussion, a library is a collection of DNA sequences that encode proteins. Two features define Q, the quality of this library: completeness, C, and uniqueness, U (Patrick et al. 2003). Completeness ensures that one does not miss a possible beneficial variant, while uniqueness avoids testing the same variant multiple times. Library quality, defined by eq. 9.11, has a maximum value of one for a library that contains all possible variants without any duplicates.

$$Q = C \cdot U \tag{9.11}$$

Completeness is the fraction of all possible proteins, given the goal of the library, that are encoded by this library. For example, the goal of saturation mutagenesis is to test all possible substitutions at the selected locations. If two locations are targeted for saturation mutagenesis, then the maximum possible is 20x20 or 400 proteins. Saturation mutagenesis using an NNK codon is complete (C = 1) since this library encodes all 400 of the possible 400 proteins. In contrast, saturation mutagenesis with an NDT codon is 36% complete (C = 0.36) since it encodes

only 12x12 = 144 of the possible 400 proteins.

Uniqueness is the fraction of non-synonymous DNA sequences in the library. For example, saturation mutagenesis using an NNK codon is only 39% unique (U = 0.39; 20x20/32x32) because using 32 codons to encode 20 amino acids creates synonymous DNA sequences. In contrast, saturation mutagenesis with an NDT codon is 100% unique (U = 1) since each DNA sequence encodes a different amino acid.

The library quality for saturation mutagenesis libraries created using the NNK or NDT codons decreases with the number of locations targeted, Table 9.6. The quality of the NNK library decreases because it contains increasing numbers of duplicates, while the quality of the NDT library decreases because it is increasingly incomplete.

Table 9.3. The quality of saturation mutagenesis libraries decreases with increasing numbers of targeted locations. The NNK library contains increasing numbers of duplicates, while the NDT library is increasingly incomplete.

number of locations	complete-ness	unique-ness	library quality	complete-ness	unique-ness	library quality
		NNK codon			NDT codon	
1	1	20/32 =	0.62	12/20	1	0.60
2	1	$(20/32)^2$	0.39	$(12/20)^2$	1	0.36
3	1	$(20/32)^3$	0.24	$(12/20)^3$	1	0.22
4	1	$(20/32)^4$	0.15	$(12/20)^4$	1	0.13

To choose the best approach for saturation mutagenesis, one should also consider the number of transformants that will be screened. If one plans to screen a small number of transformants, then the NDT library is better because a greater number of protein variants will be tested. Each transformant screened from the NDT library is likely to be unique, while the NNK library will contain duplicates. However, if one plans to screen all the transformants then the NNK library is better because it contains more total proteins. Screening the NNK library with three-fold over sampling requires 1,024 x 3 = 3072 measurements. Screening the NDT library with three-fold over sampling requires 400 x 3 = 1200 measurements. Screening greater numbers of transformants for the NNK library compensates for the low uniqueness in the library.

9.4 Error-prone PCR with a low mutation rate

The polymerase chain reaction (PCR) copies DNA strands. This copying introduces errors. By choosing the appropriate polymerase and reaction conditions, the overall error rate can be as high as ~0.5% or 5 nucleotide substitutions per thousand nucleotides. Error-prone PCR (epPCR) is a good way to introduce random mutations into a long stretch of DNA such as that encoding

an entire gene or several genes. The procedure is simple and creates a library containing a small number of mutations. Screening this library identifies variants with improved properties. (For shorter (<100 bp) DNA segments, chemical synthesis of the sequence using nucleotide mixtures is a better approach.)

One disadvantage of error-prone PCR is the imprecise control over the number of mutations introduced. At low mutation rates, the library contain many unchanged sequences (wild type) and many duplicates of single substitution variants. Screening these unchanged sequences and duplicates wastes time. A second disadvantage is that error-prone PCR creates an incomplete set of amino acid substitutions, Table 6.4. Both the way that nucleotides encode amino acids and the bias of the polymerase errors contribute to this incompleteness. Error-prone PCR is best suited to those cases where one expects many possible solutions. In these cases, an incomplete library will likely contain improved variants.

