4-2020

Group Testing Identification: Objective Functions, Implementation, and Multiplex Assays

Brianna D. Hitt
University of Nebraska-Lincoln, brianna.hitt@huskers.unl.edu

GROUP TESTING IDENTIFICATION: OBJECTIVE FUNCTIONS, IMPLEMENTATION, AND MULTIPLEX ASSAYS

by

Brianna D. Hitt

A DISSERTATION

Presented to the Faculty of

The Graduate College at the University of Nebraska

In Partial Fulfilment of Requirements

For the Degree of Doctor of Philosophy

Major: Statistics

Under the Supervision of Professor Christopher R. Bilder

Lincoln, Nebraska

April, 2020

GROUP TESTING IDENTIFICATION: OBJECTIVE FUNCTIONS, IMPLEMENTATION, AND MULTIPLEX ASSAYS

Brianna D. Hitt, Ph.D.

University of Nebraska, 2020

Adviser: Christopher R. Bilder

Group testing is the process of combining items into groups to test for a binary characteristic. One of its most widely used applications is infectious disease testing. In this context, specimens (e.g., blood, urine) are amalgamated into groups and tested. For groups that test positive, there are many algorithmic retesting procedures available to identify positive individuals. The appeal of group testing is that the overall number of tests needed is significantly less than for individual testing when disease prevalence is small and an appropriate algorithm is chosen. Group testing has a number of applications beyond infectious disease testing, such as drug discovery, food contamination detection, and diagnosis of faulty network sensors.

An important decision that needs to be made prior to implementation is the group sizes to use. In best practice, an objective function is minimized to determine the optimal set of group sizes, known as the optimal testing configuration (OTC). We examine several different objective functions and show that the OTCs and corresponding results (e.g., number of tests, accuracy) are largely the same for these functions when using standard group testing algorithms.

Both estimating the probability of disease and identifying positive individuals are goals of group testing. We present the first general R functions for identification and make these available in the new `binGroup2` package. We

also include in this package estimation functions from the `binGroup` package by creating a unified framework for them.

We developed a web-based Shiny application to assist laboratory personnel in determining how well a group testing algorithm is expected to perform before implementation. The app utilizes `binGroup2` functions to calculate the expected number of tests and diagnostic accuracy measures for a wide variety of algorithms using one- and two-disease assays. The OTC can be found with the app as well.

Most group testing research using one-disease assays makes the assumption of equal sensitivity and equal specificity values across all stages of testing. We present derivations of operating characteristics for group testing algorithms that allow the diagnostic test accuracy to differ across stages of testing. These resulting expressions are incorporated into the `binGroup2` package.

DEDICATION

This dissertation is dedicated to my husband Michael, who has sacrificed an immeasurable amount for me to pursue this degree and has always encouraged and supported me; to my parents James and Talani, who always pushed me to do my best and supported my educational goals; and to my son Brecken, may you always pursue your dreams, no matter how long they may take to realize.

ACKNOWLEDGMENTS

I would like to thank my adviser, Dr. Christopher R. Bilder, for his patience and understanding throughout my program, and for spending so much time and effort mentoring me. His attention to detail and level of professionalism have made me a better researcher. I would also like to thank my committee members, Dr. Stephen Kachman, Dr. Qi Zhang, and Dr. Brian Vander Ley for their support. I would like to thank all my friends and family who have encouraged me throughout this journey, especially my parents and parents-in-law for their love and encouragement, and their willingness to help in any way possible. Most of all, I want to express my love and gratitude to my amazing husband, without whom this would simply not be possible.

GRANT INFORMATION

This research was supported by Grant R01 AI121351 from the National Institutes of Health.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1. Fundamentals of group testing

As fears of a German World War II victory spread throughout the United States in 1940, Congress and President Franklin Delano Roosevelt worked together to pass the Selective Training and Service Act (Encyclopaedia Britannica, Inc., 2017). The law established the first peacetime draft in U.S. history and instituted screening to determine men's physical and mental fitness for war. Part of the screening process involved testing for diseases such as syphilis. With millions of people needing to be screened, Dorfman (1943) proposed a new testing algorithm meant to reduce the total number of tests needed. In his algorithm, Dorfman proposed that individual specimens be amalgamated into groups instead of being tested individually. If a group tested negative, all members of the group would be declared negative for the syphilitic antigen. If a group tested positive, all individuals would be retested to determine which were positive and which were negative. Because the prevalence of syphilis was low, it was believed that group testing would result in much fewer tests than testing each specimen separately (i.e., individual testing). Screening for diseases such as syphilis is just one of many diverse group testing applications. Throughout this dissertation, we will focus our terminology on the infectious

disease testing setting. That is, group testing will be discussed in the context of testing specimens (e.g., blood, urine) to identify which are positive or negative for a disease of interest.

Since Dorfman's original proposal, there have been many other group testing algorithms developed. Most of these can be categorized as either hierarchical or non-hierarchical in nature. Hierarchical algorithms involve testing individuals in non-overlapping groups at a particular stage of testing. The testing pattern at each subsequent stage is determined by the results in the previous stage. The Dorfman technique is a two-stage hierarchical algorithm. In contrast, non-hierarchical algorithms involve testing individuals in overlapping groups within a stage. This is done to reduce the number of retests needed at subsequent stages. The next subsections describe each of these algorithm types in detail.

1.1.1. Hierarchical algorithms

Let p be the probability that an individual is truly positive for the disease of interest. For a group of size I, the probability that the group is truly negative (all individuals in the group are truly negative) is $(1-p)^I$. Thus, the probability that the group is truly positive (at least one individual in the group is truly positive) is $1-(1-p)^I$. The expected number of tests needed to decode one group becomes $1+I[1-(1-p)^I]$. Across a population of size N, the expected total number of tests is $N/I + N\left[1-(1-p)^I\right]$, assuming that I divides evenly into N (Dorfman, 1943). In situations where I does not divide evenly into N, any remaining individuals are usually combined in another group so that the expected total number of tests is $1+b\left[1-(1-p)^b\right]+ N/I + N\left[1-(1-p)^I\right]$, where b is the size of the remainder group.

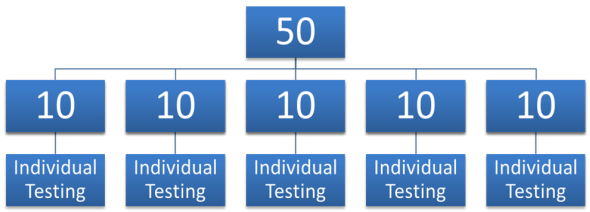
Using these equations, Dorfman (1943) showed that when compared to individual testing, group testing can lead to significant savings for the number of tests, depending on I and p. Choosing too large of an I will result in too many groups testing positive and consequently a large number of retests. On the other hand, choosing too small of an I will lead to a larger number of tests than would be needed if I was chosen better. There was an interest early in this research then to find the "optimal" group size, one that was not too large or too small for the corresponding p. For this reason, Dorfman provided tables to find the optimal group size (i.e., the smallest expected number of tests) given p and concluded that group testing is preferred to individual testing when the disease has a small prevalence (i.e., $p < 0.20$).

The Dorfman algorithm is an example of a two-stage algorithm. Figure 1.1 shows a Dorfman algorithm with an initial group of size 10 followed by individual testing. A natural extension of this technique involves repeatedly dividing groups that test positive into smaller, non-overlapping subgroups until all positive specimens are confirmed through individual testing. Finucan (1964) presented a three-stage algorithm in which an initial group is tested first, subgroups are tested second, and individual retesting is done in the third and final stage of the algorithm. The expected number of tests and minimum cost (associated with the optimal initial group size) were derived for the three-stage algorithm. Finucan (1964) also presented the optimal number of stages and minimum cost for S-stage algorithms in general.

Figure 1.2 illustrates a three-stage hierarchical algorithm used in practice for acute HIV detection in San Francisco (Sherlock et al., 2007). The first stage has an initial group of size 50, followed by five subgroups with 10 individuals each, and finally, individual testing. In current practice, three- and four-stage

Figure 1.1: Dorfman testing algorithm.

Figure 1.2: Three-stage hierarchical testing algorithm.

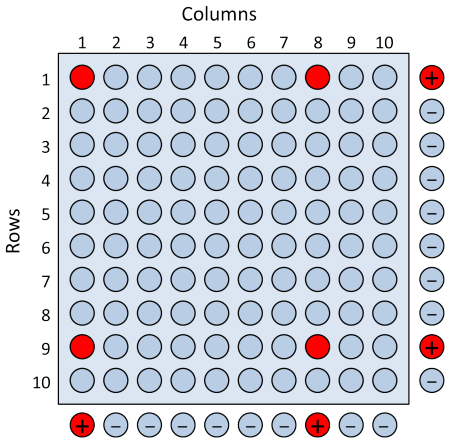


algorithms are often used (e.g. Sherlock et al., 2007; Quinn et al., 2000) because they can be more efficient (i.e., result in fewer tests) than two-stage algorithms in particular situations.

1.1.2. Non-hierarchical algorithms

The most common type of non-hierarchical algorithm is known as array testing. Array testing (Phatarfod and Sudbury, 1994) involves arranging specimens in a square grid, or array. Specimens are amalgamated by row and by column, so that each individual's specimen is included in two groups, and tested. All specimens located at the intersection of a positive row and a positive column are retested separately. In the presence of testing error, retesting is also done for all specimens in a positive row (column) where no column (row) tests positive. Figure 1.3 displays an array testing algorithm where rows 1 and 9

Figure 1.3: Array testing algorithm.



test positive, as do columns 1 and 8. Four individuals (indicated by red circles) lie at the intersections of positive rows and positive columns, so those four individuals are individually retested to determine whether they are positive or negative.

The original array testing proposal by Phatarfod and Sudbury (1994) involved just two stages of testing, the first for testing row and column groups, and the second for individual retesting. Kim et al. (2007) incorporated the possibility of testing error into the algorithm that would necessitate the retesting of entire rows/columns as aforementioned. Kim et al. (2007) also examined adding a master group test to the algorithm. Such an algorithm involves three stages, where a master group of all specimens in the array is tested first, followed by row and column testing in the second stage, and individual retesting in the third and final stage. Kim et al. (2007) provided expressions for the expected number of tests for array testing with and without master pooling. Additional work in this area includes Hudgens and Kim (2011) that studied the optimal group sizes for square array testing without master pooling. Also, Kim and Hudgens (2009) examined three-dimensional array-based test-

ing algorithms, where specimens are arranged in a three-dimensional array for testing.

1.2. Testing error for the overall algorithm

Most of the early group testing papers assumed that the assays were perfect, although this is not the case in many real-life applications. It is now common practice to incorporate testing error in infectious disease testing applications. Individual testing error occurs when a specimen that is truly positive (negative) is incorrectly identified as negative (positive) by the assay. The sensitivity (S_e) is the proportion of true positives correctly identified by the test and the specificity (S_p) is the proportion of true negatives correctly identified by the test (Altman and Bland, 1994a).

Recognizing that the true status of a specimen is not usually known in application, Altman and Bland (1994b) also defined values that describe the proportion of specimens that are correctly diagnosed with individual testing. The positive (negative) predictive value is the proportion of specimens testing positive (negative) who are correctly identified as positive (negative). The positive predictive value (PPV) and negative predictive value (NPV) can then be calculated as

$$PPV = \frac{S_e p}{S_e p + (1 - S_p)(1 - p)}$$

and

$$NPV = \frac{S_p(1 - p)}{(1 - S_e)p + S_p(1 - p)}$$

for any probability of infection p (Altman and Bland, 1994b).

Throughout a group testing algorithm, an assay may be used several times

on the same specimen (either in a group or individually) to determine whether the specimen is positive or negative. For specimens that are tested multiple times, the S_e and S_p no longer describe the probability of being correctly diagnosed by the group testing algorithm though. Instead, Johnson et al. (1991) defined the pooling sensitivity (PS_e) as the probability that an individual is identified as positive, given that the individual is truly positive. Similarly, the pooling specificity (PS_p) is the probability that an individual is identified as negative, given that the individual is truly negative. Kim et al. (2007) defined the pooling positive predictive value ($PPPV$) as the probability that an individual is truly positive, given that the individual is identified by the test as positive. The pooling negative predictive value ($PNPV$) is the probability that an individual is truly negative, given that the individual is identified by the test as negative. Applying Bayes' rule, the $PPPV$ and $PNPV$ can then be expressed as

$$PPPV = \frac{pPS_e}{(1-p)(1-PS_p) + pPS_e}$$

and

$$PNPV = \frac{(1-p)PS_p}{p(1-PS_e) + (1-p)PS_p}.$$

1.3. Informative testing

While nearly all of the previously mentioned papers assume that every individual has the same probability p of testing positive, this assumption is not reasonable in most applications. We can easily imagine that different individuals may have different probabilities of testing positive for a given disease, say p_i for $i = 1, \ldots, N$ individuals to be tested. In many situations, covariate information, such as medical history or risk behaviors, is available for individ-

uals being tested and can be used to estimate each individual's probability of testing positive. The number of tests needed for group testing can then be reduced by taking advantage of these individual probabilities.

Hwang (1975) is likely the first group testing paper that considered testing specimens that do not all have the same probability of being positive. While the paper allows for individual probabilities to differ, it does not discuss how to estimate these probabilities or account for testing error. The first paper to propose a way to identify positive specimens using available covariate information came decades later. Bilder et al. (2010) defined "informative retesting" as an algorithm in which covariate information is used to inform how retesting is implemented in groups that test positive when testing error is present. This paper showed that significant reductions in the expected number of tests for a group testing algorithm can occur when accounting for the differences among individual probabilities. McMahan et al. (2012a) examined informative retesting in the context of Dorfman's algorithm and derived expressions for the expected number of tests, PS_e, PS_p, $PPPV$, and $PNPV$. McMahan et al. (2012b) did the same for informative retesting in the context of array testing without master pooling. Both of these papers continued to show the advantages of using the available covariate information, while also showing that the informative algorithms resulted in similar or sometimes better accuracy than their non-informative ($p_i = p$ for all $i = 1, ..., N$) counterparts. Bilder and Tebbs (2012) compared a number of informative retesting algorithms and found that no single algorithm was best overall in terms of the number of tests and accuracy. Several factors including prevalence, accuracy of the assay, availability of covariate information, and heterogeneity among individual probabilities all are important factors in determining which algorithm

is best. Black et al. (2015) later provided an extension of informative testing to hierarchical testing for three or more stages.

1.4. Multiplex testing

All of the aforementioned papers dealt with group testing in the context of single-disease assays, assays that test for only one disease at a time. In recent years, group testing research has expanded to the use of multiplex assays, i.e., assays that test for more than one disease at a time. Tebbs et al. (2013) examined a two-stage hierarchical algorithm used by the State Hygienic Laboratory (SHL) at the University of Iowa to test for chlamydia and gonorrhea simultaneously via the Aptima Combo 2 Assay. For this algorithm, individual specimens are randomly assigned to groups of two or more, and each group is tested for both diseases. If a group tests negative for both diseases, all members are declared disease free. If a group tests positive for either disease, all members are retested individually for both diseases using the same assay. Tebbs et al. (2013) derived expressions for the expected number of tests and accuracy measures. Using these expressions, they were able to identify optimal group sizes for the two-stage algorithm. The authors found that pooling for multiple infections results in fewer tests than both individual testing with multiplex assays and group testing with single-disease assays.

Hou et al. (2017) generalized the methods in Tebbs et al. (2013) to hierarchical testing with three or more stages and derived closed-form expressions for the expected number of tests and accuracy measures. The authors showed a reduction in the expected number of tests can occur when using these higher-stage hierarchical algorithms rather than the two-stage algorithms. For their contribution to the statistical science and its application, both Tebbs et al.

(2013) and Hou et al. (2017) were awarded the Outstanding Statistical Application award by the American Statistical Association, and Hou et al. (2017) was also awarded the *Biometrics* paper of the year award. However, both of these papers assumed that all individuals had the same probability of testing positive. Bilder et al. (2019) provided the first group testing algorithms for multiplex assays that take advantage of individual-specific probabilities. The algorithms developed can be applied in hierarchical testing with two or more stages, and the authors derived the expected number of tests and accuracy measures for the algorithms. Bilder et al. (2019) showed that the combination of informative group testing and multiplex assays can significantly reduce the number of tests required without loss of accuracy in comparison to non-informative algorithms. Hou et al. (2020) further provided methods for array testing with multiplex assays.

1.5. Assumptions about diagnostic accuracy

The sensitivity and specificity of a diagnostic test are often assumed to be the same for each stage of testing. For example, this means that a three-stage hierarchical algorithm would have $S_e = 0.95$ for the initial group test, subsequent tests of smaller subgroups, and individual tests. This assumption may not be realistic for two reasons: 1) when larger groups are used, positive individuals can be diluted by negative individuals past the threshold of detection for an assay; and 2) different diagnostic tests may be used at different stages of the testing algorithm. In situations where dilution effects are a cause for concern, McMahan et al. (2013) removed the assumption that sensitivity and specificity are constant for all groups (pools) and derived expressions for pool-specific sensitivity and specificity that is dependent upon the size of the group,

improving prevalence estimation.

Usually though, properly calibrated assays will have the same diagnostic accuracy for each test at each stage, especially for nucleic acid amplification tests (NAATs). Kacena et al. (1998a; 1998b) found negligible loss in diagnostic accuracy with as many as 10 samples when screening for chlamydia and gonorrhea with a NAAT. Also, NAATs have been used to detect acute HIV infection in groups of 90 or more specimens with no significant dilution effects (Quinn et al., 2000; Pilcher et al., 2005). For other types of tests, Kline et al. (1989) and Tu et al. (1995) found that groups of up to 15 specimens can be used with negligible loss of diagnostic test accuracy for HIV screening with an enzyme-linked immunosorbent assay (ELISA). Soroka et al. (2003) showed the same for groups of up to size 20 with two different lateral flow rapid antibody assays. These studies support Black et al. (2015), McMahan et al. (2012a), McMahan et al. (2012b), and Kim et al. (2007) in their assumption that sensitivity and specificity do not depend on the size of the group.

In some situations, different assays may be used within a testing algorithm. For example, an ELISA test may be utilized to test an initial group size due to its lower cost and NAATs may be used to test positive groups in subsequent stages due to their frequently higher sensitivity values. For these situations, Bilder et al. (2019) allowed for differences in the diagnostic accuracy across stages of testing in their derivations of operating characteristics (e.g., expected number of tests, accuracy measures) for hierarchical testing algorithms with multiplex assays. Similarly, Hou et al. (2020) allowed for different values of sensitivity and specificity for the master array, row/column tests, and individual testing in their derivations of operating characteristics for array testing algorithms with multiplex assays.

1.6. Implementation in R

There are two primary objectives in group testing research. The first is the estimation problem: estimate the overall prevalence of disease or individual probabilities of testing positive. The `binGroup` package (Bilder et al., 2010) was the first R package to provide functions for the estimation problem for both homogeneous populations, where all individuals are assumed to have the same probability of testing positive, and heterogeneous populations, where each individual is allowed to have a different probability of testing positive. For homogeneous populations, the package provides functions that estimate the overall prevalence p, calculate a p-value and determine power for a hypothesis test involving p, and find the optimal group size for a design. For heterogeneous populations, the contributed functions fit group testing regression models and simulate group testing data.

The second objective in group testing research is the identification problem: identify all positive individuals being tested via a testing algorithm. Unfortunately, the `binGroup` package did not provide functions for the identification problem until a number of functions were added associated with this dissertation. For both the estimation and identification problems, the functions in `binGroup` were contributed by a number of different researchers. This led to an inconsistent style that can make it difficult for practitioners to use.

Several papers in the group testing literature provide R functions to calculate operating characteristics for various group testing algorithms. For single-disease assays, Black et al. (2015) provided functions for up to four-stage hierarchical testing and McMahan et al. (2012a) provided functions for informative two-stage hierarchical testing. McMahan et al. (2012b) supplied functions for

array testing without master pooling. These R functions for single-disease assays utilized the assumption that sensitivity/specificity values are equal across all stages of testing. For multiplex assays, Bilder et al. (2019) developed R functions for up to five-stage hierarchical testing and Hou et al. (2020) developed R functions for array testing with and without master pooling. The functions for hierarchical testing algorithms with multiplex assays allow sensitivity/specificity values to differ across stages of testing, while the functions for array testing with multiplex assays allow for only a single sensitivity/specificity value in the algorithm. All of the R functions mentioned here can be accessed at www.chrisbilder.com/grouptesting.

1.7. Organization of the dissertation

The order of this dissertation is as follows. Chapter 2 is a paper published in *Statistics in Medicine*. This paper compares objective functions that are used to determine the optimal testing configuration (set of group sizes that minimize an objective function) for a significant number of group testing algorithms. The goal of this paper is to settle a controversy in group testing research regarding which objective function is best to use in practice. Chapter 3 describes a new R package named `binGroup2` that provides R functions for both the estimation and identification problems of group testing. This package is built upon the estimation functions in the `binGroup` package but performs a large reorganization of these functions and creates a consistent framework that researchers will find easier to use. The `binGroup2` package also incorporates functions for the identification problem. Included are new functions that enable practitioners to find an optimal testing configuration to implement group testing. Chapter 4 describes a new Shiny application to allow

researchers without R experience to access particular identification functions from `binGroup2` without having to understand code. Chapter 5 summarizes the work completed and proposes ideas for future research.

Chapter 2

The objective function controversy for group testing: Much ado about nothing?

This chapter is published: Hitt, B., Bilder, C., Tebbs, J., and McMahan, C. (2019). The objective function controversy for group testing: Much ado about nothing? Statistics in Medicine 38(24), 4912-4923. Used with permission.

Abstract

Group testing is an indispensable tool for laboratories when testing high volumes of clinical specimens for infectious diseases. An important decision that needs to be made prior to implementation is determining what group sizes to use. In best practice, an objective function is chosen and then minimized to determine an optimal set of these group sizes, known as the optimal testing configuration (OTC). There are a few options for objective functions, and they differ based on how the expected number of tests, assay characteristics, and testing constraints are taken into account. These varied options have led to a recent controversy in the literature regarding which of two different objective functions is better. In our paper, we examine these objective functions over a number of realistic situations for infectious disease testing. We show that this controversy may be much ado about nothing because the OTCs and cor-

responding results (e.g., number of tests, accuracy) are largely the same for standard testing algorithms in a wide variety of situations.

Keywords: Binary response; Infectious disease; Pooled testing; Screening; Sensitivity; Specificity

2.1. Introduction

Laboratories throughout the world test high volumes of clinical specimens for infectious diseases, including HIV, hepatitis C, and West Nile virus. In such situations, it has become standard practice to test amalgamations of specimens as a "group" or "pool" rather than to test individual specimens. The reason is simple: members of a negative testing group can be declared negative all at once. Thus, for a group of size I, say, just one test is needed to declare all members negative, rather than the I separate tests that would be needed with individual testing. Fortunately, when disease prevalence is small, the majority of groups will test negatively when sensibly chosen group sizes are used. For members of a positive testing group, there are many algorithmic retesting procedures available to determine which specific individuals are positive. The first retesting procedure was proposed by Dorfman (1943) and simply involved individually retesting each member of a positive group. Since this seminal work, group testing has been used to efficiently test for infectious diseases in a vast number of human applications, including blood donation screening (American Red Cross, 2020), antiretroviral treatment failure detection for HIV-positive individuals (Kim et al., 2014; Tilghman et al., 2015), chlamydia and gonorrhea testing (Papp et al., 2014), and influenza outbreak surveillance (Hourfar et al., 2007). Outside of infectious disease testing in humans, group testing is used in

an extensive number of applications, including cow milk surveillance (Græsbøll et al., 2017), disease detection in cattle and buffaloes (Abdellrazeq et al., 2014), West Nile virus monitoring in mosquitoes (Khan et al., 2017), food contamination detection (Pasquali et al., 2014), drug discovery (Kainkaryam and Woolf, 2009), and diagnosis of faulty network sensors (Lo et al., 2013).

For all group testing applications, the choice of group sizes is extremely important for success. Choosing group sizes too large will lead to exceedingly many groups testing positively. This will subsequently lead to a large number of retests, perhaps even a larger number of tests overall than what would be needed for individual testing. Similarly, choosing group sizes too small will lead to a larger number of tests than would be needed if the group sizes were chosen better. In best practice, laboratories choose group sizes by minimizing an objective function that takes into account the group testing algorithm to be implemented. There are a number of different algorithms in use, and they are best characterized as being either hierarchical or non-hierarchical in nature. Hierarchical algorithms begin by testing individuals in non-overlapping groups. For a group that tests positively, subsequent retesting stages occur in smaller, non-overlapping groups. The previously described Dorfman algorithm is a two-stage algorithm. Three- and four-stage algorithms are commonly used in practice (Quinn et al., 2000; Sherlock et al., 2007) because they are often more efficient (i.e., fewer tests). Non-hierarchical algorithms involve testing each individual in overlapping groups to reduce the number of retests. The most common type of non-hierarchical algorithm is known as array testing (Phatarfod and Sudbury, 1994; Kim et al., 2007). For this algorithm, individual specimens are arranged in a two-dimensional grid. These specimens are amalgamated by row and by column and then tested. Intersecting positive

rows and columns indicate where retesting should be performed to determine which individuals are positive. For a thorough review of hierarchical and array testing algorithms, see Hughes-Oliver (2006) and Bilder (2019).

While there are many different types of group testing algorithms, all laboratories are interested in minimizing the number of tests needed to assay their specimens. For this reason, objective functions are based on the expected number of tests, so that a set of group sizes for a testing algorithm, known as the optimal testing configuration (OTC), can be found by minimizing this function. Traditionally, group testing research has focused on objective functions expressed solely as the expected number of tests per individual. This is due to a close correspondence between the number of tests and testing costs. However, using an objective function that contains only the expected number of tests leaves out an important component of infectious disease testing: accuracy. Infectious disease testing is rarely perfect. Errors can occur for reasons such as improper laboratory implementation or a specimen being collected during the window period between disease contraction and the ability to detect it. Fortunately, known mathematical expressions are available for the accuracy of most group testing algorithms. This enables laboratories to calculate the expected accuracy of a chosen testing configuration prior to implementation.

Malinovsky et al. (2016) recently proposed a new objective function that includes the expected number of tests and a measurement of accuracy. This allows laboratories to evaluate accuracy at the same time as the number of tests when choosing an OTC. As may be expected when breaking with tradition, the proposal generated controversy in the group testing research literature. Both Hudgens (2016) and McMahan et al. (2016) offered rejoinders to Malinovsky et al. (2016) that disagreed with this new objective function. All three of these

works focused only on the Dorfman algorithm in their limited evaluations. The purpose of our paper is to examine a significant number of other group testing algorithms with respect to objective functions. This is important because other algorithms are widely used and known to result in a smaller number of tests and/or higher accuracy than the Dorfman algorithm. We present findings in our paper that interestingly show both the traditional and the new objective function are actually quite similar and very often lead to the same OTC in realistic infectious disease testing situations.

The order of this paper follows. Section 2.2 explicitly defines the objective functions and provides a mathematical comparison between them. Section 2.3 calculates the OTC for each objective function along with their operating characteristics (expected number of tests and accuracy measures) in a wide variety of settings. These calculations are performed for both hierarchical and array testing algorithms. We show under what conditions these operating characteristics will be the same and when they will be different. Section 2.4 examines the objective function controversy in the context of actual assays used for infectious disease detection. To conclude, Section 2.5 summarizes our findings, discusses alternative objective functions, and provides recommendations for practice. We also discuss R functions that we provide with our paper to find the OTCs and to reproduce our work.

2.2. Objective Functions

Define T as a random variable representing the total number of tests for an overall group of size I with a hierarchical algorithm. When using the traditional objective function, the OTC is found by minimizing the expected

number of tests per individual:

$$O_{ET} = E(T)/I.$$

For example, the expected number of tests for three-stage hierarchical testing is given by

$$E(T) = 1 + m_{11}P(G_{11} = 1) + \sum_{j=1}^{c_2} m_{2j}P(G_{11} = 1, G_{2j} = 1),$$

where G_{sj} is the binary random variable (values of 1 and 0 indicate a positive and a negative test result, respectively) representing the outcome for group j at stage s, m_{sj} is the number of subgroups that would be created if group j at stage s tests positively, and c_s is the number of groups at stage s (see Black et al. (2015); an example diagram is given in Appendix A). The probabilities $P(G_{11} = 1)$ and $P(G_{11} = 1, G_{2j} = 1)$ are both functions of the number of groups and their respective sizes, the probability of being positive for each individual, and the sensitivity S_e and specificity S_p of the assay each time it is applied. We do not provide further detailed expressions for $E(T)$ here to avoid distraction from the main points of our paper and because expressions are already provided elsewhere. For example, Kim et al. (2007) provides expressions for the case of each individual having the same true probability of being positive, say p, and Black et al. (2015) provides expressions for the case of each individual potentially having a different probability of being truly positive, say p_i for $i = 1, \ldots, I$. The latter case is known as informative group testing (Bilder et al., 2010; Lewis et al., 2012; Bilder and Tebbs, 2012), because p_i can be estimated with the help of disease-risk information that may be available for each individual tested. We will refer to the former case then as

non-informative group testing in our work here. Expressions for the expected number of tests are known for array testing algorithms (Kim et al., 2007; McMahan et al., 2012b) as well, where O_{ET} is still defined as the expected number of tests per individual.

