Sample pooling methods for efficient pathogen screening: Practical implications

Tara N. Furstenau[1], Jill H. Cocking[1,2], Crystal M. Hepp[1,2], Viacheslav Y. Fofanov[1,2*]

1 School of Informatics, Computing, and Cyber Systems, Northern Arizona University, Flagstaff, Arizona, USA
2 Pathogen and Microbiome Institute, Northern Arizona University, Flagstaff, Arizona, USA

* Viacheslav.Fofanov@nau.edu

Abstract

Due to the large number of negative tests, individually screening large populations for rare pathogens can be wasteful and expensive. Sample pooling methods improve the efficiency of large-scale pathogen screening campaigns by reducing the number of tests and reagents required to accurately categorize positive and negative individuals. Such methods rely on group testing theory which mainly focuses on minimizing the total number of tests; however, many other practical concerns and tradeoffs must be considered when choosing an appropriate method for a given set of circumstances. Here we use computational simulations to determine how several theoretical approaches compare in terms of (a) the number of tests, to minimize costs and save reagents, (b) the number of sequential steps, to reduce the time it takes to complete the assay, (c) the number of samples per pool, to avoid the limits of detection, (d) simplicity, to reduce the risk of human error, and (e) robustness, to poor estimates of the number of positive samples. We found that established methods often perform very well in one area but very poorly in others. Therefore, we introduce and validate a new method which performs fairly well across each of the above criteria making it a good general use approach.

Introduction

For targeted surveillance of rare pathogens, screenings must be performed on a large number of individuals from the host population to obtain a representative sample. For pathogens present at low carriage rates of 1% or less, a typical detection scenario involves testing hundreds to thousands of samples before a single positive is identified. Although advances in molecular biology and genomic testing techniques have greatly lowered the cost of testing, the large number of negative results still renders any systematic pathogen surveillance program inefficient in terms of cost, reagents, and time. These costs can quickly become prohibitively expensive in resource-poor settings (e.g. pathogen surveillance in developing countries [1, 2], in non-human systems, such as wildlife disease surveillance [3]), or when reagents become scarce due to a rapid spike in testing demand (e.g. during the SARS-CoV-2 pandemic [4]).

Robert Dorfman first introduced a method to improve the efficiency of large-scale pathogen screening campaigns during World War II. In an effort to screen out syphilitic men from military service, the US was performing antigen-based blood tests on millions of specimens in order to detect just a few thousand cases. The large number of negative tests struck Dorfman as being extremely wasteful and expensive and he proposed that more information could be gained per test if many samples were pooled together and tested as a group [5]. If the test performed on the pooled samples was negative (which was very likely), then all individuals in the group could be cleared using a single test. If the pooled sample was positive, it would mean that at least one individual in the sample was positive and further testing could be performed to isolate the positive samples. This procedure had the potential to dramatically reduce the number of tests required to accurately screen a large population and it sparked an entirely new field of applied mathematics called group testing.

Due to practical concerns, Dorfman's group testing approach was never applied to syphilis screening because the large number of negative samples had a tendency to dilute the antigen in positive samples below the level of detection [6]. Despite this, sample pooling has proven to be highly effective when using a sufficiently sensitive, often PCR-based, diagnostic assay. In fact, ad hoc pooling strategies have long been used to mitigate the costs of pathogen detection in disease surveillance programs. For example, surveillance of mosquito vector populations in the U.S. involves combining multiple mosquitoes of the same species (typically $1 - 50$) into a single pool, prior to testing for the presence of viral pathogens [7–10]. Elsewhere, such pooling techniques have been successful in reducing the total number of tests in systems ranging from birds [11], to cows [12], to humans [13–15]. In many wildlife/livestock surveillance programs, sample pooling is used to simply determine a collective positive or negative status of a population (e.g. a herd or flock) without identifying individual positive samples. While this is often appropriate and sufficient for small-to-medium scale research experiments or surveillance programs, a well designed pooling scheme can easily provide this valuable information with little additional cost. For the purposes of this paper, we will focus on pooling methods that provide accurate classification of each sample so that infected individuals can be identified.

Group testing theory primarily focuses on minimizing the number of tests required to identify positive samples and many nearly-optimal strategies for sample pooling have been described. From a combinatorial perspective, a testing scheme begins by examining a sample space which includes all possible arrangements of exactly k positive samples in N total samples. Because the positive samples are indistinguishable from negative samples, a test must be performed on a sample or a group of samples in order to determine their status. The test is typically assumed to always be accurate, even when many samples are tested together (in practice, this is often not the case and approaches that consider test error and constraints on the number of samples per pool have been examined [16, 17]). In the worst case, all of the samples would need to be tested individually requiring N tests. The goal of group testing is to devise a strategy which tests groups of samples together in order identify the positive samples in fewer than N tests. Group testing methods are generally more efficient when positive samples are sparse. As the number of positive samples increases, the number of tests will eventually exceed individual testing for all of the methods. This point has been previously estimated to be roughly when the number of positives is greater than $\frac{N}{3}$ for sufficiently large N [18, 19]. In order to establish the most optimal testing procedure, many group testing schemes are modified based on the expected number of positive samples, $\hat{k}$. Because it is impossible to know the exact number of positive samples, problems arise when this estimate is not accurate (e.g. overestimation may require more tests to be performed than necessary, and underestimation may result in positive samples going undetected). Therefore, it is important to not only consider how different schemes scale as the number of positive samples increases but also how robust they are when the number of positive samples is misestimated.

For real-world applications, many factors should be considered when designing a pooling strategy, depending on the circumstances. Finding the best strategy often involves weighing the tradeoffs between the following factors: (a) number of tests, to minimize costs and save reagents, (b) number of sequential steps, to reduce the time it takes to complete the assay, (c) number of samples per pool, to avoid the limits of detection, (d) simplicity, to reduce the risk of human error, and (e) robustness, to poor estimates of the number of positive samples. We have identified several pooling strategies that perform well or optimally with respect to at least one of these factors. The goal of this paper is to directly compare the strengths and weaknesses of each strategy and identify the approaches that we feel are most appropriate for small-to-medium scale research experiments or surveillance programs. With this goal in mind, we favored strategies that provided the best balance across each of our criteria, particularly those that maximized the ease of performing the pooling procedure using standard laboratory equipment (i.e. defining pooling groups in ways that are easily captured using multi-channel pipettes).

We present the pros and cons of different pooling strategies by providing graphical results from computational simulations with minimal use of mathematical formulas. We focused on making the simulation results as directly comparable as possible and used realistic sample sizes (in multiples of 96 well plates) for small to medium scale experiments. The computational simulations allow us to directly compare (a) the number of tests, (b) the number of steps, (c) the number of samples per pool, (d) the number of individual pipettes, and (d) the robustness for five existing pooling strategies. We also introduce a new strategy that provides key advantages in simplicity and provides the best balance between the other criteria. Finally, we experimentally validate our strategy by testing pools of cow's milk to detect samples that are positive for the pathogen *Coxiella burnetti*.

Review of pooling strategies compared in this work

Pooling strategies often take either a non-adaptive or an adaptive approach. In non-adaptive methods, an optimal pooling strategy is designed in advance (for a given number of samples with an expected number of positives) and therefore it does not adapt based on information gained from the test results. Tests are run on each of the pools in parallel and the results are decoded when the tests are complete to determine which are positive. The ability to run all of the tests in parallel can save a lot of time and this is one of the main benefits of non-adaptive tests. Adaptive methods, on the other hand, require a series of steps that must be performed sequentially because each step relies on information gained from the outcome of a previous step. However, because more information is known at each step, adaptive algorithms often require fewer tests than non-adaptive methods. Below we describe several examples of both non-adaptive and adaptive pooling approaches and, in each case, we assume that the test applied to the pools is noiseless (the test will always be positive if a positive sample is present in the pool and negative otherwise) and it produces only a binary or two-state outcome (e.g. positive/negative or biallelic SNP typing).

DNA Sudoku

DNA-sudoku is a popular example of an optimal non-adaptive pooling strategy. This strategy is based on the idea that if a sample is present in multiple positive pools and not in any negative pools, then it is likely to be positive. However, ambiguous results can arise if multiple samples co-occur within the same positive pools because it is no longer possible to determine if one or both of the samples are truly positive. DNA Sudoku provides a more rigorous approach to avoid such ambiguity by minimizing the number of times any two samples are included in the same pool [20]. This is achieved by staggering the samples that are added to each pool in different sized windows or intervals (Fig 1); importantly, the size of the windows must be greater than $\sqrt{N}$ and co-prime to minimize the intersections between samples. The number of different pooling windows (the weight) should be one greater than the expected number (upper bound) of positive samples, $w = \hat{k} + 1$, to ensure accurate results.

Fig 1. DNA Sudoku pooling example. In this example, there are a total of $N = 96$ samples. The 96-well plates show which samples are combined into each pool for the two different window sizes ($W_1 = 10$ and $W_2 = 11$ which are greater than $\sqrt{N}$ and co-prime). By using two different window sizes, the weight of this pooling design is $w = 2$ meaning that $k = w - 1 = 1$ positive sample can be unambiguously identified in a single step using $T = W_1 + W_2 = 21$ tests. The positive samples are decoded by finding the samples that appear most often in the positive pools. For example, if G10 is the only positive sample, we can detect this from the pooling results by noticing that G10 was added to both of the positive (red) pools while other samples in those pools were added to only one or the other. Alternatively, if both G10 and D4 are positive, four samples occur with equal frequency (D4, G10, E12, and F2) in the positive pools (red and purple) and it is impossible to determine which are the true positive samples. This ambiguity is introduced because the test was designed to handle only one positive sample.

Once the samples are pooled for each window size and the pools are tested, a decoding scheme is used to identify the positive samples from the positive pools. The decoding works by identifying which samples occur the most frequently in the positive pools. If the weight is chosen correctly using a good estimate of k, all of the positive samples can be unambiguously identified. However, if the true number of positive samples exceeds $\hat{k}$, the results become ambiguous and false positives can occur. Over estimating the maximum number of positive samples can provide a buffer against ambiguous results but this comes with a large increase in the number of tests ($> N$ additional tests for each additional pooling window). Alternatively, the ambiguous samples can be tested individually, but this requires an additional round of testing which voids one of the main advantages of non-adaptive testing.

When $\hat{k}$ is estimated appropriately, DNA Sudoku is a very efficient non-adaptive approach, especially when the number of samples is very large. It was originally designed for pooling and barcoding thousands of DNA samples in preparation for high-throughput sequencing. However, because it was intended for use in large-scale sequencing facilities with robotic equipment, the pooling design is complex and intricate and therefore difficult for a human technician to perform accurately and consistently by hand.

Two Dimensional Pooling

Multidimensional pooling is another non-adaptive approach that is generally easier to perform than DNA Sudoku but can be more prone to producing ambiguous results. As the name implies, this procedure can be extended to many dimensions [21, 22], however it becomes more difficult to perform without robotics when more than two dimensions are used. In the two dimensional (2D) case, N samples are arranged in a perfectly square 2D grid or in several smaller but still square sub-grids [23]. For example, when testing 96 samples (as in Fig 2), this could be achieved through a single 10x10 grid or through 4 5x5 sub-grids (with 4 empty spaces). Once arranged, all of the samples along each individual column and each individual row are pooled. This results in 20 pools for a 10x10 grid, and 40 pools for 5x5x4 grids. Once the pools are tested, the positive samples are decoded by identifying which of them are present at the intersection of positive rows and columns [23].

In 2D pooling, ambiguous results arise when positive samples are present in multiple rows *and* multiple columns (e.g. in the top left grid in Fig 2, the two positive rows and

Fig 2. Two-Dimensional pooling example. A total 96 samples are arrayed in symmetrical 5x5 grids (with 4 empty wells in the last grid) and $k = 9$ of the samples are positive (red wells). The pooling procedure combines each row and each column of a grid into separate pools for a total of $T = 2 \times 5 \times 4 = 40$ tests. Samples that are at the intersection of a positive row and a positive column (marked with an "X") are potentially positive samples. When more than one row *and* more than one column are positive, some of the samples at the intersections are likely false positives (e.g. the top left and bottom right grids). Otherwise, the results are unambiguous and the correct positive samples can be identified (e.g. the top right and bottom left grids).

the two positive columns intersect at four wells, only two of which (red wells) are positive). When this occurs, the number of intersecting points is almost always higher than the true number of positive samples. Ambiguous results can be somewhat mitigated by decreasing the size of the grid as the expected number of positive samples increases. The chances of ambiguous intersections increase when there are more positive samples, so using more grids of smaller size (and consequently more tests), will make ambiguous arrangements less likely. Alternatively, the ambiguous samples can be tested individually in a second followup round of testing, but this again nullifies the main benefit of non-adaptive testing, which is the ability for all tests to be carried out in parallel.