Table 9.4. Amino acid substitutions introduced by error-prone PCR are biased by the preference of the polymerase for certain types of nucleotide replacement, by the encoding of amino acids in the codons, and by the spatial properties of folded proteins.

type of bias	description
non-random nucleotide substitutions by DNA polymerase	substitutions at AT sites are more common than substitutions at GC sites; replacements favor transitions and an increase in GC pairs
non-random encoding of amino acids within codons	single nucleotide substitutions in a codon yield an average of 5.7 amino acid replacements; encoding the remaining 13.3 amino acids requires two nucleotide substitutions in the codon, which are much rarer than single nucleotide substitutions
spatial constraints limit the number of amino acids that interact directly with the substrate	random amino acid substitutions rarely change amino acids in the substrate-binding site because only a few amino acids form the substrate-binding site; most amino acids lie far from the binding site

9.4.1 Overall error rate in the polymerase chain reaction

The overall error rate in the polymerase chain reaction depends not only on the error rate of the polymerase, but also on the number of copies created because the errors accumulate.

Number of doublings when copying DNA with PCR. The polymerase chain reaction copies DNA fragments exponentially typically yielding thousands to millions of copies. If the amplification is 100% efficient, then the number of DNA molecules doubles with each cycle of the polymerase chain reaction. The number of DNA molecules, N_{cycles}, is given by:

$$N_{cycles} = 2^{cycles} * N_0 \tag{9.12}$$

where cycles is the number of cycles and N_0 is starting number of DNA molecules. Thus, twenty PCR cycles can increase the amount of DNA 2^{20}-fold or approximately one-million fold. In practice the early cycles of PCR follow this formula, but later cycles are less efficient because the polymerase loses activity and the reagents are exhausted. The more general equation for the increase in the increase in product in a PCR is given by:

$$N_{cycles} = (1 + eff)^{cycles} * N_0 \tag{9.13}$$

where eff is the efficiency. When the efficiency is 1, then the equation simplifies to equation 9.9 above.

PCR reactions, especially under error-prone conditions, are not 100% efficient and are typically 70-90% efficient. The amount of DNA does not double in each cycle, but increases by a factor of 1.7-1.9. In these cases, the important value to determine the error rate is not the number of PCR cycles, but the number of doublings, D. For example, twenty cycles of a PCR reaction with 70% efficiency increases the amount of DNA by 1.7^{20} or 40,600. This increase corresponds to about 15.3 doublings since $2^{15.3} = 40{,}300$. Thus, 20 cycles of a 70% efficient PCR are equivalent to 15.3 cycles of a 100% efficient PCR.

$$2^D = \left(1+EF\right)^{cycles} \text{ or } D = cycles \cdot \log_2\left(1+EF\right) \tag{9.14}$$

Example 9.3. Frances used 30 cycles in a PCR to amplify a template DNA. Adding a dye that becomes fluorescent upon binding to double-stranded DNA revealed that the PCR procedure increased the amount of DNA 800,000-fold. How many DNA doublings does this increase correspond to? What was the average efficiency of her reaction?

Answer: Calculate the number of doublings by assuming the 800,000-fold increase comes from a 100% efficient reaction; then using equation 9.12 to calculate number of cycles, which corresponds to the number of cycles in the 100% efficient case.

$$800{,}000 = 2_{cycles} \text{ or } \ln(800{,}000) = cycles * \ln(2);\ cycles = 19.6$$

Using equation 9.14 with the actual 30 cycles to solve for the efficiency of her reaction:

$$800{,}000 = (1 + EF)^{30};\ EF = 0.57 \text{ or } 57\%$$

Error-rate of DNA polymerases. DNA polymerases sometimes make mistakes and insert an incorrect nucleotide. The error rate of DNA polymerase from *Thermus aquaticus* (*Taq* DNA polymerase) is about $4 \cdot 10^{-5}$ errors per nucleotide, 0.04 errors per thousand nucleotide added or one error per 25,000 nucleotides added (McInerney et al., 2014). This value varies with the reaction conditions in the range of 1-20 * 10^{-5} errors per nucleotide (Eckert & Kunkel, 1991a). In contrast, DNA polymerases from the archaea *Pyrococcus furosius* or *Thermococcus kodakarensis* proofread the DNA as it is copied, which lowers the error rate approximately 10-fold. These high fidelity DNA polymerases are not suitable for error-prone PCR.

In error-prone PCR, reaction conditions are adjusted to increase the error rate of *Taq*

DNA polymerase about 100-fold. Typical changes in reaction conditions are an imbalanced pool of nucleotides, increased free magnesium concentration, and addition of manganese salts (Cadwell & Joyce, 1992). Under error-prone conditions, the error rate is $6.6 * 10^{-3}$ error per nucleotide or 6.6 errors per thousand nucleotides added. Changing the reaction conditions even further in an attempt to further increase the error rate leads to poor amplification. Researchers often add a fresh aliquot of *Taq* DNA polymerase after the 15^{th} cycle to replace inactivated polymerase.