While O_{ET} is the most commonly utilized objective function, it does not directly take into account the accuracy of the algorithm. However, one will still examine separately the accuracy of the OTC to judge if it is satisfactory. As an alternative approach, Malinovsky et al. (2016) proposed an objective function that simultaneously takes into account accuracy and the expected number of tests. To examine the accuracy aspect, define Y_i as the final positive/negative (1/0) outcome based on the group testing algorithm, and define $\tilde{Y}_i$ as the true positive/negative (1/0) status of individual i, for $i = 1, \ldots, I$. Commonly used accuracy measures for a group testing algorithm as a whole are the pooling sensitivity $PS_{e,i} = P(Y_i = 1|\tilde{Y}_i = 1)$ and the pooling specificity $PS_{p,i} = P(Y_i = 0|\tilde{Y}_i = 0)$ for individual i. As an overall measure of accuracy, define C as the number of correct classifications for a group of size I. The expected number of correct classifications is

$$\begin{aligned} E(C) &= \sum_{i=1}^{I} \left\{ P(Y_i = 0, \tilde{Y}_i = 0) + P(Y_i = 1, \tilde{Y}_i = 1) \right\} \\ &= \sum_{i=1}^{I} \left\{ PS_{p,i}(1 - p_i) + PS_{e,i} p_i \right\}, \end{aligned} \tag{2.2.1}$$

where $P(\tilde{Y}_i = 1) = p_i$ is the probability that individual i is truly positive.

Malinovsky et al. (2016) proposed to find the OTC by maximizing the expected number of correct classifications per individual divided by the expected

number of tests per individual. Equivalently, this results in minimizing

$$O_{MAR} = E(T)/E(C).$$

Because C is never larger than the number of individuals I, $E(C) \leq I$. By comparing O_{MAR} and O_{ET}, we see that

$$O_{ET} = \frac{E(T)}{I} \leq \frac{E(T)}{E(C)} = O_{MAR}$$

for the same initial group size I. In fact, O_{MAR} and O_{ET} will be quite close in value. This is because infectious disease assays will only be put into use if they have high accuracy. Thus, $E(C)$ will be quite close to I in practice.

To examine this closeness more precisely, consider minimizing the logarithm of each objective function:

$$\log(O_{ET}) = \log\{E(T)\} - \log(I)$$

and

$$\log(O_{MAR}) = \log\{E(T)\} - \log\{E(C)\}. \tag{2.2.2}$$

For hierarchical testing, the pooling sensitivity is always the same for every individual tested in the same number of stages(Kim et al., 2007; Black et al., 2015). The pooling specificity is the same for every individual as well, but only for non-informative group testing with equal group sizes within a stage. Under this scenario then, we can simplify the expression for the expected number of correct classifications to be

$$E(C) = I\{PS_p(1-p) + PS_e p\}, \tag{2.2.3}$$

where PS_p and PS_e are the pooling specificity and sensitivity, respectively, but now equal for each individual. For array testing, the same simplification for $E(C)$ from Equation (2.2.1) to Equation (2.2.3) occurs when the number of rows and the number of columns are the same (i.e., a square array), which is how array testing is usually applied.

By substituting Equation (2.2.3) into Equation (2.2.2), we obtain

$$\begin{aligned} \log(O_{MAR}) &= \log\{E(T)\} - \log\left[I\left\{PS_p(1-p) + PS_e p\right\}\right] \\ &= \log(O_{ET}) - \log\left\{PS_p(1-p) + PS_e p\right\}. \end{aligned}$$

Thus, any difference between the OTCs for the two objective functions is due to the "penalty" of

$$\log\left\{PS_p(1-p) + PS_e p\right\}. \tag{2.2.4}$$

Unfortunately, further definitive statements cannot be made regarding Equation (2.2.4), and we are left with making general statements regarding what will happen most often. In particular, we see that the penalty places a large weight on PS_p in comparison to PS_e because p is small for realistic group testing applications. Also, because PS_p and PS_e tend to be close to 1 for realistic applications, the penalty tends to be close to 0. Thus, $\log(O_{MAR})$ will most often be close to $\log(O_{ET})$.

2.3. Comparisons

Because definitive statements are not possible for Equation (2.2.4) or for the more general cases of unequal group sizes and informative group testing, we provide in this section a thorough investigation of the OTCs when using the objective functions over a very large number of situations. For each of these

situations, we calculate the OTCs along with corresponding operating characteristics. Our results for both non-informative and informative group testing algorithms are described next.

2.3.1. Non-informative group testing

We include in this investigation the following group testing algorithms: two-stage hierarchical, three-stage hierarchical, array testing without a master pool (row and column groups are tested first, as described in Section 2.1), and array testing with a master pool (all specimens in the array are tested together in one group before any row or column groups are formed). For the first three algorithms, we allow the initial group sizes to range from $I = 3, ..., 40$, but allow higher initial group sizes when the overall prevalence is very small (e.g., $p = 0.005$) so that the OTC does not include our arbitrary upper bound for I. For array testing with a master pool, we use the same range of group sizes for the row and column groups, leading to a maximum master pool size of I^2. All array testing algorithms use square arrays, and we account for potential testing ambiguities that can occur in arrays (e.g., a row tests positively without any columns testing positively) by the methods described in Kim et al. (2007) We apply these group testing algorithms over thirty different values of p ranging from 0.005 to 0.150 by 0.005 and over five separate sets of accuracy levels (S_e and S_p values range from 0.90 to 0.99). These values of p, S_e, and S_p are chosen because they correspond to when group testing is used for infectious disease testing. The assay accuracies are assumed to not change based on group size, meaning that the assays have been properly tested and calibrated for group testing.

Table 2.1 displays the results for $p = 0.01$. The OTCs are the same for

both objective functions when using the hierarchical algorithms. Some small differences between OTCs exist for the array testing algorithms, but the differences are not of practical importance. For example, examine the results for array testing without master pooling and $S_e = S_p = 0.90$. The expected number of tests and the pooling sensitivities are the same to four decimal places. The pooling specificities are also quite close. In practical terms, for a testing load of 100,000 individuals, there would be 98,267 correct negatives found when using the OTC for O_{ET} and 98,307 correct negatives found when using the OTC for O_{MAR}. While 40 additional false positives would result from the OTC for O_{ET}, these false positives would most likely be discovered from follow-up confirmatory testing that normally would occur. We also provide similar tables for $p = 0.05$ and $p = 0.10$ in Appendix A. These tables show only one case with differences between the OTCs.

Table 2.2 summarizes the largest differences among the operating characteristics across all thirty different values of p included in our investigation. Most often, the OTCs found are the same for the two objective functions. When differences exist, these differences occur more often for smaller values of S_p, but again are not of practical importance. Overall, these findings help confirm what was strongly suspected in Section 2.2 through our mathematical analysis. Namely, the objective functions lead to the same OTCs or OTCs with similar operating characteristics when differences exist.

2.3.2. Informative group testing

We include in this investigation the following group testing algorithms: two-stage hierarchical implemented via the pool-specific optimal Dorfman (PSOD) method (McMahan et al., 2012a), three-stage hierarchical (Black et al., 2015),

Table 2.1: OTC summary for $p = 0.01$ under non-informative group testing. Equally sized groups are optimal at each stage; thus, an OTC of "24-6-1" means that stage 1 has a group of size 24, stage 2 has four groups of size 6, and stage 3 has twenty-four groups of size 1. Differences between O_{ET} and O_{MAR} are highlighted.

Algorithm	S_e	S_p	Objective function	OTC	$E(T)/I$	PS_e	PS_p
Two-stage hierarchical	0.99	0.99	O_{ET}	11-1	0.2035	0.9801	0.9990
			O_{MAR}	11-1	0.2035	0.9801	0.9990
	0.95	0.95	O_{ET}	11-1	0.2351	0.9025	0.9932
			O_{MAR}	11-1	0.2351	0.9025	0.9932
	0.90	0.90	O_{ET}	12-1	0.2742	0.8100	0.9816
			O_{MAR}	12-1	0.2742	0.8100	0.9816
	0.99	0.90	O_{ET}	11-1	0.2841	0.9801	0.9815
			O_{MAR}	11-1	0.2841	0.9801	0.9815
	0.90	0.99	O_{ET}	11-1	0.1941	0.8100	0.9990
			O_{MAR}	11-1	0.1941	0.8100	0.9990
Three-stage hierarchical	0.99	0.99	O_{ET}	25-5-1	0.1354	0.9703	0.9996
			O_{MAR}	25-5-1	0.1354	0.9703	0.9996
	0.95	0.95	O_{ET}	24-6-1	0.1443	0.8574	0.9973
			O_{MAR}	24-6-1	0.1443	0.8574	0.9973
	0.90	0.90	O_{ET}	24-6-1	0.1562	0.7290	0.9938
			O_{MAR}	24-6-1	0.1562	0.7290	0.9938
	0.99	0.90	O_{ET}	24-6-1	0.1708	0.9703	0.9928
			O_{MAR}	24-6-1	0.1708	0.9703	0.9928
	0.90	0.99	O_{ET}	25-5-1	0.1229	0.7290	0.9997
			O_{MAR}	25-5-1	0.1229	0.7290	0.9997
Array w/o master pooling	0.99	0.99	O_{ET}	25-1	0.1378	0.9703	0.9995
			O_{MAR}	25-1	0.1378	0.9703	0.9995
	0.95	0.95	O_{ET}	25-1	0.1475	0.8575	0.9970
			O_{MAR}	24-1	0.1475	0.8575	0.9972
	0.90	0.90	O_{ET}	25-1	0.1611	0.7291	0.9926
			O_{MAR}	24-1	0.1611	0.7291	0.9930
	0.99	0.90	O_{ET}	23-1	0.1726	0.9703	0.9923
			O_{MAR}	23-1	0.1726	0.9703	0.9923
	0.90	0.99	O_{ET}	27-1	0.1279	0.7292	0.9995
			O_{MAR}	27-1	0.1279	0.7292	0.9995
Array w/ master pooling	0.99	0.99	O_{ET}	625-25-1	0.1364	0.9606	0.9995
			O_{MAR}	625-25-1	0.1364	0.9606	0.9995
	0.95	0.95	O_{ET}	625-25-1	0.1402	0.8146	0.9972
			O_{MAR}	576-24-1	0.1402	0.8146	0.9974
	0.90	0.90	O_{ET}	625-25-1	0.1450	0.6562	0.9934
			O_{MAR}	576-24-1	0.1450	0.6562	0.9937
	0.99	0.90	O_{ET}	529-23-1	0.1708	0.9606	0.9924
			O_{MAR}	529-23-1	0.1708	0.9606	0.9924
	0.90	0.99	O_{ET}	729-27-1	0.1151	0.6563	0.9996
			O_{MAR}	729-27-1	0.1151	0.6563	0.9996

Table 2.2: Largest differences between operating characteristics for OTCs under non-informative group testing. Values of p range from 0.005 to 0.150 by 0.005. The frequency column denotes the number of times a different OTC was found for O_{ET} and O_{MAR} among these values of p. Differences between operating characteristics are rounded to four decimal places. Note that the operating characteristic value for O_{ET} is always subtracted from the operating characteristic value for O_{MAR}. Thus, a negative value (indicated with parentheses) means that the value for O_{ET} was larger than the value for O_{MAR}.

				Largest difference		
Algorithm	S_e	S_p	Frequency	$E(T)/I$	PS_e	PS_p
Two-stage hierarchical	0.99	0.99	0	-	-	-
	0.95	0.95	3	0.0018	0.0000	0.0049
	0.90	0.90	4	0.0023	0.0000	0.0054
	0.99	0.90	7	0.0056	0.0000	0.0096
	0.90	0.99	0	-	-	-
Three-stage hierarchical	0.99	0.99	0	-	-	-
	0.95	0.95	1	0.0014	0.0000	0.0051
	0.90	0.90	3	0.0015	0.0000	0.0049
	0.99	0.90	7	0.0041	(0.0098)	0.0136
	0.90	0.99	1	0.0000	0.0000	0.0002
Array w/o master pooling	0.99	0.99	0	-	-	-
	0.95	0.95	5	0.0010	0.0018	0.0026
	0.90	0.90	8	0.0028	0.0022	0.0054
	0.99	0.90	5	0.0043	0.0005	0.0076
	0.90	0.99	1	0.0000	0.0006	0.0001
Array w/ master pooling	0.99	0.99	2	0.0005	0.0006	0.0008
	0.95	0.95	4	0.0012	0.0017	0.0026
	0.90	0.90	8	0.0015	0.0018	0.0051
	0.99	0.90	5	0.0048	0.0005	0.0077
	0.90	0.99	2	0.0003	0.0026	0.0005

and array testing without a master pool implemented via the gradient method (McMahan et al., 2012b). For the PSOD method, we use a block size of 50 and replace its greedy optimization algorithm with examination of all possible testing configurations. Array testing with a master pool is not included in our investigations because there have been no informative group testing algorithms proposed for it. We continue to allow the initial group sizes to range from $I = 3, ..., 40$ and allow for higher initial group sizes when the overall prevalence is very small.

To provide different levels of heterogeneity among the p_i for $i = 1, \ldots, I$, we use the expected value of order statistics from $P_i \sim$ beta $\{\alpha, \alpha(1-p)/p\}$ for $i = 1, \ldots, I$ in the same manner as in Black et al. (2015) This beta distribution has $E(P_i) = p$, and we once again consider values of p ranging from 0.005 to 0.150 by 0.005. The amount of heterogeneity is controlled by α, where lower levels indicate a larger amount of heterogeneity (see Black et al. 2015 for further discussion regarding the choice of α).

Table 2.3 displays the results for $E(P_i) = 0.01$, and Appendix A provides the results for $E(P_i) = 0.05$ and $E(P_i) = 0.10$. The displayed pooling sensitivity, PS_e^W, and pooling specificity, PS_p^W, are weighted averages of individual pooling sensitivities and pooling specificities, respectively, for all individuals within the initial group for a hierarchical algorithm or within the entire array for an array testing algorithm. Expressions for these averages are provided in Appendix A and are based on accuracy definitions given by Altman and Bland (1994a). The largest differences for each operating characteristic across all values of p are given in Table 2.4. Overall, while differences exist more often for some algorithms than in the non-informative group testing setting, O_{ET} and O_{MAR} still result in the same or very similar OTCs the majority of

the time, and, when differences exist, the vast majority of the differences likely would not be of practical importance due to similar operating characteristic values.

For three-stage hierarchical, the maximum difference in PS_e for some settings, such as $S_e = 0.90$ and $S_p = 0.99$, may be somewhat concerning at a first examination. Further investigation revealed that this occurred when the OTC for O_{MAR} had more sub-groups in the second stage of testing with a size of 1 than did the OTC for O_{ET}. This is important because 1) a third stage of testing is unnecessary for those individuals with a sub-group size of 1 in the second stage of testing; 2) pooling sensitivity for each individual is S_e^L, where L is the number of stages that the individual is tested within (Black et al., 2015); and 3) PS_e^W is a weighted average of each individual's pooling sensitivity. Especially when p is large for three-stage hierarchical testing, the initial group size can be quite small, so each individual's pooling sensitivity plays a larger role in the weighted average. Thus, while there are some differences in the weighted averages of the pooling sensitivities, it is due to those few individuals who are not tested in the third stage. The individuals tested in the same number of stages still have the same pooling sensitivity values.

2.4. Applications

We present two different applications comparing the OTCs obtained from using O_{ET} or O_{MAR} for infectious disease testing. To find the OTC for these and other applications, the value of p or p_i for $i = 1, \ldots, I$ is needed. Of course, these quantities would most likely be unknown. Instead, some type of past experience would be used by laboratories to estimate these quantities so that an "estimated" OTC could be chosen. Also to find the OTC, the values of

Table 2.3: OTC summary for $E(P_i) = 0.01$ under informative group testing. Multiple initial group sizes for two-stage hierarchical algorithms are found within a block size of 50, so they are not displayed here. The full OTCs are provided in Appendix A. Differences between O_{ET} and O_{MAR} are highlighted.

				$\alpha = 2$				$\alpha = 0.5$			
Algorithm	S_e	S_p	Objective function	Initial group size for OTC	$E(T)/I$	PS_e^W	PS_p^W	Initial group size for OTC	$E(T)/I$	PS_e^W	PS_p^W
Two-stage hierarchical	0.99	0.99	O_{ET}	-	0.1947	0.9801	0.9991	-	0.1683	0.9801	0.9992
			O_{MAR}	-	0.1947	0.9801	0.9991	-	0.1683	0.9801	0.9992
	0.95	0.95	O_{ET}	-	0.2264	0.9025	0.9931	-	0.2019	0.9025	0.9943
			O_{MAR}	-	0.2264	0.9025	0.9931	-	0.2019	0.9025	0.9943
	0.90	0.90	O_{ET}	-	0.2657	0.8100	0.9822	-	0.2439	0.8100	0.9843
			O_{MAR}	-	0.2657	0.8100	0.9822	-	0.2439	0.8100	0.9843
	0.99	0.90	O_{ET}	-	0.2754	0.9801	0.9813	-	0.2511	0.9801	0.9837
			O_{MAR}	-	0.2754	0.9801	0.9813	-	0.2511	0.9801	0.9837
	0.90	0.99	O_{ET}	-	0.1854	0.8100	0.9990	-	0.1611	0.8100	0.9993
			O_{MAR}	-	0.1854	0.8100	0.9990	-	0.1611	0.8100	0.9993
Three-stage hierarchical	0.99	0.99	O_{ET}	26	0.1285	0.9703	0.9996	33	0.1197	0.9703	0.9996
			O_{MAR}	26	0.1285	0.9703	0.9996	33	0.1197	0.9703	0.9996
	0.95	0.95	O_{ET}	26	0.1375	0.8574	0.9974	28	0.1291	0.8574	0.9977
			O_{MAR}	26	0.1375	0.8574	0.9974	28	0.1291	0.8574	0.9977
	0.90	0.90	O_{ET}	26	0.1497	0.7290	0.9939	29	0.1422	0.7290	0.9942
			O_{MAR}	26	0.1497	0.7290	0.9939	29	0.1422	0.7290	0.9942
	0.99	0.90	O_{ET}	26	0.1638	0.9703	0.9930	28	0.1554	0.9703	0.9935
			O_{MAR}	26	0.1638	0.9703	0.9930	28	0.1554	0.9703	0.9935
	0.90	0.99	O_{ET}	26	0.1168	0.7290	0.9997	37	0.1078	0.7290	0.9997
			O_{MAR}	26	0.1168	0.7290	0.9997	37	0.1078	0.7290	0.9997
Array w/o master pooling	0.99	0.99	O_{ET}	25	0.1349	0.9703	0.9995	28	0.1277	0.9703	0.9995
			O_{MAR}	25	0.1349	0.9703	0.9995	28	0.1277	0.9703	0.9995
	0.95	0.95	O_{ET}	25	0.1448	0.8575	0.9972	28	0.1379	0.8574	0.9971
			O_{MAR}	25	0.1448	0.8575	0.9972	27	0.1379	0.8574	0.9972
	0.90	0.90	O_{ET}	25	0.1585	0.7291	0.9929	28	0.1519	0.7290	0.9927
			O_{MAR}	25	0.1585	0.7291	0.9929	27	0.1519	0.7290	0.9930
	0.99	0.90	O_{ET}	23	0.1699	0.9703	0.9926	25	0.1631	0.9703	0.9926
			O_{MAR}	23	0.1699	0.9703	0.9926	25	0.1631	0.9703	0.9926
	0.90	0.99	O_{ET}	27	0.1251	0.7293	0.9996	30	0.1181	0.7291	0.9996
			O_{MAR}	27	0.1251	0.7293	0.9996	30	0.1181	0.7291	0.9996

Table 2.4: Largest differences between operating characteristics for OTCs under informative group testing. Values of $E(P_i) = p$ range from 0.005 to 0.150 by 0.005. The frequency column denotes the number of times a different OTC was found among these values of p. Differences between operating characteristics are rounded to four decimal places. Note that the operating characteristic value for O_{ET} is always subtracted from the operating characteristic value for O_{MAR}. Thus, a negative value (indicated with parentheses) means that the value for O_{ET} was larger than the value for O_{MAR}.

					Largest difference		
Algorithm	α	S_e	S_p	Frequency	$E(T)/I$	PS_e^W	PS_p^W
Two-stage hierarchical	2	0.99	0.99	0	-	-	-
		0.95	0.95	7	0.0006	(0.0023)	0.0011
		0.90	0.90	12	0.0010	(0.0052)	0.0023
		0.99	0.90	12	0.0011	(0.0008)	0.0022
		0.90	0.99	2	0.0003	0.0052	0.0000
	0.5	0.99	0.99	0	-	-	-
		0.95	0.95	3	0.0003	(0.0035)	0.0011
		0.90	0.90	15	0.0008	(0.0103)	0.0022
		0.99	0.90	16	0.0012	(0.0011)	0.0022
		0.90	0.99	11	0.0006	0.0078	(0.0002)
Three-stage hierarchical	2	0.99	0.99	1	0.0000	(0.0019)	0.0002
		0.95	0.95	2	0.0035	0.0219	0.0033
		0.90	0.90	6	0.0044	0.0152	0.0062
		0.99	0.90	4	0.0035	0.0006	0.0066
		0.90	0.99	14	0.0180	0.0500	0.0003
	0.5	0.99	0.99	1	0.0000	0.0001	0.0001
		0.95	0.95	0	-	-	-
		0.90	0.90	3	0.0010	0.0250	0.0033
		0.99	0.90	5	0.0022	0.0034	0.0070
		0.90	0.99	9	0.0057	0.0355	0.0003
Array w/o master pooling	2	0.99	0.99	1	0.0003	0.0004	0.0005
		0.95	0.95	2	0.0011	0.0012	0.0027
		0.90	0.90	5	0.0016	0.0012	0.0040
		0.99	0.90	4	0.0028	0.0003	0.0053
		0.90	0.99	0	-	-	-
	0.5	0.99	0.99	0	-	-	-
		0.95	0.95	4	0.0003	0.0004	0.0015
		0.90	0.90	14	0.0015	0.0004	0.0032
		0.99	0.90	8	0.0024	0.0001	0.0041
		0.90	0.99	1	0.0003	0.0005	0.0003

S_e and S_p are needed because $E(T)$ and the pooling sensitivities/specificities depend upon them. Laboratories can obtain these values from a number of sources, including internal validations, research articles, product inserts for assays, and summaries provided by organizations such as the Centers for Disease Control and Prevention (CDC) and the Association of Public Health Laboratories. For each source, the sensitivity and specificity are actually observed through taking a large sample. For instance, a set of known positive specimens may be tested to evaluate the sensitivity of an assay. Alternatively, clinical-based evaluations may be performed by applying the assay in practice and using other means to validate true positive/negative statuses. The observed sensitivities and specificities usually are treated as constants and sometimes confidence intervals are stated along with them. Our purpose in this section is not to evaluate these procedures but rather use the accuracy measures as they are in practice to determine OTCs.

Group testing is used widely for HIV testing in applications including blood donation screening (American Red Cross, 2020) and health surveillance via public health clinics (Sherlock et al., 2007). Branson et al. (2014) provided the CDC's recommendations for HIV testing by laboratories. To make these recommendations, the authors examined over 30 research articles and product inserts, and they included the sensitivities and specificities associated with each assay examined. Observed sensitivities ranged from 96.3% to 100%, and observed specificities ranged from 99.03% to 100%. For our investigation here, we use the lowest values in these ranges to find the OTC. Our reason for using these particular values is because differences between OTCs would most likely occur with the lowest accuracies. Table 2.5 provides the OTCs from non-informative group testing algorithms. For these calculations, we use an

Table 2.5: OTC summary for HIV testing using $p = 0.004$, $S_e = 0.963$, and $S_p = 0.9903$, with non-informative group testing. Equally sized groups are optimal at each stage; thus, an OTC of "24-6-1" means that stage 1 has a group of size 24, stage 2 has four groups of size 6, and stage 3 has twenty-four groups of size 1. There are no differences between the OTCs.

Algorithm	Objective function	OTC	$E(T)/I$	PS_e	PS_p
Two-stage hierarchical	O_{ET}	17-1	0.1313	0.9274	0.9993
	O_{MAR}	17-1	0.1313	0.9274	0.9993
Three-stage hierarchical	O_{ET}	49-7-1	0.0732	0.8931	0.9998
	O_{MAR}	49-7-1	0.0732	0.8931	0.9998
Array w/o master pooling	O_{ET}	44-1	0.0749	0.8931	0.9997
	O_{MAR}	44-1	0.0749	0.8931	0.9997
Array w/ master pooling	O_{ET}	1936-44-1	0.0721	0.8600	0.9998
	O_{MAR}	1936-44-1	0.0721	0.8600	0.9998

overall HIV prevalence of $p = 0.004$ based on CDC estimates of HIV (Centers for Disease Control and Prevention, 2019) and Census Bureau estimates of population (U.S. Census Bureau, Population Division, 2018) in the United States from 2016. Overall, the table shows that O_{ET} and O_{MAR} lead to the same OTCs for all group testing algorithms considered. While the OTCs for array testing with master pooling are the same for O_{ET} and O_{MAR}, a master pool with a 44×44 array may be too large to use in practice (the largest group size that we have seen used for HIV testing is 128 (Sullivan et al., 2011)). A laboratory may need to choose a sub-optimal array size for such a situation.

Group testing is used widely for chlamydia testing as well. High volumes of clinical specimens are tested each year in this manner by public health laboratories across the United States as part of statewide surveillance projects (e.g., see Lewis et al. (2012) and Bilder et al. (2019)). Black et al. (2012) examined the testing performed by the Nebraska Public Health Laboratory (NPHL) with the BD ProbeTec ET CT/GC Amplified DNA Assay. A main purpose of this paper was to evaluate how well an informative group testing

algorithm could perform in comparison to their current implementation of individual testing. For our purpose here, we use the observed data from the urine specimen testing in 2009 to examine OTCs over a number of group testing algorithms. The overall observed chlamydia prevalence was 0.080 for females and 0.081 for males. We use these observed prevalences as our values for p when performing non-informative group testing by gender. To implement informative group testing, we used the beta distribution fits given by Black et al. (2015) for the individual probabilities of being positive p_i and implement methods similar to those in Section 2.3.2. We limit our maximum group sizes to be 20 due to large group sizes not being used in chlamydia testing (Mund et al., 2008). The NPHL provided assay sensitivities of $S_e = 0.805$ and $S_e = 0.93$ and specificities of $S_p = 0.96$ and $S_p = 0.95$ for females and males, respectively. This assay had an unusually low sensitivity for female urine specimens, and the laboratory eventually switched after that year to the Aptima Combo 2 Assay which has a much higher sensitivity ($S_e = 0.947$) (Food and Drug Administration, 2018). However, to be consistent with how the actual tests were performed, we use the accuracies for the BD assay. Table 2.6 provides the OTCs for non-informative and informative group testing algorithms. Overall, the table shows that O_{ET} and O_{MAR} lead to the same OTCs for all non-informative group testing algorithms considered. While differences do exist for females when using informative hierarchical testing algorithms, these small differences likely would not be of practical importance.

2.5. Conclusion

We have shown that the choice between O_{ET} and O_{MAR} most often does not change the OTC, and even when the OTC is different, there are not practical

Table 2.6: OTC summary for chlamydia testing. The test accuracies are $S_e = 0.805$ and $S_p = 0.96$ for females and $S_e = 0.93$ and $S_p = 0.95$ for males. For non-informative group testing, $p = 0.08$ for females, $p = 0.081$ for males, and equally sized groups are optimal at each stage (see Table 2.1's caption for how group sizes are denoted). For informative group testing, $P_i \sim \text{beta}(1.1, 11.54)$ for females, $P_i \sim \text{beta}(1.8, 18.20)$ for males, and full OTCs are provided in Appendix A. Results are not displayed for informative array testing with master pooling because no group testing algorithms have been proposed for it. Differences between O_{ET} and O_{MAR} are highlighted.

			Non-informative				Informative			
	Algorithm	Objective function	OTC	$E(T)/I$	PS_e	PS_p	Stage 1 size	$E(T)/I$	PS_e^W	PS_p^W
Female	Two-stage hierarchical	O_{ET}	5-1	0.5008	0.6480	0.9897	-	0.4757	0.6480	0.9901
		O_{MAR}	5-1	0.5008	0.6480	0.9897	-	0.4761	0.6480	0.9910
	Three-stage hierarchical	O_{ET}	12-4-1	0.4099	0.5217	0.9937	19	0.4102	0.5217	0.9930
		O_{MAR}	12-4-1	0.4099	0.5217	0.9937	14	0.4113	0.5479	0.9933
	Array w/o master pooling	O_{ET}	9-1	0.4327	0.5240	0.9931	10	0.4187	0.5226	0.9929
		O_{MAR}	9-1	0.4327	0.5240	0.9931	10	0.4187	0.5226	0.9929
	Array w/ master pooling	O_{ET}	81-9-1	0.3485	0.4218	0.9945	-	-	-	-
		O_{MAR}	81-9-1	0.3485	0.4218	0.9945	-	-	-	-
Male	Two-stage hierarchical	O_{ET}	4-1	0.5523	0.8649	0.9877	-	0.5462	0.8649	0.9867
		O_{MAR}	4-1	0.5523	0.8649	0.9877	-	0.5462	0.8649	0.9867
	Three-stage hierarchical	O_{ET}	9-3-1	0.4931	0.8044	0.9924	8	0.5081	0.8206	0.9905
		O_{MAR}	9-3-1	0.4931	0.8044	0.9924	8	0.5081	0.8206	0.9905
	Array w/o master pooling	O_{ET}	8-1	0.5015	0.8056	0.9901	8	0.5105	0.8053	0.9900
		O_{MAR}	8-1	0.5015	0.8056	0.9901	8	0.5105	0.8053	0.9900
	Array w/ master pooling	O_{ET}	64-8-1	0.4667	0.7492	0.9908	-	-	-	-
		O_{MAR}	64-8-1	0.4667	0.7492	0.9908	-	-	-	-

differences in the operating characteristics. Therefore, our work helps to close the case on the recent controversy regarding objective functions: both can be used in practice because they lead to very similar results. Some individuals may prefer to state that they used O_{MAR} because it directly takes into account accuracy at the beginning of the process. However, we tend to favor the traditionally used O_{ET} for one main reason. Simply, laboratories need to know the number of tests to be expected and the corresponding costs involved. In many instances, the expected costs are directly proportional to the expected number of tests. While the expected number of tests could also be stated when using O_{MAR}, this seems to be an unnecessary extra step, especially for laboratory directors and technicians who choose the OTC.