S-Stage Approach

Dorfman's original pooling design for syphilis screening was an adaptive two-stage test. Following this method, samples are partitioned and tested in g groups of size n. All of the samples in groups with negative results are considered to be negative and all of the samples in groups with positive results are tested individually. Ignoring the constraints of the actual assay, the optimal group size that minimizes the number of tests depends on the number of positive samples, k. Specifically, there should be roughly $\sqrt{Nk}$ groups of size $\sqrt{\frac{N}{k}}$ [5, 24]. Dorfman's two-stage approach was later generalized to any number of stages using Li's S-Stage algorithm [24], which can reduce the number of tests required to identify positive samples. At each stage, s_i, of the S-Stage algorithm (Fig 3), the untested samples are arbitrarily divided into g_i groups of size n_i and the test is performed on each group. The samples in pools with negative test results are deemed negative and removed from consideration. The samples in positive pools move on to the next stage where they are redivided into g_{i+1} groups of size n_{i+1}. This is repeated until the final stage, where $n_s = 1$, and all of the remaining samples are tested individually. The optimal number of samples per group at each step is $n_i = \left(\frac{N}{k}\right)^{\frac{s-i}{s}}$ and the optimal number of steps is $S = \ln(\frac{N}{k})$ which achieves an upper bound of $\frac{e}{\log_2 e} k \log_2 \left(\frac{N}{k}\right)$ tests. Li demonstrated that misestimation of the number of positive samples, $\hat{k}$, has only a small impact on the total number of tests, especially when the number of stages is high. The S-Stage algorithm can require many more steps than non-adaptive algorithms, but when the number of steps is low, it compares favorably, especially in cases when the non-adaptive methods require additional validation steps.

Fig 3. S-Stage pooling example. For 96 samples with an estimate of 3 positive samples, the S-Stage algorithm requires 4 steps. In the first step (top 96 well plate), 96 samples are tested in 6 groups (black outline) of 16. In the next step, the samples in the positive pools from the previous step are arbitrarily redivided into 5 groups of 6 or 7 samples and tested. In the third step, the samples from positive pools from step 2 are redivided into 4 groups of 3 or 4. In the final step, individual testing is performed on samples from the positive pools in step 3. The number of tests required depends on the initial arrangement of positive samples within the pools but in this example 21 tests are required to identify 3 positive samples (red wells). The number of tests is lower than the upper bound in this case due to the fortunate placement of two positive samples in the same pool in steps 1-3.

Binary Splitting by Halving

Sobel and Groll [25,26], introduced several adaptive group testing algorithms based on recursively splitting samples into groups and maximizing the information from each test result. They demonstrated that this class of algorithm is robust to inaccurate estimates of k, particularly in the case of the Binary Splitting by Halving algorithm which can be performed without any knowledge of the number of positive samples. Binary Splitting by Halving (Fig 4) begins by testing all of the samples in a single pool. If the test is negative, all of the samples are negative and testing is complete, if the test is positive, the samples are split into two roughly equal groups and only one of the groups is tested in each step. If the tested half is negative, we know that all of the samples in the tested group are negative and testing is now complete for those samples. We also know that the untested half must contain at least one positive sample (because the test containing all of the samples tested positive). Alternatively, if the tested pool is positive, we know that it contains at least one positive sample and we know nothing about the untested half. In either case, the binary splitting always continues with the group that is known to contain a positive sample until a single positive sample is identified with individual testing. At this point, all of the samples that remain untested are added to a single pool and tested, beginning the process again. This is repeated k times and stops when the initial test of all the remaining samples is negative or when all samples have been tested (either through individual testing or elimination). Using this method, k positive samples can be identified in at most $k \log_2 N$ tests. Binary splitting is only efficient when fewer than 10% of samples are positive, otherwise more tests are required than individual testing [25,26]. This is the only approach discussed here that does not rely on an estimate of k and therefore the performance is not impacted by misestimation of the number of positive samples.

Fig 4. Binary splitting by halving pooling example. In this example, there are $N = 96$ samples and two of the samples are positive (red wells). To begin, all of the samples are pooled and tested (Step 1). If the first test is negative, testing is complete and all samples are considered negative. Otherwise, half of the samples are pooled and tested (Step 2). If the tested half is negative, then all of the samples in the tested half are considered to be negative and at least one negative sample is known to be present in the other non-tested half of the samples. If the tested half is positive, then it contains at least one positive sample and no information is gained about the other untested half. In either case, the method continues by halving and testing whichever group is known to contain a positive sample until a single positive sample is identified (either by individual testing, as seen in Step 7, or by elimination, as seen in Step 16). Once a single positive sample is identified, the remaining unresolved samples (non-grey wells) are pooled and tested to determine if any positive samples remain and the process continues until all positive samples are identified. Only one test is required per round, and in this example, it takes 17 sequential rounds to recover both positive samples.

Generalized Binary Splitting

Hwang's Generalized Binary Splitting algorithm is very similar to Binary Splitting by Halving (Fig 4) except the size of the first split is optimized for the expected number of positive samples. This is important because it helps bypass some of the early and least productive tests. In the Halving method, as the number of positive samples increases, the first few tests are more likely to be positive due to chance. Positive tests, in general, provide the fewest pieces of information and do not eliminate any negative samples; consequently, positive tests are particularly inefficient early on when the potential to eliminate large groups of samples is highest. Additionally, it means that each binary search will begin with a large number of samples which will require more tests and steps to identify the first positive sample. To solve this problem, The Generalized Binary Splitting algorithm attempts to modify the size of the initial pool so that it is small enough to capture a single positive sample on average. When smaller groups are tested they are less likely to be overwhelmingly positive which means more samples can be eliminated in negative tests and a single positive sample can be found quicker using fewer tests [27]. As the ratio of samples to positive samples $(\frac{N}{k})$ increases, the number of tests required to identify k positive samples approaches $k \log_2 \left(\frac{n}{k}\right)$ which is nearly optimal; however, like Binary Splitting by Halving, the Generalized Binary Splitting approach requires many sequential steps to complete testing.

Modified 3-Stage Approach

Here we are introducing a new approach that we developed with the goal of finding a good balance between the number of tests, the number of steps, simplicity, and robustness. We found that many of the methods described previously focus on optimizing only one of these features usually to the detriment of the others. Instead of attempting to perform the best in a single area, we wanted to take a more balanced approach and find tradeoffs that allow good performance across each of these areas. Our Modified 3-Stage approach (Fig 5) is based on the S-Stage approach but it is modified so that the number of steps is constrained to a maximum of three. At three steps, this approach requires only one additional step than ambiguous non-adaptive approaches that require two steps for complete validation. Because the S-Stage algorithm is already fairly robust, constraining the number of steps does not have a large impact on the number of tests required. We also modified our method to be simpler and easier to perform by borrowing the recursive subdividing used in the binary splitting approaches. In the S-Stage approach, the remaining samples in each step are arbitrarily redivided into pools. Not only does this make it difficult to keep track of the remaining samples spread across the plate, it can also make it more difficult to collect the samples for a pool using a multichannel pipette (e.g. Step 2 in Fig 3). Instead, we opted to recursively subdivide the samples from positive pools. This makes it easier to keep track of the samples that should be pooled at each stage and, because the samples are always in close proximity, they are easier to collect using a multichannel pipette (compare 3 and 5).

Fig 5. Modified 3-Stage pooling example. For 96 samples and an estimate of 2 positive samples, the Modified 3-Stage approach begins by creating 6 pools with 16 samples each. The positive pools from the first step are then subdivided into 4 groups of 4 in the second step. In the final step, the samples from the positive pools in step 2 are tested individually. In the modified 3-Stage approach, the pools are recursively subdivided into groups instead of arbitrarily redividing the remaining samples at each step. This is simpler and keeps the samples for each subsequent pool in close proximity. The total number of tests depends on the arrangement of the positive samples, but in this example, the modified 3-stage algorithm requires 22 tests.

Materials and methods

Computational simulations

Computational simulations were carried out for each of the six pooling strategies described above. The number of samples, N, in each test were in multiples of 96 to represent 96-well plates: $1 \times 96 = 96$, $4 \times 96 = 384$, and $16 \times 96 = 1,536$. Each set of samples was represented as a binary array of size N, where 1's represented positive samples and 0's represented negative samples. For each test, 100 simulations were generated by placing k positive values in random positions in the array, with k ranging from 1 to 20. In each simulation, the number of tests, the number of sequential steps, and the number of individual pipettings required to make the pools were recorded. In cases where it was appropriate, the number of pipettings was calculated assuming either an 8- or a 16-channel pipette in addition to a single channel pipette. We only considered pooling schemes that were able to completely and accurately identify all of the positive samples in the sample set. To accomplish this, some of the pooling schemes required additional steps and tests that are accounted for in the simulation. The simulation code is available at `https://github.com/FofanovLab/sample_pooling_sims`.

DNA Sudoku Simulations

For the DNA Sudoku experiments, we tested different weights ranging from 2 to the highest value that did not exceed the number of tests required for individual testing. For example, with a sample size of 96, the maximum weight we used was 6 with window sizes of 10, 11, 13, 17, 19, and 23; this testing design required 93 tests, in the unambiguous case, and including any additional testing windows would cause the number of tests to exceed individual testing. The window sizes at the maximum weight were 20, 21, 23, 29, 31, 37, 41, 43, 47, and 53 for 384 samples; and 40, 41, 43, 47, 49, 51, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97, 101, 103, 107, 109, and 113 for 1,536 samples. For smaller weights, the window sizes were just the first w window sizes listed here for each sample size. For the first round of testing, the total number of tests was equal to the sum of the window sizes.

$$\sum_{i=1}^{w} x_i > w\sqrt{N}$$

If the result was ambiguous (i.e. any time the number of positives exceeded $w-1$), the number of steps increased to two and additional tests, equal to the number of prospective positive samples, were added to the test count. Because the samples that were added to each pool are staggered, multichannel pipettes do not provide any advantage; therefore, the number of pipettings was calculated assuming only a single channel pipette ($N \times w + |\text{Prospective Positives}|$).

2D Pooling Simulations

For the 2D pooling simulations we used square $D \times D$ grids and each of the M grids in a simulation were the same size. The samples were pooled along each row and column requiring $2DM$ tests. When the results were ambiguous, the number of steps increased by one and the number of tests increased by the number of prospective positive samples to account for the validation. Because the pooling along columns and rows can be easily and more efficiently performed using multichannel pipettes, the number of pipettings was calculated using an 8- and a 16-channel pipette, in addition to a single channel pipette. The number of pipettings was the $\sum_{i=1}^{2DM} \lceil \frac{n_i}{c} \rceil$ where n_i is the number of samples in each row or column and c is the number of channels in the pipette. We assumed that any additional pipettings required for testing the ambiguous samples was performed with a single channel pipette.

S-Stage Simulations

The S-Stage simulations were provided with an expected number of positive samples, $\hat{k}$. The number of steps was calculated as $S = \ln\left(\frac{N}{\hat{k}}\right)$ and the number of samples per group was $n_i = \frac{N}{\hat{k}}^{\frac{s-i}{s}}$. Because these calculations do not provide integer values, a nearest integer approximation was used. Optimal integer approximations of these values can be determined numerically but here we consistently applied a ceiling function. For each number of true positive samples ($k = 1 - 20$) we ran simulations with expected values, $\hat{k}$, ranging from 1-20. The number of tests was calculated as

$$T = \sum_{i=1}^{s} g_i \leq \frac{N}{n_1} + \frac{kn_1}{n+2} + \ldots + \frac{kn_{s-2}}{n_{s-1}} + kn_{s-1}$$

where $g_i = \left\lfloor \frac{N_i}{n_i} \right\rfloor$ is the number of groups tested at each step. The number of pipettings for a single channel pipette was equal to the number of samples in each of the pools that were tested. For multichannel pipettes, the number of samples in each pool was divided by the number of channels and rounded up. In cases where the samples in the pool were not in adjacent wells, additional pipettings were required.

Modified 3-Stage Simulations

Our modified 3-Stage approach is similar to the S-Stage algorithm except that the number of steps was constrained to a maximum of three: $S = \min\left(3, \left\lceil \ln \frac{N}{\hat{k}} \right\rceil\right)$. In order to recursively subdivide each pool, the number of subgroups was calculated as $g_i = \left\lfloor \frac{n_i}{n_i+1} \right\rfloor$ with n_i calculated the same way as the S-Stage simulations. For each true number of positive samples ($k = 1 - 20$) we ran simulations with expected values ranging from $\hat{k} = 1 - 20$. The number of tests and the number of pipettings were calculated the same way as the S-Stage simulations.