Overall error rate in PCR. The overall error rate in a PCR reaction, ε_{nt}, will be larger than the error rate of the polymerase, ε_p, because errors from previous cycles are copied and thus accumulate. The overall error rate is the number of nucleotide substitutions introduced during the PCR reaction divided by the total number of nucleotides copied.

$$\varepsilon_{nt} = \text{nucleotide substitutions / total nucleotides copied} \quad (9.15)$$

The total number of nucleotides after a given number of cycles, assuming 100% efficient amplification, is $2^{cycles} \cdot N_0$, given by equation 9.9 above. The numbers of errors introduced is the product of the numbers of errors introduced in one cycle ($N_0 \cdot \varepsilon_p$) and the increase in the total number of DNA molecules ($cycles \cdot 2^{cycles-1}$) (Eckert & Kunkel, 1991b).

$$\text{overall error rate in a PCR, } \varepsilon_{nt} = (N_0 \cdot \varepsilon_p)(cycles \cdot 2^{cycles-1})/(2^{cycles} \cdot N_0) = cycles \cdot \varepsilon_p/2 \quad (9.16)$$

The division by a factor of two accounts for the starting DNA, which does not contain errors. For example, the special case of one PCR cycle would create one copy of the starting DNA. Since only half of the total DNA contains errors, the overall error rates would be half of the error rate of the DNA polymerase. However, after a more typical twenty cycles, the error rate in PCR, ε_{nt}, is 10-fold higher than the polymerase error rate, ε_p, due to subsequent copying of errors made in previous cycles.

> **Problem:** Describe the contents of a PCR after two cycles and explain why the overall error rate is equal to the polymerase error rate.
>
> **Problem:** Jane amplified DNA using 20 cycles of error-prone PCR and found that the amount of DNA increased 50,000-fold. She then ligated the resulting DNA into a plasmid and transformed it into *E. coli*. She picked 10 colonies, isolated the plasmid DNA and sent the samples for sequencing. The sequencing yielded 600 bp for each sample for a total of 6000 nucleotides. The sequencing showed 97 substitutions. What was the overall error rate, ε_{nt}, and what was the error rate of the DNA polymerase, ε_p, in this experiment?

9.4.2 DNA polymerase bias

Transcribing a DNA strand with a polymerase can introduce two types of errors: 1) substitutions when the incorrect nucleotide is inserted or 2) frame shifts when an extra nucleotide is added (insertion) or a nucleotide is skipped (deletion). Substitutions can lead to proteins with altered function, while frame shifts lead to misfolded proteins since the amino acid encoding is

altered for all codons after the frame shift. Frameshift are relatively rare (<5%) among DNA polymerase errors and will be ignored in the discussion below.

For the nucleotide substitutions to be random both the location of the substitutions and the replacement nucleotide must be random. Neither one is random in the case of nucleotide substitutions introduced by DNA polymerases due to the differing chemical properties of the nucleotides themselves. This non-random nature of the substitutions is DNA polymerase bias for some types of substitutions over others. The bias of DNA polymerase away from random nucleotide substitutions reduces the likelihood of some amino acid substitutions as compared to the frequency expected in a random library. One can compensate for the reduced likelihood of some substitutions by increasing the number of colonies screened, but this extra screening tests the more common substitutions multiple times.

The frequency of substitutions at AT base pairs versus GC base pairs should be equal if the location of the substitution is random, Table 9.5. However, GC base pairs, which form three hydrogen bonds, are more stable than AT base pairs, which from only two hydrogen bonds. This chemical difference accounts for preference of *Taq* DNA polymerase for substitutions at AT locations over GC locations.

Table 9.5. Expected changes for random substitutions in a DNA strand and observed changes in an error-prone PCR experiment using *Taq* DNA polymerase.

	criteria	expected for random	observed for *Taq* DNA polymerase[a]	comment
location of substitution	$\frac{\text{change at AT site}}{\text{change at GC site}}$	1.0	$3.9\left(\frac{\text{entries 1+2+3}}{\text{entries 4+5+6}}\right)$	GC pairing is stron than AT pairing
replacement nucleotide	$\frac{\text{transition}}{\text{transversion}}$	0.5	$0.8\left(\frac{\text{entries 3+6}}{\text{entries 1+2+4+5}}\right)$	replacement with a similarly sized bas more likely
	$\frac{\text{AT} \rightarrow \text{GC}}{\text{GC} \rightarrow \text{AT}}$	1.0	$1.9\left(\frac{\text{entries 2+3}}{\text{entries 5+6}}\right)$	GC pairing is stron than AT pairing

[a] Data from Table 9.5 below.