It is important to emphasize that laboratories would not use O_{ET} without still looking at accuracy. Rather than incorporating accuracy within the objective function, they would find the OTC and then examine the accuracy associated with it. If the accuracy resulting from O_{ET} (or O_{MAR}) was unsatisfactory, a new sub-optimal testing configuration would be chosen with accuracies that are acceptable. To help laboratories and those performing research in this area, we make available a set of R functions in the `binGroup` package that can be used to find the OTC or other suitable testing configurations by using O_{ET} and O_{MAR}. Examples of how to use these functions are available on our research website at www.chrisbilder.com/grouptesting and in Appendix A.

Our evaluations of O_{ET} and O_{MAR} focus on realistic settings for infectious disease detection when group testing would be used. Thus, we focus on values of S_e and S_p close to 1 and small values of p. When smaller values of S_e and S_p and/or larger values of p are used, there can be differences in the OTCs

and associated accuracy measures. For example, when $S_e = 0.75$, $S_p = 0.80$, and $p = 0.10$ for three-stage hierarchical testing, the OTC for O_{ET} has $I = 15$ and second-stage group sizes of 5 for each sub-group. For these same settings, the OTC for O_{MAR} has $I = 12$ and second-stage group sizes of 4 for each sub-group. However, the pooling sensitivity is only $PS_e = 0.42$ for both testing configurations, which makes the use of group testing unrealistic for this situation.

Laboratories may need to limit the particular values of I for which the OTC is searched over, similar to what we did in Section 2.4 for the chlamydia testing example. This may be due to physical constraints, such as a maximum group size that can be incorporated into an automated pooling platform. Also, this limit may be due to what is known as the "dilution effect" in group testing. Because specimens are pooled together, each individual specimen becomes a smaller part of the whole as the group size increases. This reduced portion can make it more difficult for an assay to identify its target, which in turn lowers its sensitivity. Laboratories may need to place an upper limit on I in this type of situation. Properly calibrated tests are needed whenever group testing is used to make sure the dilution effect does not become a problem. Fortunately, the dilution effect is now much less likely to occur due to modern nucleic acid amplification testing methods.

There are other objective functions that could be used. For example, Malinovsky et al. (2016) considered maximizing $E(C/T)$, but concluded this to be inferior to O_{MAR}. Therefore, we focused only on their O_{MAR} proposal in our paper. Objective functions can include penalties for making classification errors. For example, Graff and Roeloffs (1972) proposed using an objective function that is a linear combination of the expected number of tests, the

number of misclassified negatives, and the number of misclassified positives. Subjectively chosen weights are used with the misclassification measures to increase or decrease their importance. As would be expected, there will be weights then that result in an OTC which is quite different than what would be obtained from using O_{ET} and O_{MAR}. We provide examples in Appendix A illustrating these differences. However, the subjectiveness of these weights can depend on the infectious disease, the laboratory, or even particular individuals at a laboratory. Therefore, for general applications and research settings, it is difficult to use this or similar types of objective functions. We say this by no means to diminish the importance of taking into account the misclassification type. Because of its importance for specific applications, we provide tools in our `binGroup` package to find the OTC in those situations when this type of control is necessary. Examples are provided again on our research website and in Appendix A.

Acknowledgements

This research was supported by Grant R01 AI121351 from the National Institutes of Health. The authors thank their colleagues at the CDC, Innovative Blood Resources, the Nebraska Public Health Laboratory, the Nebraska Veterinary Diagnostic Center, and the State Hygienic Laboratory at the University of Iowa for their discussions and collaboration to make infectious disease testing more efficient. The authors also thank the referees for helping to improve this paper.

Supporting Information

Additional supporting information may be found in Appendix A.

Chapter 3

binGroup2: Identification and estimation using group testing

3.1. Introduction

As discussed in Chapter 1, the two primary objectives of group testing research are estimation and identification. The `binGroup` package (Bilder et al., 2010) was created to provide researchers and practitioners functions to use for estimation. Because functions for the identification problem did not exist in the `binGroup` package, a primary purpose of this dissertation was to make these types of functions available as well.

Over time, a number of additions have been made to `binGroup` by multiple researchers (including myself). These additions resulted in many different styles among functions in the package, making use of the package more difficult than necessary. For this reason, the `binGroup2` package was created to align the `binGroup` functions in style and format and to incorporate new functions for the group testing identification problem. The identification functions in `binGroup2` include those that were written for the research in Chapter 2 and to incorporate new mathematical derivations that will be presented in Appendix B. These functions focus on calculating operating characteristics

(e.g., expected number of tests) and finding the optimal testing configuration (OTC) for commonly used hierarchical and array-based group testing algorithms. Minor modifications have also been made to repackage and improve documentation for most of the estimation functions in `binGroup` to include in `binGroup2`. Information on the original `binGroup` package is available in Bilder et al. (2010).

In this chapter, we show how to use four main identification functions in the `binGroup2` package. The `operatingCharacteristics1()` (`opChar1()`) function calculates operating characteristics for a specified testing configuration with a single-disease assay. The `OTC1()` function calculates the operating characteristics and finds the OTC over a range of possible initial group sizes and/or testing configurations with a single-disease assay. The `operatingCharacteristics2()` (`opChar2()`) and `OTC2()` functions provide the same calculations as their counterparts with a multiplex assay that tests for two diseases. All four of these functions perform calculations for a number of group testing algorithms. Appendix C contains the R documentation for these functions and for the `binGroup2` package.

In addition, we will briefly summarize the estimation functions included in `binGroup2` and provide tables illustrating the mapping of R functions from `binGroup` to `binGroup2`. All of the functions written for this dissertation and the supporting functions written for other papers (see Section 3.4) are included in the `binGroup2` package. This package is available from the Comprehensive R Archive Network (CRAN).

3.2. Operating characteristics for group testing algorithms

In Chapter 2, we examined operating characteristics for seven different group testing algorithms, including hierarchical and array testing algorithms for both homogeneous and heterogeneous populations. We have written an R function to calculate operating characteristics for these group testing algorithms using a single-disease assay. Also, we have written an analogous R function for multiplex assays. We discuss both of these functions next.

3.2.1. Single-disease assays

The `operatingCharacteristics1()` (`opChar1()`) function calculates operating characteristics for group testing algorithms using a single-disease assay. Suppose a laboratory wants to implement the simplest form of group testing, two-stage hierarchical testing, to detect a single disease among a continuous stream of specimens that arrive daily. The `opChar1()` function can be used to determine the expected number of tests and corresponding diagnostic accuracy for a specific initial group size.

To illustrate `opChar1()` in a simple setting, suppose a group size of 5 is used in a situation with a disease prevalence of 0.05. Note that this is the prevalence and group size used by the Nebraska Public Health Laboratory in March 2020 for its calculations when planning to test for SARS-CoV-2, the virus that causes COVID-19, via group testing (Nebraska Department of Health and Human Services, 2020; Abdalhamid et al., 2020; Bilder et al., 2020). If we assume a sensitivity of 0.99 and a specificity of 0.99 (good estimates are not available as of March 2020), the `opChar1()` function can be used the following way:

```
> library(binGroup2)
> # Example 1 - non-informative two-stage hierarchical testing
> config.mat1 <- matrix(data = c(rep(1, 5), 1:5), nrow = 2,
    ncol = 5, byrow = TRUE, dimnames = list(Stage = 1:2,
    Individual = 1:5))
> config.mat1
     Individual
Stage 1 2 3 4 5
    1 1 1 1 1 1
    2 1 2 3 4 5
> results1 <- opChar1(algorithm = "D2", p = 0.05, Se =
    rep(0.99, 2), Sp = rep(0.99, 2), hier.config = config.mat1)

  Number of minutes running: 0

> names(results1)
[1] "algorithm" "prob"      "Se"        "Sp"        "Config"
[6] "p.vec"     "ET"        "value"     "Accuracy"
> results1$ET   # One component of the returned list
[1] 2.158473
> names(results1$Config)
[1] "Stage1"
> names(results1$Accuracy)
[1] "Individual" "Overall"

> summary(results1)

Algorithm: Non-informative two-stage hierarchical testing

Testing configuration:
Stage 1: 5

Expected number of tests: 2.16
Expected number of tests per individual: 0.4317

Accuracy for individuals:
     PSe    PSp   PPPV   PNPV Individuals
1 0.9801 0.9981 0.9642 0.9990         All

Overall accuracy of the algorithm:
     PSe    PSp   PPPV   PNPV
1 0.9801 0.9981 0.9642 0.9990

PSe denotes the pooling sensitivity.
PSp denotes the pooling specificity.
PPPV denotes the pooling positive predictive value.
PNPV denotes the pooling negative predictive value.
```

We first define a matrix, `config.mat1`, that specifies the full testing configuration for the hierarchical testing algorithm. This matrix, known as a group membership matrix (Bilder et al., 2019), is provided in the `hier.config` argument of `op.Char1()`. The rows correspond to the stages of testing, the columns correspond to each individual to be tested, and the cell values specify the group number of each individual at each stage. For example, row 1 of the matrix shows that each individual is being tested as one group in stage 1, and row 2 of the matrix shows that each individual is tested separately in stage 2. While row and column names are used here for readability, these are not

required. For array testing algorithms, the row/column size is provided in the `rowcol.sz` argument of `op.Char1()` and the `hier.config` argument is not used. Only square arrays are considered for array testing algorithms because that is how array testing is most often applied.

Within the `opChar1()` function, we also specify the desired group testing algorithm. While we specified `algorithm = "D2"` here for non-informative two-stage hierarchical testing, other options include hierarchical testing up to four stages and array testing with and without master pooling. The overall prevalence of disease is specified with the `p` argument. The sensitivity and specificity of the diagnostic test are specified by the `Se` and `Sp` arguments, respectively. Their values are given in vectors, where one value is stated for each stage of testing (in order). The values in Example 1 are specified as (stage 1, stage 2). As another example, values are specified as (stage 1, stage 2, stage 3) for three-stage hierarchical testing or (master group testing, row/column testing, individual testing) for array testing with master pooling. If a single value is provided, sensitivity/specificity values are assumed to be equal for all stages of testing.

The list of results for `opChar1()` includes the algorithm (`algorithm`), overall probability of disease (`prob`), sensitivity (`Se`), specificity (`Sp`), and testing configuration (`Config`) provided by the user. The `Config` result provides group sizes for each stage of hierarchical testing algorithms or the row/column size and array size for array testing algorithms. The vector of individual probabilities (all of which are equal in Example 1) is provided by `p.vec`. Operating characteristics calculated are the expected number of tests (`ET`), expected number of tests per individual (`value`), and accuracy measures (`Accuracy`) for individuals and the overall algorithm.

Accuracy measures include the pooling sensitivity, pooling specificity, pooling positive predictive value, and pooling negative predictive value. Individual accuracy measures (`Accuracy$Individual`) are calculated for each individual specified (by number) in the optional `a` argument of `op.Char1()`. The `a` argument is not provided here, so individual accuracy measures for all individuals in the algorithm are calculated. Individual accuracy measures are displayed in a matrix, where each row is a unique set of individual accuracy measures. The "`Individuals`" column lists which individuals share those measures. In Example 1, the "`Individuals`" column displays "`All`" instead of listing every individual in the algorithm because the accuracy measures for all individuals are equal. Overall accuracy measures (`Accuracy$Overall`) for the algorithm are displayed as well. These overall measures are weighted averages of the corresponding individual accuracy measures for all individuals in the algorithm. Expressions for these averages are provided in Appendix A.

The `summary.opChar()` method function provides a concise summary for objects of class "opChar" returned by `opChar1()`. Information displayed includes the specified testing configuration, expected number of tests, expected number of tests per individual, individual accuracy measures, and overall accuracy measures of the algorithm. Note that R is referred to as an object-oriented programming language. This is because generic functions, like `summary()`, can produce different results due to the class of the object used with it. R checks an object's class when a generic function is run and looks for a method function with the name format `<generic function>.<class name>`. The `summary.opChar()` function is the method function called when `summary()` is used with an object of class "opChar".

Additional features of `opChar1()` can be demonstrated with an informative

group testing example. Suppose that, as is the case in most real-life applications, the probability of disease is not the same for all individuals being tested. For example, the Nebraska Public Health Laboratory would like to categorize specimens tested for SARS-CoV-2 as either high risk (high probability) or low risk (low probability) to maximize the potential benefit from group testing. To illustrate this approach, we will again use a two-stage hierarchical algorithm, but this time with a heterogeneous probability vector. Informative two-stage hierarchical testing is implemented in `binGroup2` via the pool-specific optimal Dorfman (PSOD) method described in McMahan et al. (2012a). Suppose a block size of 25 represents the total number of specimens to be tested in a laboratory in a single day. The first stage of the algorithm involves testing specimens in five groups of five individuals each. We will again assume a sensitivity of 0.99 and a specificity of 0.99. The `opChar1()` function can be used the following way:

```
> # Example 2 - informative two-stage hierarchical testing
> config.mat2 <- matrix(data = c(rep(x = 1:5, each = 5),
    1:25), nrow = 2, ncol = 25, byrow = TRUE, dimnames =
    list(Stage = 1:2, Individual = 1:25))
> set.seed(1002)
> p.vec <- expectOrderBeta(p = 0.05, alpha = 0.1, grp.sz = 25)
> p.vec
 [1] 1.709007e-11 5.071419e-10 7.221785e-09 6.573282e-08
 [5] 4.299270e-07 2.155123e-06 8.632776e-06 2.849657e-05
 [9] 7.953888e-05 1.923566e-04 4.133953e-04 8.108784e-04
[13] 1.490215e-03 2.622304e-03 4.534883e-03 7.483465e-03
[17] 1.224643e-02 1.966959e-02 3.108131e-02 4.847474e-02
[21] 7.493288e-02 1.154149e-01 1.785798e-01 2.817468e-01
[25] 4.702845e-01

> results2 <- opChar1(algorithm = "ID2", probabilities =
    p.vec, Se = 0.99, Sp = 0.99, hier.config = config.mat2, a
    = 1:5, print.time = FALSE)
> names(results2)
 [1] "algorithm" "prob"      "alpha"     "Se"
 [5] "Sp"        "Config"    "p.vec"     "ET"
 [9] "value"     "Accuracy"
> results2$alpha
[1] NA

> summary(results2)

Algorithm: Informative two-stage hierarchical testing

Testing configuration:
Block size: 25
Group sizes: 5,5,5,5,5

Expected number of tests: 9.50
```

```
Expected number of tests per individual: 0.3802

Accuracy for individuals:
      PSe    PSp   PPPV   PNPV Individuals
1  0.9801 0.9999 0.0000 1.0000           1
2  0.9801 0.9999 0.0000 1.0000           2
3  0.9801 0.9999 0.0001 1.0000           3
4  0.9801 0.9999 0.0006 1.0000           4
5  0.9801 0.9999 0.0042 1.0000           5

Overall accuracy of the algorithm:
     PSe    PSp   PPPV   PNPV
1 0.9801 0.9986 0.9740 0.9990

PSe denotes the pooling sensitivity.
PSp denotes the pooling specificity.
PPPV denotes the pooling positive predictive value.
PNPV denotes the pooling negative predictive value.
```

The `config.mat2` matrix provides the testing configuration and is specified in the `hier.config` argument of `opChar1()`.

The `expectOrderBeta()` function creates a heterogeneous vector of individual probabilities that are the expected value of order statistics from a beta distribution. This function is based on the `beta.dist()` function written for Black et al. (2015), where we discuss our additions/changes to it shortly. The `p` argument represents the overall disease prevalence. In the context of a beta distribution, it is the expected value of a random variable with this distribution, $p = \alpha/(\alpha + \beta)$, where $\alpha > 0$ and $\beta > 0$ are shape parameters (Casella and Berger, 2002). The `alpha` argument represents the α shape parameter for a beta distribution, and it specifies the degree of heterogeneity in the probability vector. Higher values of α correspond to lower levels of heterogeneity among the generated individual probabilities. The number of probabilities desired is given by the `grp.sz` argument.

Depending on the context that the function is used in, `grp.sz` may represent the initial group size for a group testing algorithm. Depending on the specified p, α, and number of desired probabilities, simulation may be necessary to determine the probabilities. For this reason, `expectOrderBeta()` augments `beta.dist()` by checking whether simulation is necessary. An op-

tional `num.sim` argument specifying the number of simulations can be passed to `expectOrderBeta()` through the `...` argument in `opChar1()`. While simulation is not needed for Example 2, setting a seed is good practice and ensures reproducibility of results in informative settings.

The sensitivity and specificity provided in Example 2 are single values rather than vectors, meaning that the sensitivity values are equal, and the specificity values are equal for all stages of testing. Example 2 utilizes the optional `a` argument to calculate individual accuracy measures only for the first 5 individuals. Because none of these individuals have equal accuracy measures, measures for all 5 individuals are displayed in the output. The `a` argument can also be used to specify a list of non-consecutive individuals. The `print.time` argument is also optional and determines whether the length of time for calculations to complete is printed. In Example 2, we have decided not to print the time elapsed during calculations.

The components within the returned list of `opChar1()` are the same for non-informative settings, but now with the addition of `alpha` for this informative group testing example. In Example 2, we generated a vector of individual probabilities outside the `opChar1()` function and specified it using the `probabilities` argument. Because we did not use the optional `alpha` argument, `alpha` in the list of results does not have a value. Another way to generate individual risk probabilities is to do so inside the `opChar1()` function by specifying an overall disease prevalence `p` and a shape parameter `alpha` for the beta distribution. By using the same random seed value, $p = 0.05$, and $\alpha = 0.1$, we can produce the same results as in Example 2.

```
> set.seed(1002)
> results2a <- opChar1(algorithm = "ID2", p = 0.05, alpha =
    0.1, Se = 0.99, Sp = 0.99, hier.config = config.mat2, a =
    1:5, print.time = FALSE)
```

```
> results2a$p.vec
 [1] 1.709007e-11 5.071419e-10 7.221785e-09 6.573282e-08
 [5] 4.299270e-07 2.155123e-06 8.632776e-06 2.849657e-05
 [9] 7.953888e-05 1.923566e-04 4.133953e-04 8.108784e-04
[13] 1.490215e-03 2.622304e-03 4.534883e-03 7.483465e-03
[17] 1.224643e-02 1.966959e-02 3.108131e-02 4.847474e-02
[21] 7.493288e-02 1.154149e-01 1.785798e-01 2.817468e-01
[25] 4.702845e-01

> summary(results2a)

  << OUTPUT EDITED >>

Expected number of tests: 9.50
Expected number of tests per individual: 0.3802

Accuracy for individuals:
     PSe    PSp   PPPV   PNPV Individuals
1 0.9801 0.9999 0.0000 1.0000           1
2 0.9801 0.9999 0.0000 1.0000           2
3 0.9801 0.9999 0.0001 1.0000           3
4 0.9801 0.9999 0.0006 1.0000           4
5 0.9801 0.9999 0.0042 1.0000           5

Overall accuracy of the algorithm:
     PSe    PSp   PPPV   PNPV
1 0.9801 0.9986 0.9740 0.9990

  << OUTPUT EDITED >>
```

The vector of individual probabilities (`p.vec`) in the results is the same as the vector we generated using `expectOrderBeta()` outside the `opChar1()` function.

3.2.2. Multiplex assays

Multiplex assays test for multiple diseases in a single application. If a group tests negative for both diseases, all members of the group are declared negative. If a group tests positive for at least one disease, retesting occurs according to the group testing algorithm. With multiplex assays, we have a set of joint probabilities of disease, one for each set of potential binary outcomes for the diseases. For this dissertation, we focus only on the two disease case. In this situation, there are four joint probabilities to consider: p_{00}, the probability that an individual tests negative for both diseases; p_{10}, the probability that an individual tests positive for only the first disease; p_{01}, the probability that an individual tests positive for only the second disease; and p_{11}, the probability

that an individual tests positive for both diseases. We use the ordering of $(p_{00}, p_{10}, p_{01}, p_{11})$ throughout our discussion.

The `operatingCharacteristics2()` (`opChar2()`) function calculates operating characteristics for group testing algorithms with a multiplex assay that tests for two diseases. Calculations for hierarchical group testing algorithms are performed as described in Bilder et al. (2019) and calculations for array-based group testing algorithms are performed as described in Hou et al. (2020). The required arguments are generally the same as for `opChar1()`, where we describe exceptions during our discussion as needed. We show below how to calculate operating characteristics for non-informative five-stage hierarchical testing.

```
> # Example 3 - non-informative five-stage
> # hierarchical testing
> config.mat3 <- matrix(data = c(rep(1, 20), rep(1, 10),
    rep(2, 10), rep(c(1, 2, 3, 4), each = 5), rep(1, 3),
    rep(2, 2), rep(3, 3), rep(4, 2), rep(5, 3), rep(6, 2),
    rep(7, 3), 8, 9, 1:18, NA, NA), nrow = 5, ncol = 20, byrow
    = TRUE, dimnames = list(Stage = 1:5, Individual = 1:20))
> config.mat3
      Individual
Stage 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
    1 1 1 1 1 1 1 1 1 1  1  1  1  1  1  1  1  1  1  1
    2 1 1 1 1 1 1 1 1 1  1  2  2  2  2  2  2  2  2  2
    3 1 1 1 1 1 2 2 2 2  2  3  3  3  3  3  4  4  4  4
    4 1 1 1 2 2 3 3 3 4  4  5  5  5  6  6  7  7  7  8  9
    5 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 NA NA
> results3 <- opChar2(algorithm = "D5", p.vec = c(0.95, 0.02,
    0.02, 0.01), Se = c(0.96, 0.98), Sp = c(0.99, 0.99),
    hier.config = config.mat3, ordering = matrix(data = c(0,
    1, 0, 1, 0, 0, 1, 1), nrow = 4, ncol = 2), print.time =
    FALSE)

> names(results3)
[1] "algorithm" "prob.vec"  "Se"         "Sp"         "Config"
[6] "p.mat"     "ET"        "value"      "Accuracy"
> names(results3$Config)
[1] "Stage1" "Stage2" "Stage3" "Stage4"
> names(results3$Accuracy)
[1] "Disease 1 Individual" "Disease 2 Individual" "Overall"

> results3$Se
        Stage
Disease    1    2    3    4    5
      1 0.96 0.96 0.96 0.96 0.96
      2 0.98 0.98 0.98 0.98 0.98
> results3$Sp
        Stage
Disease    1    2    3    4    5
      1 0.99 0.99 0.99 0.99 0.99
      2 0.99 0.99 0.99 0.99 0.99

> summary(results3)
```

```
Algorithm: Non-informative five-stage hierarchical testing

Testing configuration:
Stage 1: 20
Stage 2: 10,10
Stage 3: 5,5,5,5
Stage 4: 3,2,3,2,3,2,3,1,1

Expected number of tests: 7.96
Expected number of tests per individual: 0.3978

Disease 1 accuracy for individuals:
     PSe    PSp   PPPV   PNPV                        Individuals
1 0.8828 0.9989 0.9615 0.9964 1,2,3,6,7,8,11,12,13,16,17,18
2 0.8821 0.9993 0.9760 0.9964                   4,5,9,10,14,15
3 0.9048 0.9981 0.9349 0.9971                           19,20

Disease 2 accuracy for individuals:
     PSe    PSp   PPPV   PNPV                        Individuals
1 0.9388 0.9989 0.9641 0.9981 1,2,3,6,7,8,11,12,13,16,17,18
2 0.9385 0.9993 0.9778 0.9981                   4,5,9,10,14,15
3 0.9507 0.9981 0.9381 0.9985                           19,20

Overall accuracy of the algorithm:
     PSe    PSp   PPPV   PNPV
1 0.8848 0.9989 0.9630 0.9964
2 0.9399 0.9990 0.9654 0.9981

PSe denotes the pooling sensitivity.
PSp denotes the pooling specificity.
PPPV denotes the pooling positive predictive value.
PNPV denotes the pooling negative predictive value.
```

The `config.mat3` matrix describes the testing configuration for the five-stage algorithm. Note the use of "`NA`" to indicate that the 19th and 20th individuals are not tested in the fifth stage because individual testing already occurred for these individuals in the fourth stage of the algorithm.

Because this is a non-informative setting, a vector of overall joint probabilities is specified using the `p.vec` argument. The `ordering` argument specifies the ordering for the binary responses of the diseases. The columns of the matrix correspond to each disease and the rows of the matrix correspond to each of the four sets of binary outcomes. Together with the joint probabilities provided in `p.vec`, this specifies $p_{00} = 0.95$, $p_{10} = 0.02$, $p_{01} = 0.02$, and $p_{11} = 0.01$.

Because a vector of two values is specified for the sensitivity/specificity, the sensitivity/specificity values for all stages are assumed to be equal. The first value in the vector is used for the first disease and the second value in

the vector is used for the second disease. The `opChar2()` function generates a sensitivity/specificity matrix of appropriate size based on the specified group testing algorithm, available in the `Se` and `Sp` results objects. Alternatively, the sensitivity/specificity values can be specified by matrices, where one value is given for each disease at each stage of testing. The rows of the matrix correspond to each disease and the columns of the matrix correspond to each stage of testing, $s = 1, ..., S$.

The same operating characteristics are calculated in `opChar2()` as in `opChar1()`, and individual accuracy measures are now calculated for each disease. The overall joint probabilities specified by the user are provided by `prob.vec` and the matrix of joint probabilities for each individual are provided by `p.mat`. Accuracy measures for each individual specified in the `a` argument are provided in `Accuracy$'Disease 1 Individual'` and `Accuracy$'Disease 2 Individual'`. Because we did not specify a list of individuals in the `a` argument in Example 3, accuracy measures for all individuals are displayed. The overall accuracy of the algorithm is displayed in a matrix of two rows. The first row corresponds to accuracy measures for the first disease and the second row corresponds to the accuracy measures for the second disease. The `summary.opChar()` function is compatible with `opChar2()` and provides the specified testing configuration, expected number of tests, expected number of tests per individual, individual accuracy measures for each disease, and overall accuracy measures for the algorithm.

In Example 3, we assumed that all 20 individuals had the same set of joint probabilities. For heterogeneous populations, users can specify a matrix of individual probabilities in one of two ways: 1) a vector of length four containing shape parameters for a Dirichlet distribution that leads to the simulation of the

necessary joint probabilities, or 2) a matrix containing the joint probabilities for each individual of interest. Consider a group of 5 individuals. We can specify a matrix of individual probabilities in the following way:

```
> joint.p <- matrix(data = c(0.90, 0.04, 0.04, 0.02, 0.92,
    0.03, 0.03, 0.02, 0.94, 0.03, 0.02, 0.01, 0.95, 0.02,
    0.02, 0.01, 0.96, 0.02, 0.01, 0.01), nrow = 4, ncol = 5,
    byrow = FALSE)
> rownames(joint.p) <- c("00", "10", "01", "11")
> colnames(joint.p) <- 1:5
> joint.p
      1    2    3    4    5
00 0.90 0.92 0.94 0.95 0.96
10 0.04 0.03 0.03 0.02 0.02
01 0.04 0.03 0.02 0.02 0.01
11 0.02 0.02 0.01 0.01 0.01
```

The rows of the matrix correspond to the four joint probabilities and the columns correspond to each individual in the algorithm. This matrix can be specified in the `probabilities` argument of `opChar2()`. Alternatively, we can use the `alpha` argument in `opChar2()` to specify a vector of shape parameters for a Dirichlet distribution. Using the same testing configuration and diagnostic accuracy as in Example 3, we can calculate operating characteristics for an informative setting in the following way:

```
> results3a <- opChar2(algorithm = "ID5", alpha = c(18.25,
    0.75, 0.75, 0.25), Se = c(0.96, 0.98), Sp = c(0.99, 0.99),
    hier.config = config.mat3)
> results3a$alpha.vec
[1] 18.25  0.75  0.75  0.25
> results3a$p.mat
             1           2           3          4           5
00 0.9818650992 0.975220605 0.957523560 0.95518040 0.941661877
10 0.0107148978 0.002383102 0.035062204 0.01020303 0.002262310
01 0.0070155163 0.020264074 0.006309220 0.01744923 0.046596268
11 0.0004044867 0.002132218 0.001105015 0.01716733 0.009479545
  << OUTPUT EDITED >>
            16          17          18           19           20
00 0.87651645 0.866355783 0.85353355 0.7804062963 0.7421553006
10 0.06069141 0.105318444 0.10013884 0.0052134818 0.0319714149
01 0.01564643 0.021725148 0.01673743 0.2140780373 0.2249153986
11 0.04714571 0.006600624 0.02959018 0.0003021846 0.0009578858
```

Additional details on the use of the Dirichlet distribution for this purpose are provided in Bilder et al. (2019). The user-specified vector of shape parameters

is provided in `alpha.vec` of the results. The matrix of joint probabilities for each individual are provided in `p.mat`.

Additional examples using the `opChar1()` and `opChar2()` functions (e.g., array testing) are provided in the R documentation for the functions in `binGroup2`. This documentation is also provided in Appendix C of this dissertation.