Binary Splitting by Halving

The Binary Splitting by Halving simulations did not require any estimate of the number of positive samples. The simulation performed repeated binary searches for positive samples until no more positive samples remained. Only one test was performed at each step and, because each step depended on information gained in the previous step, none of the steps were performed in parallel. Therefore, the number of tests was equal to the number of steps. The number of pipettings was equal to the size of each pool divided by the number of channels in the pipette, rounded up.

Generalized Binary Splitting

The Generalized Binary Splitting simulations were similar to the Binary Splitting by Halving simulations except that the initial group size was calculated based on the number of expected positive samples ($\hat{k}$). More specifically, the initial group size was calculated as 2^a where $a = \left\lfloor \log_2\left(\frac{n-\hat{k}+1}{k}\right) \right\rfloor$. Binary Splitting by Halving (as described above) was performed on the initial group until a positive sample was identified at which point the value N was updated to reflect the number of remaining untested samples and the value $\hat{k}$ was decremented by 1 if a positive sample was found. The next group of 2^a was calculated using updated values of N and $\hat{k}$. This continued until either $N \leq 2\hat{k} - 2$, at which point the remaining samples were tested individually, or $\hat{k} = 0$, at which point all of the suspected positive samples were identified. Because the standard algorithm only guarantees that up to $\hat{k}$ positive samples will be found, we added additional rounds of binary splitting to ensure all of the positive samples were identified. The number of tests, steps, and pipettings were calculated the same way as the Binary Splitting by Halving simulations.

Experimental validation of modified 3-stage approach

We set up rare pathogen detection experiments in complex microbiome backgrounds to test our Modified 3-Stage approach. We used a total of 768 samples (eight 96-well plates) that contained a background of 2 μL of DNA extraction from cow's milk and 8 μL of molecular grade water. These samples originated from 24 distinct cow milk samples and were replicated (32 replicates each) to fill eight 96-well plates – a total of 24 unique microbiome backgrounds. *C. burnetti* DNA (1 μL) was added to 10 randomly chosen background samples ($\sim 1.3\%$ carriage rate) as we verified that the spike-in was successful using a highly sensitive Taqman assay designed to target the IS1111 repetitive element in *Coxiella burnetti* [28]. Using the same Taqman assay, we also verified that the target pathogen was not present in any of the 24 unique microbiome backgrounds prior to the spike-in. To ensure a consistent amount of background DNA, the milk extractions were tested to determine the amount of bacteria with a real-time PCR assay that detects the 16S gene and compares it to a known standard [29].

The pooling procedure was carried out by a typical researcher looking to identify samples that are positive for the pathogen of interest, *C. burnetti.* Assuming $N = 96$ and $\hat{k} = \lceil 96 \times 0.013 \rceil = 2$ the pooling scheme recommended by our modified 3-stage approach is depicted in Fig 5. In the first step, 6 pools consisted of 16 samples each, collected along every 2 columns of the 96 well plate using an 8-channel pipette. The 2 μL aliquots from each sample were collected in a plastic reservoir and then pipetted back into a single well in a new 96 well plate. The *C. burnetti* Taqman assay was used to test each of the pools. For the reaction, the following were combined for a final volume of 10 μL: 1 μL from the pool, 2 μL of Life Technologies TaqMan® Universal PCR Master Mix for a final concentration of 1X, 0.3 μL each of the forward and reverse primers for a concentration of 0.6 µM, 0.13 μL of the probe for a concentration of 0.25 µm and molecular grade water to a final volume of 10 μL. The reaction was run on an Applied Biosystems 7900 Real Time PCR system with the following conditions: 50 °C for 2 minutes, 95 °C for 10 minutes, and 40 cycles of 95°C for 15 seconds and 60°C for 1 minute. The second pooling step was carried out by subdividing the samples from the positive pools in the previous step into four groups of four samples. Again, 2 μL from each well was combined into the pool. These pools were subjected to Taqman *C. burnetti* assay as described above. Finally, the individual samples belonging to pools positive in the second pooling step, were tested as described above.

Results and Discussion

Number of tests

Because minimizing the number of tests is one of the primary goals of group testing, we begin by comparing the number of tests required for each method using a range of sample sizes: 96, 384, and 1,536. The number of positive samples ranged from 1 to 20 which resulted in minimum positive rates of 1.04%, 0.26%, and 0.07%; and maximum positive rates of 20.83%, 5.20%, and 1.30% for 96, 284, and 1,536 samples, respectively. Fig 6 directly compares the average number of tests for each method using the optimal parameter settings. For the S-Stage, Modified 3-Stage, and General Binary Splitting approaches, the results shown are for simulations where the expected number of positive samples was the same as the true number of positives ($k = \hat{k}$). For DNA Sudoku and 2D Pooling, the results shown are for simulations with parameters that resulted in the lowest average number of tests.

As expected, the General Binary Splitting method consistently required the fewest number of tests in all cases because it is nearly optimal according to group testing theory. Also expectedly, all of the pooling methods were most efficient positive samples were sparse (Fig. 6, top row where $k = 1$). The two non-adaptive methods (DNA Sudoku and 2D Pooling) required the highest number of tests when $k = 1$. DNA Sudoku performed slightly worse than 2D pooling (by one test) owing to the fact that

Fig 6. Comparison of the number of tests required for each pooling method. The bar graphs show the average number of tests required for each method and the error bars are the standard deviation across 100 simulations. The top row shows simulations with one positive sample and the bottom row shows simulations with 20 positive samples. The columns are different sample sizes from left to right: 96, 384, and 1,596. For the S-Stage, Modified 3-Stage, and General Binary Splitting approaches, the results shown are for simulations where the expected number of positive samples was the same as the true number of positives. For DNA Sudoku and 2D Pooling, the results shown are for simulations with parameters that resulted in the lowest average number of tests (DNA Sudoku: $w = 2$ when $k = 1$, and when $k = 20$, $w = 3$ for 96 samples and $w = 4$ for 384 and 1,536 samples; 2D Pooling: when $k = 1$, the grid sizes shown are 1x10x10 for 96 samples, 1x20x20 for 394 samples, and 1x40x40 for 1536 samples, and when $k = 20$ the grid sizes are 11x3x3 for 96 samples, 24x4x4 for 384 samples, and 96x4x4 for 1,536 samples).

the window sizes were co-prime instead of symmetrical like 2D Pooling. At the maximum number of positive samples for our simulations ($k = 20$), Binary Splitting by Halving performed the worst when the positive rate was high and exceeded individual testing for 96 samples (along with many DNA Sudoku simulations). The number of tests required for our Modified 3-Stage approach typically fell somewhere in the middle except for when the number of total samples and positive samples were highest. When $N = 1,536$ and $k = 20$, the Modified 3-Stage simulations required only slightly fewer tests, on average, than 2D Pooling, which performed the worst (Table 1).

Table 1. Summary of the performance of pooling methods for each of our areas of interest: number of tests, number of steps, number of samples per pool, robustness, and simplicity. For sample sizes $N = 96$, 384, and 1,536, the table shows the average number for each feature when $k = 1$ and for $k = 20$.

	No. of Samples	**DNA Sudoku**	**2D Pooling**			**S-Stage**			**Halving**			**Generalized Binary Splitting**			**Modified 3-Stage**		
Avg. No. Tests ($k = 1$, $\hat{k} = 1$)	**96**	21	20			13			8.65			7.99			14		
	384	41	40			16.59			10.66			9.99			22.1		
	1,536	81	80			20.61			12.60			12.00			34.68		
Avg. No. Tests ($k = 20$, $\hat{k} = 20$)	**96**	97.77	85.18			80.84			132.05			70.34			80.99		
	384	147.19	152.16			149.63			171.37			112.60			155.84		
	1,536	214.43	250.67			224.82			211.14			152.96			248.78		
Avg. No. Steps ($k = 1$, $\hat{k} = 1$)	**96**	1	1			5			8.65			7.99			3		
	384	1	1			6			10.66			9.99			3		
	1,536	1	1			8			12.60			12.00			3		
Avg. No. Steps ($k = 20$, $\hat{k} = 20$)	**96**	2	2			2			132.05			69.56			2		
	384	2	2			3			171.37			112.54			3		
	1,536	2	2			5			211.14			152.94			3		
Max No. Samples per Pool	**96**	10 ($W_i = 10$)	10 (10x10x1)			3 (2 steps, $\hat{k} = 13 - 20$) 48 (5 steps, $\hat{k} = 1$)			96			2 ($\hat{k} = 20$) 64 ($\hat{k} = 1$)			3 ($\hat{k} = 13 - 20$) 24 (k= 1)		
	384	20 ($W_i = 20$)	20 (20x20x1)			8 (3 steps, $\hat{k} = 20$) 192 (6 steps, $\hat{k} = 1 - 2$)			384			16 ($\hat{k} = 20$) 256 ($\hat{k} = 1$)			8 ($\hat{k} = 17 - 20$) 55 ($\hat{k} = 1$)		
	1,536	39 ($W_i = 40$)	40 (40x40x1)			34 (5 steps, $\hat{k} = 11 - 20$) 768 (8 steps, $\hat{k} = 1$)			1,536			64 ($\hat{k} = 20$) 1024 ($\hat{k} = 1$)			20 ($\hat{k} = 19 - 20$) 140 ($\hat{k} = 1$)		
Change in No. Tests with Overestimate of Positive Samples ($k = 1$, $\hat{k} = 20$)	**96**	+935	+45			+22 (-3 steps)			N/A			+68.77 (+32.19 steps)			+21 (-1 step)		
	384	+1187	+217			+37.41 (-3 steps)			N/A			+109.30 (+73.08 steps)			+31.89		
	1,536	+1455	+945			+35.39 (-3 steps)			N/A			+146.96 (+110.89 steps)			+54.65		
Change in No. Tests with Underestimate of Positive Samples ($k = 20$, $\hat{k} = 1$)	**96**	-857.68 (+1 step)	+11.5 (+1 step)			+14.53 (+3 steps)			N/A			+61.99 (+62.57 steps)			+11.75 (+1 step)		
	384	-1027.82 (+1 step)	-60.79 (+1 step)			+11.34 (+3 steps)			N/A			+58.40 (+58.40 steps)			+54.74		
	1,536	-1212.81 (+1 step)	-691.86 (+1 step)			+7.34 (+3 steps)			N/A			+56.80 (+56.80 steps)			+108.85		
No. of Channels		**1**	**1**	**8**	**16**	**1**	**8**	**16**	**1**	**8**	**16**	**1**	**8**	**16**	**1**	**8**	**16**
Avg. No. Pipettes ($k = 1$, $\hat{k} = 1$)	**96**	192	192	39	20	172	26	17	238.48	32.92	19.07	177.87	24.79	14.63	126	22	18
	384	768	768	111	79	675.66	91.59	50.59	973.63	124.83	65.05	682.72	87.88	46.18	447.96	70.11	43.11
	1,536	3072	3072	384	222	2762.68	352.61	181.61	3790.28	476.88	241.04	2781.30	350.20	177.36	1688.32	232.68	122.68
Avg. No. Pipettes ($k = 20$, $\hat{k} = 20$)	**96**	351.77	229.18	85.18	85.18	144.84	80.84	80.84	2096.29	332.10	214.15	156.24	70.57	70.34	144.99	80.99	80.99
	384	1590.19	840.16	223.16	151.16	575.92	160.5	160.5	8442.96	1141.23	627.38	815.90	145.06	115.07	592.76	155.84	155.84
	1,536	6187.43	3162.67	556.67	399.67	2519.24	476.96	349.81	33899.45	4340.01	2236.15	3382.25	465.78	273.17	1970.68	404.98	324.98

Number of steps

The number of sequential steps is one of the major factors that differentiates pooling methods. The major benefit of non-adaptive pooling methods is that, in some cases, all of the tests can be run at the same time which means that testing can be completed faster. Clearly, the nonadaptive tests required the fewest number of steps even when the results were ambiguous, necessitating a second round of validation 7. For 96 samples, the highest weight that we tested was 6 which meant that any simulation with 5 or more positive samples was ambiguous. Although higher weights could be used to avoid ambiguous results, the number of tests required would have exceeded individual testing. For 384 and 1,536 samples, up to 9 and 20 positive samples, respectively, were unambiguously identified without exceeding individual testing. The ability to unambiguously identify the positive samples in a single step, however, came with a high cost in the number of tests that needed to be performed. For example, the cost of saving one step when using $w = 6$ versus $w = 2$ for 5 positive samples was on average 56.56 tests for 96 samples. This increased to an additional 1,321.57 tests on average for 1,536 samples using $w = 21$ (1 step) versus $w = 4$ (2 steps) when $k = 20$.

