

Supplementary File S2

Céline Trébeau, Jacques Boutet de Monvel, Fabienne Wong Jun Tai, Christine Petit, Raphaël Etournay.

1. The maximum value of the entropy of a barcode distribution

Given an integer $N \geq 1$, let $C = (b_1, \dots, b_N)$ be any list chosen among n available barcode indices ($b_i \in \{1, \dots, n\}$ for $1 \leq i \leq N$), and let $x = (x_1, \dots, x_n)$ be the corresponding empirical distribution, defined by $x_l = \#\{i \leq N : b_i = l\}$ for $1 \leq l \leq n$, that is, x_l is the number of times the index l occurs in (b_1, \dots, b_N) . To x we associate its Shannon entropy $S(x) = S(x_1, \dots, x_n) = -\sum_{l=1}^n p_l \log(p_l)$, where $p_l = \frac{x_l}{N}$ for $1 \leq l \leq n$. We will denote $S(C) = S(x) = S(p)$ and talk indifferently about the entropy of C , x or p . The purpose of this section is to prove the following:

Proposition. Given any n -tuple $x = (x_1, \dots, x_n)$ of positive integers such that $\sum_{l=1}^n x_l = N$, the maximum possible value of $S(x)$ is

$$S_{max} = -(n-r) \frac{\lfloor N/n \rfloor}{N} \log\left(\frac{\lfloor N/n \rfloor}{N}\right) - r \frac{\lceil N/n \rceil}{N} \log\left(\frac{\lceil N/n \rceil}{N}\right) \quad (1)$$

where r denotes the rest of the division of N by n , and $\lfloor \alpha \rfloor$ and $\lceil \alpha \rceil$ denote the lower and upper integer parts of α , respectively.

Proof. Note first that, by euclidean division of N by n , we may write $N = n \lfloor N/n \rfloor + r = (n-r) \lfloor N/n \rfloor + r \lceil N/n \rceil$, where $r \in \{0, \dots, n-1\}$. It follows that we may form a distribution $x^* = (x_1^*, \dots, x_n^*)$ such that $\sum_{l=1}^n x_l^* = N$ by setting $x_1^* = \dots = x_{n-r}^* = \lfloor N/n \rfloor$, and $x_{n-r+1}^* = \dots = x_n^* = \lceil N/n \rceil$. Clearly $S(x^*) = S_{max}$, and similarly $S(x) = S_{max}$ for any distribution x obtained by a permutation of x^* . Let now $x = (x_1, \dots, x_n)$ be any distribution satisfying $\sum_{l=1}^n x_l = N$. We may then proceed to perform a sequence of changes that will bring x to one of the permuted versions of x^* , as follows:

- i) Search for an index $s \in \{1, \dots, n\}$ such that $x_s \geq \lceil N/n \rceil + 1$. If no such s exists, stop.
- ii) If such an s is found, there must exist another index s' such that $x_{s'} \leq \lfloor N/n \rfloor - 1$ (for otherwise one would deduce that $\sum_{l=1}^n x_l \geq N + 1$). Define the new distribution $y = (y_1, \dots, y_n)$ by setting $y_s = x_s - 1$, $y_{s'} = x_{s'} + 1$, and $y_l = x_l$ for $l \neq s, s'$. Set $x \leftarrow y$ and proceed to step i).

It should be clear that a finite number of iterations of steps i) and ii) will produce a distribution x whose entries are all equal either to $\lfloor N/n \rfloor$ or to $\lceil N/n \rceil$. Hence this distribution must be a permutation of x^* as claimed. To complete the proof of the proposition, it suffices to note that, as a result of well known concavity properties of the entropy, for each of these steps we have $S(y) \geq S(x)$. For the convenience of the reader, we detail the proof below:

Lemma. Let $x = (x_1, \dots, x_n)$ be any integer distributions satisfying $\sum_{l=1}^n x_l = N$, and let $y = (y_1, \dots, y_n)$ be a distribution obtained from x by application of the steps i) and ii) above. Then $S(y) \geq S(x)$.

Proof. Setting $p_l = \frac{x_l}{N}$, $p'_l = \frac{y_l}{N}$ for $1 \leq l \leq n$, note that we have $p'_s = p_s - \frac{1}{N}$, $p'_{s'} = p_{s'} + \frac{1}{N}$, and $p'_l = p_l$ for $l \neq s, s'$. It follows that

$$\begin{aligned} S(y) - S(x) &= -\sum_{l=1}^n p'_l \log(p'_l) + \sum_{l=1}^n p_l \log(p_l) \\ &= \sum_{l=1}^n p_l (\log(p_l) - \log(p'_l)) + \frac{1}{N} (\log(p'_s) - \log(p'_{s'})) \\ &= \sum_{l=1}^n p_l \log\left(\frac{p_l}{p'_l}\right) + \frac{1}{N} \log\left(\frac{p'_s}{p'_{s'}}\right). \end{aligned}$$

In the last expression, the first term $\sum_{l=1}^n p_l \log\left(\frac{p_l}{p'_l}\right)$ is nothing but the relative entropy (or Kullback-Leibler divergence) of the distribution $p = (p_l)$ with respect to $p' = (p'_l)$, which is well known to be positive; the second term is also positive, since $p'_s \geq p'_{s'}$ by construction. Therefore, we have $S(y) - S(x) \geq 0$, as was to be shown.

The above lemma is best understood in the context of majorization theory. Namely, the inequality $S(y) \geq S(x)$ follows from the fact that the sequence y is “less spread” than x , and that the Shannon entropy increases under spreading. In technical terms, y is majorized by x , and S is a Schur-concave function. (For precise definitions and a detailed account of majorization theory, see Marshall, Olkin and Arnold, 2011.) Here, y is obtained from x by a “ T -transform” (also called “Dalton transfer” or “Robin Hood transfer”), that is an operator of the form $y = Tx = \lambda x + (1 - \lambda)x'$, where $0 \leq \lambda \leq 1$ and x' is obtained from x by exchanging two entries. The effect of such a transform is to mix two entries of a vector, making them closer without changing their sum, which can only increase the entropy.