3.3. Optimal testing configurations for group testing algorithms

In Chapter 2, we discussed the importance of choosing group sizes when implementing group testing. In practice, group sizes are chosen by minimizing an objective function, usually the expected number of tests per individual. The set of group sizes corresponding to the minimum value of the objective function is the optimal testing configuration (OTC). Chapter 2 examined several different objective functions and provided comparisons for seven different group testing algorithms, including hierarchical and array testing algorithms for both homogeneous and heterogeneous populations. We have written an R function to find the optimal testing configuration over every possible configuration for those group testing algorithms using a single-disease assay. We have also written a comparable R function for multiplex assays. We discuss both of these functions next.

3.3.1. Single-disease assays

The `OTC1()` function (a.k.a., the `OTC()` function in Chapter 2 for `binGroup`) finds the OTC for group testing algorithms with a single-disease assay and computes the associated operating characteristics, as described in Chapter 2. The required arguments are the same as for `opChar1()` with two exceptions: 1)

initial group sizes are specified instead of a specific testing configuration and 2) a list of objective functions is provided for finding the OTC. An initial group size or range of initial group sizes (or row/column sizes) is specified using the `group.sz` argument. The `obj.fn` argument specifies a list of objective functions which are minimized to find the OTC.

Options for the objective functions include the expected number of tests per individual; the expected number of tests divided by the expected number of correct classifications (Malinovsky et al., 2016); and a linear combination of the expected number of tests, the number of misclassified negatives, and the number of misclassified positives (Graff and Roeloffs, 1972). These objective functions are referred to as `ET`, `MAR`, and `GR`, respectively, where the latter two objective functions are termed for the authors of the papers in which the objective functions were presented. The `GR` objective function requires a matrix of weights, specified in the `weights` argument, for the number of misclassified negatives and misclassified positives. The rows of the matrix correspond to each set of weights and up to six sets of weights are allowed. The available group testing algorithms include up to three-stage hierarchical testing and array testing with and without master pooling.

We show below how to use the `OTC1()` function to find the OTC for an informative three-stage hierarchical group testing algorithm with sensitivity and specificity that vary across stages of testing and individual probabilities that are the expected values of order statistics from a beta(0.5, 49.5) distribution. Note that random variables from this beta distribution have an expected value of $E(P_i) = 0.01$. Just as in the `opChar1()` function, we can alternatively specify a vector of individual probabilities using the `probabilities` argument.

```
> # Example 4 - OTC for informative three-stage
> # hierarchical testing
```

```
> set.seed(1234)
> results4 <- OTC1(algorithm = "ID3", p = 0.01, alpha = 0.5,
    Se = c(0.95, 0.95, 0.98), Sp = c(0.96, 0.96, 0.99),
    group.sz = 3:30, obj.fn = c("ET", "MAR", "GR"), weights =
    matrix(data = c(1, 1, 10, 10, 1, 10), nrow = 3, ncol = 2,
    byrow = TRUE))
Initial Group Size = 3
Initial Group Size = 4

 << OUTPUT EDITED >>

Initial Group Size = 29
Initial Group Size = 30

  Number of minutes running: 0.34

> names(results4)
 [1] "algorithm"   "prob"        "alpha"       "Se"
 [5] "Sp"          "opt.ET"      "opt.MAR"     "opt.GR1"
 [9] "opt.GR2"     "opt.GR3"     "Configs"     "Top.Configs"
> names(results4$opt.ET)
[1] "OTC"      "p.vec"    "ET"       "value"    "Accuracy"
> names(results4$opt.ET$OTC)
[1] "Stage1" "Stage2"

> results4$Configs
    I   config     ET  value    PSe    PSp   PPPV   PNPV
1  28 13,7,5,3 3.5120 0.1254 0.8844 0.9996 0.9532 0.9988
2  29 14,7,5,3 3.6378 0.1254 0.8844 0.9995 0.9516 0.9988
3  30 15,7,5,3 3.7692 0.1256 0.8844 0.9995 0.9499 0.9988
4  27 13,7,4,3 3.3934 0.1257 0.8844 0.9996 0.9549 0.9988
5  26 13,6,4,3 3.2710 0.1258 0.8844 0.9996 0.9567 0.9988

  << OUTPUT EDITED >>

24  7      5,2 1.4067 0.2010 0.8844 0.9998 0.9780 0.9988
25  6      5,1 1.3339 0.2223 0.8935 0.9994 0.9339 0.9989
26  5      4,1 1.2604 0.2521 0.8944 0.9994 0.9362 0.9989
27  4      3,1 1.2011 0.3003 0.8955 0.9994 0.9358 0.9989
28  3      2,1 1.1553 0.3851 0.8970 0.9993 0.9303 0.9990

> head(results4$Top.Configs)
   I   config     ET  value    PSe    PSp   PPPV   PNPV
1 28 13,7,5,3 3.5120 0.1254 0.8844 0.9996 0.9532 0.9988
2 29 14,7,5,3 3.6378 0.1254 0.8844 0.9995 0.9516 0.9988
3 28 14,7,4,3 3.5132 0.1255 0.8844 0.9996 0.9531 0.9988
4 28 14,6,5,3 3.5146 0.1255 0.8844 0.9996 0.9531 0.9988
5 30 15,7,5,3 3.7692 0.1256 0.8844 0.9995 0.9499 0.9988
6 27 13,7,4,3 3.3934 0.1257 0.8844 0.9996 0.9549 0.9988

> summary(results4)

Algorithm: Informative three-stage hierarchical testing

Optimal testing configuration:
    Stage 1  Stage 2
ET       28 13,7,5,3
MAR      28 13,7,5,3
GR1      28 13,7,5,3
GR2      23 11,5,4,3
GR3      28 13,7,5,3

Expected number of tests:
    E(T)  Value
ET  3.51 0.1254
MAR 3.51 0.1256
GR1 3.51 0.1270
GR2 2.91 0.1412
GR3 3.51 0.1373

E(T) denotes the expected number of tests.
Value denotes the objective function value per individual.

Overall accuracy of the algorithm:
       PSe    PSp   PPPV   PNPV
ET  0.8844 0.9996 0.9532 0.9988
MAR 0.8844 0.9996 0.9532 0.9988
GR1 0.8844 0.9996 0.9532 0.9988
```

```
GR2 0.8844 0.9996 0.9616 0.9989
GR3 0.8844 0.9996 0.9532 0.9988

PSe denotes the pooling sensitivity.
PSp denotes the pooling specificity.
PPPV denotes the pooling positive predictive value.
PNPV denotes the pooling negative predictive value.
```

Because we specified a range of group sizes ($I = 3, ..., 30$) over which to find the OTC, an appropriately-sized vector of individual probabilities is newly generated for each initial group size in the range using the `expectOrderBeta()` function described in Section 3.2.1. We compare OTCs for all three objective functions by specifying `obj.fn = c("ET","MAR","GR")`, where the GR objective function from Graff and Roeloffs (1972) uses three different sets of weights specified in the `weights` argument.

The results produced by the `OTC1()` function are similar to those produced by the `opChar1()` function, and include the algorithm, overall probability of disease, α shape parameter for the beta distribution (for informative testing settings only), sensitivity, and specificity as specified by the user. Results for each requested objective function (provided in `opt.ET`, for example) include the OTC (`OTC`) and the corresponding vector of individual probabilities (`p.vec`), expected number of tests (`ET`), objective function value per individual (`value`), and overall accuracy measures for the algorithm (`Accuracy`). Similar to the `Config` result produced by `opChar1()` and `opChar2()`, the `OTC` result details the full testing configuration (group sizes for each stage in hierarchical testing algorithms or row/column sizes for array testing algorithms).

Additionally, the `OTC1()` function provides results for some configurations other than the OTC. For algorithms where there is only one configuration for each initial group size (e.g., non-informative two-stage hierarchical or array testing), results for each initial group size are provided (`Configs`). For algorithms where there is more than one possible configuration for each initial

group size (e.g., informative two-stage hierarchical or three-stage hierarchical testing), two sets of configurations are provided: 1) the best configuration for each initial group size (`Configs`), and 2) the top 10 configurations for each initial group size (`Top.Configs`). The `summary.OTC()` function provides a concise summary for objects of class "OTC" returned by `OTC1()`. Information displayed includes the optimal testing configuration, expected number of tests, objective function value per individual, and overall accuracy measures of the algorithm for each objective function.

Notice that the OTC for the `GR2` objective function (with weights of $D_1 = 10$ and $D_2 = 10$) differs from the others. The OTC for `GR2` has an initial group size of 23 with stage 2 group sizes of 11, 5, 4, and 3 and $2.91/23 = 0.1265$ expected tests per individual. In contrast, the OTC for all other objective functions is an initial group size of 28 with stage 2 group sizes of 13, 7, 5, and 3 and $3.51/28 = 0.1254$ expected tests per individual. The pooling predictive values are slightly higher for `GR2` than for the other objective functions. All other accuracy measures are the same as for other objective functions up to four decimal places.

The optional `trace` argument determines whether the progress of calculations should be printed for each initial group size provided by the user. The optional `print.time` argument is available for the `OTC1()` function as well.

3.3.2. Multiplex assays

The `OTC2()` function finds the OTC for group testing algorithms with a multiplex assay that tests for two diseases and computes the associated operating characteristics. Calculations are performed the same as in the `opChar2()` function. The required arguments are the same as for `OTC1()`, except that no list

of objective functions is provided. The OTC is found only for the expected number of tests per individual. This simplification was made due to the results described in Chapter 2. The joint probabilities are specified in the same way as for `opChar2()`. Available group testing algorithms include up to three-stage hierarchical testing and array testing with and without master pooling. We show below how to find the OTC for non-informative array testing with master pooling.

```
> # Example 5 - non-informative array testing
> # with master pooling
> Se <- matrix(data = rep(0.95, 6), nrow = 2, ncol = 3,
    dimnames = list(Infection = 1:2, Stage = 1:3))
> Sp <- matrix(data = rep(0.99, 6), nrow = 2, ncol = 3,
    dimnames = list(Infection = 1:2, Stage = 1:3))
> results2 <- OTC2(algorithm = "A2M", p.vec = c(0.90, 0.04,
    0.04, 0.02), Se = Se, Sp = Sp, ordering = matrix(data =
    c(0, 1, 0, 1, 0, 0, 1, 1), nrow = 4, ncol = 2), group.sz =
    3:20)
Row/Column Size=3, Array Size=9
Row/Column Size=4, Array Size=16

 << OUTPUT EDITED >>

Row/Column Size=19, Array Size=361
Row/Column Size=20, Array Size=400

 Number of minutes running: 0.15

> names(results5)
[1] "algorithm" "prob.vec"  "Se"        "Sp"
[5] "opt.ET"    "Configs"
> names(results5$opt.ET)
[1] "OTC"      "p.mat"    "ET"       "value"    "Accuracy"
> names(results5$opt.ET$OTC)
[1] "Array.dim" "Array.sz"

> summary(results5)

Algorithm: Non-informative array testing with master pooling

Optimal testing configuration:
   Row/column size Array size
ET               8         64

Expected number of tests:
    E(T)  Value
ET 34.77 0.5432

E(T) denotes the expected number of tests.
Value denotes the objective function value per individual.

Overall accuracy of the algorithm:
     PSe    PSp   PPPV   PNPV
1 0.8919 0.9976 0.9601 0.9931
2 0.8919 0.9976 0.9601 0.9931

PSe denotes the pooling sensitivity.
PSp denotes the pooling specificity.
PPPV denotes the pooling positive predictive value.
PNPV denotes the pooling negative predictive value.
```

The sensitivity and specificity values (specified in matrices) are equal for all stages of testing and all individuals share the same set of joint probabilities. For array testing algorithms, the `group.sz` argument specifies the row/column size for testing. All of the functions described in this chapter consider only square arrays for array testing algorithms, which is how array testing is usually applied. Because we have specified a range of row/column sizes over which to find the OTC, the matrix of joint probabilities for each individual is newly generated for each array so that the correct number of individuals are specified.

The same operating characteristics are calculated as in `OTC1()`. Because the only objective function option available is the expected number of tests per individual, all OTC results are provided in `opt.ET`. For array testing algorithms, the `OTC` results include the row/column size (`Array.dim`) and the array size (`Array.sz`). Similar configurations are provided as well. The `summary.OTC()` function is compatible with `OTC2()` and provides the optimal testing configuration, expected number of tests, expected number of tests per individual, and overall accuracy measures for the algorithm. Overall accuracy measures for each disease are displayed in a matrix, where each row corresponds to a disease.

3.4. Calculation details

Calculations of operating characteristics in `binGroup2` are performed by utilizing code written for several papers by other authors. New code was also written for array testing with master pooling. This section provides details on the mathematical expressions and R functions used for the group testing algorithms available in `OTC1()`, `OTC2()`, `opChar1()`, and `opChar2()`.

3.4.1. Single-disease assays

Operating characteristics for hierarchical testing algorithms, with the exception of informative two-stage hierarchical testing, are calculated based on expressions provided in Black et al. (2015). The `opChar1()` and `OTC1()` functions utilize the `hierarchical.desc()` function, which implements up to four-stage hierarchical testing and was written for the same paper. Informative two-stage hierarchical testing (Dorfman) is implemented via the pool-specific optimal Dorfman (PSOD) method described in McMahan et al. (2012a), where the greedy algorithm proposed for PSOD is replaced by considering all possible testing configurations. For this algorithm, the `opChar1()` and `OTC1()` functions utilize the `characteristics.pool()` and `accuracy.dorf()` functions written for McMahan et al. (2012a). The former function is used to calculate the expected number of tests and the latter function calculates the individual accuracy measures for the algorithm.

Operating characteristics for array testing without master pooling are calculated based on expressions provided in McMahan et al. (2012b). Informative array testing without master pooling is implemented using the gradient arrangement (the most efficient array design) for the matrix of individual risk probabilities, where higher-risk individuals are grouped in the leftmost columns of the array. The gradient arrangement is executed with the `informativeArrayProb()` function (originally `Informative.array.prob()`) and operating characteristics are calculated using the `Array.Measures()` function written for McMahan et al. (2012b). Operating characteristics for non-informative array testing with master pooling are calculated based on expressions provided in Kim et al. (2007) using the `MasterPool.Array.Measures()`

function written for Chapter 2.

All of the R functions and papers mentioned in this section make the assumption that the sensitivity/specificity values are equal for all stages of testing. In Appendix B, we present derivations that allow the sensitivity/specificity values to vary across stages of testing for all seven group testing algorithms in the `opChar1()` and `OTC1()` functions. The functions written for Black et al. (2015), McMahan et al. (2012a), McMahan et al. (2012b), and Chapter 2 have been revised to match the derivations presented in Appendix B and incorporated into the `binGroup2` package.

3.4.2. Multiplex assays

Operating characteristics for hierarchical testing algorithms with multiplex assays that test for two diseases are calculated by the `ET.all.stages.new()` and `PSePSpAllStages()` functions written for Bilder et al. (2019). These functions allow the sensitivity/specificity values to vary across stages of testing for hierarchical testing up to five stages. As a result, no new derivations were needed. Only minor modifications (e.g., formatting of returned values) were made to the functions from Bilder et al. (2019) for inclusion in the `binGroup2` package.

Operating characteristics for array testing algorithms with multiplex assays that test for two diseases are calculated based on expressions provided in Hou et al. (2020). While the derivations presented in that paper allow for the sensitivity/specificity values to vary across stages of testing, the corresponding `ARRAY()` function written for the paper does not. Revisions were made to the `ARRAY()` function to allow the sensitivity/specificity to vary across stages of testing. These revisions involved editing the C++ code that was used for

`ARRAY()`. No derivations for array testing algorithms with multiplex assays are presented in this dissertation.

3.5. Estimation functions

To provide a coherent structure and reconcile the diverse styles present in the `binGroup` package, most of the estimation functions were reorganized and their documentation revised for inclusion in the `binGroup2` package. The `propCI()` function integrates three functions from `binGroup` into one to calculate point estimates and confidence intervals for a single proportion when only group responses are observed. It provides a number of different methods for point estimation and confidence interval calculation, and it allows for both equal and unequal group sizes. The `propDiffCI()` function calculates point estimates and confidence intervals for a difference of proportions in a similar setting.

Additional information on methods for estimation of and inference on proportions for group testing algorithms is available in Schaarschmidt (2007), Biggerstaff (2008), and Tebbs and Bilder (2004). The expected width of a confidence interval can be calculated using the `gtWidth()` function. The `gtTest()` and `gtPower()` functions calculate the p-value and power associated with a hypothesis test, respectively. Minor modifications were made to these three functions for inclusion in `binGroup2`. The `designPower()` function combines two functions from `binGroup` that help determine the number of groups or group size needed to achieve a specified level of power. The `designEst()` function helps to find the optimal group size for a design and was included in `binGroup2` with only minor modifications. Additional information on determining the optimal group testing design is available in Schaarschmidt (2007).

The `gtSim()` function merges three separate simulation functions from

`binGroup`, simulating group testing data for simple pooling (two-stage hierarchical testing), halving (an S-stage algorithm where each group that tests positive is split into two subgroups), and array testing designs. The `gtReg()` function joins three distinct regression functions from `binGroup` and fits group testing regression models for simple pooling, halving, and array testing designs. Additional information on group testing simulation and regression functions is available in Zhang (2012). Print and summary method functions were combined, as necessary, so that a single print and/or summary method function exists for the estimation functions included in the `binGroup2` package.

3.6. Additional functions

In addition to the `opChar1()`, `opChar2()`, `OTC1()`, and `OTC2()` functions discussed in this chapter, we authored 32 functions to perform calculations for each of the group testing algorithms internally. For example, `NI.Dorf.calc1()` and `NI.Dorf.calc2()` perform calculations for non-informative two-stage hierarchical (Dorfman) testing to support the `opChar1()` and `opChar2()` functions. Similarly, `NI.Dorf.OTC1()` and `NI.Dorf.OTC2()` support the `OTC1()` and `OTC2()` functions by finding the OTC for non-informative Dorfman testing. Analogous functions provide support for the other group testing algorithms available in the `opChar1()`, `opChar2()`, `OTC1()`, and `OTC2()` functions.

Operating characteristics for the halving protocol can be calculated with the `halving()` function written for Black et al. (2012). The `Sterrett()` function written for Bilder et al. (2010) performs calculations for the unique Sterrett (1957) algorithm that was adapted for informative retesting by Bilder et al. (2010). Functions described by McMahan et al. (2012a) for informa-

Table 3.1: New functions for `binGroup2`.

binGroup2
`operatingCharacteristics1` (`opChar1`)
`operatingCharacteristics2` (`opChar2`)
`summary.opChar`
`OTC2`
`summary.OTC`
`Sterrett`

tive two-stage hierarchical testing implemented via methods other than PSOD were previously included in the `binGroup` package but are not included in the `binGroup2` package.

Table 3.1 provides a list of functions that did not exist in the `binGroup` package and are new to the `binGroup2` package. Table 3.2 illustrates the mapping of exported functions in the `binGroup` package to the `binGroup2` package. The mapping of internal functions is shown in Table 3.3. Table 3.4 provides a list of `binGroup` functions without direct counterparts in the `binGroup2` package.

Table 3.2: Mapping of R functions from `binGroup` to `binGroup2` (exported functions only).

binGroup	**binGroup2**
p.vec.func*	expectOrderBeta
Informative.array.prob*	informativeArrayProb
OTC*	OTC1
halving*	halving
sim.gt sim.halving sim.mp	gtSim
gtreg gtreg.halving gtreg.mp	gtReg
gt.control	gtRegControl
summary.gt summary.gt.mp	summary.gtReg
print.summary.gt print.summary.gt.mp	print.summary.gtReg
predict.gt	predict.gtReg
bgtCI bgtvs pooledBin	propCI
summary.poolbin print.bgtCI print.bgtvs print.poolbin	print.propCI
pooledBinDiff	propDiffCI
summary.poolbindiff print.poolbindiff	print.propDiffCI
bgtPower	gtPower
bgtTest	gtTest
print.bgtTest	print.gtTest
bgtWidth	gtWidth
estDesign	designEst
nDesign sDesign	designPower
print.nDesign print.sDesign	print.designPower

*denotes functions that were incorporated into the `binGroup` package by myself before the decision to create `binGroup2` was made.

Table 3.3: Mapping of R functions from `binGroup` to `binGroup2` (internal/hidden functions).

binGroup	**binGroup2**
`beta.dist`* (called by `p.vec.func`)	`beta.dist` / `beta.dist2` (called by `expectOrderBeta`)
`NI.Dorf`*	`NI.Dorf.OTC1`, `NI.Dorf.calc1`
`Inf.Dorf`*	`Inf.Dorf.OTC1`, `Inf.Dorf.calc1`
`NI.D3`*	`NI.D3.OTC1`, `NI.D3.calc1`
`Inf.D3`*	`Inf.D3.OTC1`, `Inf.D3.calc1`
`NI.Array`*	`NI.Array.OTC1`, `NI.Array.calc1`
`Inf.Array`*	`Inf.Array.OTC1`, `Inf.Array.calc1`
`NI.A2M`*	`NI.A2M.OTC1`, `NI.A2M.calc1`
`hierarchical.desc2`* (called by `NI.Dorf`, `NI.D3`, `Inf.D3`)	`hierarchical.desc2` (called by `NI.Dorf.OTC1`, `NI.Dorf.calc1`, `NI.D3.OTC1`, `NI.D3.calc1`, `Inf.D3.OTC1`, `Inf.D3.calc1`)
`inf.dorf.measures`* (called by `Inf.Dorf`)	`inf.dorf.measures` (called by `Inf.Dorf.OTC1`, `Inf.Dorf.calc1`)
`characteristics.pool`* (called by `inf.dorf.measures`)	`characteristics.pool` (called by `inf.dorf.measures`)
`accuracy.dorf`* (called by `inf.dorf.measures`)	`accuracy.dorf` (called by `inf.dorf.measures`)
`Array.Measures`* (called by `NI.Array`, `Inf.Array`)	`Array.Measures` (called by `NI.Array.OTC1`, `NI.Array.calc1`, `Inf.Array.OTC1`, `Inf.Array.calc1`)
`MasterPool.Array.Measures`* (called by `NI.A2M`)	`MasterPool.Array.Measures` (called by `NI.A2M.OTC1`, `NI.A2M.calc1`)
`bgtAC`, `bgtBlaker`, `bgtCP`, `bgtSOC`, `bgtWald`, `bgtWilson` (called by `bgtCI`)	`bgtAC`, `bgtBlaker`, `bgtCP`, `bgtSOC`, `bgtWald`, `bgtWilson` (called by `propCI`)
`bgtPowerI` (called by `bgtPower`)	`bgtPowerI` (called by `gtPower`)
`bgtWidthI` (called by `bgtWidth`)	`bgtWidthI` (called by `gtWidth`)
`gtreg.fit` (called by `gtreg`)	`gtreg.fit` (called by `gtReg`)
`EM` (called by `gtreg`)	`EM` (called by `gtReg`)
`EM.ret` (called by `gtreg`)	`EM.ret` (called by `gtReg`)
`EM.halving` (called by `gtreg`)	`EM.halving` (called by `gtReg`)
`EM.mp` (called by `gtreg.mp`)	`EM.mp` (called by `gtReg`)

*denotes functions that were incorporated into the `binGroup` package by myself before the decision to create `binGroup2` was made.

Table 3.4: `binGroup` functions not included in `binGroup2`

binGroup
`opt.info.dorf`, `pool.specific.dorf`, `opt.pool.size`, `thresh.val.dorf`
`binCI` (`binAC`, `binBlaker`, `binCP`, `binSOC`, `binWald`, `binWilson`)
`plot.poolbin`, `print.binCI`
`binDesign`
`plot.binDesign`, `print.binDesign`
`plot.nDesign`, `plot.sDesign`
`binPower`, `binTest`, `binWidth`
`print.binTest`
`print.gt`
`residuals.gt`

Chapter 4

A Shiny app for pooled testing

4.1. Background

Laboratories around the world frequently need to test a high volume of clinical specimens for disease. For these situations, a decision needs to be made about whether group testing will reduce testing loads compared to testing specimens individually. Additionally, laboratories need to determine the set of group sizes to use. We developed a Shiny application (available at www.chrisbilder.com/shiny) to assist directors and technicians in laboratories with making these decisions.

A Shiny application (a.k.a., Shiny app) is an interactive web-based application built using the `Shiny` package in R. The app seeks input via a user-friendly interface in a web browser. The app then takes those inputs and performs calculations or produces plots by calling functions in R. Our Shiny app utilizes functions in the `binGroup2` package to calculate the expected number of tests and diagnostic accuracy measures for a wide variety of group testing algorithms using both one- and two-disease assays. The optimal testing configuration for an algorithm can be identified with the app as well.

Our app focuses on group testing in homogeneous (i.e., non-informative) settings because that is how group testing is most often utilized in laboratories.

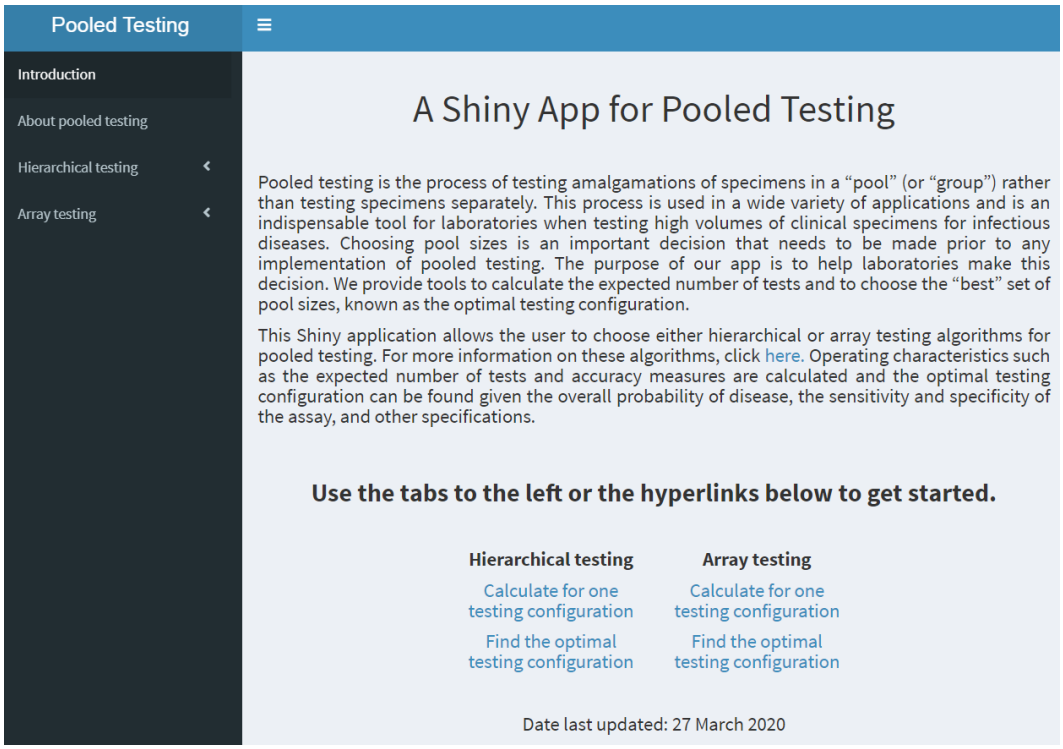
While the use of the `binGroup2` package requires reading the R documentation and some prior knowledge of programming, the goal of the Shiny app is to make this research accessible to those with non-statistical backgrounds who are using group testing in practice. The Shiny app uses simple questions, fill-in text boxes, and slider/radio button inputs to elicit the same specifications used for the `binGroup2` functions. We illustrate the use of our app with a simple example for single-disease assays, and with regard to the Aptima Combo 2 Assay used to test for chlamydia and gonorrhea throughout the United States.

4.2. Methods

Users can calculate operating characteristics for a single testing configuration or find the optimal testing configuration over a range of initial group sizes for hierarchical or array testing algorithms by following links available 1) in the side bar panel, 2) on the Introduction (A Shiny App for Pooled Testing) page (Figure 4.1), and 3) on the About pooled testing page (Figure 4.2) of the app. Each of the calculation pages (pages that calculate operating characteristics or find the OTC) pose questions to draw out the user's testing specifications. These specifications are:

1. The number of diseases for the assay,
2. The overall disease prevalence for a one-disease assay or a set of joint probabilities for a two-disease assay,
3. Sensitivity and specificity values for each stage of testing for each disease,
4. The number of stages for the algorithm, and
5. The testing configuration.

Figure 4.1: Introduction page in the Shiny app.



Like the functions in `binGroup2`, the Shiny app allows for sensitivity and specificity values to differ across stages of testing. Examples of the app specifications are shown in Figures 4.3, 4.4, and 4.5 for hierarchical testing and Figures 4.6 and 4.7 for array testing. Examples are given for when the operating characteristics of a specific testing configuration are of interest and for when the optimal testing configuration is to be found. After providing the required inputs, the user clicks CALCULATE and the Shiny app uses the `binGroup2` package to perform calculations.

Operating characteristics for a single testing configuration are calculated using the `opChar1()` and `opChar2()` functions in the `binGroup2` package. While the computed results are the same as with using these functions in R, the testing configuration, expected number of tests, expected number of tests per individual, and accuracy measures are displayed nicely in paragraph form

Figure 4.2: About pooled testing page in the Shiny app.