Fig 7. Comparison of the number of steps required for each pooling method. The bar plot shows the average number of steps required for each method and the error bars are the standard deviation across 100 simulations (note the different scales for the y-axis). The top row shows simulations with one positive sample and the bottom row shows simulations with 20 positive samples. The columns are different sample sizes from left to right: 96, 384, and 1,596. These results are from the same set of simulations as shown in Fig 6.

For 2D Pooling, ambiguous results occurred more frequently as the number of positive samples increased and were more likely in larger grid arrangements (Fig. 11). Unlike the DNA Sudoku results, which were always ambiguous or unambiguous based on the weight and the number of positives, ambiguous results for 2D Pooling depended on the random arrangement of the positive samples in the grid and therefore were not always consistent for a given window size. For 96 samples, up to 16 positive samples could be identified in a single step but this only occurred in 1% of the simulations. At 5 positive samples (the highest number that could be unambiguously identified using DNA Sudoku), 66% of the simulations required only one step using 3x3x11 grids (66 tests). For 384 samples, up to 20 positive samples were unambiguously identified using 3x3x43 grids (258 tests) in 15% of the simulations. For 1,536 samples, 20 positive samples were unambiguously identified in 60% of the 3x3x171 grid simulations (1,026 tests), and in only 6% of the 6x6x43 grid simulations (516 tests). This shows that reducing the grid size increases the chances of an unambiguous result but, again, it comes with a large increase in the number of tests (1,026 tests for 3x3x171 grid vs. 160 tests for 20x20x4 at 1,536 samples).

For the adaptive methods, the trade off for being more efficient in the number of tests is often an increase in the number of sequential steps. This is most striking in the case of the Generalized Binary Splitting method which performed the best overall in the number of tests but, in some cases, required over 100 steps. Even so, the Binary Splitting by Halving method required even more steps than the General Binary Splitting method. This is partially due to the fact that the number of tests is highly correlated with the number of steps for both of these methods and the General Binary Splitting algorithm does a better job of minimizing the number of tests (the General Binary Splitting method also switches to individual testing in some cases and, because individual tests can be completed in parallel, this can also reduce the total number of steps). The number of steps required for the S-Stage approach were much more moderate compared to the binary splitting algorithms; however, they did get as high as 8 sequential steps. Our Modified 3-stage approach performed the best among the adaptive methods because it enforced a maximum of 3 steps.

Number of samples per pool

The number of samples that are combined in a single pool is a very important practical concern because it can determine whether the assay can produce accurate results. Typically, assays can fail to identify positive samples if the positive signal is diluted beyond the limit of detection. This means that pooling approaches that limit the number of samples per pool are more likely to perform better in practice. Fig 8 compares the maximum number of samples per pool for each pooling method.

Fig 8. Comparison of the maximum number of samples per pool for each pooling method. The bar graphs plot the maximum number of samples in a single pool for each method (note the different scales for the y-axis). The top row shows simulations with one positive sample and the bottom row shows simulations with 20 positive samples. The columns are different sample sizes from left to right: 96, 384, and 1,596. These results are from the same set of simulations as shown in Fig 6.

For the DNA Sudoku simulations, the number of samples per pool was determined by the pooling interval. Because the size of the pooling interval determined the number of pools that the samples would be divided into, a smaller interval resulted in more samples per pool. For 96 samples, the smallest interval was 10 which resulted in up to 10 samples per pool, for 394 samples, the smallest interval was 20 with 20 samples per pool, and for 1,536 samples the smallest interval was 40 with 39 samples per pool (Table 1). For 2D Pooling, the number of samples per pool was equal to the number of samples in each column or row for a given grid layout. The largest grid layouts had the largest number of samples per pool: 10 for 96 samples, 20 for 384 samples, and 40 for 1,536 samples. The number of samples per pool was very consistent for both DNA Sudoku and 2D pooling. For the adaptive approaches, the number of samples per pool varied at each step and tended to be the largest for the first step. For the S-Stage, the Modified 3-stage, General Binary Splitting approaches, the maximum number of samples per pool was lowest when the number of expected positive samples was high and increased as the number of expected positive samples decreased (Table 1). The Modified 3-Stage approach always had the same or fewer samples per pool compared to the S-stage approach. This was particularly true when the number of positive samples was low, resulting in a reduction of 24, 137, and 628 samples per pool for N = 96, 384, 1,536, respectively. The Binary Splitting by Halving method required the largest number of samples per pool at 96, 384, and 1536, due to the need to pool and test all of the samples together as the first step.

Simplicity of pooling method

The simplicity of a pooling method can be somewhat subjective. However, one of the major points of failure when combining pools by hand, is mistakes in pipetting the wrong samples. Therefore, we used the number of individual pipetting actions required for each method using 1-, 8-, and 16-channel pipettes as an indicator for the simplicity and reproducibility of each method. Fig 9 compares the number of pipettings required for each method with either 1 or 20 positive samples for each of the sample sizes.

Fig 9. Comparison of the average number of pipettings for each pooling method. The number of pipettings required for each pooling method is an indicator of method simplicity and reproducibility. The bar charts indicate the average number of pipettings required to create pools for each method using 1-, 8-, and 16-channel pipettes (columns). The rows are the results from simulations using N = 96, 384, and 1,536 samples and $k = 1$ or 20 positive samples (note the different scale on the y-axes). The error bars represent the standard deviation of 100 simulations. The DNA Sudoku method does not benefit from the use of multichannel pipettes so the number of pipettings is the same across each row. These results are from the same set of simulations as shown in Fig 6.

In most methods, using a multichannel pipette reduced the number of pipettings by an order of magnitude in some cases. Compared to the 8-channel pipette, the 16-channel pipette reduced the number of pipettings only for schemes where the size of the pools were large. Our Modified 3-Stage method required the fewest pipettings with a single-channel pipette, compared to the other methods; however, the S-Stage method performed similarly well in cases where the number of steps happened to be similar (i.e.

in Fig 9, both methods required the same number of steps when $k = 20$ for $N = 96$ and 384). This generally translated into the best performance when using multichannel pipettes; although, the General Binary Splitting simulations performed slightly better in cases where the size of the pools were large, making the multichannel pipettes more efficient.

Binary Splitting by Halving is the least efficient method in number of pipettings, likely because the method requires many samples to be pooled at each step for many steps. The performance is slightly improved when multichannel pipettes are used but it is still the least efficient in many cases. Using a single pipette, DNA Sudoku was not the most inefficient compared to the other methods. However, because the samples that are combined in each pool are spaced out in different intervals instead of in consecutive groups, the number of pipettings did not improve by using multichannel pipettes. This means that, in the best case, a laboratory technician would need to correctly pipette ~ 200 ($N = 96$) to $\sim 6,000$ ($N = 1,536$) times to combine the samples into pools.

Sensitivity to misestimation of positive rate

Most of the methods described here required an estimate of the positive rate in order to design the pooling scheme. In some cases, over or underestimating the number of positives can have large impacts on the number of tests and/or the number of steps required to complete the assay. Methods that minimize the impact of misestimation are more robust to fluctuating rates in the sample population which is common in outbreak scenarios.

Binary Splitting by Halving was, by default, the most robust of the approaches because the protocol was not modified based on any estimate of the number of positive samples (Fig 10, second from right). Although the number of tests increased when there were more positive samples, assuming a fixed sample size, knowledge of the number of positive samples did not have any impact on the performance of this approach. In contrast, the size of the initial pool for the Generalized Binary Splitting method depended on $\hat{k}$ and, among the adaptive approaches, misestimation of the true value resulted in the largest impact on the number of tests and steps (Fig 10, right). The consequences of extreme overestimation ($k = 1$ and $\hat{k} = 20$) and underestimation ($k = 20$ and $\hat{k} = 1$) are provided in Table 1 which shows that method is more sensitive to overestimation than to underestimation. Of the methods that depended on an estimate of the number of positive samples, the S-Stage (Fig 10, left) and our Modified 3-Stage approach (Fig 10, second from left) were the most robust to misestimations of k. The number of steps was more robust in the Modified 3-Stage approach than the S-Stage due to the 3-Step constraint; however, the Modified 3-Stage was more sensitive in the number of tests in some cases (Table 1).

Fig 10. Changes in the number of tests and steps given different estimated positive rates for the adaptive methods. The figure shows the number of tests (y-axis) and the number of steps (marker size) required to recover all positive samples (x-axis) in simulations with $N = 384$ samples using each of the adaptive methods. For each method, except for Binary Splitting by Halving, the pooling scheme was optimized around the expected number of positive samples (marker color) provided to each simulation. Each point represents a single simulation and the lines are the average number of tests for a given number of expected positives. The black dashed line in the S-Stage, 3-Stage, and Binary Splitting by Halving figures represents the upper bound of the number of tests (assuming that the number of positive samples is estimated correctly, where applicable). For the Generalized Binary Splitting figure, the number of tests approaches the lower bound (black dashed line) when $\frac{N}{k}$ is large.

DNA Sudoku (Fig 11, left) was the most sensitive method overall. Overestimating of the number of positive samples caused the weight of the pooling design ($w = \hat{k} + 1$) to be set higher than it needed to be. When this happened, all of the positive samples were still unambiguously identified but each unnecessary increase in the weight required more than $\sqrt{N}$ additional tests. When the number of positive samples was underestimated, fewer tests were performed but the pooling scheme was no longer able to unambiguously identify the positive samples in a single step and a second round of verification was required. A similar pattern occurred in the 2D Pooling simulations (Fig 11, right). While the grid dimensions did not directly depend on k, generally larger grids were more efficient when the number of positive samples was low and smaller grids reduced ambiguous results when the number of positive samples was high but at the cost of many more tests. However, because 2D Pooling was constrained to two dimensions, the number of tests did not vary as drastically as DNA Sudoku.

Fig 11. Changes in the number of tests and steps given different estimated positive rates for the non-adaptive methods. The figure shows the number of tests (y-axis) and the number of steps (marker size) required to recover all positive samples (x-axis) in simulations with $N = 384$ samples using DNA Sudoku and 2D Pooling methods. Each point is the average number of tests required for 100 simulations and the width of the bands is the standard deviation. The simulations were run using different weights for DNA Sudoku and different symmetrical 2D grid sizes for 2D Pooling. Small markers indicate unambiguous results that required only a single round of testing and the larger markers indicate ambiguous results that required a second validation step to correctly identify the positive samples. The grey dashed line is the number of tests required for individual testing.

Experimental validation of modified 3-Stage approach

To experimentally validate our modified 3-Stage approach, we set up a controlled experiment with *C. burnetti* DNA spiked into complex microbiome background samples. All of the 24 background samples used in the validation experiment were negative for the *C. burnetti* pathogen prior to the spike-in and each background extraction was found to have similar amounts of the 16S gene (CT values of 29 to 31), indicating similar background bacterial loads. *C. burnetti* was detected in the spike-in samples prior to pooling. The random placement of the *C. burnetti* positive samples within the eight 96-well plates is shown in Table 2. Although the expected number of positive samples per plate was ~ 2 given the 1.3% carriage rate, the actual number of positives ranged from 0 to 3 and none of the plates had exactly 2 positive samples (Table 2). The TaqMan assay was able to accurately identify the positive pools without any false positives or false negatives even during the first step when the number of samples per pool was the largest at 16. Using an 8-channel pipette where appropriate, a total of 180 pipettings was required to pool the samples. A total of 120 TaqMan assays were performed which is $\sim 84\%$ fewer than would be required to individually test 768 samples.

Conclusions

Picking the right pooling approach for a given pathogen surveillance campaign can be a complicated decision, which is often driven by a set of conflicting constraints and priorities, including budgetary limitations, complexity of the procedure (and thus likelihood of human error), and time-to-answer requirements. As is evident from the data presented in this manuscript, no single group theory approach is a clear winner under all these possible constraints – the correct choice depends on the predominant constraints placed on the surveillance campaign. Below we present some of the practical implications of the various group theory approaches outlined in this manuscript.

When minimizing the total number of tests is the absolute overriding goal and time-to-answer is not an important constraint, Generalized Binary Splitting is the optimal choice. This approach minimized the total number of tests while maintaining a

Table 2. Using the modified 3-Stage approach we were able to accurately recover all of the positive *C. burnetii* samples. Eight 96-well plates were filled with background DNA from complex cow milk microbiome samples and 10 randomly chosen samples had *C. burnetii* DNA spiked in. The table shows the number and location of the positive samples in each 96-well plate and the number of tests and pipettings required to identify the positive samples using our Modified 3-Stage pooling approach.

