Supplemental Material for

Indel-correcting DNA barcodes for high-throughput sequencing

John A. Hawkins[1,2,3], Stephen K. Jones Jr.[2,3], Ilya J. Finkelstein[2,3,4,*], and William H. Press[1,3,5,*]

[1] *Institute for Computational Engineering and Science, The University of Texas at Austin, Austin, TX 78712, USA*

[2] *Department of Molecular Biosciences, The University of Texas at Austin, Austin, TX 78712, USA*

[3] *Institute for Cellular and Molecular Biology, The University of Texas at Austin, Austin, TX 78712, USA*

[4] *Center for Systems and Synthetic Biology, The University of Texas at Austin, Austin, TX 78712, USA*

[5] *Department of Integrative Biology, The University of Texas at Austin, Austin, TX 78712, USA*

[*] *Correspondence: wpress@cs.utexas.edu, ifinkelstein@cm.utexas.edu*

1 Error Correcting Code and Free Divergence Properties

1.1 Minimum error correcting barcode length

An error correcting code which corrects m substitutions, deletions, or insertions must be at least $2m+1$ bp long. Proof: Suppose the contrary. Let $L \leq 2m$ be the length of the barcode. Then by definition every barcode is at most L substitutions from any other barcode by substituting all of the bases. For any two barcodes B_1 and B_2 define B_{mid} to be the barcode with the first m bases of B_1 and the remaining $L - m \leq m$ bases of B_2. Then $B_{mid} \in DecodeSphere_m(B_1)$ and $B_{mid} \in DecodeSphere_m(B_2)$. Since B_1 and B_2 were arbitrary, it is thus impossible to have two disjoint decode spheres. Therefore, it is impossible to have a non-trivial (i.e. more than one barcode) m-error correcting code of length less than $2m + 1$ bp.

www.pnas.org/cgi/doi/10.1073/pnas.1802640115

1.2 Calculating FREE divergence

$FreeDiv(X, Y)$ can be efficiently calculated with a modified Needleman-Wunsch algorithm [1], where the last row and column of the matrix have zero penalty for insertion and deletion corresponding to right-end fill or truncation respectively.

1.3 Symmetry and minimum paths

FREE divergence is symmetric because any minimum filled/truncated right end edit path (FREE path) is invertible by inverting all the edits and then inverting the fill/truncation step. Substitutions are invertible with substitutions, while insertions and deletions are invertible with each other in the natural way, so edits by themselves are invertible. Invertibility with the fill/truncation step is less obvious, and requires no edit be truncated off the end. For example, a substitution in the last position followed by any insertion results in the substitution getting truncated off the end. Minimum FREE paths never have any edits truncated off the end, because any truncated edit can be omitted to create a shorter edit path.

Let X and Y be barcodes and let P be any minimum FREE path from X to Y. If P has no fill or truncation, then the fill/truncation step is trivially invertible by doing nothing. Suppose P has a fill step which fills f bases at the end. Then starting at Y and inverting the edits results in exactly those f bases being outside the barcode window, so they are truncated to arrive at X. Suppose P has a truncation step which truncates t bases. Since P is a minimum edit path, none of the truncated bases were edited bases, so they are not needed for the inverted edit path starting at Y. After inverting the edits, t bases need to be filled, which we fill with the last t bases of X. Hence, any minimum FREE path can be inverted in the same number of edits. Furthermore, since all minimum FREE paths from X to Y and from Y to X are invertible, $FreeDiv(X, Y) \leq FreeDiv(Y, X)$ and $FreeDiv(Y, X) \leq FreeDiv(X, Y)$. Therefore, $FreeDiv(X, Y) = FreeDiv(Y, X)$ and any inverted minimum FREE path is itself a minimum FREE path.

1.4 FREE divergence is not a metric

We use the counter-example shown in the right column of Figure 1c. For $FreeDiv(\,TAGA,\ ACGC\,)$, the modified Needleman-Wunsch algorithm described above produces

$$\begin{array}{c} \\ \text{T} \\ \text{A} \\ \text{G} \\ \text{A} \end{array} \begin{array}{c} \begin{array}{ccccc} \text{A} & \text{C} & \text{G} & \text{C} & \end{array} \\ \begin{pmatrix} 0 & 1 & 2 & 3 & 4 \\ 1 & 1 & 2 & 3 & 4 \\ 2 & 1 & 2 & 3 & 4 \\ 3 & 2 & 2 & 2 & 3 \\ 4 & 3 & 3 & 3 & 3 \end{pmatrix} \end{array},$$

so, from the value in the last row and column, $FreeDiv(\,TAGA,\ ACGC\,) = 3$. Hence, the following is a minimum filled/truncated right end edit path (FREE path) between TAGA and ACGC:

$$TAGA \xrightarrow{ins.} TACG\,|\,A \xrightarrow{sub.} TACG\,|\,C \xrightarrow{del.} ACGC$$

The vertical bars ("|") show the end of the barcode window, though the truncation step would not happen until after all actual edits. Now, the above FREE path shows that $FreeDiv(\,TAGA,\ TACG\,) = FreeDiv(\,TACG,\ ACGC\,) = 1$. But $FreeDiv(\,TAGA,\ ACGC\,) = 3$, a violation of the triangle inequality.

We note that this was the error made in the paper defining Sequence-Levenshtein codes [2]. That code generation technique depends on the Sequence-Levenshtein distance function being a metric, which it is not.

1.5 Repeated sub-barcode decoding behavior

Using concatenated barcodes consisting of r repeats of a single sub-barcode is a strategy to decrease decoding error rates. With this strategy, the output is filtered to accept only barcodes in which all sub-barcodes decode to the same sub-barcode. We here calculate the probability of erroneously accepting a barcode with errors, as well as the probability of detecting and filtering barcodes with sub-barcode errors.

Let p be the probability of a sub-barcode decoding error, let r be the number of repeated sub-barcoces in one concatenated barcode, and let q be the probability that r wrong decodes all decode to the same wrong sub-barcode. Then, assuming independent errors, the full-barcode decoding error probability is

$$P(decoding\ error) = qp^r.$$

We characterize p throughout this paper, but it remains to calculate q. q, the probability that

r wrong decodes all decode to the same wrong sub-barcode, is a function of the effective number of neighbors of a typical sub-barcode. If the whole space of k-mers were filled with equidistant sub-barcodes and their decode spheres, then each wrong sub-barcode would be equally likely and q would be $(\frac{1}{n-1})^{r-1}$, where n is the number of sub-barcodes in the library. However, in practice both of those assumptions are bad. First, the sub-barcode space is not completely filled with decode spheres. Many decode errors return 'None' rather than wrong sub-barcodes, and 'None' values are always filtered. Second, the sub-barcodes are not equidistant. Each sub-barcode has a few neighboring sub-barcodes whose decode spheres are some minimal number of edits away, and these dominate observed wrong sub-barcodes. The former effect decreases q relative to the above estimate by increasing filtering, while the latter effect increases q. In this way, the number of effective neighbors of a typical sub-barcode, and hence q, depends on the the efficiency of the sphere packing and the size of the space. It also depends to a lesser degree on the per-base probability of substitutions, insertions, and deletions, in the same way the effective number of neighbors for a child depends on whether the child is crawling, walking, or riding a bicycle.