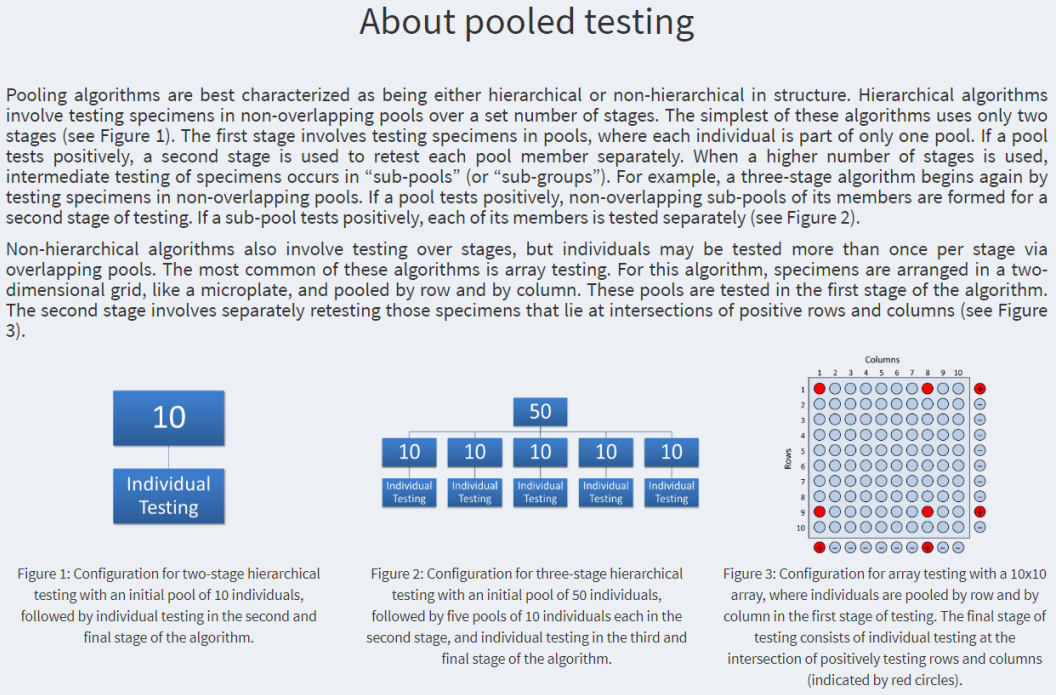


Figure 4.3: Specifications for two-stage hierarchical testing with a one-disease assay prior to specifications being included.

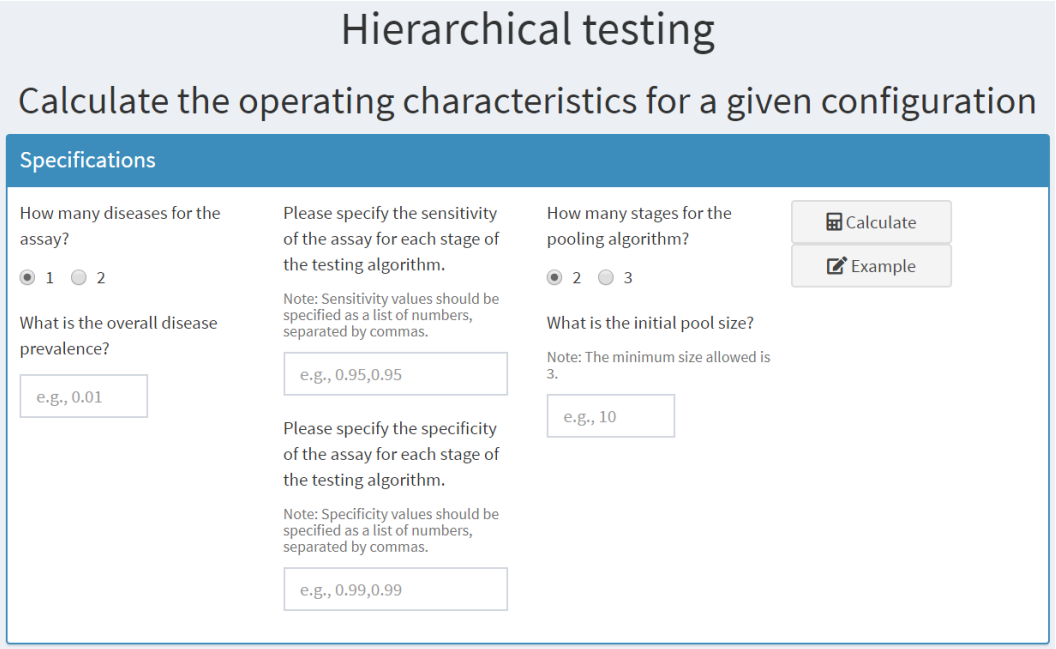


Figure 4.4: Specifications for three-stage hierarchical testing with a one-disease assay prior to specifications being included.

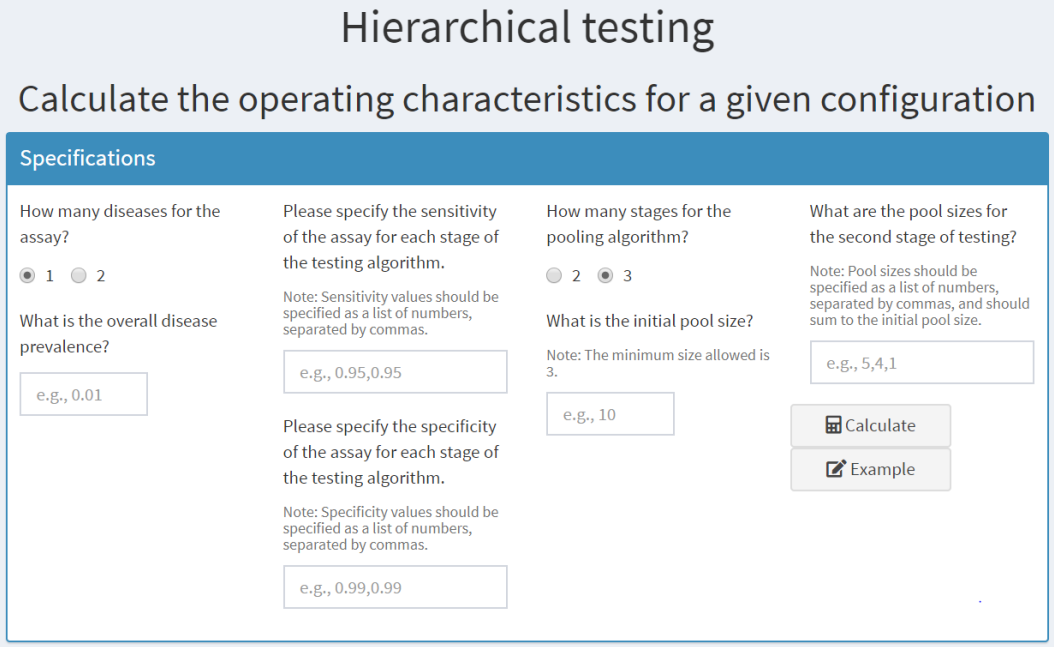


Figure 4.5: Specifications for two-stage hierarchical testing with a two-disease assay prior to specifications being included.

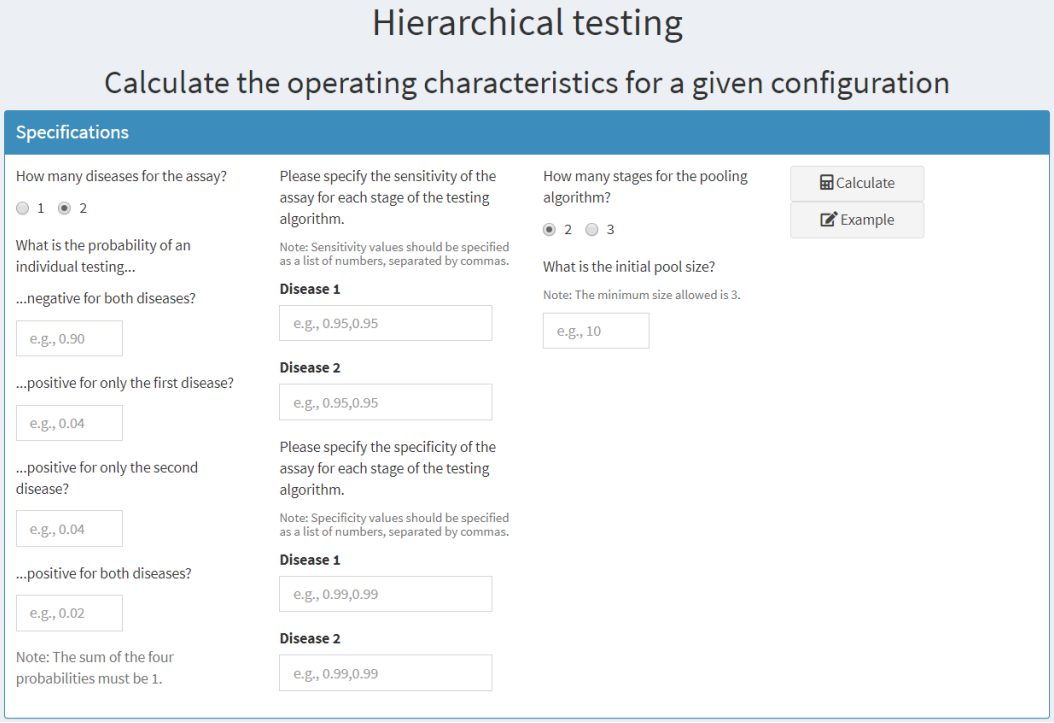


Figure 4.6: Specifications for array testing with a one-disease assay prior to specifications being included.

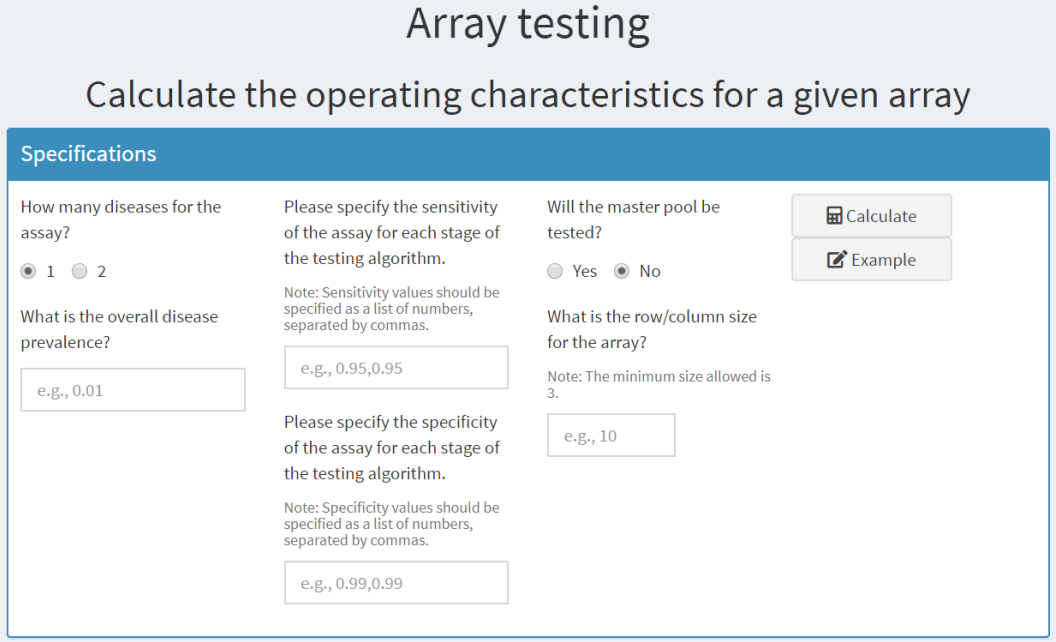
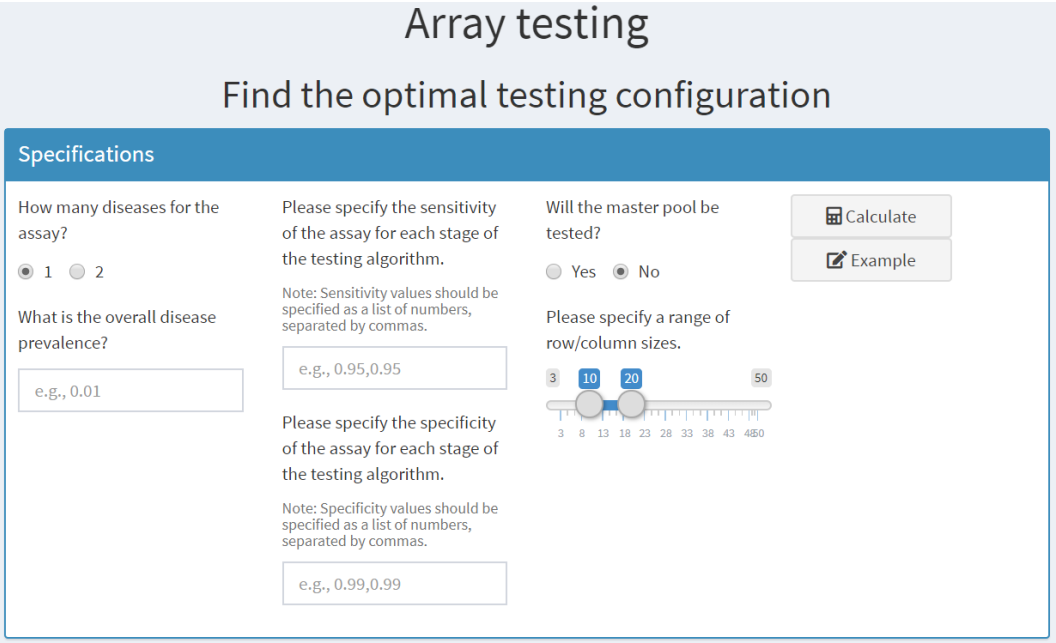


Figure 4.7: Specifications for array testing when finding the OTC prior to specifications being included.



in the app.

Additionally, the app provides the percent reduction in the number of tests when compared to individual testing and displays the input values used for calculations. An algorithm diagram is also produced to illustrate the testing configuration. When finding the optimal testing configuration, the app displays operating characteristics corresponding to the best configuration for each initial group size provided by the user. A plot of the the expected number of tests per individual for each of these similar configurations is also displayed.

4.3. Examples

4.3.1. Single-disease assays

Consider an example involving a two-stage hierarchical testing algorithm with an overall prevalence of $p = 0.03$ and an initial group size of 15. The sensitivity is $S_e = 0.95$ and the specificity is $S_p = 0.99$ for each stage of testing. Figure 4.8 displays the specifications. After a user selects CALCULATE, the corresponding operating characteristics are displayed as shown in Figure 4.9 and the algorithm diagram is displayed as shown in Figure 4.10.

The expected number of tests per individual is $E(T)/I = 6.32/15 = 0.42$. Thus, two-stage hierarchical testing reduces the expected number of tests by $(1 - 0.42) \times 100 = 58\%$ when compared to individual testing.

4.3.2. Multiplex assays

Consider an example involving the Aptima Combo 2 Assay that is used to test for both chlamydia and gonorrhea simultaneously. We focus here on how the State Hygienic Laboratory (SHL) at the University of Iowa uses the assay, where details are provided in Hou et al. (2020). While the SHL uses group

Figure 4.8: Example 1 - Specifications for two-stage hierarchical testing.

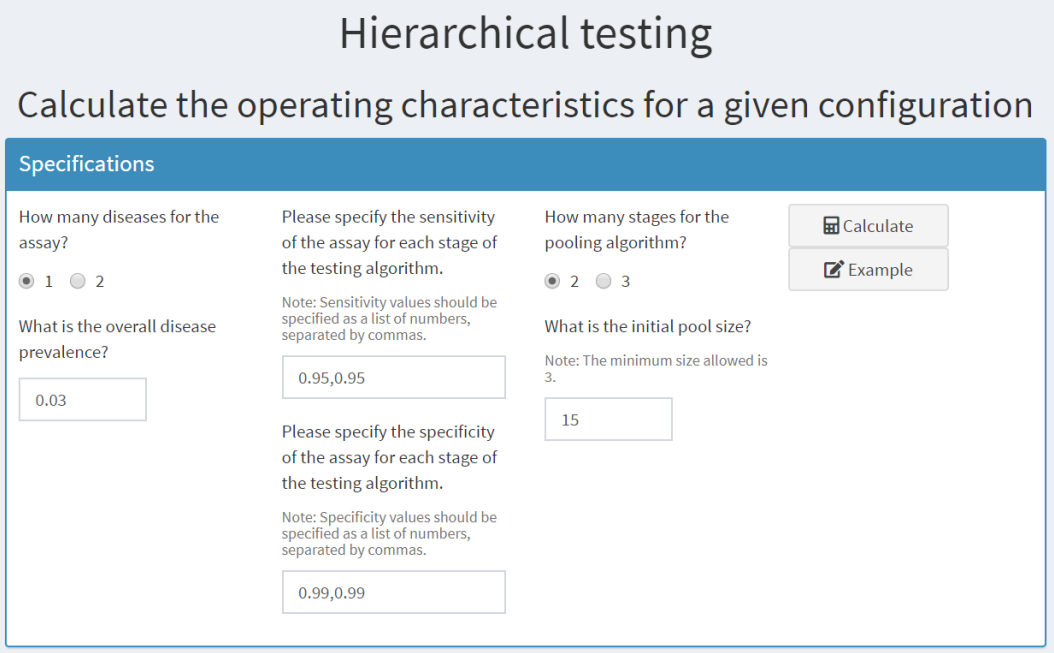


Figure 4.9: Example 1 - Operating characteristics for two-stage hierarchical testing.

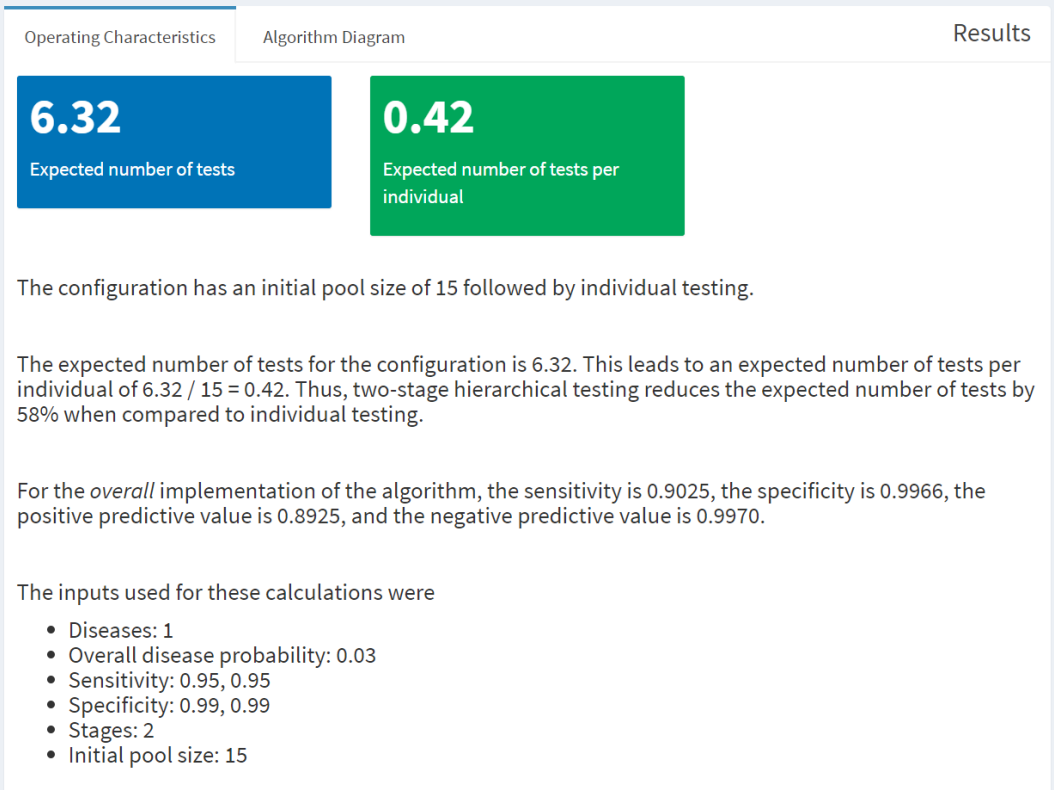
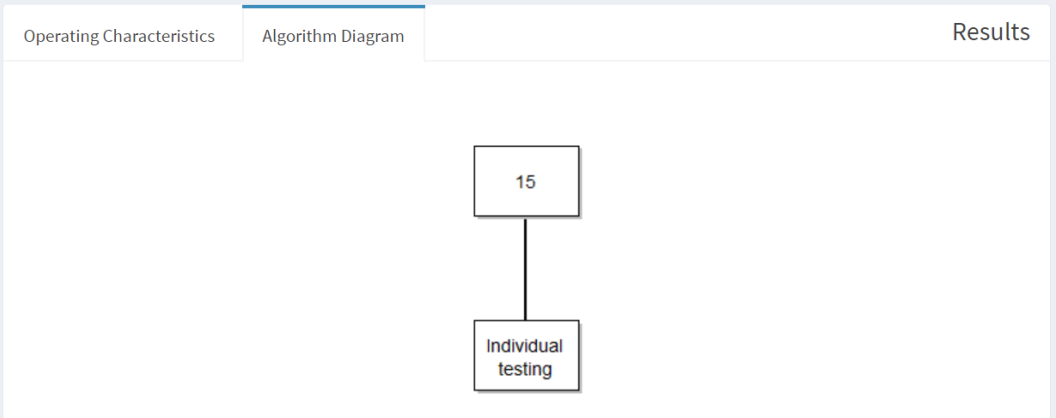


Figure 4.10: Example 1 - Algorithm diagram for two-stage hierarchical testing.

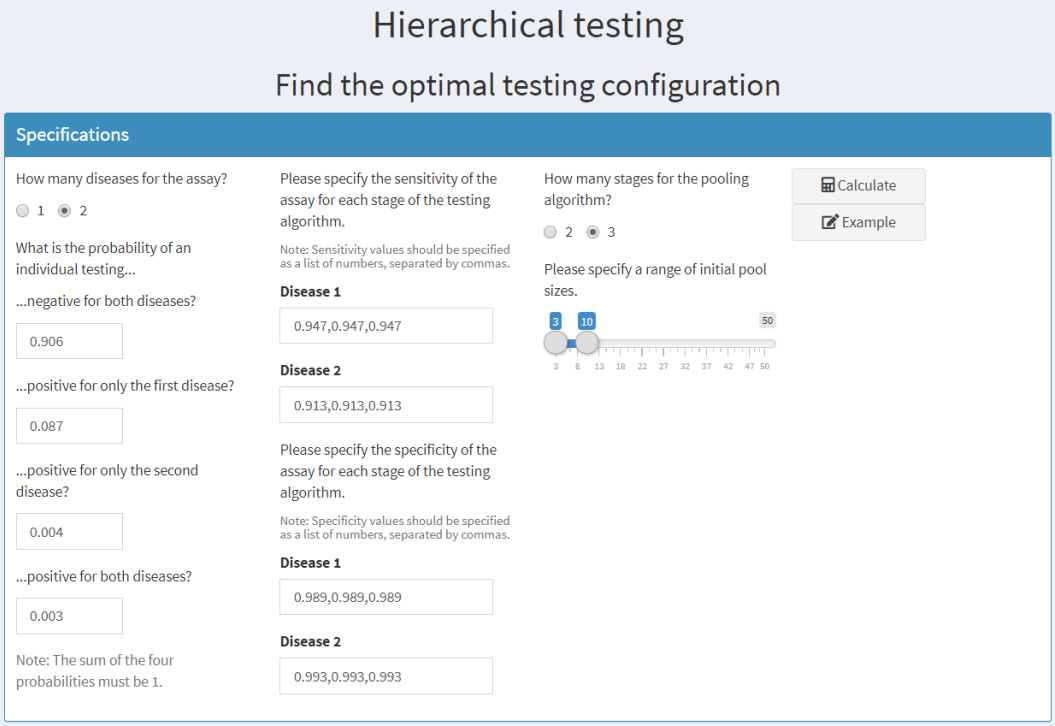


testing for female swab specimens, it uses individual testing for female urine specimens because they are concerned that the probability of having at least one disease may be too large for group testing to work well. The purpose here is to determine if group testing would be beneficial for the female urine specimens.

Figure 4.11 displays the specifications for a three-stage hierarchical testing algorithm. The joint probabilities of disease were estimated based on previous testing results for 5,998 individuals, where the first disease is chlamydia and the second disease is gonorrhea in our specifications. Sensitivity and specificity values were obtained from the Aptima Combo 2 Assay product insert (Food and Drug Administration, 2018), and we assumed these accuracy measures to be equal for all stages of testing. A maximum group size of 10 is used here because this is the largest size that we have seen used with group testing for these diseases (e.g., see Mund et al. (2008)).

Figure 4.12 shows the optimal testing configuration has an initial group size of 9 individuals with groups of size 3 in the second stage of testing. The expected number of tests per individual is $E(T)/I = 4.83/9 = 0.54$. Thus, three-stage hierarchical testing reduces the expected number of tests by

Figure 4.11: Example 2 - Specifications for three-stage hierarchical testing.



$(1 - 0.54) \times 100 = 46\%$ when compared to individual testing. Figure 4.13 displays the algorithm diagram for the OTC and results for similar configurations are displayed in Figure 4.14.

The specifications are the same for array testing without master pooling. Figure 4.15 shows the optimal array is 7×7, with an expected number of tests per individual of $E(T)/I = 26.50/49 = 0.54$. Thus, array testing without master pooling also reduces the expected number of tests by 46% when compared to individual testing. Figure 4.16 shows the algorithm diagram for the OTC and results for similar configurations are shown in Figure 4.17. Both algorithms offer significant savings over the individual testing currently being used by the SHL to test female urine specimens.

Because both algorithms provide a 46% reduction in the number of tests compared to individual testing, we can examine the accuracy of each algorithm

Figure 4.12: Example 2 - OTC and operating characteristics for three-stage hierarchical testing.

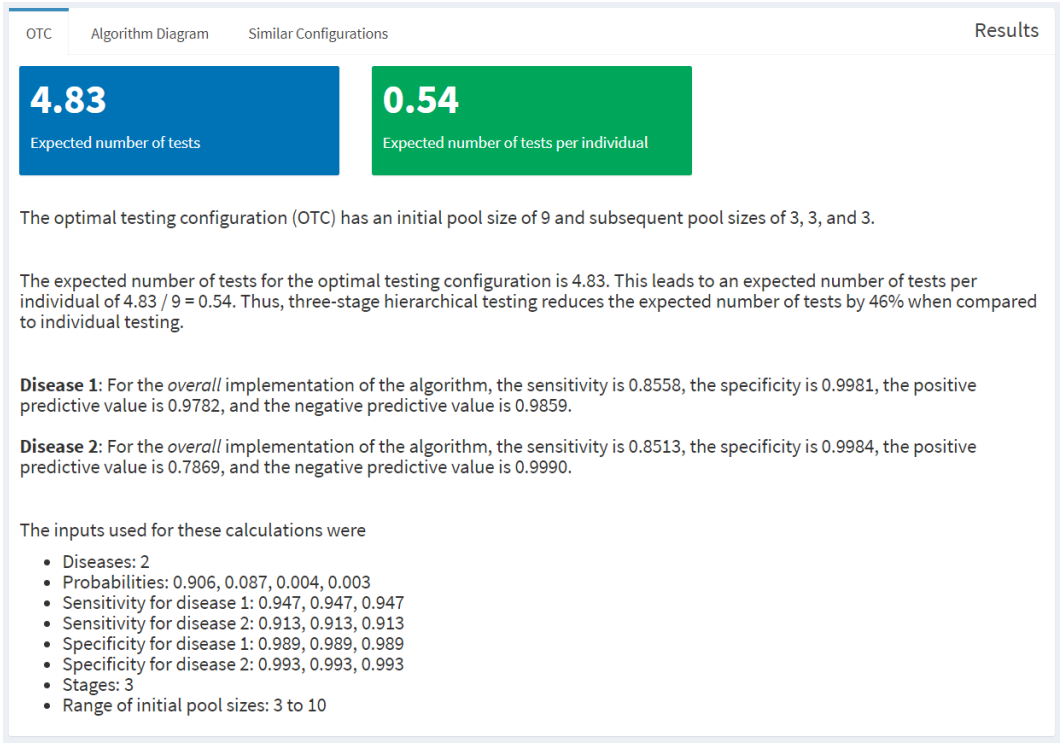


Figure 4.13: Example 2 - Algorithm diagram for three-stage hierarchical testing.

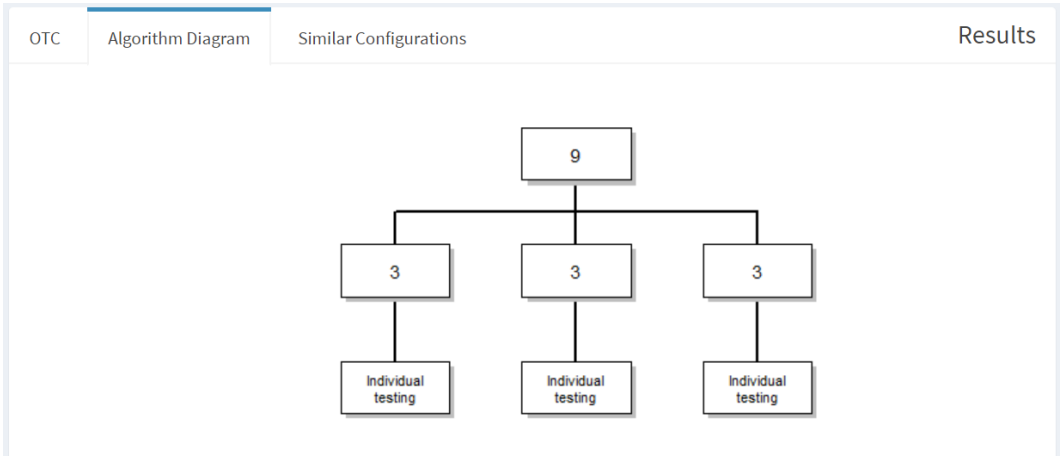
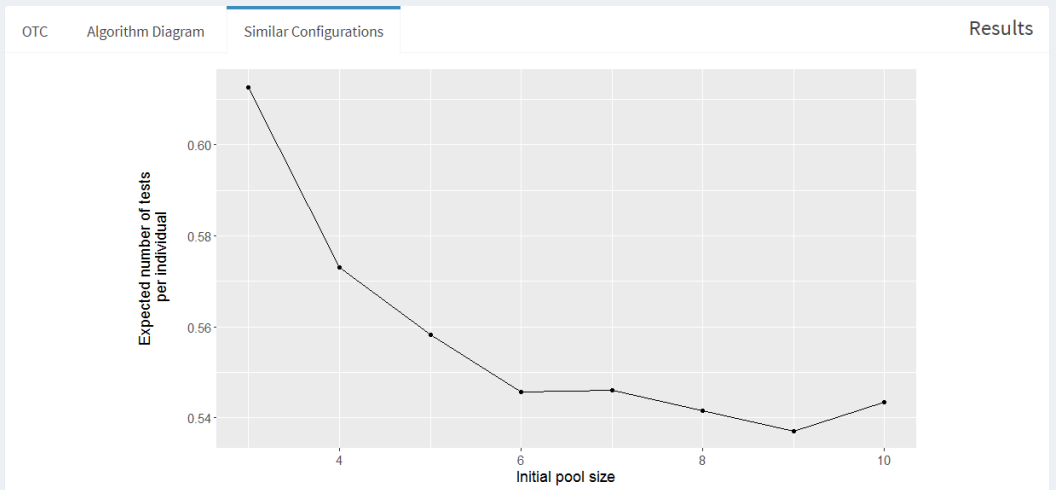


Figure 4.14: Example 2 - Similar configurations for three-stage hierarchical testing.