Two criteria measure whether the replacement nucleotide is random. The first criteria is that the number of transitions must be half the number of transversions. Transitions are substitutions where the original and new nucleotide have similar-sized bases (purine to purine or pyrimidine to pyrimidine exchange). Transversions are substitutions where base sizes differ (purine to pyrimidine or pyrimidine to purine exchange), Figure 9.4. There are four possible transitions and eight possible transversions, so random substitutions requires half the number of transitions as transversions. Replacement with a differently sized base changes the width of the base pair and is expected to be be less likely. Once again the chemical properties (size in this case) predict non-random substitutions.

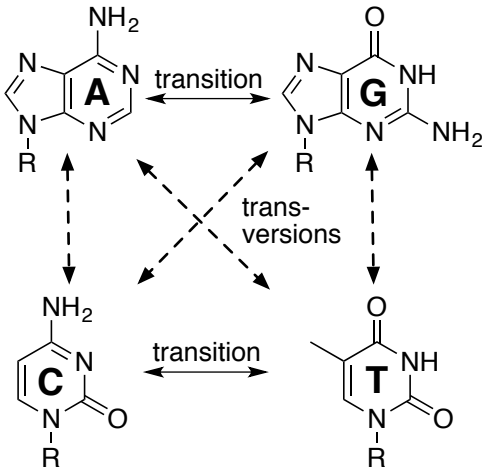

Figure 9.4. Transitions substitute a purine with a purine (A with G and vice versa) or a pyrimidine with a pyrimidine (C with T and vice versa), while transversions substitute a purine with a pyrimidine (A or G with C or T) or a pyrimidine with a purine (C or T with A or G). There are four possible transitions and eight possible transversions.

Exercise: Write out all the possibilities to show that random mutagenesis should yield twice as many transversions as transitions.

The second criteria for whether the replacement nucleotide is random is that the replacement base pair must be equally likely to be an AT pair as a GC pair. In other words, the AT→GC replacements must the equal to the GC→AT replacements. The GC content of the DNA remains unchanged if the substitutions are random. Because GC pairs form three hydrogen bonds as compared to two hydrogen bonds formed by AT pairs, replacements of AT pairs with GC pairs is about twice as common as replacement of GC pairs with AT pairs.

Sequencing of the DNA from an error-prone PCR using *Taq* DNA polymerase confirmed that both the site of substitution and the replacement nucleotide are non-random (Shafikhani et al., 1997), Table 9.6. First, most (>95%) of the errors were substitutions, not frame shifts, as mentioned above. The most common substitutions were replacement of AT with TA (40.9% of the total) and of AT with GC (27.6% of the total). Another study using different reaction conditions reported a different, but still non-random distribution of substitutions (Lin-Goerke et al., 1997). Replacement of AT with TA was only 16.1% of the total, while replacement of AT with GC was the most common substitution (63.2% of the total).

Table 9.6. Fraction of each substitution type expected for random substitutions and observed in error-prone PCR using *Taq* DNA polymerase.[a]

entry	substitution	fraction expected for random substitutions, %	fraction observed, *Taq* polymerase, %[a]	fraction observed, Mutazyme II, %[b]
1	A→T and T→A	16.7	40.9	28.5
2	A→C and T→G	16.7	7.3	4.7
3	A→G and T→C	16.7	27.6	17.5
4	G→C and C→G	16.7	1.4	4.1
5	G→T and C→A	16.7	4.5	14.1
6	G→A and C→T	16.7	13.6	25.5

[a] Data from DNA sequencing of 9800 nucleotides, which identified 286 errors (ε_{nt} = 2.9%). The sum of the substitution errors is 95.3%. The remaining errors were frame shifts: insertions: 0.3%, deletions: 4.2% and the missing 0.2% is due to rounding errors. The data are from Shafikhani *et al.* (1997). *b* from GeneMorph II Random Mutagenesis kit manual; https://www.agilent.com/cs/library/usermanuals/public/200550.pdf.

The observed substitutions failed the randomness test under all three criteria in Table 9.5. The location of substitutions favored AT sites by a factor of almost four (75.8/19.5 = 3.9), the replacements favored transitions over transversions (41.2/54.1 = 0.8 versus 0.5 for random) and replacement with GC was 1.9-fold higher (34.9/18.1) than replacement with AT such that the overall GC content of the DNA increased. These biases in error-prone PCR do not prevent any substitutions, but skew the library by making some substitutions less common than expected in a random distribution. It requires more screening to find the less common substitutions.

