[Long version for academic venues]

**Lots of Group Testing and a Little Computer Science Can Improve COVID-19 Screening**
by Alan G. Labouseur, Ph.D.

In his New York Times column of May 7, 2020 [1], Jordan Ellenberg wrote about a World War II era technique called group (or “pooled”) testing. It can be used to reduce the tests needed to screen a large population by simultaneously testing multiple samples. Back then there was a different sort of social virus causing trouble: syphilis. US Army economists Robert Dorfman and David Rosenblatt created a group testing method for detecting (and rejecting) syphilitic draft candidates [2]. In today’s environment of scarce COVID-19 tests, this technique has renewed importance.

Assuming we have an unbiased and uniform population to test, and assuming the test is sufficiently accurate, sensitive, and specific to signal the presence of COVID-19, combining biomarkers (samples) from multiple patients into single test can reduce the overall number of tests we need. It’s based on a core principle of computer science: divide and conquer. Here’s how it works: Randomly divide the population into small groups and test (or “conquer” in this metaphor) each group for infection. If the test comes out negative then nobody in the group is infected. If the test comes out positive then one or more members of the group are infected, but we don’t know which one(s) so we have to test some more by testing everybody in the group or subdividing into smaller groups and repeating. Here’s a recipe for the test plan:



```
Start with groups of size g ∈ {4, 8, 16, 32, 64}

test g elements
if infection found
   divide into two equal size groups and test each group
   if one group shows infection and the other does not
      test all members of the infected group and clear the others // done with 1 + 2 + g/2 tests (the rest of the cases)
   else // both groups show infection
      test all members of both groups // done with 1 + 2 + g tests (worst case)
   end if
else
   // done with 1 test (best case)
end if
```

We can form a mental picture of the divide and conquer approach by thinking about choose-your-own-adventure books (or more recently, *Black Mirror: Bandersnatch* on Netflix) where each decision lets you to branch off in one of many directions. Were we to illustrate all of the possibilities starting with the first, we’d draw something that looks (a little) like an upside-down tree, starting from the beginning and branching off over and over. That gets quite complex, but we can simplify things by limiting each decision to two choices. That’s called a binary tree, and that’s all we need. Figure 1 shows an example. For binary trees, as with many aspects of computer science, powers of 2 (that is: 2, 4, 8, 16, 32, 64, etc.) are important.

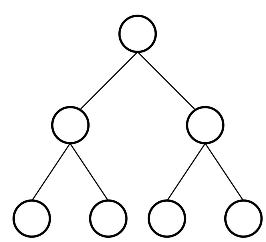


Figure 1 - Binary Tree

There are three possibilities to consider when testing a group of samples:
(1) there are no infected samples
(2) there is exactly one infected sample
(3) there are two or more infected samples

Given these three cases, applying our binary divide and conquer approach, and following the recipe above, we could test groups of 8 (for example) and then test subgroups of 4 if any infection is found. The best case scenario is that we determine all 8 are infection-free with 1 test. See *Case (1)* in Figure 2. A second case occurs when we find 1 infection with 7 tests: 1 for the group of eight, 2 for subgroups of four, and 4 individual tests for the (single) infected subgroup. See *Case (2)* in Figure 2. The worst case scenario is when there are two or more infected samples in the group and we divide them into subgroups of 4 and end up using 11 tests: 1 for the group of eight, 2 for the groups of four, and 8 individual tests. See *Case (3)* in Figure 2. Thankfully, this worst case scenario is rare when the infection rate is low, but that changes with the infection rate.