Assuming we intend all barcodes with equal probability, we can write

$$\begin{aligned} q &= P(r \text{ same decode errors} \mid r \text{ errors from same intended sub-}BC) \\ &= \sum_{i=1}^{n} P(r \text{ same decode errors} \mid r \text{ errors, intended sub-}BC_i)\, P(\text{intended sub-}BC_i) \\ &= \frac{1}{n} \sum_{i=1}^{n} P(r \text{ same decode errors} \mid r \text{ errors, intended sub-}BC_i) \end{aligned}$$

where

$$\begin{aligned} &P(r \text{ same decode errors} \mid r \text{ errors, intended sub-}BC_i) \\ &\qquad = \sum_{j \neq i} P(\text{observed sub-}BC_j \mid \text{decoding error, intended sub-}BC_i)^r. \end{aligned}$$

This expression we estimate through direct simulation: randomly selecting intended barcodes, adding errors, and decoding. We estimate q, the average, by subsampling intended sub-barcodes. Estimates for q over a range of barcode lengths, per-base error levels, and numbers of repeats are shown in Figure S8.

The probability of filtering a barcode due to sub-barcode errors is the probability of any error

decoding any sub-barcode except for the previous case where all sub-barcodes decode to the same other sub-barcode. This is given by

$$\begin{aligned} P(filtered\ barcode) &= 1 - P(correct\ decode) - P(decoding\ error) \\ &= 1 - (1-p)^r - qp^r \end{aligned}$$

For all expected use cases, p is small and r is a small integer, so the first-order approximation of this expression is a good one, given by:

$$P(filtered\ barcode) = rp.$$

2 Barcode Generation

2.1 Sphere iterator

Central to our generation and decoding algorithms is the ability to deterministically iterate over decode and encode spheres. Recursive iteration is far too slow for practical use due to redundancy. For example, attempting to find $DecodeSphere_2(B)$ by finding $DecodeSphere_1(W)$ of all words W in $DecodeSphere_1(B)$ results in iterating over each 2-error word at least twice, by switching the order of added edits. As the number of edits, m, grows, the redundancy grows as $m!$ due to edit permutations.

So, to iterate over a sphere centered at barcode B, we instead built a method to iterate over all words at a given FREE divergence d from B and then iterate over d as needed. We additionally exploit the following identities regarding substitution (sub), insertion (ins), and deletion (del) edits to optimize iteration: sub-del = del, ins-del = del-ins = sub, ins-sub = sub-ins, and in the last position, ins = del = sub. Note that use of these identities assumes we are only interested in solid spheres, so will for example iterate over a sequence at divergence d with an ins-del sequence during the previous $d-1$ divergence sphere with a sub.

2.2 Use of encode spheres

The algorithm used for efficient code generation relies on the fact that if a word W is in $EncodeSphere(B)$, then $DecodeSphere(B)$ and $DecodeSphere(W)$ overlap. That is, there exists a word U such that $U \in DecodeSphere(B)$ and $U \in DecodeSphere(W)$. Let $W \in EncodeSphere(B)$. If $W \in DecodeSphere(B)$,

then $U = W$ and by symmetry we are done. Suppose $W \notin EncodeSphere(B)$. By the definition of encode spheres, there exists a filled/truncated right end edit path (FREE path) with at most $2m$ edits from B to W. Let U be the filled or truncated sequence m edits along this path from B. With this choice, $FreeDiv(B, U) \leq m$, where the less than or equal sign is in case of any fill/truncation effects. Furthermore, there are at most m more edits along the path to W by choosing the fill or truncation for word U to allow use of the same FREE path to W. Then, by use of symmetry, $FreeDiv(W, U) \leq m$, and we are done.

2.3 Code efficiency

Code efficiency is measured, where possible, in terms of a code rate, defined as the number of usable "message" bits that can be encoded in a single barcode divided by the actual number of bits in the sent barcode. In many standard codes, k message bits have r bits added for error correction, giving a code rate of $k/(k + r)$. For n-mer barcodes, each sent base is two bits of information, so the denominator is $2n$. The numerator is the effective number of message bits: the length of the largest binary number smaller than the number of barcodes, given by $\lfloor \log_2(Number\ of\ barcodes) \rfloor$. However, for our purposes the number of message bits does not need to be an integer, so we will refer to the previous as the actual message bits, while we are more interested in the "raw" message bits: $\log_2(Number\ of\ barcodes)$ without a floor function. These correspond to raw and actual code rates, shown in Figure S1.

The code rate of FREE codes increases with barcode length, and appears to asymptotically approach a maximal code rate determined by the properties of the decode sphere packing. We observe in Fig. 2b that after some boundary effects at short barcode lengths, the number of raw message bits (log of the number of barcodes) increases linearly with the length of the barcodes. The slope of this line, up to a factor of 2 for the x-axis due to using base-4 instead of base-2, is an empirical estimate for the asymptotic code rate—message bits over sent bits—for our packing method. We show estimated asymptotic values for our single- and double-error correcting codes as dashed lines in Figure S1c.

2.4 Generating Linear Hamming Codes

Generating a linear Hamming code for DNA strings of length n encoding raw messages of length k and which corrects up to e errors is equivalent to finding a parity check matrix $H = (-P^T \,|\, I_{n-k})$ over Galois field $\mathbb{F}_4$ such that any subset of $2e$ columns is linearly independent [3]. Given such a

matrix H, all barcodes can be expressed as mG, where m is any raw message vector of length k and G is the generation matrix $G = (I_k \,|\, P)$. We found matrices P_1 and P_2, corresponding to single- and double-error correcting linear Hamming codes, via lexicographical search through possible columns of H, accepting new columns if they were linearly independent from all previous subsets of $2e-1$ columns. We found such P matrices for k up to length 100. The submatrices corresponding to codes up to length 14, as used in Figure 2e, are given by

$$P_1 = \begin{pmatrix} 0 & 1 & 1 \\ 0 & 1 & 2 \\ 0 & 1 & 3 \\ 1 & 0 & 1 \\ 1 & 0 & 2 \\ 1 & 0 & 3 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \\ 1 & 1 & 2 \\ 1 & 1 & 3 \\ 1 & 2 & 0 \end{pmatrix}, \qquad P_2 = \begin{pmatrix} 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 2 \\ 0 & 1 & 1 & 0 & 2 & 1 \\ 0 & 1 & 1 & 2 & 1 & 0 \\ 0 & 1 & 2 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 3 \\ 1 & 0 & 1 & 0 & 2 & 3 \\ 1 & 0 & 1 & 2 & 3 & 1 \end{pmatrix}.$$

3 Experimental Validation

3.1 Primer processing

Primers were used both chemically for library amplification and informatically to distinguish left from right sides. However, the possibility of insertions and/or deletions in these primer sites introduced some uncertainty in the starting position of the DNA barcodes. To address this, we wrote a custom adaptation of the Smith-Waterman algorithm for overhanging sequences. The user specifies an expected primer sequence, a full-length observed read, and a maximum allowable number of errors, which we chose to be 2 for both the left (19 bp) and right (18 bp) primers. Using the modified Smith-Waterman algorithm with unity penalties for all error types, we identified the highest scoring prefix of the observed sequence which matches the expected sequence. If two or more possible lengths had the same score, we chose the one closest to the expected length. If the number of edits is less than or

equal to 2, this best inferred length then determines the position to be used as the start of the barcode sequence.