Plate #	Positive Samples (k)	Exp. Positive Samples ($\hat{k}$)	Samples (N)	Positive Wells	Tests	Pipettings
1	3	2	96	F1, A4, G9	30	48
2	1	2	96	H2	14	20
3	1	2	96	D9	14	20
4	1	2	96	E11	14	20
5	0	2	96	NA	6	12
6	3	2	96	F3, A10, C10	22	28
7	0	2	96	NA	6	12
8	1	2	96	D4	14	20

reasonable complexity, as measured by the number of distinct pipetting actions, but sacrifices speed due to significantly increase in the number of serial steps. On the other hand, when speed is the predominant constraint, DNA Sudoku can offer a single-step pooling approach with the minimum number of tests, at the cost of significant complexity. DNA Sudoku, however, is far from optimal for monitoring rapidly changing pandemics due to its extreme sensitivity to misestimation of the carriage rate of the pathogen in population.

A good middle ground between the adaptive and non-adaptive pooling approaches is the Modified 3-Stage approach – our preference in our own surveillance applications. While it is never the absolute best in any one category, it is always nearly optimal in terms of number of serial steps (2nd best), complexity (2nd best), number of tests (4th best), and extremely resilient to misestimation of the carriage rate (2nd best). The latter is particularly important, as it allows this approach to be useful for surveillance in situations with rapidly changing pathogen carriage rates (e.g. in pandemic or seasonal outbreaks), while keeping number of serial steps as low as possible for an adaptive method.

References

1. Abdurrahman ST, Mbanaso O, Lawson L, Oladimeji O, Blakiston M, Obasanya J, et al. Testing Pooled Sputum with Xpert MTB/RIF for Diagnosis of Pulmonary Tuberculosis To Increase Affordability in Low-Income Countries. Journal of clinical microbiology. 2015;53(8):2502–2508. doi:10.1128/JCM.00864-15.

2. Ray KJ, Zhou Z, Cevallos V, Chin S, Enanoria W, Lui F, et al. Estimating Community Prevalence of Ocular Chlamydia trachomatis Infection using Pooled Polymerase Chain Reaction Testing. Ophthalmic Epidemiology. 2014;21(2):86–91. doi:10.3109/09286586.2014.884600.

3. Stallknecht DE. Impediments to wildlife disease surveillance, research, and diagnostics. Curr Top Microbiol Immunol. 2007;315:445–461. doi:10.1007/978-3-540-70962-6_17.

4. Evaluating and testing persons for coronavirus disease 2019 (COVID-19). Centers for Disease Control and Prevention; 2020. Available from: `https://www.cdc.gov/coronavirus/2019-ncov/hcp/clinical-criteria.html`.

5. Dorfman R. The Detection of Defective Members of Large Populations. Annals of Mathematical Statistics. 1943;14(4):436–440. doi:10.1214/aoms/1177731363.

6. Du Dz, Kwang-ming Hwang F. Combinatorial group testing and its applications. 2nd ed. World Scientific; 1993.

7. Ramírez AL, van den Hurk AF, Meyer DB, Ritchie SA. Searching for the proverbial needle in a haystack: advances in mosquito-borne arbovirus surveillance. Parasites & Vectors. 2018;11(1):320. doi:10.1186/s13071-018-2901-x.

8. Hepp CM, Cocking JH, Valentine M, Young SJ, Damian D, Samuels-Crow KE, et al. Phylogenetic analysis of West Nile Virus in Maricopa County, Arizona: Evidence for dynamic behavior of strains in two major lineages in the American Southwest. PLOS ONE. 2018;13(11):1–12. doi:10.1371/journal.pone.0205801.

9. Sutherland GL, Nasci RS. Detection of West Nile Virus in Large Pools of Mosquitoes. Journal of the American Mosquito Control Association. 2007;23(4):389 – 395. doi:10.2987/5630.1.

10. West Nile Virus in the United States: Guidelines for Surveillance, Prevention, and Control. Centers for Disease Control and Prevention; 2013. Available from: `https://www.cdc.gov/westnile/resources/pdfs/wnvGuidelines.pdf`.

11. Pannwitz G, Wolf C, Harder T. Active surveillance for avian influenza virus infection in wild birds by analysis of avian fecal samples from the environment. J Wildl Dis. 2009;45(2):512–518. doi:10.7589/0090-3558-45.2.512.

12. Muñoz-Zanzi CA, Johnson WO, Thurmond MC, Hietala SK. Pooled-Sample Testing as a Herd-Screening Tool for Detection of Bovine Viral Diarrhea Virus Persistently Infected Cattle. Journal of Veterinary Diagnostic Investigation. 2000;12(3):195–203. doi:10.1177/104063870001200301.

13. Hogan CA, Sahoo MK, Pinsky BA. Sample Pooling as a Strategy to Detect Community Transmission of SARS-CoV-2. JAMA. 2020;323(19):1967–1969. doi:10.1001/jama.2020.5445.

14. Taylor SM, Juliano JJ, Trottman PA, Griffin JB, Landis SH, Kitsa P, et al. High-Throughput Pooling and Real-Time PCR-Based Strategy for Malaria Detection. Journal of Clinical Microbiology. 2010;48(2):512–519. doi:10.1128/JCM.01800-09.

15. Pilcher CD, McPherson JT, Leone PA, Smurzynski M, Owen-O'Dowd J, Peace-Brewer AL, et al. Real-time, Universal Screening for Acute HIV Infection in a Routine HIV Counseling and Testing Population. JAMA. 2002;288(2):216–221. doi:10.1001/jama.288.2.216.

16. Aldridge M, Johnson O, Scarlett J. Group Testing: An Information Theory Perspective. Foundations and Trends® in Communications and Information Theory. 2019;15(3-4):196–392. doi:10.1561/0100000099.

17. Laurin E, Thakur K, Mohr PG, Hick P, Crane MSJ, Gardner IA, et al. To pool or not to pool? Guidelines for pooling samples for use in surveillance testing of infectious diseases in aquatic animals. Journal of Fish Diseases. 2019;42(11):1471–1491. doi:10.1111/jfd.13083.

18. Hu MC, Hwang FK, Wang JK. A Boundary Problem for Group Testing. SIAM Journal on Algebraic Discrete Methods. 1981;2(2):81–87. doi:10.1137/0602011.

19. Riccio L, Colbourn CJ. Sharper Bounds in Adaptive Group Testing. Taiwanese J Math. 2000;4(4):669–673. doi:10.11650/twjm/1500407300.

20. Erlich Y, Chang K, Gordon A, Ronen R, Navon O, Rooks M, et al. DNA Sudoku–harnessing high-throughput sequencing for multiplexed specimen analysis. Genome Res. 2009;19(7):1243–53. doi:10.1101/gr.092957.109.

21. Chi X, Zhang Y, Xue Z, Feng L, Liu H, Wang F, et al. Discovery of rare mutations in extensively pooled DNA samples using multiple target enrichment. Plant Biotechnol J. 2014;12(6):709–17. doi:10.1111/pbi.12174.

22. Cao HX, Schmidt R. Screening of a Brassica napus bacterial artificial chromosome library using highly parallel single nucleotide polymorphism assays. BMC Genomics. 2013;14:603. doi:10.1186/1471-2164-14-603.

23. Zuzarte PC, Denroche RE, Fehringer G, Katzov-Eckert H, Hung RJ, McPherson JD. A two-dimensional pooling strategy for rare variant detection on next-generation sequencing platforms. PLoS One. 2014;9(4):e93455. doi:10.1371/journal.pone.0093455.

24. Li CH. A Sequential Method for Screening Experimental Variables. Journal of the American Statistical Association. 1962;57(298):455–477. doi:10.1080/01621459.1962.10480672.

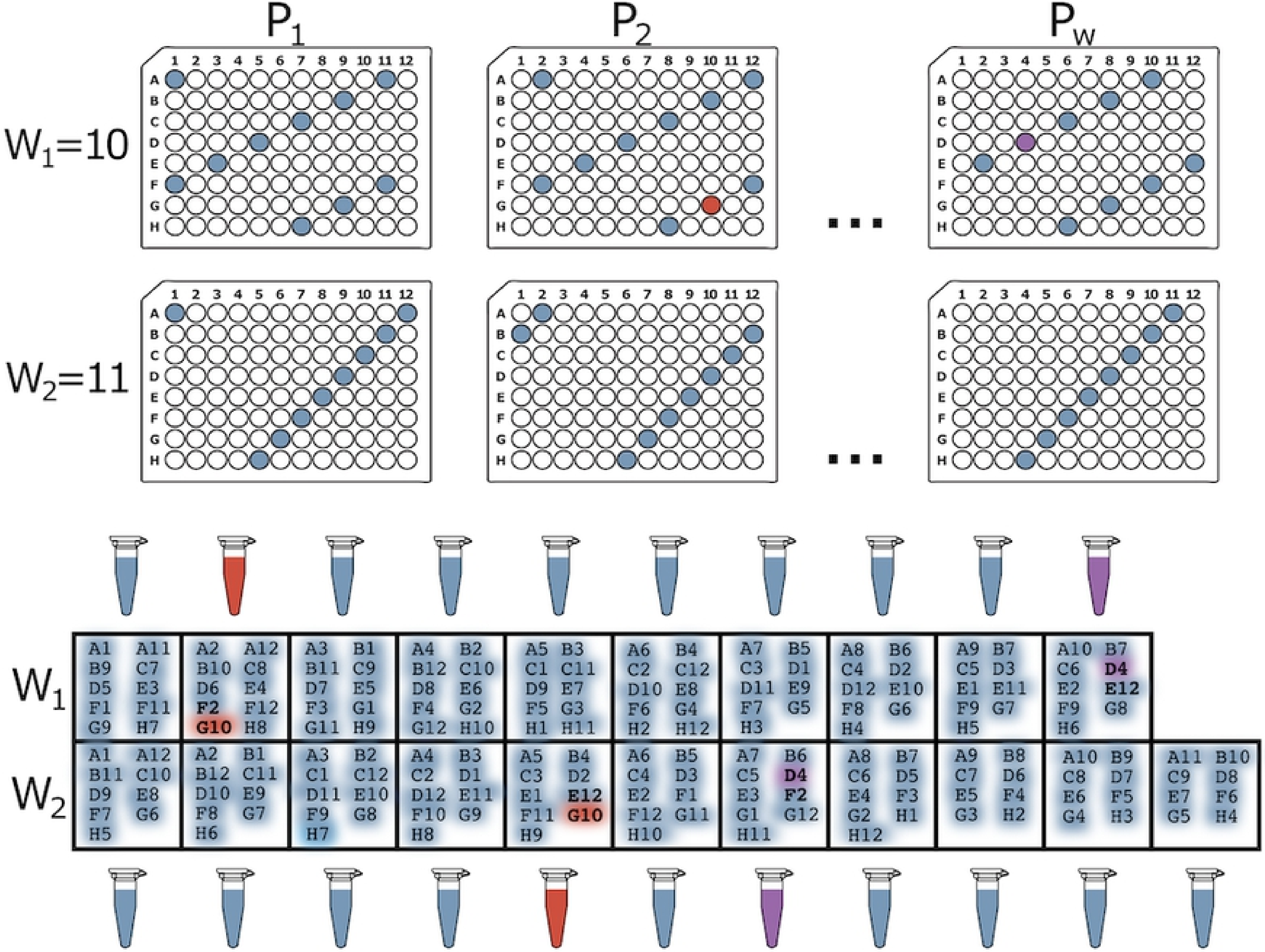
25. Sobel M, Groll PA. Group testing to eliminate efficiently all defectives in a binomial sample. The Bell System Technical Journal. 1959;38(5):1179–1252.

26. Sobel M, Groll PA. Binomial Group-Testing with an Unknown Proportion of Defectives. Technometrics. 1966;8(4):631–656.

27. Hwang FK. A Method for Detecting all Defective Members in a Population by Group Testing. Journal of the American Statistical Association. 1972;67(339):605–608. doi:10.1080/01621459.1972.10481257.

28. Loftis AD, Reeves WK, Szumlas DE, Abbassy MM, Helmy IM, Moriarity JR, et al. Rickettsial agents in Egyptian ticks collected from domestic animals. Exp Appl Acarol. 2006;40(1):67–81. doi:10.1007/s10493-006-9025-2.

29. Liu CM, Aziz M, Kachur S, Hsueh PR, Huang YT, Keim P, et al. BactQuant: an enhanced broad-coverage bacterial quantitative real-time PCR assay. BMC Microbiol. 2012;12:56. doi:10.1186/1471-2180-12-56.