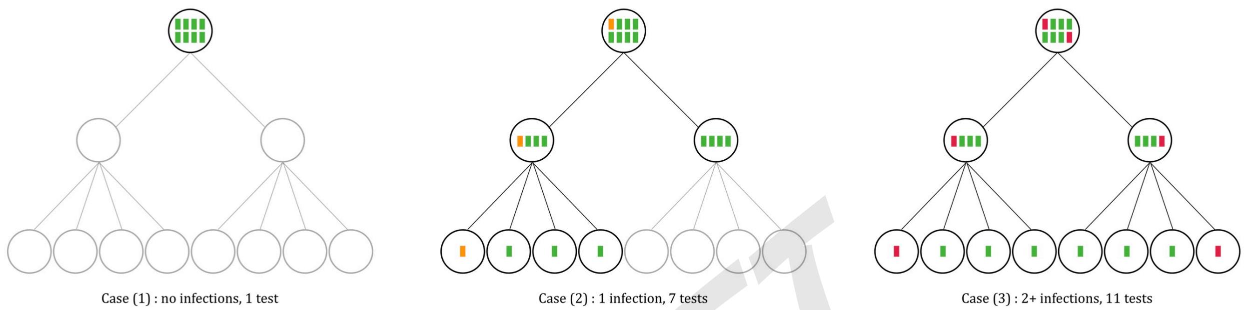



Figure 2 - Testing Possibilities - Each black circle is one test.



How do we know this works? We can determine the likelihood of each testing possibility based on the number of samples we pool into each group and the infection rate of the disease. For example, when testing groups of 8 for a disease with an infection rate of 2% in the manner noted above . . .

Case (1) is expected to happen 85% of the time. This is because a 2% infection rate means that, on average, 98% of the population is uninfected. The likelihood of randomly choosing 8 uninfected samples is 0.98 × 0.98 × 0.98 × 0.98 × 0.98 × 0.98 × 0.98 × 0.98 (put another way, $0.98^8$), which is 0.85 or 85%. When this occurs only one test is needed.

Case (3) happens slightly less than 0.04% of the time because the likelihood of randomly choosing two infected samples is 0.02 × 0.02, or 0.0004 or 0.04%. (It's actually less, but it's safe to err on the side of an upper bound value. Also, the likelihood of randomly choosing more than two infected samples is even lower, so again we are safe with this upper bound.) In this case, 11 tests are needed.

Case (2), the only other possibility, happens the rest of the time, which is 14.96%, and 7 tests are needed.

So, for 1000 people where 20 of them (2%) are infected and 980 are infection-free, we could make 125 groups of 8 samples each and work out what we expect based on the percentages:

```
Case (1): 125 × 0.8500 = 106.25 instances requiring 107 tests (since there are no partial tests)
Case (2): 125 × 0.1496 =  18.70 instances requiring 131 tests
Case (3): 125 × 0.0004 =   0.05 round up to 1 instance requiring 11 tests
```

That's 249 tests to screen a population of 1000 people for a disease with an infection rate of 2%.

Note: This assumes 100% testing accuracy. Since tests are rarely perfect, we would be wise to incorporate test reliability into our model by introducing conditional probability and Bayes' theorem. But that's another discussion.

Can we do better? What about those other powers of two I mentioned earlier? The likelihood of the worst case scenario is fixed, so we want to balance the best case with the "rest case" to minimize the tests. Figure 3 charts the analysis we did above.