3.2 Experimental decode errors

Decode errors are detected by whether or not the left and right barcodes, as shown in Figure 3a, match an intended left/right barcode pair. There are two possible ways to decode incorrectly: either by decoding to a wrong barcode or by decoding to "None" if the observed barcode is not in any decode sphere at all. If a barcode decodes to "None", then that decode is obviously an error. If a barcode decodes to an incorrect barcode, then the observed output is that the left and right barcodes mismatch but it is unclear which is actually the decode error. We determine which barcode is in error by measuring the edit distance of the entire oligo against the two possible intended sequences, accepting the one with lowest edit distance. To measure the 0- and 1-Error correction data in Figure 5, we then measured the edit distance of each observed barcode to the intended barcode using the primer processing algorithm described above.

This analysis resulted in the detection of chimera oligos, oligos with the left side of one intended oligo and the right side of another. Most of the barcodes which decoded to wrong barcodes matched the wrong barcode with zero errors, which was very unexpected. The decode spheres for 17-mer, 2-error correcting codes contain $\sim 10^4$ barcodes, of which $\sim 10^2$ are 1-error away and exactly 1 is the 0-error wrong barcode (Fig. S1a). Furthermore, the wrong barcode with zero errors is the center word, furthest from the sphere boundary and other barcodes. Thus, seeing a barcode decode to a wrong barcode with zero errors should be vanishingly rare compared with 1 and 2 errors. These together imply that we are not observing random errors. We instead appear to be generating chimera oligos. This is likely explained by degeneracy in the spacer region: the spacers are all different, but have stretches of identical sequences around 20 bp long. Incomplete PCR products could then act as primers for this sequence in later rounds of PCR, creating chimera sequences. To correct for these chimeras, we conservatively assumed the distribution of the number of errors in chimera barcodes is the same as that for correct barcodes, though it is likely higher. The observed number of wrong barcodes with zero errors was 9,628, the approximate size of a decode sphere, so we accepted that as an approximation for how many chimera oligos had barcodes with zero errors. We then used this number and the distribution of correct barcode errors to approximate how many of the wrong barcodes 1-error and 2-errors away from the wrong barcode were chimera oligos. The 0-, 1-, and 2-error correct barcode

counts on the left were 1,258,928, 104,892, and 9,129 on the left and 1,205,545, 140,566, and 18,007 on the right. The wrong barcodes with zero errors were attributed to the two sides proportional to the total wrong barcodes found on each side. This resulted in 0-, 1-, and 2-error inferred chimera barcode counts of 2,078, 173, and 15 on the left and 7,550, 880, and 113 on the right. These were then omitted.

3.3 Decode error rate model

The decoding error rate of an m-error correction code is the probability of seeing more than m errors in a given barcode. For error analysis, we model each barcode as a queue of intended bases. At each read position, an intended base is popped off the queue and attempted to be added. One of four things will happen: 1) the correct base will be added, 2) an incorrect base will be added, 3) the base will be deleted, or 4) another base will be inserted and the intended base will go back to the top of the queue. The first three options do not return the base to the queue, resulting in the same structure of expected output $\mapsto$ observed output. However, insertions cause the intended base to return to the top of the queue, and the output was never expected in the first place. For this reason, it must be modeled differently from the other three. Assuming independent errors of all types and positions, we model insertions with a negative binomial distribution and the correct bases, deletions, and substitutions with a multinomial distribution, using our measured error rates per reference base, shown in Figure 3.

Let a barcode be given and let B be the 1-by-4 row vector with counts of each of the bases ACGT in the given barcode. Let I be the 1-by-4 row vector of insertion counts for each of the four bases. Further, let CDS be the 3-by-4 matrix with columns corresponding to the DNA bases, and rows corresponding to all non-insertion outputs: correct bases, deletions, and substitutions. We will occasionally refer to the rows of CDS individually as C, D, and S, but we leave it in matrix form as they are tightly connected. In fact, it must be true that $C + D + S = B$.

We use the measured error rates given reference base shown in Figure 3. Insertion and deletion rates, $p_i(b)$ and $p_d(b)$, are taken directly from synthesis error rate measurements. Substitution rates, $p_s(b)$, are calculated as the probability of not observing the event {no synthesis substitution and no sequencing substitution} nor the event {synthesis substitution to another base c and correcting synthesis substitution back to b}, and are thus given by

$$p_s(b) = 1 - (1 - p_{s,synth}(b))(1 - p_{s,seq}(b)) \; - \sum_{\substack{c \in \{ACGT\} \\ c \neq b}} \frac{1}{3} p_{s,synth}(b) \cdot \frac{1}{3} p_{s,seq}(c)$$

Now let $p_c(b) = 1 - p_d(b) - p_s(b)$ be the probability of correctly adding a base, let N be the random variable for the total number of errors, let n_{err} be given, and let n_i, n_d, and n_s be the number of insertions, deletions, and substitutions respectively. Then, from our assumption of independent error rates given reference base, we get for each reference base the previously mentioned negative binomial distribution for insertions and multinomial distribution for the rest:

$$p(N = n_{err} \mid B) = \sum_{\substack{CDS \text{ with } n_d, n_s \\ I \text{ with } n_i \\ n_d + n_s + n_i = n_{err}}} \prod_{b \in \{ACGT\}} \left[\binom{I_b + B_b - 1}{I_b} (1 - p_i(b))^{B_b} \, p_i(b)^{I_b} \right. \\ \left. \times \binom{B_b}{C_b + D_b + S_b} p_c(b)^{C_b} p_d(b)^{D_b} p_s(b)^{S_b} \right]$$

Finally, we marginalize the above over barcode identity,

$$p(N = n_{err}) = \sum_{B \in Barcodes} p(N = n_{err} \mid B) \, p(B)$$

and sum over n_{err} as required.