W_1	A1 A11 B9 C7 D5 E3 F1 F11 G9 H7	A2 A12 B10 C8 D6 E4 **F2** F12 **G10** H8	A3 B1 B11 C9 D7 E5 F3 G1 G11 H9	A4 B2 B12 C10 D8 E6 F4 G2 G12 H10	A5 B3 C1 C11 D9 E7 F5 G3 H1 H11	A6 B4 C2 C12 D10 E8 F6 G4 H2 H12	A7 B5 C3 D1 D11 E9 F7 G5 H3	A8 B6 C4 D2 D12 E10 F8 G6 H4	A9 B7 C5 D3 E1 E11 F9 G7 H5	A10 B7 C6 **D4** E2 **E12** F9 G8 H6	
W_2	A1 A12 B11 C10 D9 E8 F7 G6 H5	A2 B1 B12 C11 D10 E9 F8 G7 H6	A3 B2 C1 C12 D11 E10 F9 G8 H7	A4 B3 C2 D1 D12 E11 F10 G9 H8	A5 B4 C3 D2 E1 **E12** F11 **G10** H9	A6 B5 C4 D3 E2 F1 F12 G11 H10	A7 B6 C5 **D4** E3 **F2** G1 G12 H11	A8 B7 C6 D5 E4 F3 G2 H1 H12	A9 B8 C7 D6 E5 F4 G3 H2	A10 B9 C8 D7 E6 F5 G4 H3	A11 B10 C9 D8 E7 F6 G5 H4

Fig 1

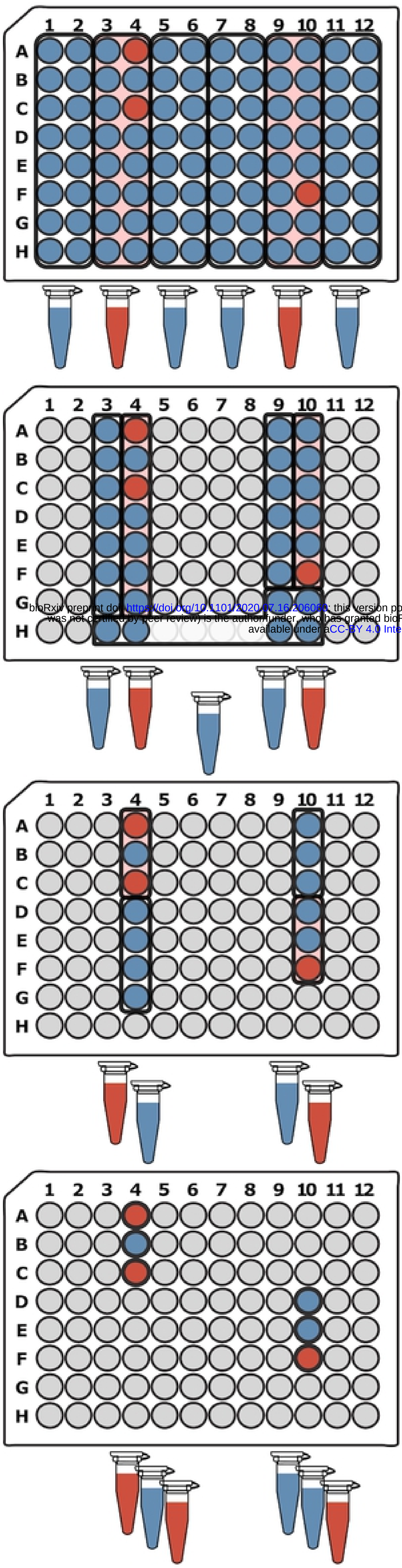


Fig 3

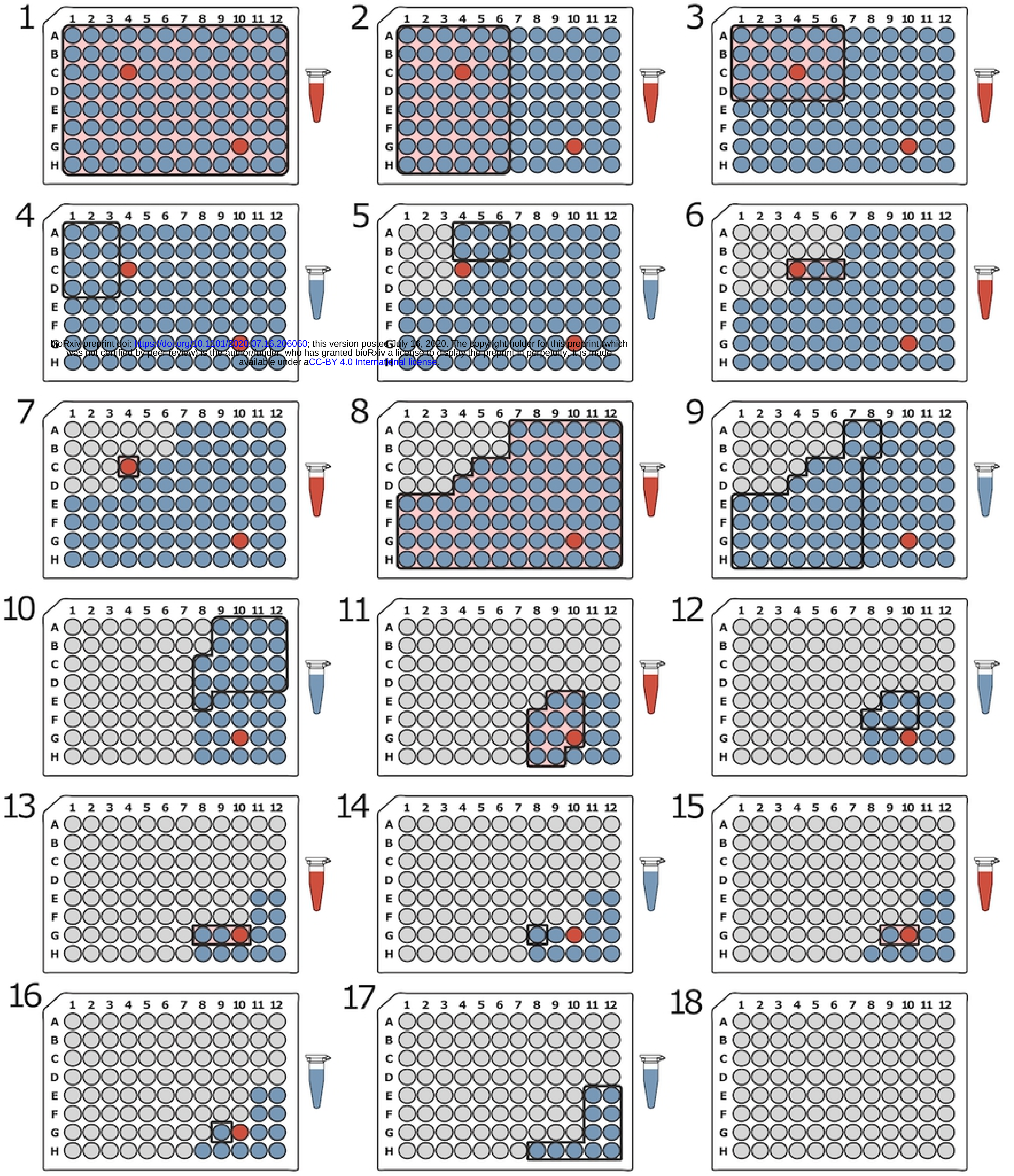

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
1 2 3 4 5 6 7 8 9 10 11 12
A
B
C
D
E
F
G
H

Fig 4

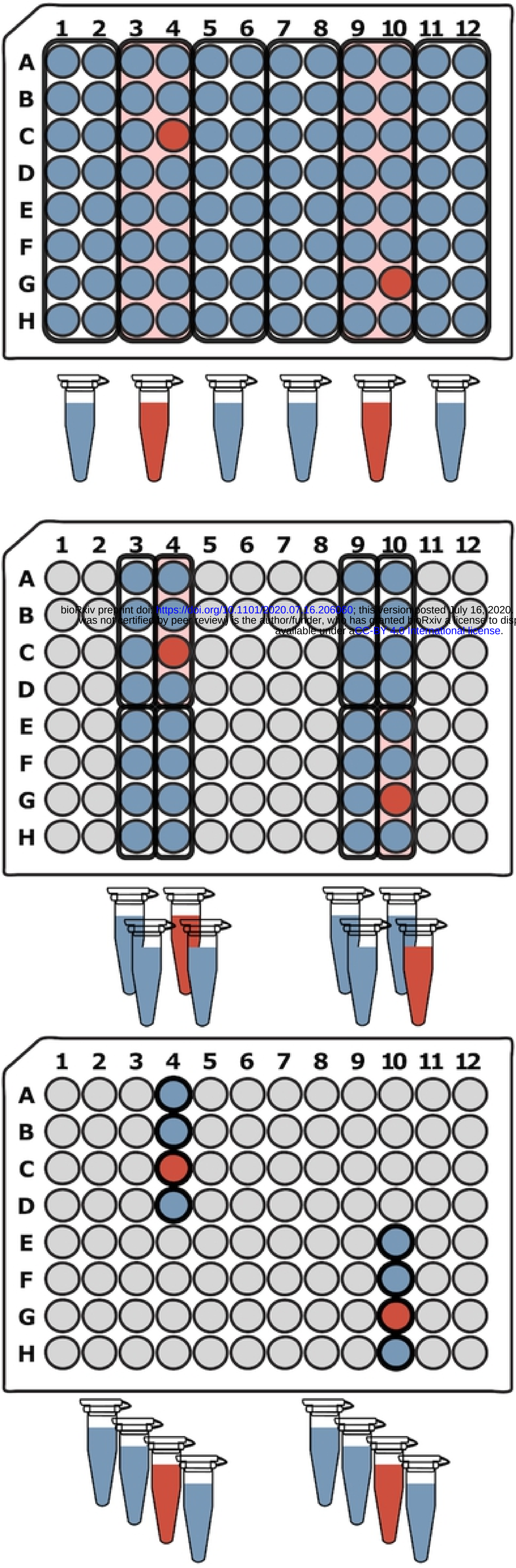


Fig 5

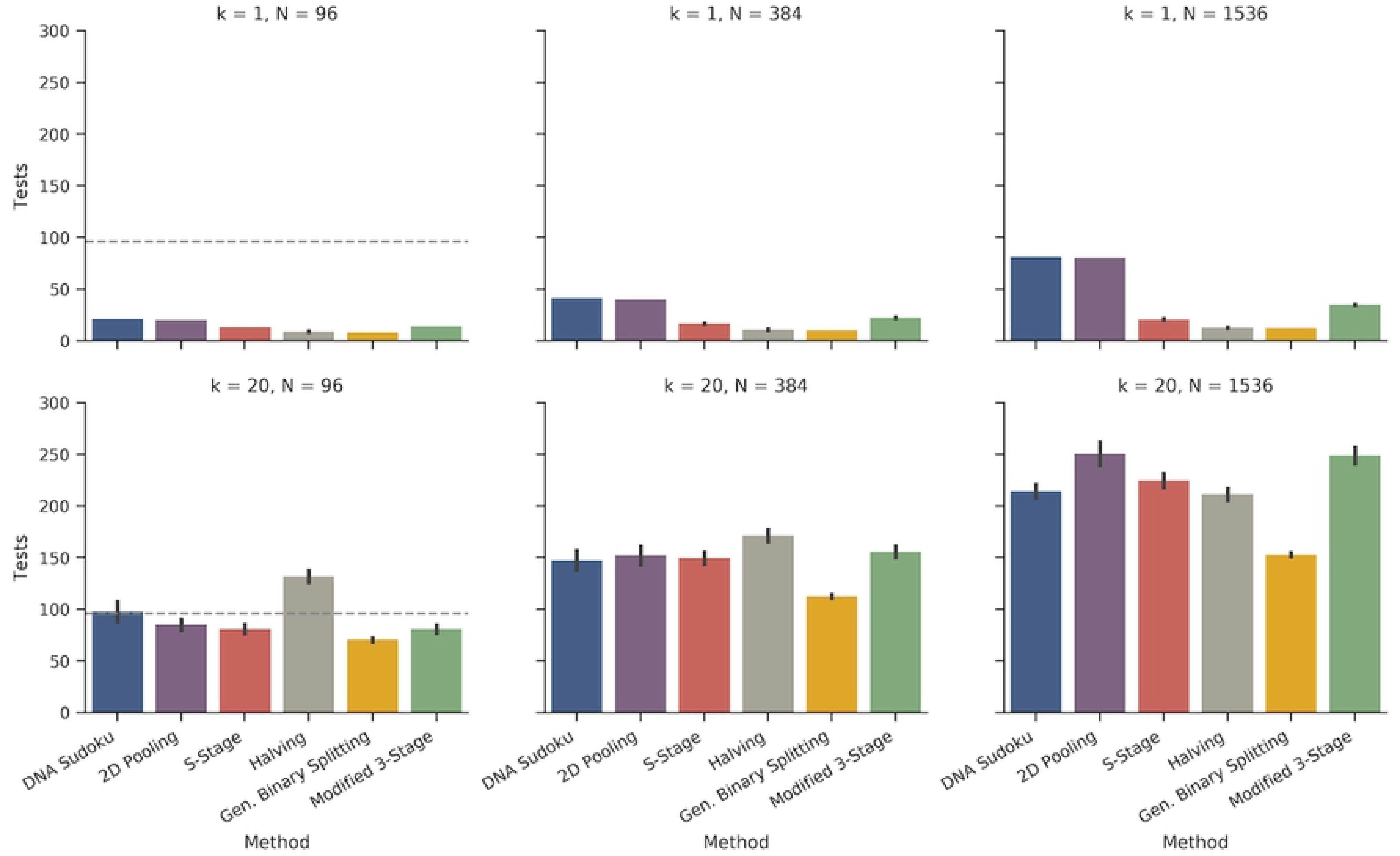

k = 1, N = 96
k = 1, N = 384
k = 1, N = 1536
k = 20, N = 96
k = 20, N = 384
k = 20, N = 1536
Tests
300
250
200
150
100
50
0
DNA Sudoku
2D Pooling
S-Stage
Halving
Gen. Binary Splitting
Modified 3-Stage
Method