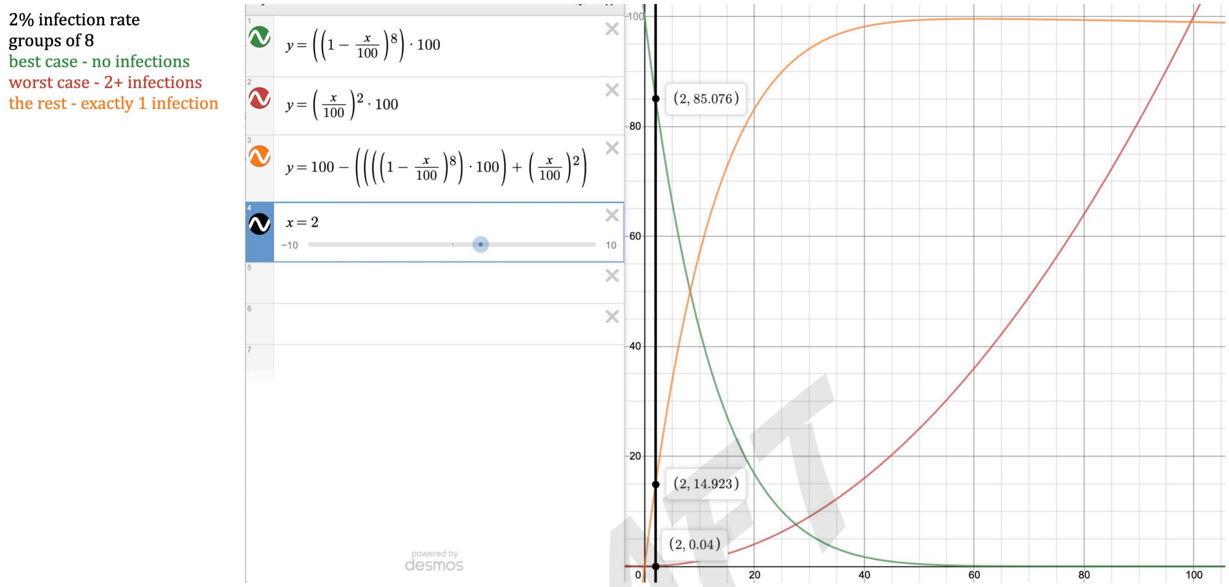



Figure 3 - Groups of 8, Infection Rate of 2%



In Figure 3, the black vertical line represents the infection rate of 2%. (Meaning `x=2` on the horizontal axis between 0 and 100.) The green curve represents the likelihood of the best case scenario occurring as `x` (the infection rate) varies from 1 to 100 percent. Where it crosses the black line (meaning, when the infection rate is 2%) we see that the likelihood of encountering the best case is 85.076%. Note that we can confirm this value with by the calculations we did above. (It's always nice when the numbers work out.) The red line is the worst case curve, which we see intersects the black line at 0.04%, again matching our earlier calculations. Finally, the orange line is the curve for the rest of the cases, crossing the black line at 14.923%, as we saw above. (We can ignore minuscule differences; these are due to rounding.) As it turns out, groups of 8 are very effective. Doing the same analysis on groups of 4 and 16 (the powers of 2 on either side of 8) show that those testing protocols need 335 and 257 tests respectively to screen a population of 1000. (See the appendix for details.)

There are other analyses we could do along these lines. For example, we could fix the group size and vary the infection rate to explore that relationship. There are many fun ways to go. If you're interested in diving even deeper, the seminal treatise on the topic is a book from 1993 (and updated in 2000) by D. Du and F. Hwang called *Combinatorial Group Testing and Its Applications* [2]. There is also a recent paper by M. Aldridge, O. Johnson, and J. Scarlett on the topic called *Group Testing: An Information Theory Perspective* [3].

In summary: efficient and widespread testing of large populations is critical in these pandemic times; we simply do not have enough tests. Lots of group testing and a little computer science can improve COVID-19 screening.

REFERENCES / LINKS


[1] Ellenberg, J; "Five People. One Test. This Is How You Get There." New York Times, May 7, 2020.
https://www.nytimes.com/2020/05/07/opinion/coronavirus-group-testing.html
Accessed May 11, 2020

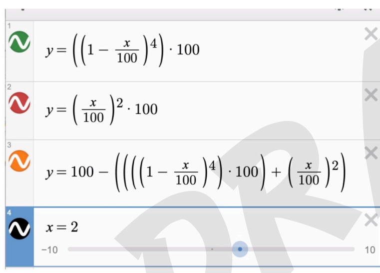
[2] D. Du and F. Hwang. 2000. *Combinatorial Group Testing and Its Applications*. World Scientific.
https://books.google.com/books?id=80hqDQAAQBAJ

[3] M. Aldridge, O. Johnson, and J. Scarlett. "Group Testing: An Information Theory Perspective." arXiv preprint (2019). https://arxiv.org/abs/1902.06002


# APPENDIX



2% infection rate
groups of 4
best case - no infections
worst case - 2+ infections
the rest - exactly 1 infection