3.4 Maximum error run lengths

The error models used in this paper, both the simpler binomial model and that derived above, assume independent errors at each position, understanding that this is an oversimplification. A quick way to see that the errors in our experimental data are definitely not independent and to show why this impacts our work directly is to check the distribution of maximum error run lengths, i.e., the maximum number of consecutive errors in a given oligo. We consider the binomial model where each position is either an error with probability p or correct with probability $q = 1 - p$. We wish to know the probability that in a sequence of length n the maximum run of errors will be r bases long. This is a well-studied problem, and we use Simpson's solution as presented by Hald [4]. Briefly, let Z_n be the probability that the maximum run of errors in a sequence of length n is at least r bases long and let z_n be the probability that the first run of r errors ends at the n^{th} position. Then

$$Z_n = z_1 + z_2 + \cdots + z_n.$$

For $n < r$, $Z_n = z_n = 0$ trivially, since there are not enough bases, and for $n = r$, $Z_n = z_n = p^r$ since they must all be errors. For $n > r$,

$$z_n = (1 - Z_{n-r-1})qp^r$$

by definition of z_n, since this is the probability that no run of length $\geq r$ in the first $n - r - 1$ bases, then there is a correct base followed by r errors. One can then recursively find Z_{cr+i} for increasing c, resulting in the general formula

$$Z_n = \sum_{c=1}^{\lfloor n-1/r \rfloor} (-1)^{c+1} \left[p^r \binom{n - cr}{c - 1} (qp^r)^{c-1} + \binom{n - cr}{c} (qp^r)^c \right].$$

Finally, for fixed n and p, $\mathbb{P}(\text{maximum run length} = r) = Z_n(r) - Z_n(r+1)$, shown in Figure S7 using our oligo length, $n = 116$, and measured probability of error, $p = 0.005$ (Fig. 3). Our experimental data are similar to this model for maximum runs of zero and one errors, but deviate significantly for maximum runs of more than one error because our errors are not, in fact, independent. This helps explain the deviation of our experimental barcode decoding error rates from the model predictions in Figure 5, since our experimental errors clump together, increasing the probability of having more than two, say, in a single barcode.

References

[1] Saul B. Needleman and Christian D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48(3):443–453, March 1970.

[2] Tilo Buschmann and Leonid V. Bystrykh. Levenshtein error-correcting barcodes for multiplexed DNA sequencing. *BMC Bioinformatics*, 14:272, September 2013.

[3] W. Cary Huffman and Vera Pless. *Fundamentals of Error-Correcting Codes*. Cambridge University Press, February 2010.

[4] Anders Hald. *A History of Probability and Statistics and Their Applications Before 1750*. John Wiley & Sons, September 2003.

Barcode Length	1-Error Correction	2-Error Correction
3	2	-
4	3	-
5	10	2
6	27	2
7	67	4
8	213	7
9	554	12
10	1,903	31
11	6,161	75
12	17,214	179
13	56,736	468
14	157,197	1,156
15	518,509	3,183
16	1,636,418	8,777
17	-	23,025

Table S1: Numbers of FREE barcodes. Number of FREE barcodes for each barcode set included with this paper, by barcode length and number of errors corrected.

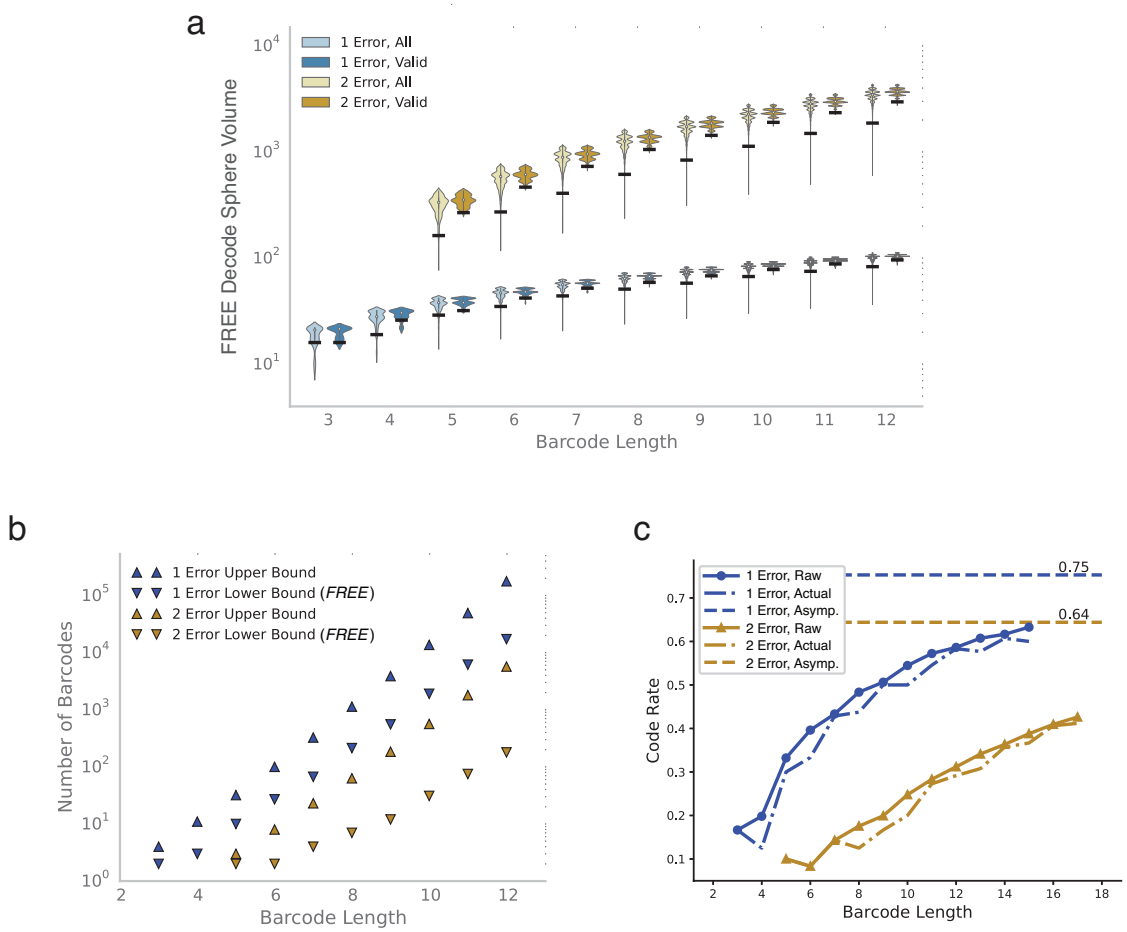

Figure S1: Decode sphere volumes and code efficiency. a. Unlike Hamming decode spheres, FREE divergence decode spheres do not have uniform volume due to degeneracy of insertions and deletions. For example, the sequence AACT only has three unique deletions because a deletion of either A generates the same resulting sequence. Sphere volumes of 1- and 2-error codes are shown for all words and for only valid code words after our FREE code synthesis and sequencing filters (no homopolymer runs, no triplet complementarity, etc.). Black lines explained in (b). **b.** Optimal sphere packing bounds. The optimal packing for an error-correcting code is not known in general. Typical code generating algorithms, including ours, are instead heuristics for finding relatively good codes. For reference, one can usually find a (typically impossible) upper bound on the maximum number of code words by calculating the volume of the space divided by the volume of a single decode sphere. As shown in (a), the volumes of FREE divergence decode spheres are not uniform, so we instead find the volume of every sphere in the space, sort them, and find the minimum number of barcodes at which the cumulative sum of barcode sphere volumes is smaller than the space. We show the upper bound calculated for valid code words. The volume at which that happens for each code is shown in (a) as black lines. The lower bound is the best efficiency achieved by any code generation method to date, which for FREE codes is simply the number of barcodes reported in this paper. The actual maximum possible number of barcodes is somewhere between the two. **c.** Raw and actual code rates for each FREE barcode set included with this paper as well as the asymptotic values they approach, as described in the Supplemental Materials.