Initial pool size	Second stage pool sizes	Expected number of tests	Expected number of tests per individual	Overall sensitivity for disease 1	Overall specificity for disease 1	Overall positive predictive value for disease 1	Overall negative predictive value for disease 1	Overall sensitivity for disease 2	Overall specificity for disease 2
9	3,3,3	4.8331	0.537	0.8558	0.9981	0.9782	0.9859	0.8513	0.9984
8	3,3,2	4.3331	0.5416	0.8554	0.9983	0.9807	0.9859	0.8486	0.9985
10	4,3,3	5.4341	0.5434	0.8562	0.9978	0.9748	0.9859	0.8542	0.9982
6	3,3	3.2739	0.5457	0.8549	0.9982	0.9786	0.9858	0.8454	0.9984
7	3,2,2	3.822	0.546	0.855	0.9986	0.9838	0.9858	0.8456	0.9987

to determine which is best. The overall sensitivity of the algorithm is slightly higher for both diseases when using array testing. The overall specificity and positive predictive value of the algorithm is slightly higher for both diseases when using three-stage hierarchical testing. The overall negative predictive value corresponding to chlamydia is the same for both diseases and the overall negative predictive value corresponding to gonorrhea is slightly higher for array testing. Depending on which overall accuracy measure is considered most important for these diseases, this information can help to make a decision on which algorithm is preferred.

Sometimes laboratories face testing constraints and may not be able to im-

Figure 4.15: Example 2 - Optimal testing configuration and operating characteristics for array testing without master pooling.

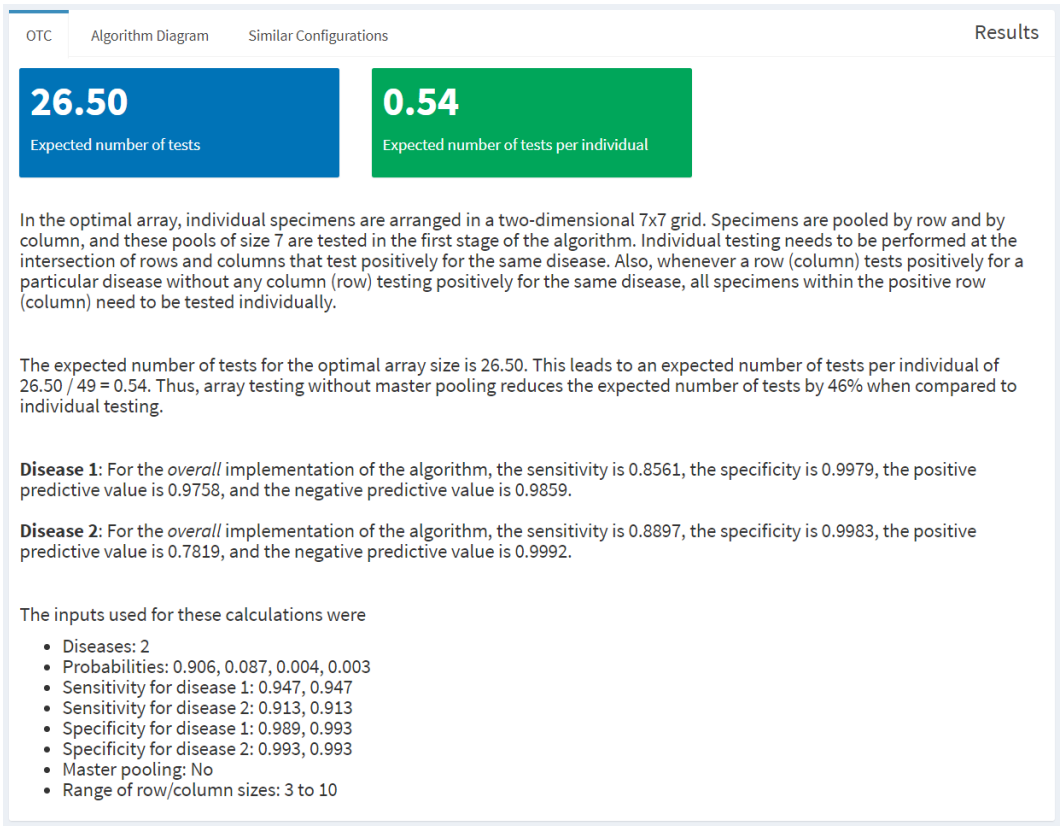


Figure 4.16: Example 2 - Algorithm diagram for array testing without master pooling.

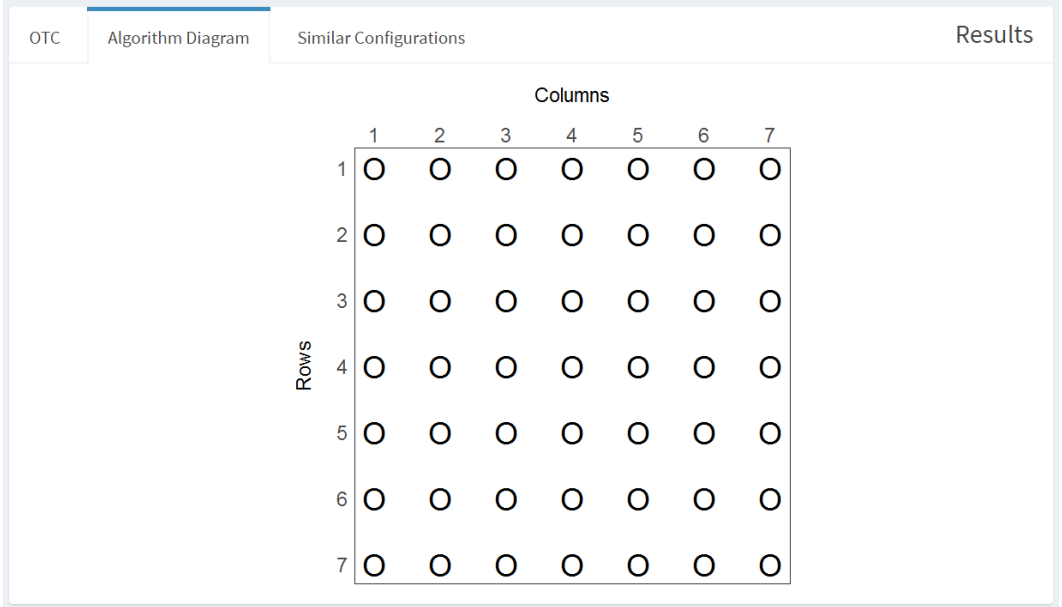
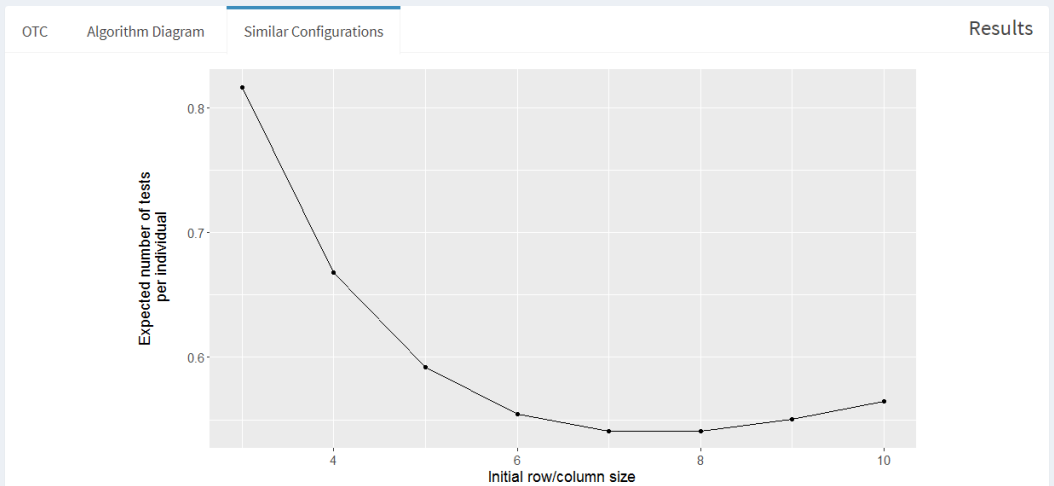


Figure 4.17: Example 2 - Similar configurations for array testing without master pooling.



Initial row/column size	Array size	Expected number of tests	Expected number of tests per individual	Overall sensitivity for disease 1	Overall specificity for disease 1	Overall positive predictive value for disease 1	Overall negative predictive value for disease 1	Overall sensitivity for disease 2	Overall specificity for disease 2
3	9	7.3486	0.8165	0.906	0.9992	0.9915	0.9908	0.905	0.999
4	16	10.6907	0.6682	0.8843	0.999	0.9885	0.9887	0.9016	0.9989
5	25	14.7992	0.592	0.8689	0.9987	0.9848	0.9872	0.8977	0.9987
6	36	19.9686	0.5547	0.8601	0.9983	0.9805	0.9863	0.8937	0.9985
7	49	26.4954	0.5407	0.8561	0.9979	0.9758	0.9859	0.8897	0.9983

plement the optimal testing configuration. This may be due to physical limits, such as a restriction on the group size that can be used in an automated pooling platform, or due to concerns over dilution effects. In these situations, the results for similar configurations can assist laboratories in choosing a different set of group sizes as close as possible to the optimal. For example, if the SHL chooses to utilize three-stage hierarchical testing but an initial group size of 9 is too large for implementation, an initial group of 6 individuals might be a good alternative (see Figure 4.17). This configuration allows for a smaller initial group size and provides similar savings in the number of tests over individual testing.

4.4. Conclusions

There are several additional features in our Shiny app that contribute to a more user-friendly experience:

- After clicking CALCULATE, a pop-up indicator bar displays the progress of the calculations as shown in Figure 4.18. The proportion of calculations completed is based on the range of initial group sizes provided by the user.

- An EXAMPLE button allows users to quickly populate input fields with sample values to demonstrate the app's capabilities. As the number of diseases or the testing configuration inputs change, the button can be clicked again and new sample values will populate as appropriate.

- Popover text provides helpful information about the sensitivity, specificity, and testing configurations and how to correctly specify their values. Figure 4.19 illustrates the popover text that appears when the user's mouse hovers over a fill-in box for the test sensitivity. Helpful explanations also appear over the overall accuracy measures in the results section.

- The results for similar configurations are displayed in an interactive data table and can be sorted by any column in increasing or decreasing order. Additionally, these results are available for download in a .csv file. For three-stage hierarchical testing, two sets of similar configurations (the best configuration for each initial group size and the top 10 configurations for each initial group size) are accessible.

Figure 4.18: Progress indicator for the three-stage hierarchical testing algorithm in Example 2.

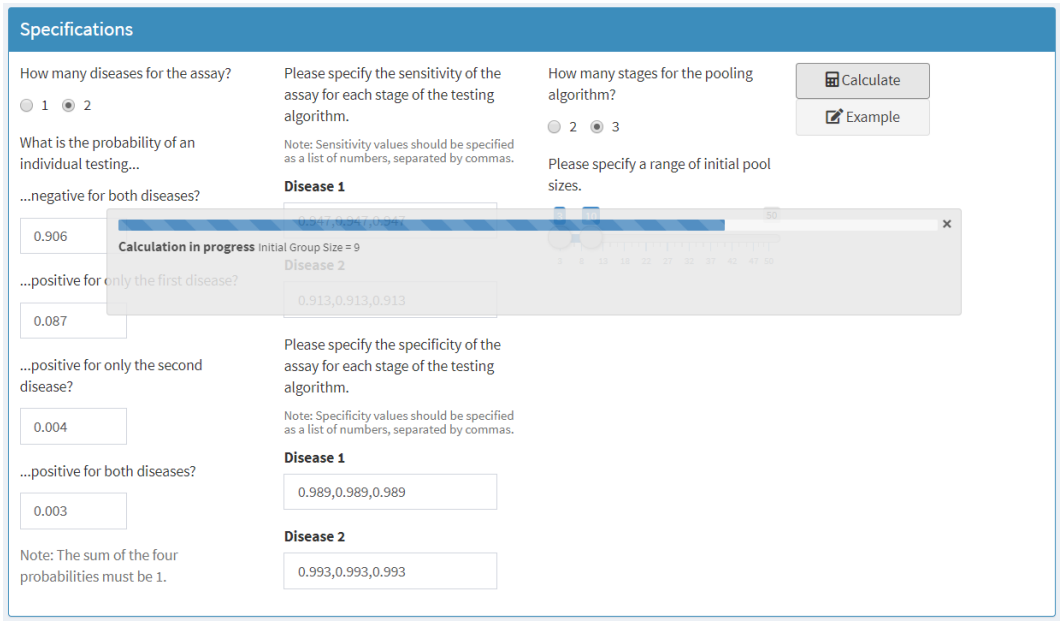
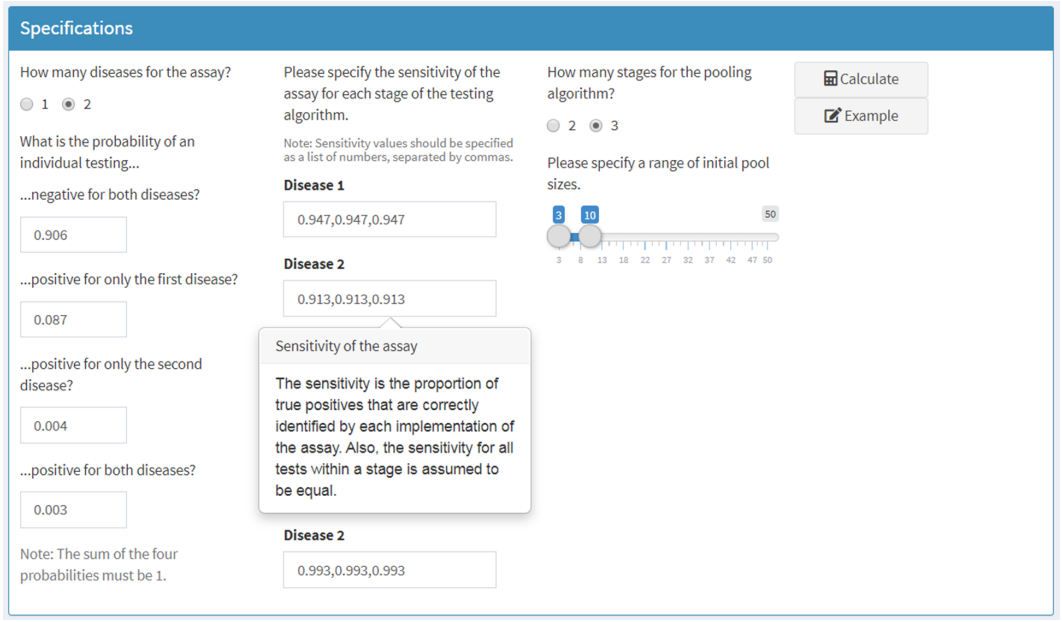


Figure 4.19: Popover text for the test sensitivity in Example 2.



Our Shiny app implements group testing algorithms for hierarchical and array-based testing algorithms with one- and two-disease assays over a wide variety of settings. It performs calculations for specified testing configurations and finds the optimal testing configuration over a range of initial group sizes. The app allows laboratories to do an initial investigation and determine if group testing would be beneficial for their situation, making the `binGroup2` functions available to researchers with no statistical or programming background.

Chapter 5

Additional Research

While developing this dissertation, we considered some investigations additional to those already presented in other chapters. We present here a few of these explorations related to the work in Chapter 2 and Chapter 4. We then discuss potential topics of future mathematical investigations related to multiplex assays. Finally, we propose ideas for future work on the `binGroup2` package and Shiny app.

5.1. Additional investigations for "The objective function controversy for group testing"

In an initial effort to find the OTC in Chapter 2, we considered the use of simulated annealing implemented via the `optim()` function in the `stats` package in R. Simulated annealing is a method for combinatorial optimization that gets its name from the process of slowly heating and cooling metals. The algorithm, described in detail within Givens and Hoeting (2013), involves randomly selecting moves and accepting them with a probability dependent on the amount the solution is worsened and a temperature parameter. Simulated annealing is best used with large candidate spaces (e.g., the set of all possible configurations for three-stage hierarchical testing); however, it can be extremely slow

to converge and may require a significant amount of skilled tuning to improve the performance of the algorithm (Givens and Hoeting, 2013).

We investigated the use of simulated annealing to find the OTC for both three-stage hierarchical and array testing algorithms. In the end, we discovered that searching over all possible testing configurations is faster in non-informative settings and in informative settings when the individual probabilities are ordered, which is how informative group testing is applied in practice (Black et al., 2012). In informative settings where risk probabilities are not ordered, simulated annealing might provide a benefit.

Expanding on our search for the OTC, we examined much larger prevalences than those given in Chapter 2. These investigations ranged up to a value of $p = 0.30$. Overall, the patterns discussed in Chapter 2 were similar for these larger prevalences. In addition, the optimal initial group size decreased as the overall prevalence increased in most cases. Occasionally, the optimal group size for larger probabilities was found to be 40 instead, which was the maximum group size allowed in our investigations. This result coincided with what Malinovsky et al. (2016) showed, that the optimal initial group size can be infinite for certain combinations of p, S_e, and S_p. For this reason, and because $p > 0.15$ rarely occurs in application, we focused on a range of prevalences from 0.005 to 0.150 by 0.005 in that chapter.

5.2. Additional considerations for the Shiny app

While developing the Shiny app, a significant amount of work was put into two features of the app: 1) a progress indicator that was only briefly mentioned in Chapter 4, and 2) a RESET button. In this section, we provide additional details on the efforts related to these features.

In Chapter 4, we introduced the progress indicator available in the Shiny app. To make this feature available, the app utilizes several functions from the `shiny` package to create and update the progress indicator as calculations are completed. Before any calculations are started, `Progress$new()` creates a new progress object and `progress$set()` is used to initialize the value of the progress bar to 0. The message argument provides the text to be displayed on the progress bar, which is "Calculation in progress" for the app. When the app calculates operating characteristics for a specified testing configuration, the progress is set to a value of 0.5 right before the `opChar1()` or `opChar2()` function is called and the progress is set to a value of 1 after the calculations are complete.

When the app finds the OTC, we are able to provide an additional level of detail in the progress indicator. Prior to the `OTC1()` or `OTC2()` function being called, a simple progress function is created. This function is passed as an additional argument to the OTC functions in `binGroup2`. For every initial group size in the range provided by the user, the progress bar is incremented by $1/(m+1)$, where m is the number of initial group sizes considered. The text for the progress bar is also updated, displaying the initial group size (or row/column size) for which calculations were just completed. When all calculations are completed, the progress is set to a value of 1 and the pop-up window containing the progress indicator closes. We added code to update Shiny progress objects in the `binGroup2` functions so that the functions would not have to be revised for use with the Shiny app. This code allows Shiny to interact with the `binGroup2` functions and update the progress indicators without making `binGroup2` dependent on the `shiny` package.

Previous versions of the Shiny app attempted to provide a RESET button

to complement the EXAMPLE and CALCULATE buttons. The goal of a RESET button was to clear the specified input values and erase any displayed results. Issues arose when resetting the specified inputs caused the disabling of the CALCULATE button. Additionally, the effects of a RESET button were not isolated to the page on which the button appeared. For example, resetting inputs on the calculation page for hierarchical testing disabled the CALCULATE button on all pages of the app. While we believe a RESET button could prove a useful addition to the Shiny app, additional research is needed to implement this feature successfully.

5.3. Future research

In this dissertation, we focused on calculating operating characteristics and finding OTCs for a large number of group testing algorithms using one- or two-disease assays. We presented an R package and Shiny app to this end and provided derivations to allow diagnostic accuracy to vary across stages of testing. We next describe extensions to our proposed methods and programming work, and provide recommendations for future research. We discuss additional mathematical investigations for group testing using multiplex assays, especially with three or more diseases. Additionally, the `binGroup2` package and Shiny app will continue to be advanced with the goal of enhancing the user experience. Additions and revisions are proposed to improve the app for statisticians and researchers who use group testing in practice.

5.3.1. Mathematical investigations

This dissertation focused on the implementation of group testing algorithms with assays that test for one or two diseases. One of the largest areas for new

research in group testing is with multiplex assays that test for more diseases. Closed-form expressions are available to calculate operating characteristics for hierarchical and array-based group testing algorithms with two-disease assays. Research could be performed to derive operating characteristics for the same algorithms using multiplex assays that test for three or more diseases.

At the time of publication, the authors of Bilder et al. (2019) were only aware of multiplex assays for up to $K = 3$ diseases being used in practice. For example, the American Red Cross uses a multiplex assay to screen blood donations for HBV, HCV, and HIV (American Red Cross, 2020), and the BD Max CT/GC/TV assay tests for chlamydia, gonorrhea, and trichomonas (Van Der Pol et al., 2016). However, there exist several multiplex assays to test for more than three diseases. Rumyantseva et al. (2015) evaluated a multiplex assay that detects chlamydia, gonorrhea, trichomonas, and M.gen, and the Tick-Borne Disease Serochip tests for eight major tick-borne pathogens including Lyme disease, babesiosis, anaplasmosis, and Powassan virus disease (Tokarz et al., 2018). While the implementation of these tests does not appear to utilize group testing procedures at this time, it serves as motivation for developing statistical methodology to show how best to take advantage of such an assay with group testing. Also, it is of interest to determine at what point the probability of testing positive for at least one disease becomes too high for realistic group testing applications.

Bilder et al. (2019) developed algorithms for hierarchical testing with two or more stages, but their derivations focused primarily on the $K = 2$ disease setting. Work is needed to explore settings for $K \geq 3$ diseases and to develop functions that implement the associated calculations. It may be possible to find closed-form expressions for operating characteristics in the $K = 3$ disease

case, but the derivations will likely be cumbersome because there are eight joint probabilities and many more probability terms to deal with than for the $K = 2$ disease setting. In the case that closed-form expressions for $K \geq 3$ are not attainable, a simulation approach could be used instead. It is important to note that simulation raises questions of whether finding the OTC would be feasible in regards to the time it takes to find a solution. Additional explorations potentially involve four-stage hierarchical testing, finding the OTC using combinatorial algorithms. Overall, further research is needed to determine how to calculate operating characteristics and find the OTC for $K \geq 3$ diseases and to evaluate how well hierarchical testing works as K increases.

Additionally, there are a number of investigations associated with the research in Bilder et al. (2019) that could be valuable. First, the effect of the correlation between diseases for $K > 2$ needs to be examined. As the correlation increases, it is more likely that positive responses for diseases will occur together and, hence, fewer tests should be needed. Investigation is also needed to determine what happens as the correlation between diseases approaches zero. Second, research is needed in regards to the pooling sensitivity. With a higher number of diseases, the pooling sensitivity increases. Work is needed to determine whether there is a mathematical reason for this phenomenon and to observe contextually what happens with simulation.

5.3.2. binGroup2

The `binGroup2` package is available on CRAN, but development of the package continues. The OTC and operating characteristic functions make available a large number of hierarchical and array-based group testing algorithms. The `halving()` and `Sterrett()` functions are available separately in `binGroup2`,

but these are only available for single-disease assays and do not allow for diagnostic accuracy to vary across stages of testing. One potential addition to the package involves revising these functions to allow sensitivity and specificity values to differ across stages of testing. These algorithms could also be added as options to the OTC and operating characteristic functions.

Another possible addition to the package involves a revision to the `obj.fn` argument in the `OTC1()` function. The `OTC1()` function only calculates operating characteristics and finds the OTC for a limited set of objective functions. Rather than use the objective functions currently available, we could allow the user to provide their own objective function. This capability would require the user to understand how the `OTC1()` and associated internal functions operate, but would allow the user to customize `OTC1()` for their needs.

The `OTC1()` and `OTC2()` functions provide results for configurations similar to the OTC. For algorithms that have only one testing configuration per initial group size (i.e., non-informative two-stage hierarchical testing, array testing), the set of similar configurations includes results for every initial group size specified by the user. For algorithms that have more than one testing configuration per initial group size (i.e., informative two-stage hierarchical testing, three-stage hierarchical testing), two sets of similar configurations are provided: 1) the best configuration for each initial group size specified by the user, and 2) the top 10 configurations for each initial group size specified by the user. A potential addition to the OTC functions involves a new argument, say `sim.config`, that allows the user to specify how many testing configurations to display for each initial group size. Another method for providing similar configurations could utilize a threshold value ϵ, specified by the user. All configurations where the objective function value per individual is within

ϵ of the OTC could be displayed by the function.

Currently, the functions for array testing with a two-disease assay do not allow for individuals with different risk probabilities p_i. Once methods for informative array testing with multiplex assays are developed, the corresponding functions in `binGroup2` could be revised to include this option and calculate individual accuracy measures as appropriate. Additionally, R functions could be written to perform simulation for hierarchical and array-based testing algorithms using multiplex assays. Bilder et al. (2019) made available R functions to implement simulation using hierarchical testing with multiplex assays that test for two diseases. These functions could be incorporated into the `binGroup2` package. Additional work could be done to expand these programs for multiplex assays that test for three or more diseases. New functions that implement simulation for array testing algorithms could also be included in `binGroup2` when they become available.

5.3.3. Shiny app

The most significant future development for our Shiny app is to incorporate programs for simulation that were mentioned in Section 5.3.2. However, we mentioned in Section 5.3.1 that these simulations may not be feasible with regards to the time required to find the OTC. For incorporation of these simulations in the Shiny app, we could omit the OTC and only calculate $E(T)/I$, the expected number of tests per individual. If it is determined that analytical derivations for the $K \geq 3$ disease case are not possible, the Shiny app could be limited to $K = 2$ diseases. Functions for simulation could either be incorporated on another page in the existing app or in a completely separate app that focuses on simulation.

Our Shiny app currently provides only overall accuracy measures when calculating operating characteristics for a specified testing configuration and only allows non-informative settings. The corresponding functions in `binGroup2` provide overall and individual accuracy measures. Individual accuracy measures could be added to the results in the Shiny app, particularly for hierarchical testing. Individual accuracy measures for hierarchical testing algorithms can vary depending on the set of group sizes used, but accuracy measures for array testing algorithms will be the same for all individuals. Additionally, future versions of the Shiny app could incorporate informative group testing settings. The expansion of the app in this manner would need to be carefully implemented though, because laboratory directors and technicians would need to decide on what risk probabilities to use and/or a distribution of risk probabilities.

The current version of the Shiny app needs a true disease prevalence(s) to perform calculations. In practice, laboratories most often won't know this information. Instead, they will have the proportion of individuals declared as positive from previous tests. To be more accurate, users of the app could specify which probability they have. If they provide the positive test proportion, the app could calculate a maximum likelihood estimate of the true disease prevalence using this proportion.

Minor revisions could be made to the app pertaining to long running computations and displayed error messages. When performing long running computations (e.g., finding the OTC for three-stage hierarchical testing over a large range of initial group sizes), the only way to stop calculations is to close the app and reopen it. To make the app more user-friendly, we could modify it to allow long running computations to be halted without restarting the app.

In addition, multiple copies of an error message (e.g., "Please specify a sensitivity value for each stage of testing.") are displayed when input values are incorrectly specified. This occurs because multiple rendered outputs depend on the calculations and each generates its own copy of the error message. A simple revision to the app could clean up the display so that only a single copy of an error message is displayed when necessary. Finally, development of a phone app would provide an exciting new way for users to access our research.

Bibliography

Abdalhamid, B., C. R. Bilder, E. L. McCutchen, S. H. Hinrichs, S. A. Koepsell, and P. C. Iwen (2020). Assessment of specimen pooling to conserve SARS CoV-2 testing resources. *American Journal of Clinical Pathology*.

Abdellrazeq, G. S., M. M. El-Naggar, S. Khaliel, and A. Gamal-Eldin (2014). Detection of Mycobacterium avium subsp. paratuberculosis from cattle and buffaloes in Egypt using traditional culture, serological and molecular based methods. *Veterinary World 7*, 586–593.

Altman, D. G. and J. M. Bland (1994a). Diagnostic tests 1: Sensitivity and specificity. *BMJ 308*, 1552.

Altman, D. G. and J. M. Bland (1994b). Diagnostic tests 2: Predictive values. *BMJ 309*, 102.