2. Optimization of the entropy of barcode distributions by a Random-Greedy search algorithm

Let us first fix some notations. We denote $I = \{b_1, \dots, b_n\}$ the set of available barcodes in some RNAseq experiment, and $\mathcal{C}_{comp} = \{c_1, \dots, c_{N_{comp}}\}$ a set of N_{comp} compatible barcode combinations, such as the set produced by Step 1 of the DNABarcodeCompatibility algorithm (§2.1). For a multiplexing level k , each combination can be seen as a k -tuple $c = (b_{l_1}, \dots, b_{l_k})$, i.e. an element of I^k , whose coordinates are distinct and ordered indifferently. We consider here the problem of maximizing the entropy a set of $a = N/k$ barcode combinations selected from \mathcal{C}_{comp} (Step 2 of the algorithm). Each selection is itself a a -tuple $C = (c_{i_1}, \dots, c_{i_a})$ of compatible combinations, or an element of $(\mathcal{C}_{comp})^a$ without repetitions, which can also be seen as a vector of $N = ak$ barcodes, $C = (b_1, \dots, b_N) \in I^N$, this time with possible repetitions. The entropy of C is defined as above as the Shannon entropy of the empirical distribution $x = x(C) = (x_1, \dots, x_n)$ corresponding to C . (That is, $S(C) = S(x) = S(x_1, \dots, x_n)$ where $x_l = \#\{i \leq N : b_i = l\}$ for $1 \leq l \leq n$.) Because \mathcal{C}_{comp} is a subset of the set of all $\binom{n}{k}$ possible combinations, finding a selection C of maximum entropy in $(\mathcal{C}_{comp})^a$ is a non trivial combinatorial optimisation problem. Even though N_{comp} is usually much smaller than $\binom{n}{k}$, in general the total number $\binom{N_{comp}}{a}$ of possible selections is prohibitively large, making an exhaustive search impractical. Fortunately, it turns out that a simple random-search algorithm performs very well in maximizing the entropy. The random search can moreover be supplemented by a greedy-like strategy, making the algorithm highly effective in practice. The question of finding an exact entropy maximization algorithm is thus of limited practical interest for applications to RNAseq experiments, and was left for further work. Two random-greedy search algorithms (referred to as the greedy-descent and the greedy-exchange algorithms, described in pseudo-code below) are implemented in the DNABarcodeCompatibility package. In the implementation, the chosen algorithm (either greedy descent or greedy exchange) is iterated a number of times in order to ensure a high probability of attaining the combination of highest entropy among the possible choices of $C \subset \mathcal{C}_{comp}$. At the end of the iterations, the solution of highest entropy that has been found is chosen.

These two greedy algorithms were tested extensively on sets \mathcal{C}_{comp} that were randomly chosen from the compatible barcode combinations generated either for the Illumina barcode set (which contains 48 barcodes), or an artificially generated set of barcodes that was disjoint from the Illumina set and constrained to be robust against a single base calling error. The results

of these simulations are described in the Supplementary File S1 (R Markdown document “Simulations”) accompanying the package’s documentation on GitHub. Qualitatively similar results were obtained on sets \mathcal{C}_{comp} that were fully randomly generated (by picking N_{comp} combinations among the $\binom{n}{k}$ possible choices). In brief, the random-greedy algorithms appear to be near-optimal in the sense that in most of the cases tested they either find a solution reaching the maximum entropy bound S_{max} (given by Eq. 1 above), or a solution having an entropy S_{opt} close to S_{max} , which is, with high probability, the selection of maximum entropy attainable for the set \mathcal{C}_{comp} at hand. In a few exceptional, or “degenerate” cases, the probability to find an optimal combination turned out to be less than 1%. While not guaranteed to be optimal, the combination obtained after 100 iterations of the greedy algorithm in such cases was still better optimized (with a barcode distribution much closer to the uniform distribution) than the combination of highest entropy found among 100 combinations picked at random. For details see the Supplementary File S1 on <https://github.com/comotopasteur-fr/DNABarcodeCompatibility>.

Algorithm 1: Procedure Greedy Descent:

Set $N' > a$ (by default $N' = \min(N_{comp}, 120)$);
Generate random initial selection $C' = (c_1, \dots, c_{N'}) \subset \mathcal{C}_{comp}$;
while $N' > a$ **do**:
 Solve $l^* = \arg \max_{1 \leq l \leq N'} (S(C' - c_l))$;
 ($C' - c_l$ denotes the selection obtained from C' by removing c_l)
 Set $N' \leftarrow N' - 1$ and $C' \leftarrow C' - c_{l^*}$;
end while
Output C'

Algorithm 2: Procedure Greedy Exchange

Set $N' > a$ (by default $N' = \min(N_{comp}, 120)$);
Generate random subset $C' = (c_1, \dots, c_{N'}) \subset \mathcal{C}_{comp}$;
Generate random initial selection $C = (c_1, \dots, c_a) \subset C'$;
while C is not 2-exchange optimal **do**:
 search for $l \in \{1, \dots, a\}$ **and for** $i \in \{1, \dots, N'\}$ **such that** $S(C - c_l + c_i) > S(C)$:
 ($C - c_l + c_i$ denotes the selection obtained from C by exchanging c_l with c_i)
 if l and i are found, **set** $C \leftarrow C - c_l + c_i$ **and escape search**;
 else declare C **to be 2-exchange optimal**.
end while
Output C