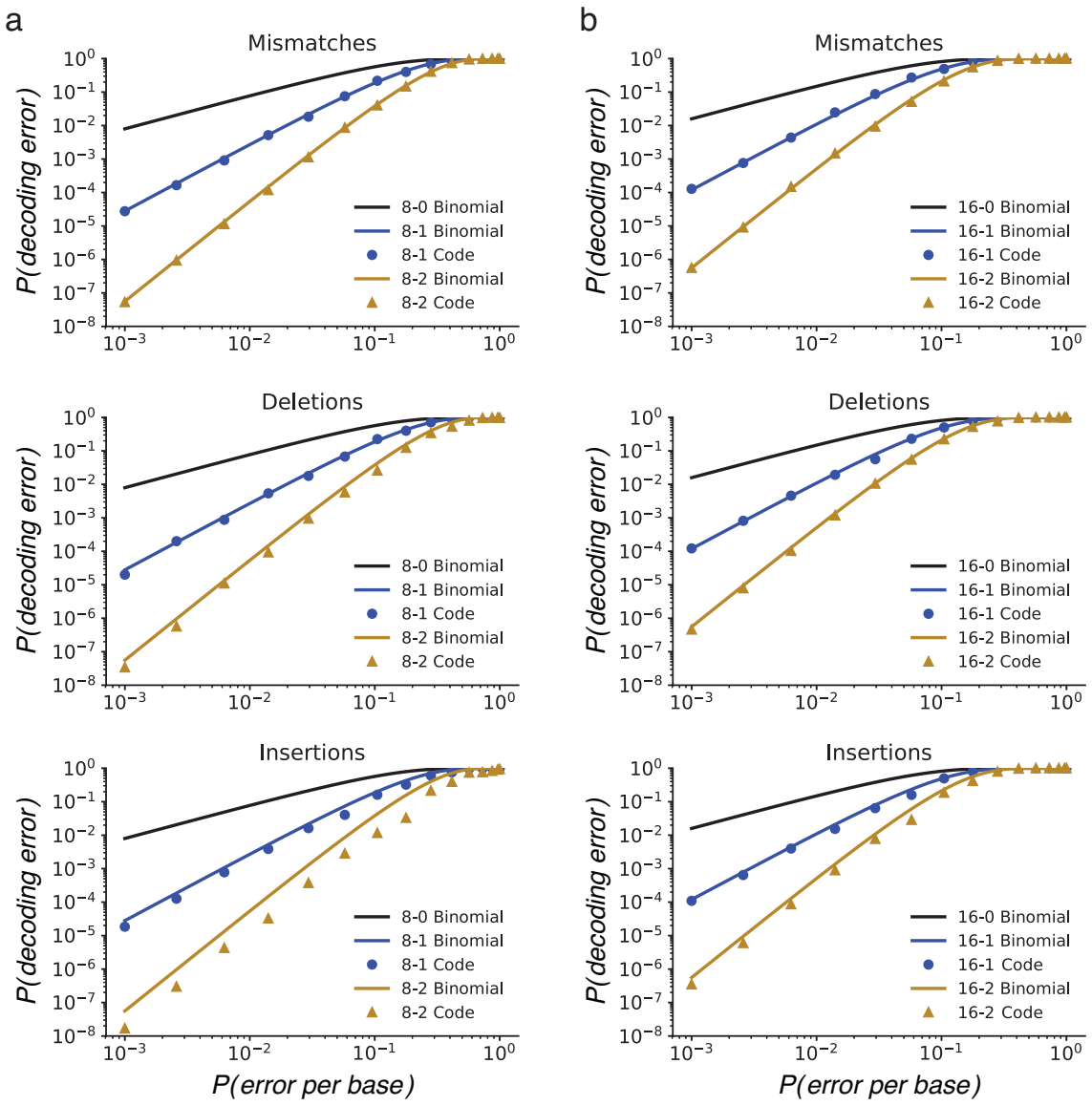

Figure S2: Error rate simulations by error type. a-b. The simulations performed for Figure 4, repeated for each error type—substitutions, deletions, insertions—individually. Shown for length (a) 8 and (b) 16 barcodes. Barcode sets are labeled according to length and number of errors corrected; for example, the 16-2 code is length 16 and corrects up to 2 errors. Mismatches follow the binomial approximation closely, while deletions and especially insertions perform slightly better than the binomial approximation.