American Red Cross (2020). Infectious disease testing. https://www.redcrossblood.org/biomedical-services/blood-diagnostic-testing/blood-testing.html. Accessed March 23, 2020.

Biggerstaff, B. (2008). Confidence interval for the difference of proportions estimated from pooled samples. *Journal of Agricultural, Biological, and Environmental Statistics 13*, 478–496.

Bilder, C. R. (2019). Group testing for identification. *Wiley StatsRef: Statistics Reference Online*.

Bilder, C. R., P. C. Iwen, B. Abdalhamid, J. M. Tebbs, and C. S. McMahan (2020, April). Increasing testing capacity for SARS-CoV-2 by pooling specimens. Online. https://www.significancemagazine.com/science/651-increasing-testing-capacity-for-sars-cov-2-by-pooling-specimens. Accessed April 21, 2020.

Bilder, C. R. and J. M. Tebbs (2012). Pooled-testing procedures for screening high volume clinical specimens in heterogeneous populations. *Statistics in Medicine 31*, 3261–3268.

Bilder, C. R., J. M. Tebbs, and P. Chen (2010). Informative retesting. *Journal of the American Statistical Association 105*, 942–955.

Bilder, C. R., J. M. Tebbs, and C. S. McMahan (2019). Informative group testing for multiplex assays. *Biometrics 75*, 278–288.

Bilder, C. R., B. Zhang, F. Schaarschmidt, and J. M. Tebbs (2010). binGroup: A package for group testing. *The R Journal 2*, 56–60.

Black, M. S., C. R. Bilder, and J. M. Tebbs (2012). Group testing in heterogeneous populations by using halving algorithms. *Journal of the Royal Statistical Society. Series C: Applied Statistics 61*, 277–290.

Black, M. S., C. R. Bilder, and J. M. Tebbs (2015). Optimal retesting configurations for hierarchical group testing. *Journal of the Royal Statistical Society. Series C: Applied Statistics 64*, 693–710.

Branson, B. M., S. M. Owen, L. G. Wesolowski, B. Bennett, B. G. Werner, K. E. Wroblewski, and M. A. Pentella (2014). Laboratory testing for the diagnosis of HIV infection: Updated recommendations. http://stacks.cdc.gov/view/cdc/23447, Accessed March 23, 2020.

Casella, G. and R. L. Berger (2002). *Statistical Inference* (Second ed.). Duxbury Advanced Series. Pacific Grove, CA: Duxbury.

Centers for Disease Control and Prevention (2019). Estimated HIV incidence and prevalence in the United States, 2010-2016. HIV surveillance supplemental report 2019; 24 (no. 1). http://www.cdc.gov/hiv/library/reports/hiv-surveillance.html. Published February 2019. Accessed March 23, 2020.

Dorfman, R. (1943). The detection of defective members of large populations. *The Annals of Mathematical Statistics 14*, 436–440.

Encyclopaedia Britannica, Inc. (2017). Selective service acts. https://www.britannica.com/event/Selective-Service-Acts. Accessed March 23, 2020.

Finucan, H. M. (1964). The blood testing problem. *Journal of the Royal Statistical Society. Series C (Applied Statistics) 13*, 43–50.

Food and Drug Administration (2018). Gen-Probe® Aptima Combo 2® Assay package insert. https://www.fda.gov/media/74033/download. Accessed March 23, 2020.

Givens, G. H. and J. A. Hoeting (2013). *Computational Statistics* (Second Ed. ed.). Wiley Series in Computational Statistics. Hoboken, NJ: John Wiley & Sons, Inc.

Græsbøll, K., L. O. Andresen, T. Halasa, and N. Toft (2017). Opportunities and challenges when pooling milk samples using ELISA. *Preventive Veterinary Medicine 139*, 93–98.

Graff, L. E. and R. Roeloffs (1972). Group testing in the presence of test error; an extension of the Dorfman procedure. *Technometrics 14*, 113–122.

Hou, P., J. M. Tebbs, C. R. Bilder, and C. S. McMahan (2017). Hierarchical group testing for multiple infections. *Biometrics 73*, 656–665.

Hou, P., J. M. Tebbs, D. Wang, C. S. McMahan, and C. R. Bilder (2020). Array testing with multiplex assays. To appear in *Biostatistics*.

Hourfar, M. K., A. Themann, M. Eickmann, P. Puthavathana, T. Laue, E. Seifried, and M. Schmidt (2007). Blood screening for influenza. *Emerging Infectious Diseases 13*, 1081–1083.

Hudgens, M. G. (2016). Rejoinder to 'Reader reaction: A note on the evaluation of group testing algorithms in the presence of misclassification'. *Biometrics 72*, 304.

Hudgens, M. G. and H. Y. Kim (2011). Optimal configuration of a square array group testing algorithm. *Communications in Statistics - Theory and Methods 40*, 436–448.

Hughes-Oliver, J. M. (2006). Pooling experiments for blood screening and drug discovery. In A. Dean and S. Lewis (Eds.), *Screening: Methods for Experimentation in Industry, Drug Discovery, and Genetics*, pp. 48–68. New York, NY: Springer.

Hwang, F. K. (1975). A generalized binomial group testing problem. *Journal of the American Statistical Association 70*, 923–926.

Johnson, N., S. Kotz, and X. Wu (1991). *Inspection Errors for Attributes in Quality Control*. New York: Chapman and Hall Ltd.

Kacena, K. A., S. B. Quinn, S. C. Hartman, T. C. Quinn, and C. A. Gaydos (1998a). Pooling of urine samples for screening for Neisseria gonorrhoeae by ligase chain reaction: Accuracy and application. *Journal of Clinical Microbiology 36*, 3624–3628.

Kacena, K. A., S. B. Quinn, R. M. Howell, G. E. Madico, T. C. Quinn, and C. A. Gaydos (1998b). Pooling urine samples for ligase chain reaction screening in genital Chlamydia trachomatis infection in asymptomatic women. *Journal of Clinical Microbiology 36*, 481–485.

Kainkaryam, R. M. and P. J. Woolf (2009). Pooling in high-throughput drug screening. *Current Opinion in Drug Discovery and Development 12*, 339–350.

Khan, S. A., P. Chowdhury, P. Choudhury, and P. Dutta (2017). Detection of West Nile virus in six mosquito species in synchrony with seroconversion among sentinel chickens in India. *Parasites & Vectors 10*, 13.

Kim, H. Y. and M. G. Hudgens (2009). Three-dimensional array-based group testing algorithms. *Biometrics 65*, 903–910.

Kim, H. Y., M. G. Hudgens, J. M. Dreyfuss, D. J. Westreich, and C. D. Pilcher (2007). Comparison of group testing algorithms for case identification in the presence of test error. *Biometrics 63*, 1152–1163.

Kim, S. B., H. W. Kim, H.-S. Kim, H. W. Ann, J. K. Kim, H. Choi, M. H. Kim, J. E. Song, J. Y. Ahn, N. S. Ku, D. H. Oh, Y. C. Kim, S. J. Jeong, S. H. Han, J. M. Kim, D. M. Smith, and J. Y. Choi (2014). Pooled nucleic acid testing to identify antiretroviral treatment failure during HIV infection in Seoul, South Korea. *Scandinavian Journal of Infectious Diseases 46*, 136–40.

Kline, R. L., T. A. Brothers, R. Brookmeyer, S. Zeger, and T. C. Quinn (1989). Evaluation of human immunodeficiency virus seroprevalence in population surveys using pooled sera. *Journal of Clinical Microbiology 27*, 1449–1452.

Lewis, J. L., V. M. Lockary, and S. Kobic (2012). Cost savings and increased efficiency using a stratified specimen pooling strategy for Chlamydia trachomatis and Neisseria gonorrhoeae. *Sexually Transmitted Diseases 39*, 46–48.

Litvak, E., X. M. Tu, and M. Pagano (1994). Screening for the presence of a disease by pooling sera samples. *Journal of the American Statistical Association 89:426*, 424–434.

Lo, C., M. Liu, J. P. Lynch, and A. C. Gilbert (2013). Efficient sensor fault detection using combinatorial group testing. *2013 IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS)*, 199–206. IEEE.

Malinovsky, Y., P. S. Albert, and A. Roy (2016). Reader reaction: A note on the evaluation of group testing algorithms in the presence of misclassification. *Biometrics 72*, 299–302.

McMahan, C. S., J. M. Tebbs, and C. R. Bilder (2012a). Informative Dorfman screening. *Biometrics 68*, 287–296.

McMahan, C. S., J. M. Tebbs, and C. R. Bilder (2012b). Two-dimensional informative array testing. *Biometrics 68*, 793–804.

McMahan, C. S., J. M. Tebbs, and C. R. Bilder (2013). Regression models for group testing data with pool dilution effects. *Biostatistics 14*, 284–298.

McMahan, C. S., J. M. Tebbs, and C. R. Bilder (2016). Rejoinder to 'Reader reaction: A note on the evaluation of group testing algorithms in the presence of misclassification'. *Biometrics 72*, 303–304.

Mund, M., G. Sander, P. Potthoff, H. Schicht, and K. Matthias (2008). Introduction of Chlamydia trachomatis screening for young women in Germany. *Journal der Deutschen Dermatologischen Gesellschaft 6*, 1032–1037.

Nebraska Department of Health and Human Services (2020, April). Health alert network advisory: Availability of COVID-19 test ordering at Nebraska Public Health Lab (NPHL) by Nebraska healthcare providers. Online. http://dhhs.ne.gov/hanApril 1, 2020. Accessed April 3, 2020.

Papp, J. R., J. Schachter, C. A. Gaydos, and B. Van Der Pol (2014). Recommendations for the laboratory-based detection of Chlamydia trachomatis and Neisseria gonorrhoeae-2014. Technical Report RR-02, Centers for Disease Control and Prevention.

Pasquali, F., A. De Cesare, A. Valero, J. E. Olsen, and G. Manfreda (2014). Improvement of sampling plans for Salmonella detection in pooled table eggs by use of real-time PCR. *International Journal of Food Microbiology 184*, 31–34.

Phatarfod, R. M. and A. Sudbury (1994). The use of a square array scheme in blood testing. *Statistics in Medicine 13*, 2337–2343.

Pilcher, C. D., S. A. Fiscus, T. Q. Nguyen, E. Foust, L. Wolf, D. Williams, R. Ashby, J. Owen-O'Dowd, J. T. McPherson, B. Stalzer, L. Hightow, W. C. Miller, J. Joseph J. Eron, M. S. Cohen, and P. A. Leone (2005). Detection of acute infections during HIV testing in North Carolina. *The New England Journal of Medicine 352*, 1873–1883.

Quinn, T. C., R. Brookmeyer, R. Kline, M. Shepherd, R. Paranjape, S. Mehendale, D. A. Gadkari, and R. Bollinger (2000). Feasibility of pooling sera for HIV-1 viral RNA to diagnose acute primary HIV-1 infection and estimate HIV incidence. *AIDS 14*, 2751–2757.

R Core Team (2017). *R: A language and environment for statistical computing*. Vienna, Austria: R Foundation for Statistical Computing. https://www.R-project.org.

Rumyantseva, T., D. Golparian, C. S. Nilsson, E. Johansson, M. Falk, H. Fredlund, A. Van Dam, A. Guschin, and M. Unemo (2015). Evaluation of the new AmpliSens multiplex real-time PCR assay for simultaneous detection of Neisseria gonorrhoeae, Chlamydia trachomatis, Mycoplasma genitalium, and Trichomonas vaginalis. *APMIS 123*, 879–886.

Schaarschmidt, F. (2007). Experimental design for one-sided confidence intervals or hypothesis tests in binomial group testing. *Communications in Biometry and Crop Science 2*, 32–40.

Sherlock, M., N. Zetola, and J. Klausner (2007). Routine detection of acute

HIV infection through RNA pooling: Survey of current practice in the United States. *Sexually Transmitted Diseases 34*, 314–316.

Soroka, S. D., T. C. Granade, S. Phillips, and B. Parekh (2003). The use of simple, rapid tests to detect antibodies to human immunodeficiency virus types 1 and 2 in pooled serum specimens. *Journal of Clinical Virology 27*, 90–96.

Sterrett, A. (1957). On the detection of defective members of large populations. *The Annals of Mathematical Statistics 28*, 1033–1036.

Sullivan, T. J., P. Patel, A. Hutchinson, S. F. Ethridge, and M. M. Parker (2011). Evaluation of pooling strategies for acute HIV-1 infection screening using nucleic acid amplification testing. *Journal of Clinical Microbiology 49*, 3667–3668.

Tebbs, J., C. McMahan, and C. Bilder (2013). Two-stage hierarchical group testing for multiple infections with application to the Infertility Prevention Project. *Biometrics 69*, 1064–1073.

Tebbs, J. M. and C. R. Bilder (2004). Confidence interval procedures for the probability of disease transmission in multiple-vector-transfer designs. *Journal of Agricultural, Biological, and Environmental Statistics 9*, 75–90.

Tilghman, M., D. Tsai, T. P. Buene, M. Tomas, S. Amade, D. Gehlbach, S. Chang, C. Ignacio, G. Caballero, S. Espitia, S. May, E. V. Noormahomed, and D. M. Smith (2015). Pooled nucleic acid testing to detect antiretroviral treatment failure in HIV-infected patients in Mozambique. *Journal of Acquired Immune Deficiency Syndromes 70*, 256–61.

Tokarz, R., N. Mishra, T. Tagliafierro, S. Sameroff, A. Caciula, L. Chauhan, J. Patel, E. Sullivan, A. Gucwa, B. Fallon, M. Golightly, C. Molins, M. Schriefer, A. Marques, T. Briese, and W. I. Lipkin (2018). A multiplex serologic platform for diagnosis of tick-borne diseases. *Scientific Reports 8*.

Tu, X. M., E. Litvak, and M. Pagano (1995). On the informativeness and accuracy of pooled testing in estimating prevalence of a rare disease: Application to HIV screening. *Biometrika 82*, 287–297.

U.S. Census Bureau, Population Division (2018). Annual estimates of the resident population: April 1, 2010 to july 1, 2018. https://www.census.gov/data/tables/time-series/demo/popest/2010s-national-total.html. Released December 2018. Accessed March 23, 2020.

Van Der Pol, B., J. A. Williams, D. Fuller, S. N. Taylor, and I. Edward W. Hook (2016). Combined testing for chlamydia, gonorrhea, and trichomonas by use of the BD Max CT/GC/TV assay with genitourinary specimen types. *Journal of Clinical Microbiology 55*, 155–164.

Zhang, B. B. (2012). *Group testing regression models*. Ph. D. thesis.

Appendix A

Supporting information for Chapter 2

This appendix provides supporting information for the published manuscript in Chapter 2: *Hitt, B., Bilder, C., Tebbs, J., and McMahan, C., (2019). The objective function controversy for group testing: Much ado about nothing? Statistics in Medicine 38(24), 4912-4923. Used with permission.*

A.1. Notation for Section 2.2

In Section 2.2, we provided the following expression for the expected number of tests for three-stage hierarchical testing:

$$E(T) = 1 + m_{11}P(G_{11} = 1) + \sum_{j=1}^{c_2} m_{2j}P(G_{11} = 1, G_{2j} = 1).$$

To help explain the expression's notation, Figure A.1 provides a visual representation of this type of algorithm as it is used for HIV testing in San Francisco (Sherlock et al., 2007). The binary random variable G_{sj} indicates the positive (1) or negative (0) outcome for group j at stage s. For the initial group of 50 individuals in the first stage of Figure A.1, there will be a single group testing result for G_{11}. When $G_{11} = 1$, $c_2 = 5$ subgroups of size $m_{11} = 10$ are formed for a second stage of testing. These five subgroups have binary

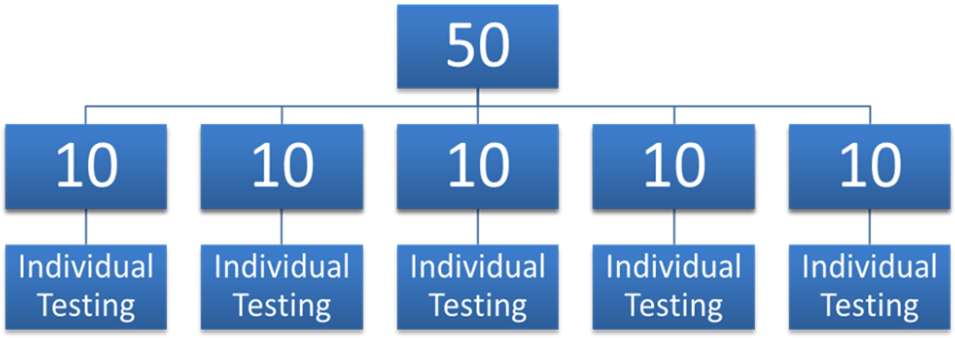


Figure A.1: Diagram of the three-stage hierarchical testing algorithm used for HIV testing in San Francisco.

testing outcomes of G_{21}, G_{22}, G_{23}, G_{24}, and G_{25}. If $G_{2j} = 1$ for some subgroup $j = 1, \ldots, 5$, $m_{2j} = 10$ individual tests are performed in the third and final stage of testing.

A.2. Additional results for Section 2.3.1

We provide additional results to coincide with our investigations in Section 2.3.1. Overall, these additional results continue to show that the OTCs have the same or very similar operating characteristics when using either objective function. We also include in our summaries the pooling positive predictive value, $PPPV$, and the pooling negative predictive value, $PNPV$, as additional accuracy measures. The pooling positive (negative) predictive value is the probability that an individual who is determined to be positive (negative) by the testing algorithm is truly positive (negative). Predictive values simply provide an alternative way of looking at accuracy in comparison to the pooling sensitivity and pooling specificity. Expressions for all accuracy measures are

available in Kim et al. (2007), McMahan et al. (2012a,b), and Black et al. (2015).

Table 2.1 provides a summary of the optimal testing configurations (OTCs) and their operating characteristics when $p = 0.01$. This table is reproduced here as Table A.1 with the addition of the predictive values. Similar tables for $p = 0.05$ and $p = 0.10$ are shown in Tables A.2 and A.3, respectively. The largest differences between operating characteristics for OTCs are shown in Table 2.2. Table A.4 displays the same results with the addition of the predictive values.

A.3. Additional results for Section 2.3.2

A.3.1. Tables

We provide additional results to coincide with our investigations in Section 2.3.2. Once again, these additional results show that the same or very similar operating characteristics are obtained regardless of which objective function is used. Table A.5 displays the same results as Table 2.3 but with the addition of the predictive values. Similar tables for $E(P_i) = 0.05$ and $E(P_i) = 0.10$ are provided in Tables A.6 and A.7, respectively. Table A.8 displays the same findings as Table 2.4 but with the addition of the predictive values.

Because informative group testing results in potentially different accuracy measures for each individual tested, we formed weighted averages across all individuals tested to present one overall value for each accuracy measure. These weighted averages are developed from accuracy definitions given by Altman and Bland (1994a,b) and were used by Black et al. (2015). The pooling sensi-

tivity is defined as

$$PS_e^W = \frac{\sum_i p_i PS_{e,i}}{\sum_i p_i}, \tag{A.3.1}$$

and the pooling specificity is defined as

$$PS_p^W = \frac{\sum_i (1-p_i) PS_{p,i}}{\sum_i (1-p_i)}. \tag{A.3.2}$$

Similarly, the pooling positive predictive value is defined as

$$PPPV^W = \frac{\sum_i p_i PS_{e,i}}{\sum_i p_i PS_{e,i} + (1-p_i)(1-PS_{p,i})}, \tag{A.3.3}$$

and the pooling negative predictive value is defined as

$$PNPV^W = \frac{\sum_i (1-p_i) PS_{p,i}}{\sum_i (1-p_i) PS_{p,i} + p_i(1-PS_{e,i})}. \tag{A.3.4}$$

Expressions (A.3.1) through (A.3.4) represent weighted averages over all I individuals within the initial group for a hierarchical algorithm or all I^2 individuals within the array for an array testing algorithm.

A.3.2. OTCs for informative group testing

Due to the lack of available space, Tables A.5, A.6, and A.7 display at most only the initial (stage 1) group size for the informative hierarchical algorithms. We display their full algorithms in Tables A.9 - A.14. Define I_{sj} as the size of group j at stage s. For two-stage hierarchical testing, individuals are assembled into blocks (McMahan et al., 2012a), where we use a block size of 50. Thus, we have $\sum_j I_{1j} = 50$ by design.

To better understand the displayed OTCs in the tables, consider the OTC given in the first row of results in Table A.12. The algorithm is performed

over $S = 3$ stages with an initial group size of $I_{11} = 10$ individuals. If this initial group tests positively, four new groups are formed for the second stage of testing with sizes $I_{21} = 4$, $I_{22} = 3$, $I_{23} = 2$, and $I_{24} = 1$. Informative group testing always orders individuals by their probabilities of positivity. Therefore, the first group consists of the individuals with the four smallest probabilities, and the last group consists of the individual with the largest probability. If any of these groups test positively and has a size greater than 1, individual testing is performed upon its group members. For the first group in stage 2, this means that individual tests would be performed on each of its members in stage 3 ($I_{31} = I_{32} = I_{33} = I_{34} = 1$). For the last group in stage 2, no subsequent retesting would be performed. Figure A.2 provides a pictorial representation of this group testing algorithm.

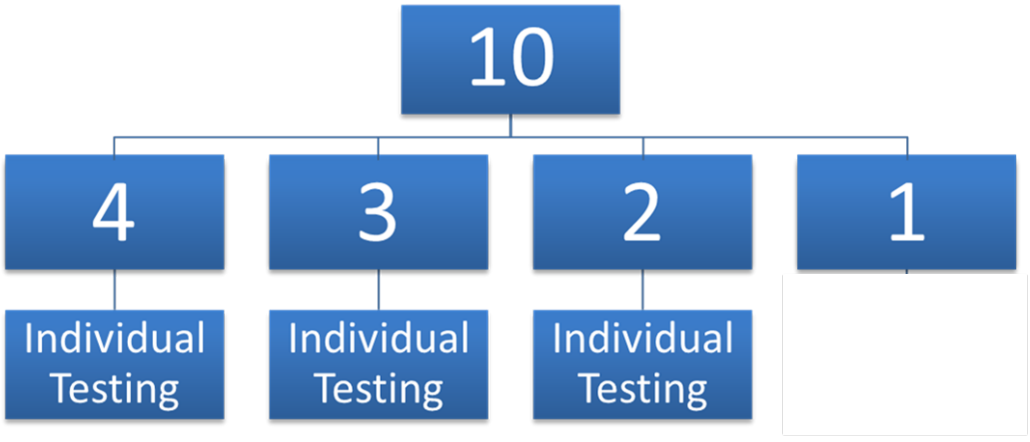


Figure A.2: Diagram of the group testing algorithm described in Section A.3.2. Group sizes are provided within nodes.

A.4. Additional results for Section 2.4

We provide additional results to coincide with our investigations in Section 2.4. Tables A.15 and A.16 display the same results as Tables 2.5 and 2.6, respectively, but with the addition of the predictive values. Due to the lack of available space, Table A.16 displays at most only the initial (stage 1) group size for the informative hierarchical algorithms. We display their full algorithms in Tables A.17 and A.18.

A.5. Additional results for Section 2.5

Graff and Roeloffs (1972) proposed an objective function that is a linear combination of the expected number of tests, the number of misclassified negative individuals (FN_1), and the number of misclassified positive individuals (FP_1). This linear combination can be expressed as

$$\begin{aligned} & E(T) + D_1 \times FN_1 + D_2 \times FP_1 \\ &= \quad E(T) + \sum_{i=1}^{I} \left\{ D_1(1 - PS_{p,i})(1 - p_i) + D_2(1 - PS_{e,i})p_i \right\}, \end{aligned}$$

where D_1 and D_2 are subjectively chosen weights. The OTC is found by minimizing the value of this linear combination per individual, denoted by O_{GR}. Because weights are subjectively chosen, there will be weights that result in an OTC different than what is obtained by using O_{ET} or O_{MAR}. We provide results in Tables A.19 and A.20 to illustrate these differences. Overall, the value of D_1 has a larger effect than the value of D_2, because there are many more individuals who are truly negative than positive due to the small probability of being positive.

A.6. R examples

To reproduce the research in this paper, we make available a set of R functions in the `binGroup` package that

- Calculate $E(T)$ and associated accuracy measures for different objective functions, and
- Find the OTC over a wide variety of settings.

All calculations for the paper were performed in version 3.4.1 of R (R Core Team, 2017).

The examples provided next show how to use `binGroup` to reproduce results from Tables 2.1 and 2.3. Examples 1 and 2 use non-informative group testing with an overall disease prevalence of $p = 0.01$. Examples 3 and 4 use informative group testing with an overall disease prevalence of $E(P_i) = 0.01$. Estimated running times for each example were calculated using a computer with 16 GB of RAM and one core of an Intel i7-6500U processor.

```
> library(binGroup)
> # Example 1
> # Finding the OTC using non-informative
> # three-stage hierarchical testing, where
> # p denotes the overall prevalence of disease,
> # Se denotes the sensitivity of the diagnostic test,
> # Sp denotes the specificity of the diagnostic test,
> # group.sz denotes the range of initial pool sizes
> # for consideration, and obj.fn specifies
> # the objective functions for which to find results.

> # This example takes approximately 2.5 minutes to run.
> results1 <- OTC(algorithm="D3", p=0.01, Se=0.99, Sp=0.99,
    group.sz=3:40, obj.fn=c("ET", "MAR"))

You have specified an overall probability of disease.
A probability vector will be generated based on the algorithm
    specified.
Algorithm: Non-informative three-stage hierarchical testing
Initial Group Size = 3
Initial Group Size = 4
Initial Group Size = 5

<OUTPUT EDITED>

Initial Group Size = 38
Initial Group Size = 39
Initial Group Size = 40
```

```
 Number of minutes running: 2.429667

> # Print the results.
> data.frame("Obj.Fn"=c("O_ET", "O_MAR"),
    "OTC"=c(paste(results1$opt.ET$OTC$Stage1,
    results1$opt.ET$OTC$Stage2[1], 1, sep="-"),
    paste(results1$opt.MAR$OTC$Stage1,
    results1$opt.MAR$OTC$Stage2[1], 1, sep="-")),
    "ET.I"=c(round(results1$opt.ET$ET /
    results1$opt.ET$OTC$Stage1, 4),
    round(results1$opt.MAR$ET/results1$opt.MAR$OTC$Stage1,
    4)), "PSe"=c(round(results1$opt.ET$PSe, 4),
    round(results1$opt.MAR$PSe, 4)),
    "PSp"=c(round(results1$opt.ET$PSp, 4),
    round(results1$opt.MAR$PSp, 4)))
  Obj.Fn     OTC   ET.I    PSe    PSp
1   O_ET 25-5-1 0.1354 0.9703 0.9996
2  O_MAR 25-5-1 0.1354 0.9703 0.9996


> # Example 2
> # Finding the OTC using non-informative
> # array testing with master pooling.
> # The OTC differs for the ET and MAR
> # objective functions in this example.

> # This example takes approximately 2 minutes to run.
> results2 <- OTC(algorithm="A2M", p=0.01, Se=0.90, Sp=0.90,
    group.sz=3:30, obj.fn=c("ET", "MAR"))
You have specified an overall probability of disease.
A probability vector will be generated based on the algorithm
    specified.
Algorithm: Non-informative square array testing with master
    pooling
Row/Column Size = 3, Array Size = 9
Row/Column Size = 4, Array Size = 16
Row/Column Size = 5, Array Size = 25

<OUTPUT EDITED>

Row/Column Size = 28, Array Size = 784
Row/Column Size = 29, Array Size = 841
Row/Column Size = 30, Array Size = 900
 Number of minutes running: 1.745667

> # Print the results.
> data.frame("Obj.Fn"=c("O_ET", "O_MAR"),
    "OTC"=c(paste(results2$opt.ET$OTC$Array.sz,
    results2$opt.ET$OTC$Array.dim, 1, sep="-"),
    paste(results2$opt.MAR$OTC$Array.sz,
    results2$opt.MAR$OTC$Array.dim, 1, sep="-")),
    "ET.I"=c(round(results2$opt.ET$ET /
    results2$opt.ET$OTC$Array.sz, 4),
    round(results2$opt.MAR$ET/results2$opt.MAR$OTC$Array.sz,
    4)), "PSe"=c(round(results2$opt.ET$PSe, 4),
    round(results2$opt.MAR$PSe, 4)),
    "PSp"=c(round(results2$opt.ET$PSp, 4),
    round(results2$opt.MAR$PSp, 4)))
  Obj.Fn       OTC  ET.I    PSe    PSp
1   O_ET 625-25-1 0.145 0.6562 0.9934
2  O_MAR 576-24-1 0.145 0.6562 0.9937


> # Example 3
> # Finding the OTC using informative two-stage
> # hierarchical testing, implemented via the
> # pool-specific optimal Dorfman (PSOD) method
> # described in McMahan et al. (2012), where
> # alpha denotes the level of heterogeneity in
> # the beta distribution used to generate the
> # vector of individual probabilities.
```

```
> # Depending on the specified probability, level of
> # heterogeneity, and initial group size, simulation
> # may be necessary in order to generate an ordered
> # vector of individual probabilities. This is done
> # with the beta.dist() function (see Black et al. 2015)
> # using 10,000 simulated data sets. The user will
> # need to set a seed in order to reproduce results.

> # This examples takes approximately 2.5 minutes to run.
> set.seed(1002)
> results3 <- OTC(algorithm="ID2", p=0.01, Se=0.95, Sp=0.95,
    group.sz=50, obj.fn=c("ET", "MAR"), alpha=2)
You have specified an overall probability of disease.
A probability vector will be generated based on the algorithm
    specified.
A single group size was provided. The optimal testing
    configuration will be found
 over all possible testing configurations for the specified
     group size.
NOTE: You have specified a maximum group size of 50 or larger.
This function may take a VERY long time to run.
Press 'ESC' if you wish to cancel the submitted statements.
Algorithm: Informative Dorfman testing
[1] "Using simulation"
Block Size = 50
[1] "Using simulation"
[1] "Using simulation"
 Number of minutes running: 2.617833

> # Print the results.
> data.frame("Obj.Fn"=c("O_ET", "O_MAR"),
    "OTC"=c(results3$opt.ET$OTC$Block.sz,
    results3$opt.MAR$OTC$Block.sz),
    "ET.I"=c(round(results3$opt.ET$ET /
    results3$opt.ET$OTC$Block.sz, 4),
    round(results3$opt.MAR$ET / results3$opt.MAR$OTC$Block.sz,
    4)), "PSe"=c(round(results3$opt.ET$PSe, 4),
    round(results3$opt.MAR$PSe, 4)),
    "PSp"=c(round(results3$opt.ET$PSp, 4),
    round(results3$opt.MAR$PSp, 4)))
  Obj.Fn OTC   ET.I    PSe    PSp
1   O_ET  50 0.2264 0.9025 0.9931
2  O_MAR  50 0.2264 0.9025 0.9931

> # Second-stage of OTCs
> results3$opt.ET$OTC$pool.szs
[1] 18 13 11 8
> results3$opt.MAR$OTC$pool.szs
[1] 18 13 11 8


> # Example 4
> # Finding the OTC using non-informative two-stage
> # hierarchical testing

> # This example takes approximately 2.5 minutes to run.
> set.seed(1002)
> results4 <- OTC(algorithm="ID3", p=0.01, Se=0.95, Sp=0.95,
    group.sz=3:40, obj.fn=c("ET", "MAR"), alpha=0.5)
You have specified an overall probability of disease.
A probability vector will be generated based on the algorithm
    specified.
Algorithm: Informative three-stage hierarchical testing
Initial Group Size = 3
Initial Group Size = 4
Initial Group Size = 5

<OUTPUT EDITED>

Initial Group Size = 38
Initial Group Size = 39
Initial Group Size = 40
 Number of minutes running: 2.614333
```

```
> # Print the results.
> data.frame("Obj.Fn"=c("O_ET", "O_MAR"),
    "OTC"=c(results4$opt.ET$OTC$Stage1,
    results4$opt.MAR$OTC$Stage1),
    "ET.I"=c(round(results4$opt.ET$ET /
    results4$opt.ET$OTC$Stage1, 4), round(results4$opt.MAR$ET
    / results4$opt.MAR$OTC$Stage1, 4)),
    "PSe"=c(round(results4$opt.ET$PSe, 4),
    round(results4$opt.MAR$PSe, 4)),
    "PSp"=c(round(results4$opt.ET$PSp, 4),
    round(results4$opt.MAR$PSp, 4)))
  Obj.Fn OTC   ET.I    PSe    PSp
1   O_ET  28 0.1291 0.8574 0.9977
2  O_MAR  28 0.1291 0.8574 0.9977
```