One can reduce some of the bias by replacing *Taq* DNA polymerase with Mutazyme II DNA polymerase, which is a mixture of two polymerases (Agilent; Cline & Hogrefe, 2000). Mutazyme II reduces the bias for mutations at AT versus GC sites from almost four to 1.2 (50.7/43.7 =1.2; 1.0 is ideal), but slightly increases the bias in the ratio of transitions to transversions from 0.8 to 0.9 (43.0/51.7 = 0.9; 0.5 is ideal) and reverses the bias in ratio of replacement with GC over replacement by AT from 1.9 to 0.6 (22.2/39.6 = 0.6; 1.0 is ideal).

9.4.3 Codon bias

Translation of the 64 possible nucleotide codons yields only twenty possible amino acids. Multiple nucleotide codons encode the same amino acid; that is, they are synonymous. For example, the GGA codon codes for glycine, Figure 9.5. All three possible replacements of the third nucleotide, A, yield codons that still code for glycine. In this case, 33% of the new codons still encode glycine. On average, 30% of the single nucleotide substitutions in a codon yield synonymous codons. Some substitutions yield a stop codon (3.6% on average) and only 66.3% of the single nucleotide substitutions yield a codon that encodes a new amino acid (a missense mutation). Single nucleotide substitutions in a codon yields, on average, only 5.7 new amino acid codons. The inability to create all possible amino acid substitutions with a single nucleotide substitution is called codon bias.

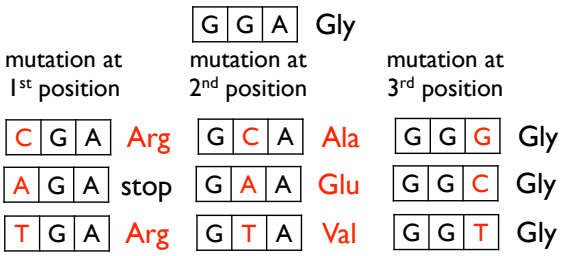

Figure 9.5. Replacement of one nucleotide in the glycine codon GGA yields nine new codons and only four amino acid substitutions. The three new codons created by replacement of the third nucleotide, A, also code for glycine, so 33% of the single nucleotide substitutions for this codon are synonymous. On average 30% of the single nucleotide substitutions are synonymous. The non-synonomous codons encode only four new amino acids. On average, single substitutions in a codon yield codons for new 5.7 amino acids.

Converting a codon for one amino acid into a codon for any of the nineteen other amino acids requires changing more than one nucleotide. Low mutation rates typical of error prone PCR only rarely create two substitutions in one codon. If the mutation rate is 0.5% at each nucleotide position, then the chance of one nucleotide substitution at any one of the three nucleotides in a codon is 3*0.5% = 1.5%. The chance of a second substitution within this codon at the remaining two nucleotides is 1.5% *(2*0.5%) = 0.015%, which is 100-fold lower. This is so low that it can be ignored.

Changing only one nucleotide yields only nine new codons, thus, at most one could encode nine new amino acids. As explained above, some of the new codons are synonymous with the original codon, so the nine new codons will encode fewer than nine amino acids. In the glycine codon example, single nucleotide substitutions encode only four new amino acids. On average, single nucleotide substitutions in a codon yields 5.7 new amino acids. Thus, random mutagenesis that changes one nucleotide in a codon does not create codons for nineteen new amino acids, but only for 5.7 amino acids. This codon bias is one reason that random mutagenesis libraries created by error-prone PCR are often incomplete.

9.4.4 Location bias

Error-prone PCR and many other random mutagenesis methods target all amino acid positions equally. Counterintuitively, this random process favors mutations far from the active site (Morley & Kazlauskas, 2005). There are greater numbers of amino acids far from the active site, so greater numbers of mutations will occur far from the active site. Only a few amino acid residues (<10%) form the substrate-binding site and most amino acid residues lie far away. The inherent preference of random mutagenesis methods for mutations far from the active site is known as location bias.

Imagine a spherical enzyme with the active site at the center, where the amino acid residues form spherical, concentric shells around the active site. As one moves outward, each shell is larger, so it contains more amino acids. The number of amino acids, n, in each 1-Å-thick shell of outer radius r is given by the volume of the shell divided by the average volume occupied by an

amino acid residue, which is 141 Å^3, Figure 9.6. The number of amino acids in each shell increases as one moves outward from the center. The data for a typical 30 kDa protein fit this prediction well for shells out to ~15 Å, but beyond this value is smaller than predicted because the shells become larger than the protein.