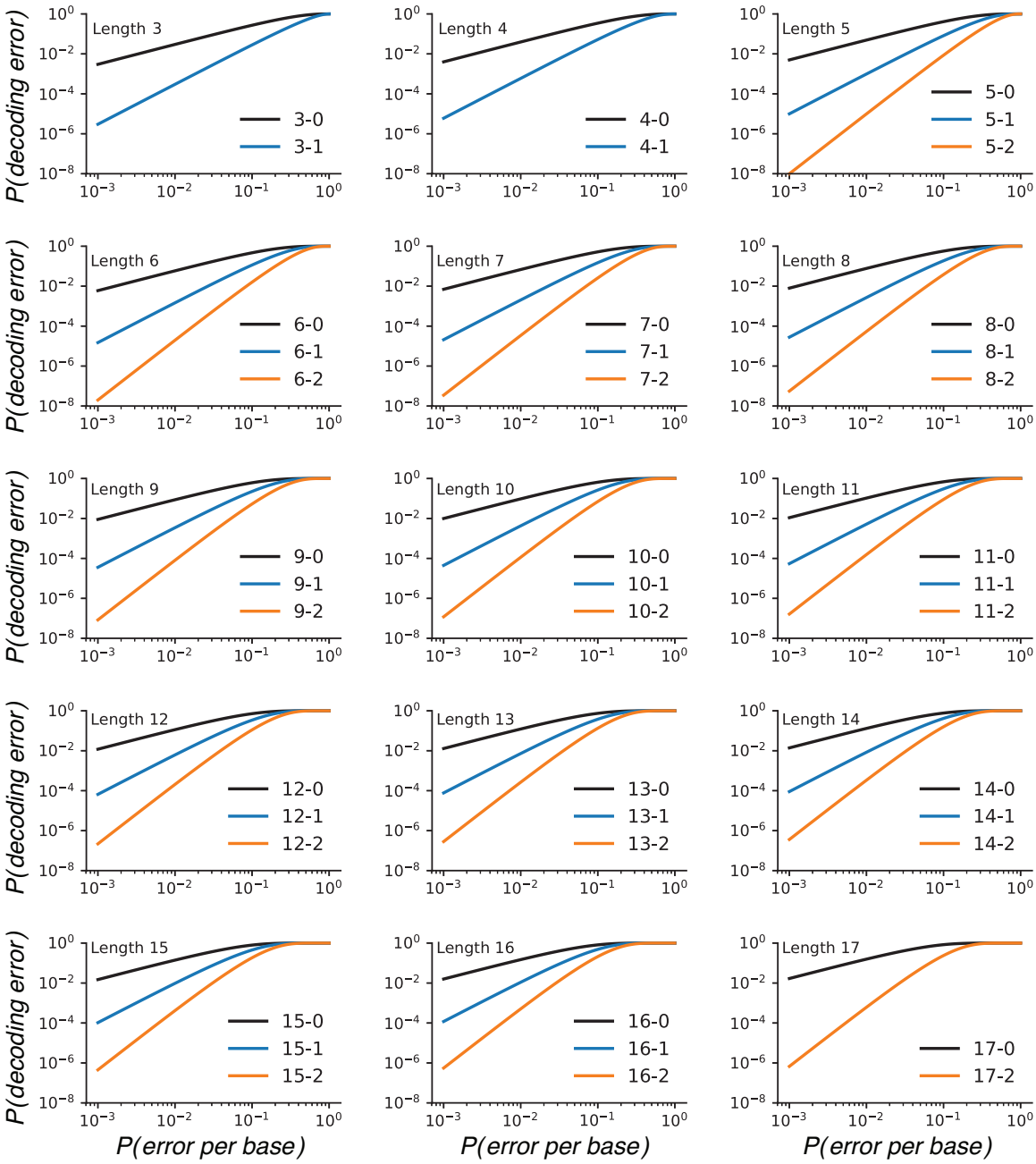

Figure S3: Error rate comparison with constant barcode length. The binomial approximation of the decode error rate as a function of the error rate per base, grouped by given barcode length.

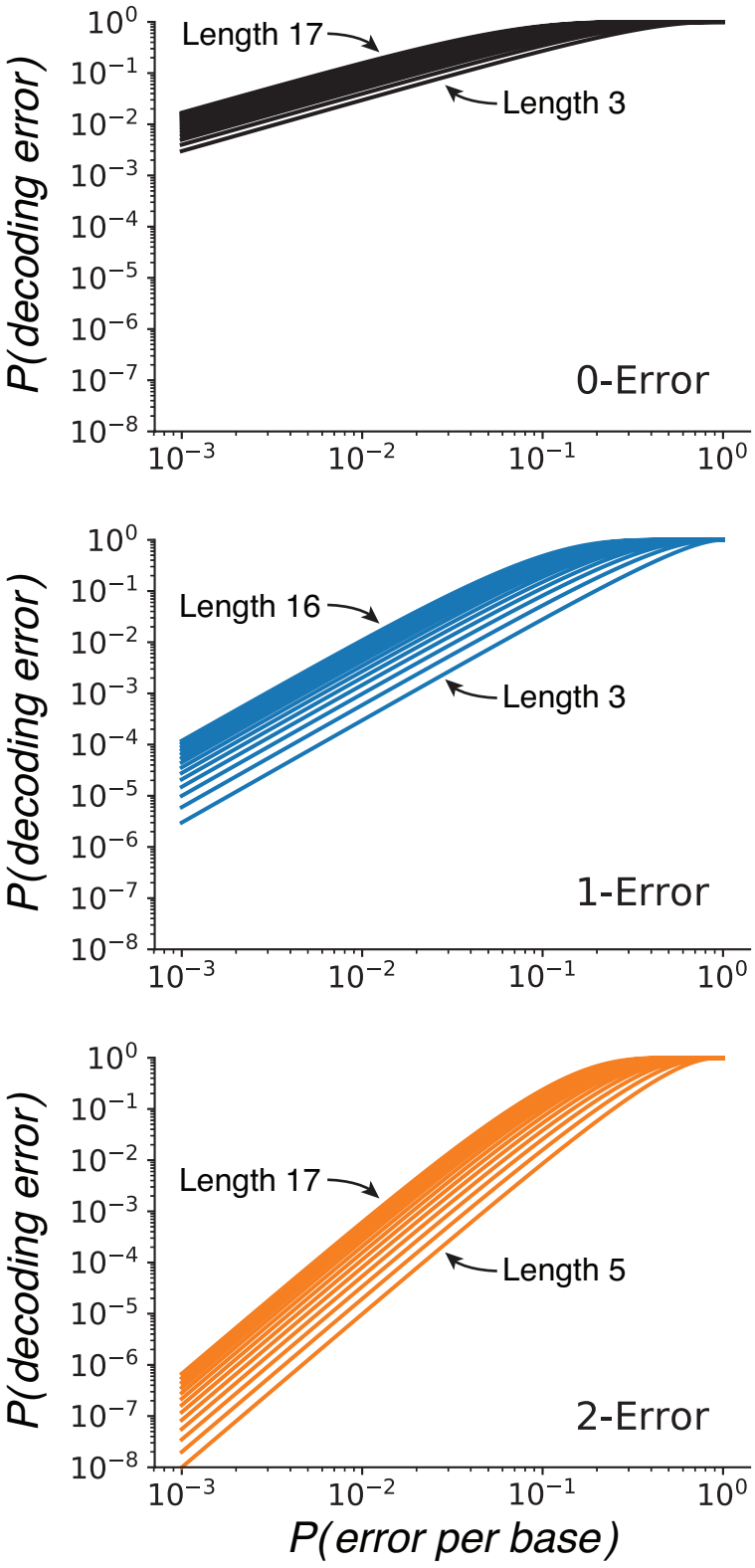

Figure S4: Error rate comparison with constant barcode number of errors corrected. The binomial approximation of the decode error rate as a function of the error rate per base, grouped by given number of errors corrected.