Fig 6

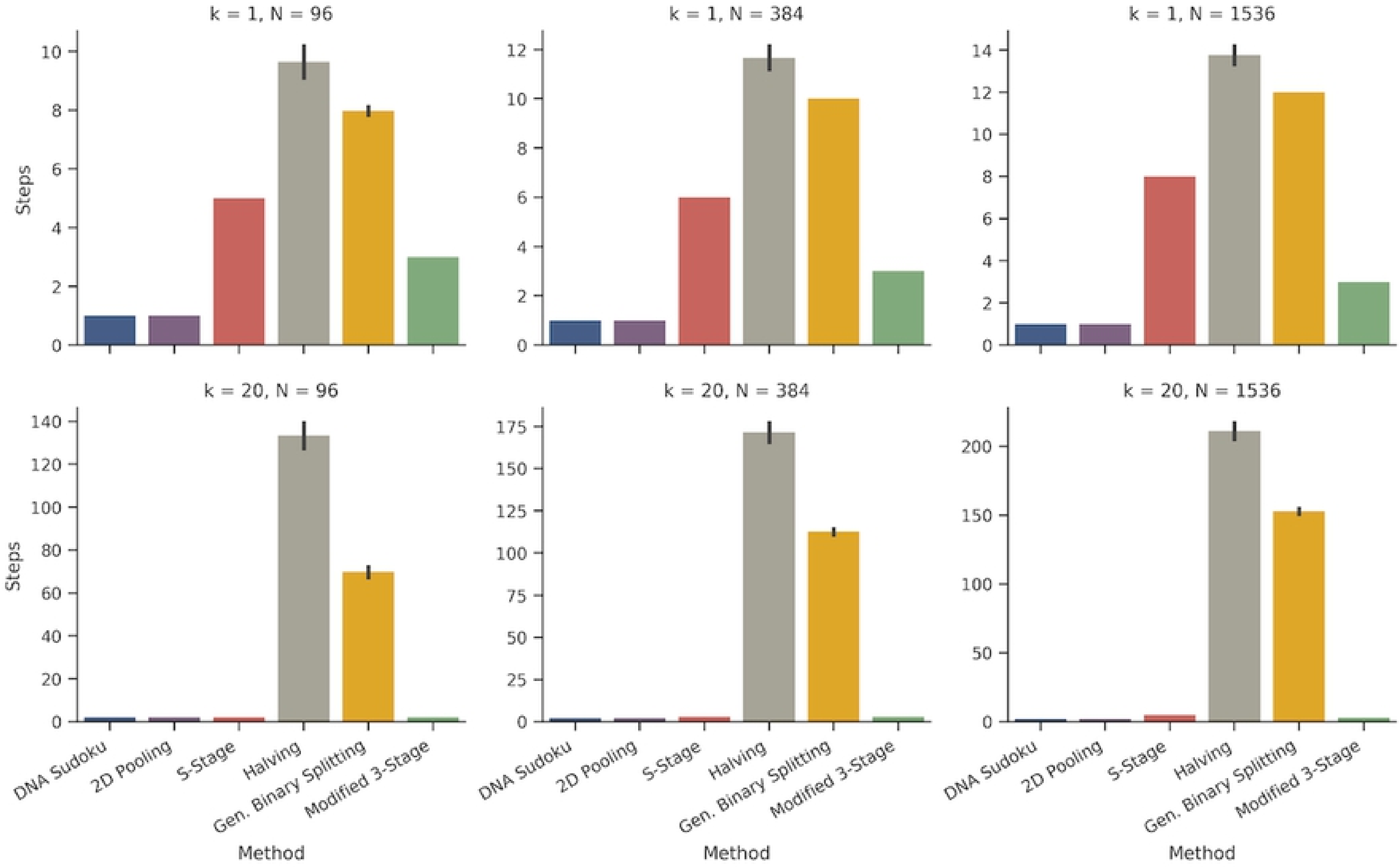

k = 1, N = 96
k = 1, N = 384
k = 1, N = 1536
k = 20, N = 96
k = 20, N = 384
k = 20, N = 1536
Steps
Method
DNA Sudoku
2D Pooling
S-Stage
Halving
Gen. Binary Splitting
Modified 3-Stage

Fig 7

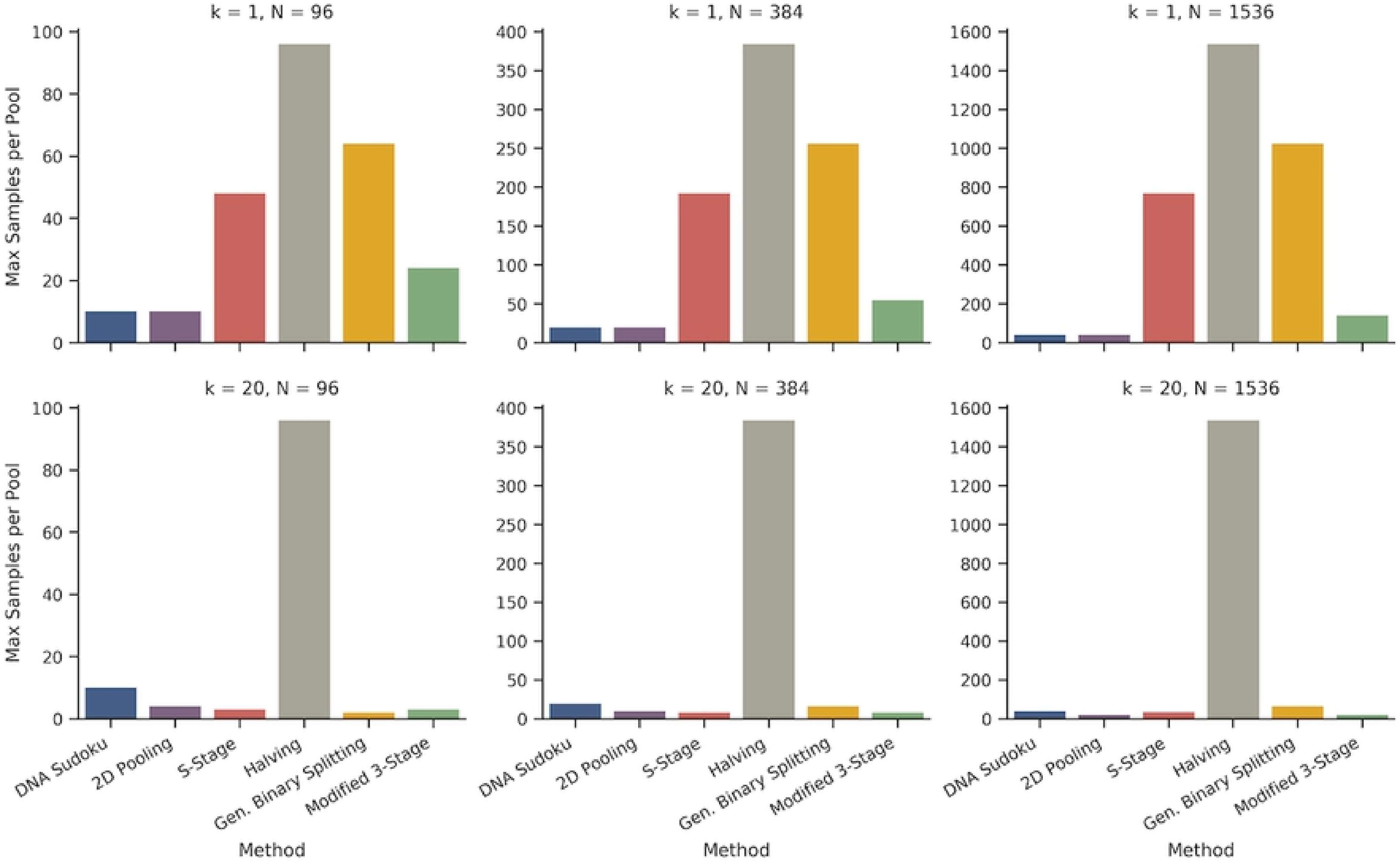

k = 1, N = 96
k = 1, N = 384
k = 1, N = 1536
k = 20, N = 96
k = 20, N = 384
k = 20, N = 1536
Max Samples per Pool
Method
DNA Sudoku
2D Pooling
S-Stage
Halving
Gen. Binary Splitting
Modified 3-Stage

Fig 8

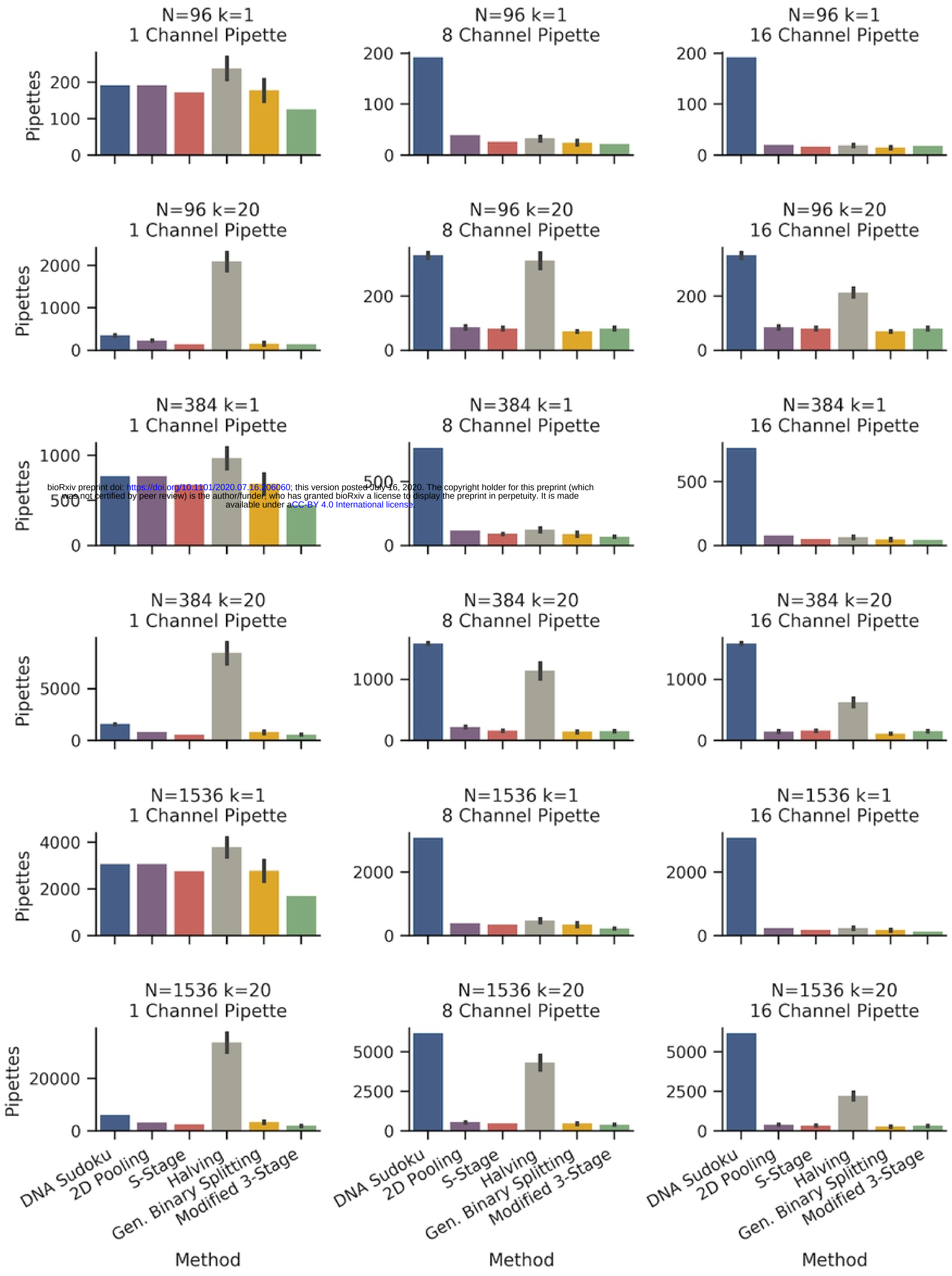

N=96 k=1
1 Channel Pipette
8 Channel Pipette
16 Channel Pipette
N=96 k=20
N=384 k=1
N=384 k=20
N=1536 k=1
N=1536 k=20
Pipettes
DNA Sudoku
2D Pooling
S-Stage
Halving
Gen. Binary Splitting
Modified 3-Stage
Method