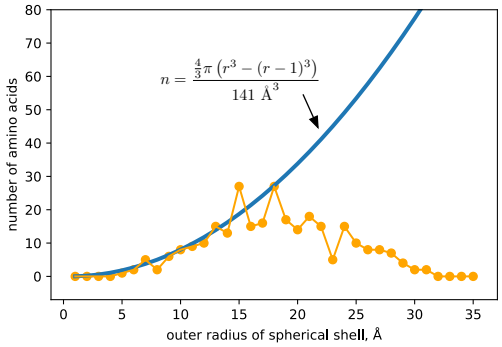

Figure 9.6. The number of amino acids within concentric, spherical shells increases as one moves outward from the center of a protein. The shells are 1-Å thick and have an outer radius of r. The blue line plots the values predicted by the equation shown, while the orange circles indicate observed values for a typical 30 kDa protein (esterase, pdb id 1va4). The agreement between the predicted and observed values is good out to ~15 Å, but then reaches the outside edge of the protein so the number of amino acids are fewer than predicted by the equation.

More than half of the amino acid residues in this typical protein lie 13-22 Å from the active site and <10% lie within 10 Å of the active site. For larger proteins, the edges lie further out, so most amino acids will lie even further from the active site. There is a greater number of distant amino acids and therefore methods that target all amino acids create a greater number of distant mutations. Substitutions anywhere in the protein can contribute to protein stability, so this bias toward substitutions far from the active site is not a disadvantage. However, substitutions near the active site are more effective at changing protein properties like substrate specificity and selectivity. In these cases, the bias toward distant substitutions is a disadvantage.

9.5 Conclusions

The most reliable way to search for single substitutions that improve protein properties it to makes all possible single substitution variants and test them for the desired properties. This approach is a lot of work, so researchers simplify their task by making assumptions about their solutions. One assumption is that there are many possible substitutions that can lead to the improvement and that they are located throughout the protein. For example, substitutions that increase the thermal stability of a protein often fits these criteria. In these cases, epPCR is a convenient approach. It targets the entire protein, but makes an incomplete set of single substitutions. Since there are likely many solutions, the library will likely contain some of them. A different assumption is that substitutions near the active site are most likely to yield improved properties. In these cases saturation mutagenesis at selected locations in the substrate binding site is a good choice.

Table 9.7. Recommended approaches to make single-substitution variants for different protein engineering goals.

goal	approach	rationale
increase stability	epPCR	many solutions likely; solutions are likely distributed throughout the protein
expand substrate range (increase reactivity)	saturation mutagenesis of residues in substrate-binding site	solutions may be rare and most likely within or close to the substrate binding site
increase selectivity	saturation mutagenesis of residues in substrate-binding site	solutions may be rare and most likely within or close to the substrate binding site

References

R. C. Cadwell, G. F Joyce (1992) Randomization of genes by PCR mutagenesis. *PCR Meth. Appl.* **2**, 28-33.

M. Chodorge, L. Fourage, C. Ullmann, V. Duvivier, J.-M. Masson, F. Lefèvre (2005) Rational strategies for directed evolution of biocatalysts – application to Candida antarctica lipase B (CALB), *Adv. Synth. Catal.*, **347**, 1022-6; https://doi.org/10.1002/adsc.200505055

J. Cline, H. Hogrefe (2000) Randomize gene sequences with new PCR mutagenesis kit, *Strategies* **13**, 157–62; https://www.agilent.com/cs/library/usermanuals/public/200550.pdf

G. DeSantis, K. Wong, B. Farwell, K. Chatman, Z. Zhu, G. Tomlinson, H. Huang, X. Tan, L. Bibbs, P. Chen, K. Kretz, M. J. Burk (2003) Creation of a productive, highly enantioselective nitrilase through gene site saturation mutagenesis (GSSM). *J. Am. Chem. Soc.* **125**, 11476-7; https://doi.org/10.1021/ja035742h

K. A. Eckert, T. A. Kunkel (1991a) DNA polymerase fidelity and the polymerase chain reaction. *PCR Meth. Appl.* **1**, 17-24;

K. A. Eckert, T. A. Kunkel (1991b) The fidelity of DNA polymerases used in the polymerase chain reactions. *PCR A Practical Approach*, M. J. McPhearson, P. Quirke, G. R. Taylor, Eds., Oxford University Press: Oxford, pp 225–44.