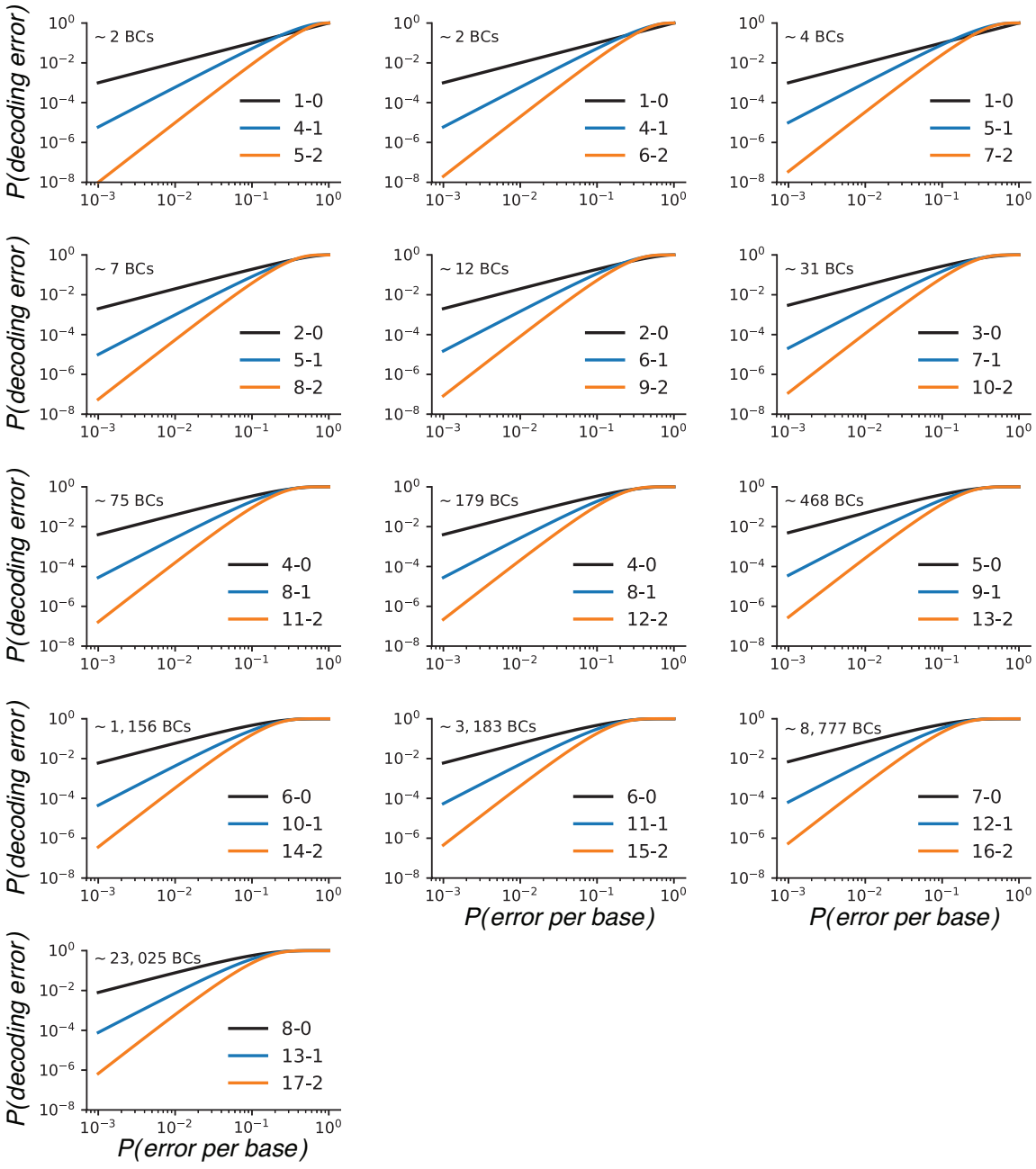

Figure S5: Error rate comparison with constant number of barcodes. The binomial approximation of the decode error rate as a function of the error rate per base, grouped by number of barcodes. Numbers of barcodes were not precisely equal. Rather, each panel starts with the number of 2-error correcting barcodes and uses the smallest 0- and 1-error correcting barcode sets with at least as many barcodes.

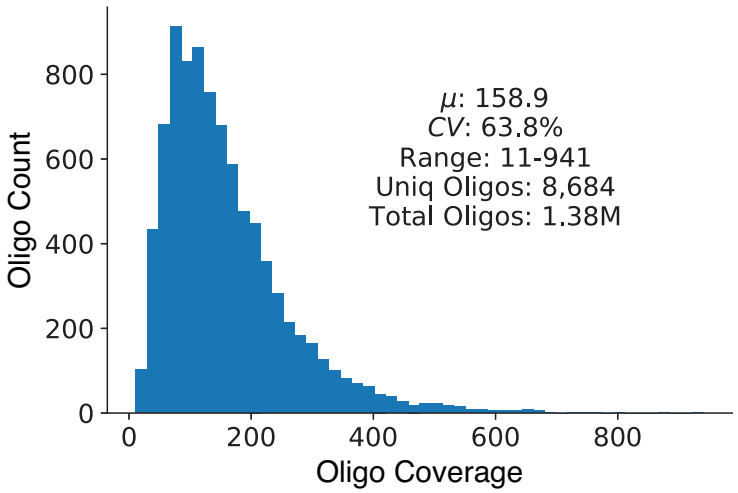

Figure S6: Coverage. Coverage histogram and statistics for the FREE code validation experiment. Each of the 8,684 oligos was observed with average coverage of 159x.

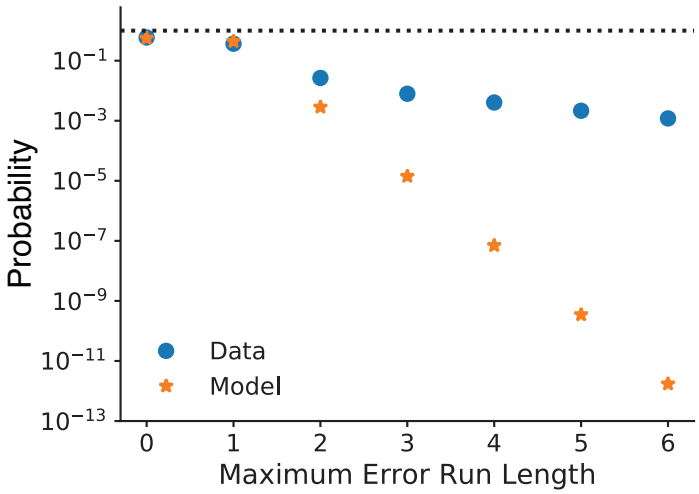

Figure S7: Maximum error run length probabilities. The probability distribution of maximum consecutive-error run lengths from a model assuming independent errors (Supplemental Materials) and from our data. The two differ significantly because errors in our data are not independent.