Fig 9

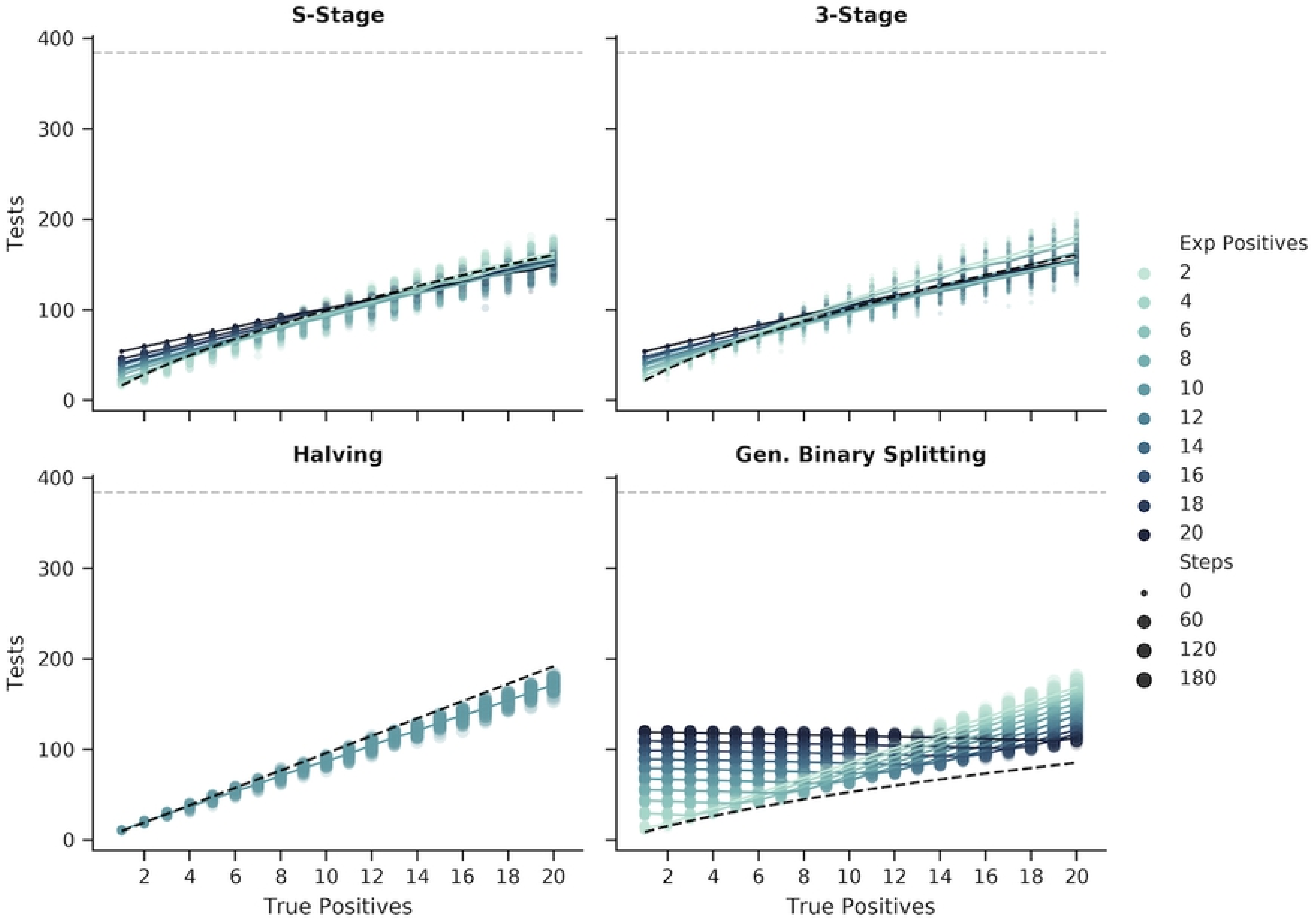

S-Stage
3-Stage
Halving
Gen. Binary Splitting
Tests
True Positives
0
100
200
300
400
2
4
6
8
10
12
14
16
18
20
Exp Positives
2
4
6
8
10
12
14
16
18
20
Steps
0
60
120
180

Fig 10

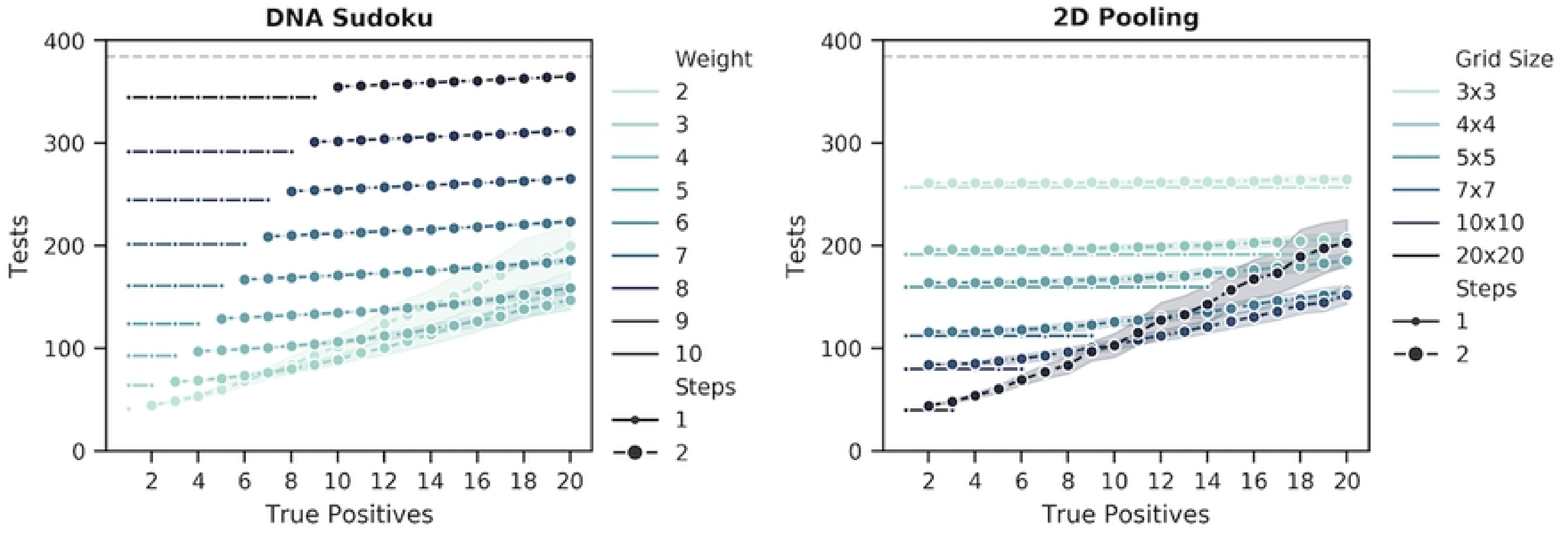

DNA Sudoku
2D Pooling
Tests
True Positives
Weight
2
3
4
5
6
7
8
9
10
Steps
1
2
Grid Size
3x3
4x4
5x5
7x7
10x10
20x20
0
100
200
300
400
2
4
6
8
10
12
14
16
18
20

Fig 11

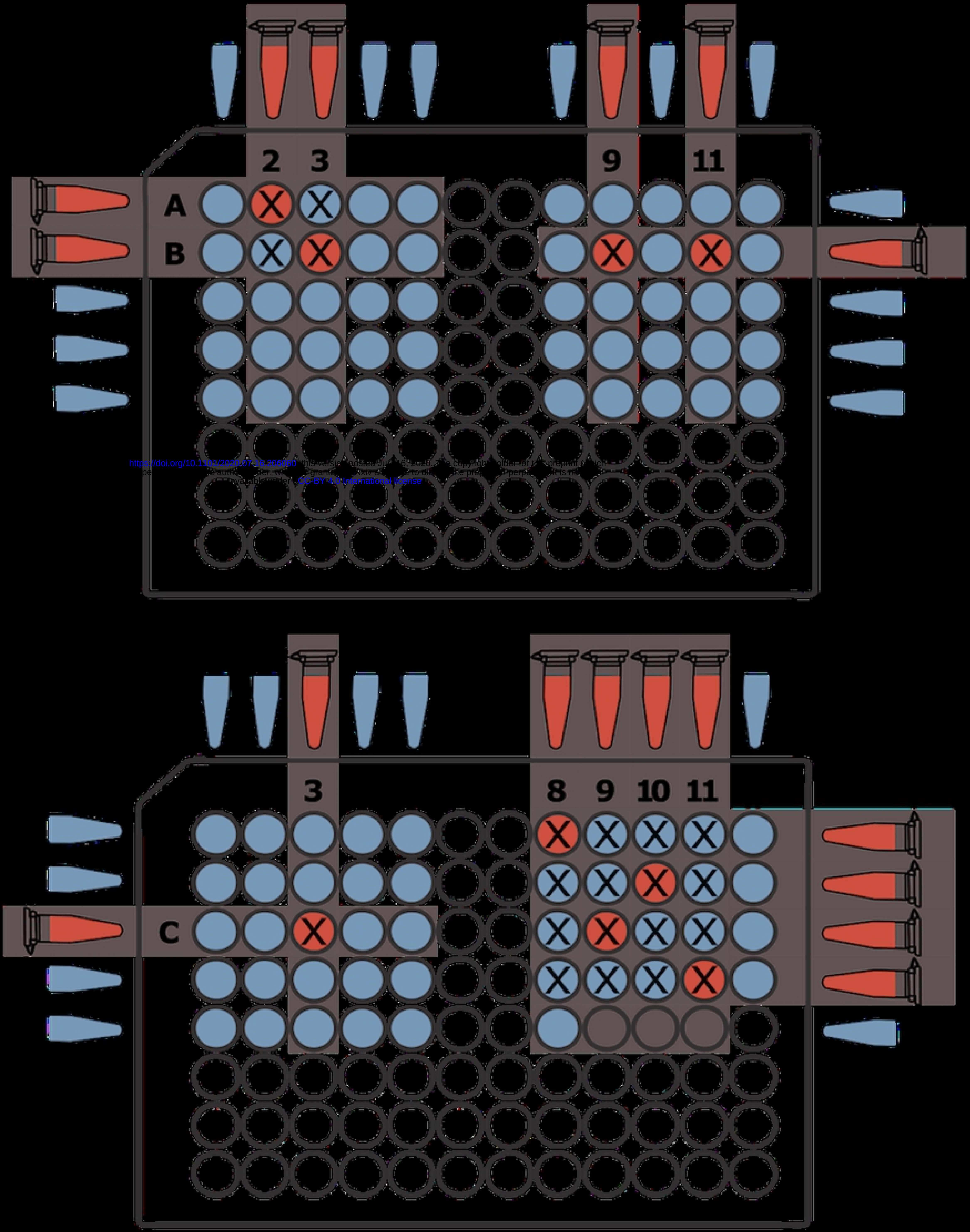


Fig 2