M. Eigen, W. Gardiner (1984) Evolutionary molecular engineering based on RNA replication. *Pure Appl. Chem.* **56**, 967–78. https://doi.org/10.1351/pac198456080967

D. A. Estell, T. P. Graycar, J. A. Wells (1985) Engineering an enzyme by site-directed mutagenesis to be resistant to chemical oxidation. *J. Biol. Chem.* **260**, 6518–21; https://https://www.jbc.org/content/260/11/6518.long

E. Kreyszig (1972) *Advanced Engineering Mathematics*, 3rd ed. Section 19.6 Permutation and Combinations, Wiley, 1972.

J. L. Lin-Goerke, D. J. Robbins, J. D. Burczak (1997) PCR-based random mutagenesis using manganese and reduced dNTP concentration. *Biotechniques*, **23**, 409-12.

P. McInerney, P. Adams, M. Z. Hadi (2014) Error rate comparison during polymerase chain reaction by DNA polymerase. *Mol. Biol. Int.* **2014**, 287430; https://doi.org/10.1155/2014/287430

K. L. Morley, R. J. Kazlauskas (2005) Improving enzyme properties: when are closer mutations better? *Trends Biotechnol.* **23**, 231–7. https://doi.org/10.1016/j.tibtech.2005.03.005

W. M. Patrick, A. E. Firth, J. M. Blackburn (2003) User-friendly algorithms for estimating completeness and diversity in randomized protein-encoding libraries. *Protein Eng.* **16**, 451–7. https://doi.org/10.1093/protein/gzg057

S. Shafikhani, R. A. Siegel, E. Ferrari, V. Schellenberger (1997) Generation of large libraries of random mutants in *Bacillus subtilis* by PCR-based plasmid multimerization. *Biotechniques*, **23**, 304–10.

J. H. Spee, W. M. de Vos, O. P. Kuipers (1993) Efficient random mutagenesis method with adjustable mutation frequency by use of PCR and dITP. *Nucleic Acids Res.* **21**, 777–8.

L. Tang, H. Gao, X. Zhu, X. Wang, M. Zhou, R. Jiang (2012) Construction of "small-intelligent" focused mutagenesis libraries using well-designed combinatorial degenerate primers, *Biotechniques*, **52**, 149–58; https://www.biotechniques.com/multimedia/archive/00175/BTN_A_000113820_O_175322a.pdf

J.-P. Vartanian, M. Henry, S. Wain-Hobson (1996) Hypermutagenic PCR involving all four transitions and a sizeable proportion of transver- sions. *Nucleic Acids Res.* **24**, 2627–31;

Supporting Information for Chapter 9

```
from math import factorial   #
def Vk(L, n, k):             # L = length of sequence; n = 19 for
proteins, 3 for
                             # nucleotides; k = number of
substitutions
   ratio = 1
   for i in range(0,k):      # Loop calculates L!/(L-k)! =
L*(L-1)*(L-2)*...(L-k-1)
      ratio = ratio*(L-i)
                             # runs from i = 0 to i = k-1
   nlp = ratio/factorial(k)  # number of location possibilities =
L!/[k!*(L-k)!]
   nsp = n**k                # number of substitution
possibilities = n^k
   Vk = (nlp*nsp)
   print ('The number of possible variants for',k,'substitutions
using',n,'replacements anywhere in a sequence of
length',L,'is','{:.3G}'.format(Vk))
                           # example
Vk(4000, 3, 3)
```

```
The number of possible variants for 3 substitutions using 3
replacements anywhere in a sequence of length 4000 is 2.88E+11
```

Script 9.1. Python script to calculate Vk, the number of variants with k substitutions. Vk = number of location possibilities * number of substitution possibilities. nlp = L!/[k!*(L-k)!] and nsp = n^k) where L = length of sequence, k = number of substitutions and n = the number of replacements, usually 19 for proteins and 3 for nucleotides. Adjust these values as needed. The loop runs from i = 0 to i = k-1 and calculates the ratio = L!/(L-k)! = L*(L-1)*(L-2)*…(L-k-1).