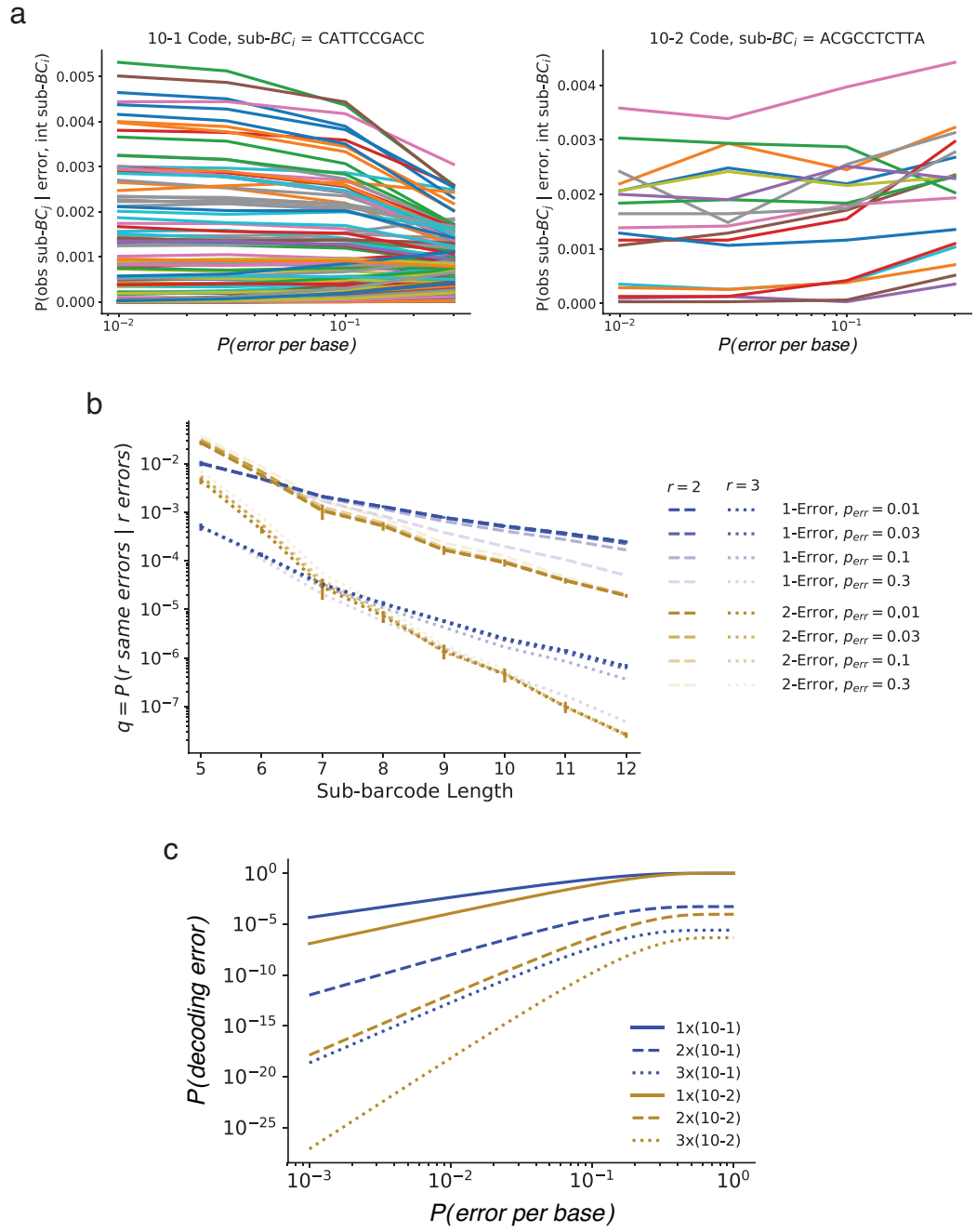

Figure S8: Repeated, concatenated sub-barcode behavior. **a.** Example values of P(observed sub-BC_j | decoding error, intended sub-BC_i) for sub-BC_i's from the 1- and 2-error correcting codes of length 10 across a range of per-base error rates. Individual lines represent different sub-BC_j's. **b.** Estimated values for q from direct simulation. $p_{err} = P(error\ per\ base)$. Error bars give the S.E.M., shown only for $p_{err} = 0.01$. **c.** Decoding error probabilities for repeated, concatenated sub-barcodes after filtering mismatching decoded sub-barcodes. The value of q was taken as that estimated for $p_{err} = 0.01$, since p_{err} is observed above to have a relatively small effect on q except in a few cases where $p_{err} = 0.01$ gives the most conservative estimate of q. Code label format example: 2x(10-1) is 2 repeated sub-barcodes of length 10 and 1-error correction.

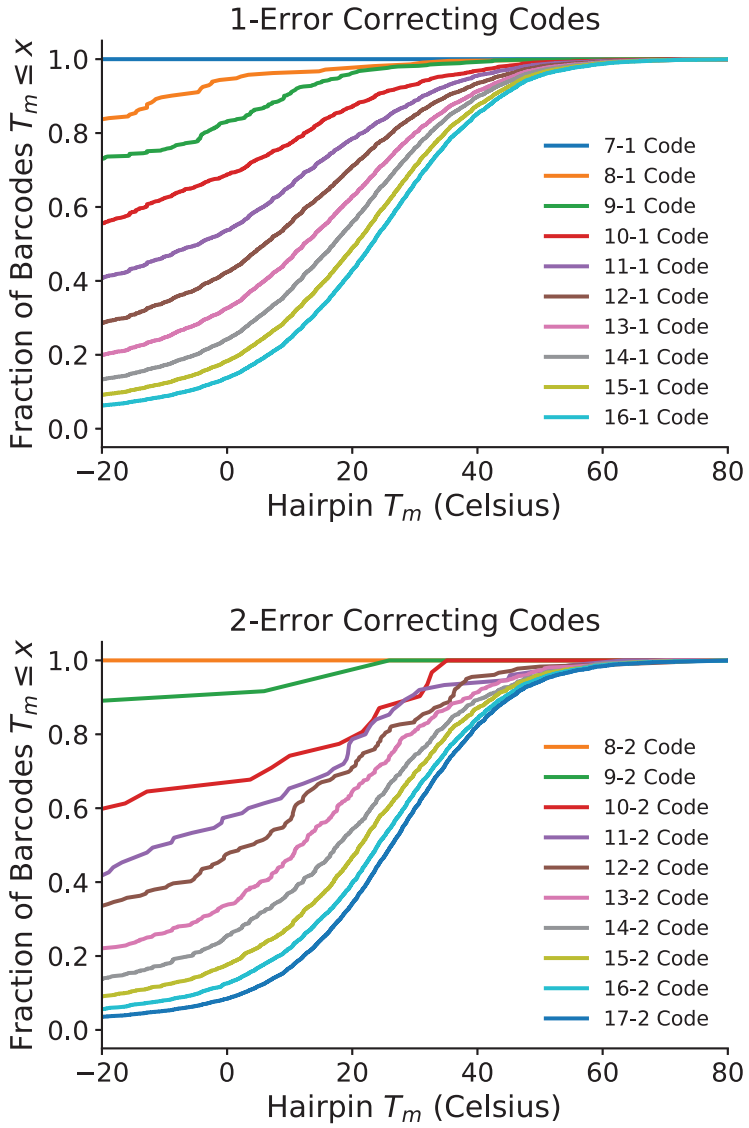

Figure S9: Hairpin melting temperatures. Hairpin melting temperature CDFs are shown for all barcodes libraries included with this manuscript. The barcodes included here nearly all have $T_m < 60°C$, and users can further filter the barcode sets to avoid hairpins in their specific experimental conditions. The calculated T_m of each barcode is included in the Supplemental Data.