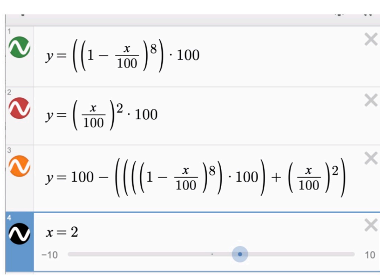


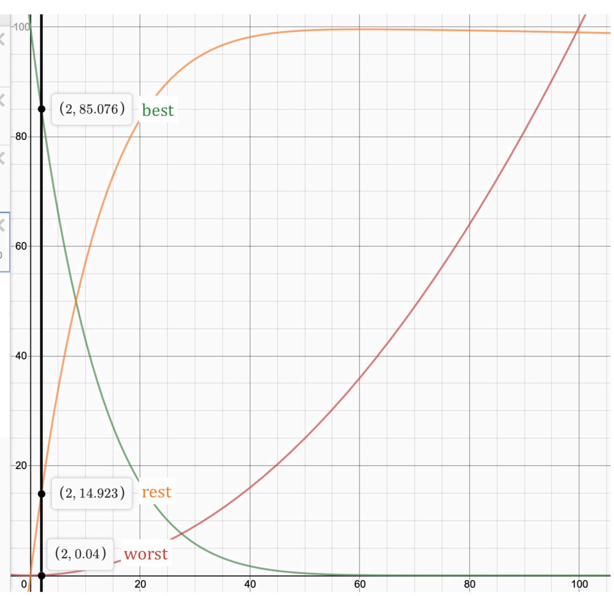


```
For 1000 people (20 infected and 980 infection-free):
Make 250 groups of 4
best case : 250 * 0.9224 = 230.6 instances requiring 231 tests
worst case: 250 * 0.0004 = round up to 1 instance requiring 7 tests
the rest  : 250 * 0.0772 = 19.3 instances requiring tests 97 tests
```

That's **335** tests for 1000 people
(given an infection rate of 2% and 100% testing accuracy)

2% infection rate
groups of 8
best case - no infections
worst case - 2+ infections
the rest - exactly 1 infection

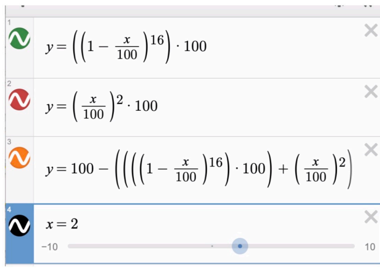


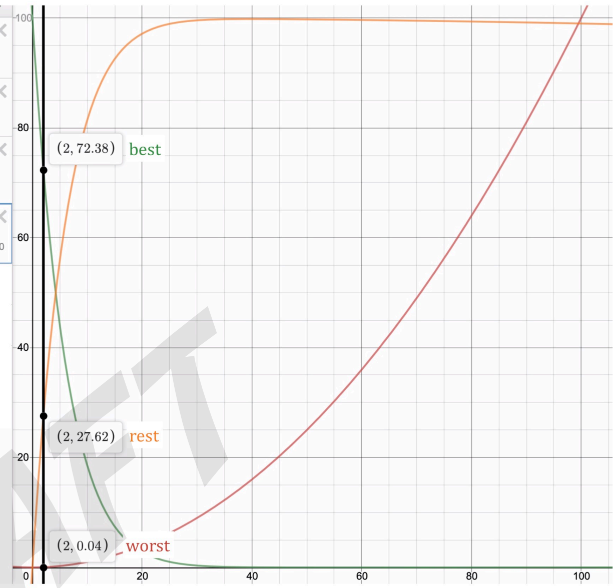




```
For 1000 people (20 infected and 980 infection-free):
Make 125 groups of 8
best case : 125 * 0.8500 = 106.25 instances requiring 107 tests
worst case: 125 * 0.0004 = round up to 1 instance requiring 11 tests
the rest  : 125 * 0.1496 = 18.7 instances requiring 131 tests
```

That's **249** tests for 1000 people
(given an infection rate of 2% and 100% testing accuracy)

2% infection rate
groups of 16
best case - no infections
worst case - 2+ infections
the rest - exactly 1 infection




```
For 1000 people (20 infected and 980 infection-free):
Make 62.5 (call it 63) groups of 16
best case : 63 * 0.7238 = 45.6 instances requiring 46 tests
worst case: 63 * 0.0004 = round up to 1 instance requiring 19 tests
the rest  : 63 * 0.2758 = 17.4 instances requiring 192 tests
```

---

That's **257** tests for 1000 people
(given an infection rate of 2% and 100% testing accuracy)