Table 9.S1. The mixtures of thirty-two codons represented by NNK or NNS both contain the sixteen NNG codons, but differ in the remaining sixteen. NNK contains the sixteen NNT codons, while NNS contains the sixteen NNC codons. In each case, the thirty-two codons encode the same amino acids, which includes all twenty amino acids, one stop codon and eleven redundant codons.

sequence NNK	sequence NNS	encoded amino acid	comment
	AAG	Lys	only codon for this amino acid
AAT	AAC	Asn	only codon for this amino acid
	ACG	Thr	rare codon for yeast
ACT	ACC	Thr	
	AGG	Arg	rare codon for *E. coli* & yeast
AGT	AGC	Ser	
	ATG	Met	only codon for this amino acid
ATT	ATC	Ile	only codon for this amino acid
	CAG	Gln	only codon for this amino acid
CAT	CAC	His	only codon for this amino acid
	CCG	Pro	rare codon for yeast
CCT	CCC	Pro	CCC is a rare codon for *E. coli*
	CGG	Arg	rare codon for *E. coli* & yeast
CGT	CGC	Arg	
	CTG	Leu	
CTT	CTC	Leu	
	GAG	Glu	only codon for this amino acid
GAT	GAC	Asp	only codon for this amino acid
	GCG	Ala	rare codon for yeast
GCT	GCC	Ala	
	GGG	Gly	
GGT	GGC	Gly	
	GTG	Val	
GTT	GTC	Val	
	TAG	stop	
TAT	TAC	Tyr	only codon for this amino acid
	TCG	Ser	
TCT	TCC	Ser	
	TGG	Trp	only codon for this amino acid
TGT	TGC	Cys	only codon for this amino acid
	TTG	Leu	
TTT	TTC	Phe	only codon for this amino acid

Rarely-used codons can create protein expression problems, but the NNK and NNS codons do not encounter this potential problem. There are eight rare codons in *E. coli*: CGA, CGG, AGA, AGG for Arg; CTA for Leu; ATA for Ile; GGA for Gly; CCC for Pro (Zhang et al., 1991). The tRNAs that recognize rare codons are also present in low amounts. Normally, this low abundance has no effect, but over expressed proteins may deplete the existing pool of charged tRNAs and cause premature termination or misincorporation of amino acids. The two underlined codons occur in the NNK library and both code for Arg. However, the NNK library includes a third codon for Arg (CGT), which is not a rare codon. There are eight rare codons in yeast: AGG, CGA, CGG, CGC for Arg; CCG for Pro, CTC for Leu, GCG for Ala; ACG for Thr (Zhang et al., 1991). The five underlined rare codons appear in the NNK library, but, like the *E. coli* case, the NNK library also includes synonymous non-rare codons for each of the amino acids. Similar analysis indicates that NNS also encodes at least one non-rare codon for both *E. coli* and yeast.

Reference

S. Zhang G. Zubay, E. Goldman (1991) Low-usage codons in *Escherichia coli*, yeast, fruit fly and primates, *Gene* **105**, 61-72; https://doi.org/10.1016/0378-1119(91)90514-C

```
from math import log           # log = natural logarithm
def numTransformants(N, P):    # N = number of codon variations, P
= probability
   T_rarest = log(1-P)/log(1-(1/N))
   print('Testing',round(T_rarest),'colonies gives
a','{:.0f}'.format(100*P)+'% probability that any individual
variant in a collection of','{:.0f}'.format(N),'variants has been
tested.')
   T_all = log(1-(P**(1/N)))/log(1-(1/N))
   print('Testing',round(T_all),'colonies gives
a','{:.0f}'.format(100*P)+'% probability that every variant in a
collection of','{:.0f}'.format(N),'variants has been tested.')

          # example
numTransformants(32*32, 0.95)
```

```
Testing 3066 colonies gives a 95% probability that any individual
variant in a collection of 1024 variants has been tested.
Testing 10134 colonies gives a 95% probability that every variant
in a collection of 1024 variants has been tested.
```

Script 9.2. Python function to calculate the number of transformants, T, needed to have tested the rarest one in a collection of N variations with a probability of P. T = ln(1-P)/ln(1-(1/N)). In the example, N = 32.0 corresponds to site-saturation mutagenesis of a single site using an NNK or NNS degenerate codon and P = 0.95 corresponds to a 95% probability. Adjust values for N and P as needed.

```
# Calculates D, the number of doublings
# 2^D = (1+EF)^cycles or D = cycles * log2(1+EF)
```

```
# EF = efficiency of PCR (ranges from 0 to 1)
# cycles = number of inefficient PCR cycles

from math import log
EF = 0.7
cycles = 30.0

D = cycles * log(1.0+EF,2)

print(format(cycles,'.0f'),' PCR cycles with an efficiency of
',format(100*EF,'.0f'),'% yields ',format(D,'.1f'),' doublings.',
sep ='')

   # example
30 PCR cycles with an efficiency of 70% yields 23.0 doublings.
```

Script 9.2. Python script to calculate the number of doublings, or 100% efficient cycles, for a known number of inefficient cycles when the efficiency is known.