The next example shows how to use `binGroup` to reproduce results from Table A.19.

```
> # Example 5
> # Finding the OTC using two-stage
> # hierarchical testing with O_GR

> # This example takes less than 1 second to run.
> results5 <- OTC(algorithm="D2", p=0.01, Se=0.99, Sp=0.99,
    group.sz=3:40, obj.fn="GR", weights=matrix(data=c(1, 1,
    1000, 1000), nrow=2, ncol=2, byrow=TRUE))
You have specified an overall probability of disease.
A probability vector will be generated based on the algorithm
    specified.
Algorithm: Non-informative two-stage hierarchical (Dorfman)
    testing

Initial Group Size = 3
Initial Group Size = 4
Initial Group Size = 5

<OUTPUT EDITED>

Initial Group Size = 38
Initial Group Size = 39
Initial Group Size = 40
 Number of minutes running: 0.0001666667

> names(results5)
[1] "prob"    "Se"      "Sp"      "opt.ET"  "opt.GR1"
[6] "opt.GR2" "Configs"

> data.frame("Obj.Fn"=c("O_GR", "O_GR"),
    "OTC"=c(paste(results5$opt.GR1$OTC$Stage1, 1, sep="-"),
    paste(results5$opt.GR2$OTC$Stage1, 1, sep="-")),
    "ET.I"=c(round(results5$opt.GR1$ET /
    results5$opt.GR1$OTC$Stage1, 4), round(results5$opt.GR2$ET
    / results5$opt.GR2$OTC$Stage1, 4)),
    "PSe"=c(round(results5$opt.GR1$PSe, 4),
    round(results5$opt.GR2$PSe, 4)),
    "PSp"=c(round(results5$opt.GR1$PSp, 4),
    round(results5$opt.GR2$PSp, 4)))
  Obj.Fn  OTC   ET.I    PSe    PSp
1   O_GR 11-1 0.2035 0.9801 0.9990
2   O_GR  3-1 0.3724 0.9801 0.9997
```

Table A.1: OTC summary for $p = 0.01$ under non-informative group testing. Equally sized groups are optimal at each stage; thus, an OTC of "24-6-1" means that stage 1 has a group of size 24, stage 2 has four groups of size 6, and stage 3 has twenty-four groups of size 1. Differences between O_{ET} and O_{MAR} are highlighted.

Algorithm	S_e	S_p	Objective function	OTC	$E(T)/I$	PS_e	PS_p	$PPPV$	$PNPV$
Two-stage hierarchical	0.99	0.99	O_{ET}	11-1	0.2035	0.9801	0.9990	0.9052	0.9998
			O_{MAR}	11-1	0.2035	0.9801	0.9990	0.9052	0.9998
	0.95	0.95	O_{ET}	11-1	0.2351	0.9025	0.9932	0.5727	0.9990
			O_{MAR}	11-1	0.2351	0.9025	0.9932	0.5727	0.9990
	0.90	0.90	O_{ET}	12-1	0.2742	0.8100	0.9816	0.3081	0.9980
			O_{MAR}	12-1	0.2742	0.8100	0.9816	0.3081	0.9980
	0.99	0.90	O_{ET}	11-1	0.2841	0.9801	0.9815	0.3485	0.9998
			O_{MAR}	11-1	0.2841	0.9801	0.9815	0.3485	0.9998
	0.90	0.99	O_{ET}	11-1	0.1941	0.8100	0.9990	0.8959	0.9981
			O_{MAR}	11-1	0.1941	0.8100	0.9990	0.8959	0.9981
Three-stage hierarchical	0.99	0.99	O_{ET}	25-5-1	0.1354	0.9703	0.9996	0.9604	0.9997
			O_{MAR}	25-5-1	0.1354	0.9703	0.9996	0.9604	0.9997
	0.95	0.95	O_{ET}	24-6-1	0.1443	0.8574	0.9973	0.7634	0.9986
			O_{MAR}	24-6-1	0.1443	0.8574	0.9973	0.7634	0.9986
	0.90	0.90	O_{ET}	24-6-1	0.1562	0.7290	0.9938	0.5437	0.9973
			O_{MAR}	24-6-1	0.1562	0.7290	0.9938	0.5437	0.9973
	0.99	0.90	O_{ET}	24-6-1	0.1708	0.9703	0.9928	0.5780	0.9997
			O_{MAR}	24-6-1	0.1708	0.9703	0.9928	0.5780	0.9997
	0.90	0.99	O_{ET}	25-5-1	0.1229	0.7290	0.9997	0.9564	0.9973
			O_{MAR}	25-5-1	0.1229	0.7290	0.9997	0.9564	0.9973
Array w/o master pooling	0.99	0.99	O_{ET}	25-1	0.1378	0.9703	0.9995	0.9529	0.9997
			O_{MAR}	25-1	0.1378	0.9703	0.9995	0.9529	0.9997
	0.95	0.95	O_{ET}	25-1	0.1475	0.8575	0.9970	0.7456	0.9986
			O_{MAR}	24-1	0.1475	0.8575	0.9972	0.7566	0.9986
	0.90	0.90	O_{ET}	25-1	0.1611	0.7291	0.9926	0.4996	0.9973
			O_{MAR}	24-1	0.1611	0.7291	0.9930	0.5112	0.9973
	0.99	0.90	O_{ET}	23-1	0.1726	0.9703	0.9923	0.5614	0.9997
			O_{MAR}	23-1	0.1726	0.9703	0.9923	0.5614	0.9997
	0.90	0.99	O_{ET}	27-1	0.1279	0.7292	0.9995	0.9410	0.9973
			O_{MAR}	27-1	0.1279	0.7292	0/9995	0.9410	0.9973
Array w/ master pooling	0.99	0.99	O_{ET}	625-25-1	0.1364	0.9606	0.9995	0.9529	0.9996
			O_{MAR}	625-25-1	0.1364	0.9606	0.9995	0.9529	0.9996
	0.95	0.95	O_{ET}	625-25-1	0.1402	0.8146	0.9972	0.7458	0.9981
			O_{MAR}	576-24-1	0.1402	0.8146	0.9974	0.7569	0.9981
	0.90	0.90	O_{ET}	625-25-1	0.1450	0.6562	0.9934	0.4997	0.9965
			O_{MAR}	576-24-1	0.1450	0.6562	0.9937	0.5115	0.9965
	0.99	0.90	O_{ET}	529-23-1	0.1708	0.9606	0.9924	0.5618	0.9996
			O_{MAR}	529-23-1	0.1708	0.9606	0.9924	0.5618	0.9996
	0.90	0.99	O_{ET}	729-27-1	0.1151	0.6563	0.9996	0.9410	0.9965
			O_{MAR}	729-27-1	0.1151	0.6563	0.9996	0.9410	0.9965

Table A.2: OTC summary for $p = 0.05$ under non-informative group testing. Equally sized groups are optimal at each stage; thus, a "24-6-1" means that stage 1 has a group of size 24, stage 2 has four groups of size 6, and stage 3 has twenty-four groups of size 1. There are no differences between the OTCs.

Algorithm	S_e	S_p	Objective function	OTC	$E(T)/I$	PS_e	PS_p	$PPPV$	$PNPV$
Two-stage hierarchical	0.99	0.99	O_{ET}	5-1	0.4317	0.9801	0.9981	0.9642	0.9990
			O_{MAR}	5-1	0.4317	0.9801	0.9981	0.9642	0.9990
	0.95	0.95	O_{ET}	5-1	0.4536	0.9025	0.9892	0.8141	0.9948
			O_{MAR}	5-1	0.4536	0.9025	0.9892	0.8141	0.9948
	0.90	0.90	O_{ET}	6-1	0.4786	0.8100	0.9719	0.6027	0.9898
			O_{MAR}	6-1	0.4786	0.8100	0.9719	0.6027	0.9898
	0.99	0.90	O_{ET}	5-1	0.5013	0.9801	0.9735	0.6605	0.9989
			O_{MAR}	5-1	0.5013	0.9801	0.9735	0.6605	0.9989
	0.90	0.99	O_{ET}	5-1	0.4113	0.8100	0.9982	0.9605	0.9901
			O_{MAR}	5-1	0.4113	0.8100	0.9982	0.9605	0.9901
Three-stage hierarchical	0.99	0.99	O_{ET}	9-3-1	0.3773	0.9703	0.9990	0.9812	0.9984
			O_{MAR}	9-3-1	0.3773	0.9703	0.9990	0.9812	0.9984
	0.95	0.95	O_{ET}	9-3-1	0.3798	0.8574	0.9950	0.8993	0.9925
			O_{MAR}	9-3-1	0.3798	0.8574	0.9950	0.8993	0.9925
	0.90	0.90	O_{ET}	12-4-1	0.3806	0.7290	0.9853	0.7227	0.9857
			O_{MAR}	12-4-1	0.3806	0.7290	0.9853	0.7227	0.9857
	0.99	0.90	O_{ET}	9-3-1	0.4227	0.9703	0.9874	0.8023	0.9984
			O_{MAR}	9-3-1	0.4227	0.9703	0.9874	0.8023	0.9984
	0.90	0.99	O_{ET}	12-4-1	0.3409	0.7290	0.9988	0.9701	0.9859
			O_{MAR}	12-4-1	0.3409	0.7290	0.9988	0.9701	0.9859
Array w/o master pooling	0.99	0.99	O_{ET}	10-1	0.3809	0.9705	0.9986	0.9735	0.9984
			O_{MAR}	10-1	0.3809	0.9705	0.9986	0.9735	0.9984
	0.95	0.95	O_{ET}	10-1	0.3852	0.8581	0.9926	0.8597	0.9925
			O_{MAR}	10-1	0.3852	0.8581	0.9926	0.8597	0.9925
	0.90	0.90	O_{ET}	10-1	0.3907	0.7302	0.9842	0.7086	0.9858
			O_{MAR}	10-1	0.3907	0.7302	0.9842	0.7086	0.9858
	0.99	0.90	O_{ET}	9-1	0.4243	0.9705	0.9839	0.7602	0.9984
			O_{MAR}	9-1	0.4243	0.9705	0.9839	0.7602	0.9984
	0.90	0.99	O_{ET}	11-1	0.3511	0.7301	0.9986	0.9659	0.9860
			O_{MAR}	11-1	0.3511	0.7301	0.9986	0.9659	0.9860
Array w/ master pooling	0.99	0.99	O_{ET}	100-10-1	0.3772	0.9608	0.9986	0.9736	0.9979
			O_{MAR}	100-10-1	0.3772	0.9608	0.9986	0.9736	0.9979
	0.95	0.95	O_{ET}	100-10-1	0.3660	0.8152	0.9930	0.8600	0.9903
			O_{MAR}	100-10-1	0.3660	0.8152	0.9930	0.8600	0.9903
	0.90	0.90	O_{ET}	100-10-1	0.3517	0.6572	0.9858	0.7091	0.9820
			O_{MAR}	100-10-1	0.3517	0.6572	0.9858	0.7091	0.9820
	0.99	0.90	O_{ET}	81-9-1	0.4201	0.9608	0.9842	0.7617	0.9979
			O_{MAR}	81-9-1	0.4201	0.9608	0.9842	0.7617	0.9979
	0.90	0.99	O_{ET}	121-11-1	0.3160	0.6571	0.9988	0.9660	0.9823
			O_{MAR}	121-11-1	0.3160	0.6571	0.9988	0.9660	0.9823

Table A.3: OTC summary for $p = 0.10$ under non-informative group testing. Equally sized groups are optimal at each stage; thus, a "24-6-1" means that stage 1 has a group of size 24, stage 2 has four groups of size 6, and stage 3 has twenty-four groups of size 1. Differences between O_{ET} and O_{MAR} are highlighted.

Algorithm	S_e	S_p	Objective function	OTC	$E(T)/I$	PS_e	PS_p	$PPPV$	$PNPV$
Two-stage hierarchical	0.99	0.99	O_{ET}	4-1	0.5970	0.9801	0.9972	0.9753	0.9978
			O_{MAR}	4-1	0.5970	0.9801	0.9972	0.9753	0.9978
	0.95	0.95	O_{ET}	4-1	0.6095	0.9025	0.9853	0.8722	0.9891
			O_{MAR}	4-1	0.6095	0.9025	0.9853	0.8722	0.9891
	0.90	0.90	O_{ET}	4-1	0.6251	0.8100	0.9683	0.7396	0.9787
			O_{MAR}	4-1	0.6251	0.8100	0.9683	0.7396	0.9787
	0.99	0.90	O_{ET}	4-1	0.6561	0.9801	0.9659	0.7614	0.9977
			O_{MAR}	4-1	0.6561	0.9801	0.9659	0.7614	0.9977
	0.90	0.99	O_{ET}	4-1	0.5661	0.8100	0.9975	0.9728	0.9793
			O_{MAR}	4-1	0.5661	0.8100	0.9975	0.9728	0.9793
Three-stage hierarchical	0.99	0.99	O_{ET}	9-3-1	0.5836	0.9703	0.9981	0.9827	0.9967
			O_{MAR}	9-3-1	0.5836	0.9703	0.9981	0.9827	0.9967
	0.95	0.95	O_{ET}	9-3-1	0.5733	0.8574	0.9905	0.9091	0.9843
			O_{MAR}	9-3-1	0.5733	0.8574	0.9905	0.9091	0.9843
	0.90	0.90	O_{ET}	9-3-1	0.5619	0.7290	0.9808	0.8081	0.9702
			O_{MAR}	9-3-1	0.5619	0.7290	0.9808	0.8081	0.9702
	0.99	0.90	O_{ET}	9-3-1	0.6295	0.9703	0.9772	0.8254	0.9966
			O_{MAR}	6-3-1	0.6295	0.9703	0.9786	0.8345	0.9966
	0.90	0.99	O_{ET}	9-3-1	0.5188	0.7290	0.9984	0.9809	0.9707
			O_{MAR}	9-3-1	0.5188	0.7290	0.9984	0.9809	0.9707
Array w/o master pooling	0.99	0.99	O_{ET}	7-1	0.5821	0.9705	0.9978	0.9800	0.9967
			O_{MAR}	7-1	0.5821	0.9705	0.9978	0.9800	0.9967
	0.95	0.95	O_{ET}	7-1	0.5776	0.8585	0.9888	0.8950	0.9843
			O_{MAR}	7-1	0.5776	0.8585	0.9888	0.8950	0.9843
	0.90	0.90	O_{ET}	7-1	0.5722	0.7310	0.9772	0.7808	0.9703
			O_{MAR}	7-1	0.5722	0.7310	0.9772	0.7808	0.9703
	0.99	0.90	O_{ET}	7-1	0.6250	0.9704	0.9732	0.8009	0.9966
			O_{MAR}	7-1	0.6250	0.9704	0.9732	0.8009	0.9966
	0.90	0.99	O_{ET}	7-1	0.5335	0.7324	0.9982	0.9778	0.9711
			O_{MAR}	7-1	0.5335	0.7324	0.9982	0.9778	0.9711
Array w/ master pooling	0.99	0.99	O_{ET}	49-7-1	0.5767	0.9608	0.9978	0.9800	0.9957
			O_{MAR}	49-7-1	0.5767	0.9608	0.9978	0.9800	0.9957
	0.95	0.95	O_{ET}	49-7-1	0.5491	0.8156	0.9894	0.8952	0.9797
			O_{MAR}	49-7-1	0.5491	0.8156	0.9894	0.8952	0.9797
	0.90	0.90	O_{ET}	49-7-1	0.5154	0.6579	0.9795	0.7812	0.9626
			O_{MAR}	49-7-1	0.5154	0.6579	0.9795	0.7812	0.9626
	0.99	0.90	O_{ET}	49-7-1	0.6191	0.9607	0.9735	0.8013	0.9955
			O_{MAR}	49-7-1	0.6191	0.9607	0.9735	0.8013	0.9955
	0.90	0.99	O_{ET}	49-7-1	0.4806	0.6592	0.9983	0.9778	0.9635
			O_{MAR}	49-7-1	0.4806	0.6592	0.9983	09778	0.9635

Table A.4: Largest differences between operating characteristics for OTCs under non-informative group testing. Values of p range from 0.005 to 0.150 by 0.005. The frequency column denotes the number of times a different OTC was found for O_{ET} and O_{MAR} among these values of p. Differences between operating characteristics are rounded to four decimal places. Note that the operating characteristic value for O_{ET} is always subtracted from the operating characteristic value for O_{MAR}. Thus, a negative value (indicated with parentheses) means that the value for O_{ET} was larger than the value for O_{MAR}.

				Largest difference				
Algorithm	S_e	S_p	Frequency	$E(T)/I$	PS_e	PS_p	$PPPV$	$PNPV$
Two-stage hierarchical	0.99	0.99	0	-	-	-	-	-
	0.95	0.95	3	0.0018	0.0000	0.0049	0.0262	0.0001
	0.90	0.90	4	0.0023	0.0000	0.0054	0.0345	0.0001
	0.99	0.90	7	0.0056	0.0000	0.0096	0.0382	0.0000
	0.90	0.99	0	-	-	-	-	-
Three-stage hierarchical	0.99	0.99	0	-	-	-	-	-
	0.95	0.95	1	0.0014	0.0000	0.0051	0.0296	0.0001
	0.90	0.90	3	0.0015	0.0000	0.0049	0.0575	0.0001
	0.99	0.90	7	0.0041	(0.0098)	0.0136	0.0580	(0.0015)
	0.90	0.99	1	0.0000	0.0000	0.0002	0.0097	0.0000
Array w/o master pooling	0.99	0.99	0	-	-	-	-	-
	0.95	0.95	5	0.0010	0.0018	0.0026	0.0195	0.0003
	0.90	0.90	8	0.0028	0.0022	0.0054	0.0305	0.0005
	0.99	0.90	5	0.0043	0.0005	0.0076	0.0317	0.0001
	0.90	0.99	1	0.0000	0.0006	0.0001	0.0042	0.0000
Array w/ master pooling	0.99	0.99	2	0.0005	0.0006	0.0008	0.0046	0.0001
	0.95	0.95	4	0.0012	0.0017	0.0026	0.0198	0.0003
	0.90	0.90	8	0.0015	0.0018	0.0051	0.0307	0.0005
	0.99	0.90	5	0.0048	0.0005	0.0077	0.0327	0.0001
	0.90	0.99	2	0.0003	0.0026	0.0005	0.0048	0.0004

Table A.5: OTC summary for $E(P_i) = 0.01$ under informative group testing. Multiple initial group sizes for two-stage hierarchical algorithms are found within a block size of 50, so they are not displayed here. The full OTCs are given in Tables A.9 and A.10. Differences between O_{ET} and O_{MAR} are highlighted.

				$\alpha = 2$						$\alpha = 0.5$					
Algorithm	S_e	S_p	Objective function	Initial group size for OTC	$E(T)/I$	PS_e^W	PS_p^W	$PPPV^W$	$PNPV^W$	Initial group size for OTC	$E(T)/I$	PS_e^W	PS_p^W	$PPPV^W$	$PNPV^W$
Two-stage hierarchical	0.99	0.99	O_{ET}	-	0.1947	0.9801	0.9991	0.9204	0.9998	-	0.1683	0.9801	0.9992	0.9245	0.9998
			O_{MAR}	-	0.1947	0.9801	0.9991	0.9204	0.9998	-	0.1683	0.9801	0.9992	0.9245	0.9998
	0.95	0.95	O_{ET}	-	0.2264	0.9025	0.9931	0.5687	0.9990	-	0.2019	0.9025	0.9943	0.6115	0.9990
			O_{MAR}	-	0.2264	0.9025	0.9931	0.5687	0.9990	-	0.2019	0.9025	0.9943	0.6115	0.9990
	0.90	0.90	O_{ET}	-	0.2657	0.8100	0.9822	0.3143	0.9981	-	0.2439	0.8100	0.9843	0.3389	0.9981
			O_{MAR}	-	0.2657	0.8100	0.9822	0.3143	0.9981	-	0.2439	0.8100	0.9843	0.3389	0.9981
	0.90	0.99	O_{ET}	-	0.2754	0.9801	0.9813	0.3459	0.9998	-	0.2511	0.9801	0.9837	0.3735	0.9998
			O_{MAR}	-	0.2754	0.9801	0.9813	0.3459	0.9998	-	0.2511	0.9801	0.9837	0.3735	0.9998
	0.90	0.99	O_{ET}	-	0.1854	0.8100	0.9990	0.8937	0.9981	-	0.1611	0.8100	0.9993	0.9167	0.9981
			O_{MAR}	-	0.1854	0.8100	0.9990	0.8937	0.9981	-	0.1611	0.8100	0.9993	0.9167	0.9981
Three-stage hierarchical	0.99	0.99	O_{ET}	26	0.1285	0.9703	0.9996	0.9635	0.9997	33	0.1197	0.9703	0.9996	0.9637	0.9997
			O_{MAR}	26	0.1285	0.9703	0.9996	0.9635	0.9997	33	0.1197	0.9703	0.9996	0.9637	0.9997
	0.95	0.95	O_{ET}	26	0.1375	0.8574	0.9974	0.7655	0.9986	28	0.1291	0.8574	0.9977	0.7883	0.9986
			O_{MAR}	26	0.1375	0.8574	0.9974	0.7655	0.9986	28	0.1291	0.8574	0.9977	0.7883	0.9986
	0.90	0.90	O_{ET}	26	0.1497	0.7290	0.9939	0.5422	0.9973	29	0.1422	0.7290	0.9942	0.5558	0.9973
			O_{MAR}	26	0.1497	0.7290	0.9939	0.5422	0.9973	29	0.1422	0.7290	0.9942	0.5558	0.9973
	0.90	0.99	O_{ET}	26	0.1638	0.9703	0.9930	0.5772	0.9997	28	0.1554	0.9703	0.9935	0.5990	0.9997
			O_{MAR}	26	0.1638	0.9703	0.9930	0.5772	0.9997	28	0.1554	0.9703	0.9935	0.5990	0.9997
	0.90	0.99	O_{ET}	26	0.1168	0.7290	0.9997	0.9597	0.9973	37	0.1078	0.7290	0.9997	0.9548	0.9973
			O_{MAR}	26	0.1168	0.7290	0.9997	0.9597	0.9973	37	0.1078	0.7290	0.9997	0.9548	0.9973
Array w/o master pooling	0.99	0.99	O_{ET}	25	0.1349	0.9703	0.9995	0.9556	0.9997	28	0.1277	0.9703	0.9995	0.9542	0.9997
			O_{MAR}	25	0.1349	0.9703	0.9995	0.9556	0.9997	28	0.1277	0.9703	0.9995	0.9542	0.9997
	0.95	0.95	O_{ET}	25	0.1448	0.8575	0.9972	0.7548	0.9986	28	0.1379	0.8574	0.9971	0.7491	0.9986
			O_{MAR}	25	0.1448	0.8575	0.9972	0.7548	0.9986	27	0.1379	0.8574	0.9972	0.7581	0.9986
	0.90	0.90	O_{ET}	25	0.1585	0.7291	0.9929	0.5089	0.9973	28	0.1519	0.7290	0.9927	0.5017	0.9973
			O_{MAR}	25	0.1585	0.7291	0.9929	0.5089	0.9973	27	0.1519	0.7290	0.9930	0.5113	0.9972
	0.90	0.99	O_{ET}	23	0.1699	0.9703	0.9926	0.5702	0.9997	25	0.1631	0.9703	0.9926	0.5698	0.9997
			O_{MAR}	23	0.1699	0.9703	0.9926	0.5702	0.9997	25	0.1631	0.9703	0.9926	0.5698	0.9997
	0.90	0.99	O_{ET}	27	0.1251	0.7293	0.9996	0.9444	0.9973	30	0.1181	0.7291	0.9996	0.9439	0.9973
			O_{MAR}	27	0.1251	0.7293	0.9996	0.9444	0.9973	30	0.1181	0.7291	0.9996	0.9439	0.9973

Table A.6: OTC summary for $E(P_i) = 0.05$ under informative group testing. Multiple initial group sizes for two-stage hierarchical algorithms are found within a block size of 50, so they are not displayed here. The full OTCs are given in Tables A.11 and A.12. Differences between O_{ET} and O_{MAR} are highlighted.

				$\alpha = 2$						$\alpha = 0.5$					
Algorithm	S_e	S_p	Objective function	Initial group size for OTC	$E(T)/I$	PS_e^W	PS_p^W	$PPPV^W$	$PNPV^W$	Initial group size for OTC	$E(T)/I$	PS_e^W	PS_p^W	$PPPV^W$	$PNPV^W$
Two-stage hierarchical	0.99	0.99	O_{ET}	-	0.4101	0.9801	0.9981	0.9645	0.9990	-	0.3584	0.9801	0.9984	0.9705	0.9990
			O_{MAR}	-	0.4101	0.9801	0.9981	0.9645	0.9990	-	0.3584	0.9801	0.9984	0.9705	0.9990
	0.95	0.95	O_{ET}	-	0.4321	0.9025	0.9892	0.8152	0.9948	-	0.3830	0.9025	0.9908	0.8371	0.9948
			O_{MAR}	-	0.4321	0.9025	0.9892	0.8152	0.9948	-	0.3830	0.9025	0.9908	0.8371	0.9948
	0.90	0.90	O_{ET}	-	0.4586	0.8100	0.9733	0.6149	0.9898	-	0.4124	0.8100	0.9761	0.6403	0.9899
			O_{MAR}	-	0.4586	0.8100	0.9733	0.6149	0.9898	-	0.4124	0.8100	0.9761	0.6403	0.9899
	0.99	0.90	O_{ET}	-	0.4798	0.9801	0.9736	0.6619	0.9989	-	0.4308	0.9812	0.9746	0.6703	0.9990
			O_{MAR}	-	0.4798	0.9801	0.9736	0.6619	0.9989	-	0.4311	0.9801	0.9767	0.6885	0.9989
	0.90	0.99	O_{ET}	-	0.3898	0.8100	0.9983	0.9609	0.9901	-	0.3411	0.8100	0.9986	0.9675	0.9901
			O_{MAR}	-	0.3898	0.8100	0.9983	0.9609	0.9901	-	0.3411	0.8100	0.9986	0.9675	0.9901
Three-stage hierarchical	0.99	0.99	O_{ET}	10	0.3687	0.9725	0.9990	0.9803	0.9986	11	0.3365	0.9757	0.9988	0.9768	0.9987
			O_{MAR}	10	0.3687	0.9725	0.9990	0.9803	0.9986	11	0.3365	0.9757	0.9988	0.9768	0.9987
	0.95	0.95	O_{ET}	11	0.3709	0.8574	0.9940	0.8821	0.9925	11	0.3433	0.8822	0.9934	0.8763	0.9938
			O_{MAR}	11	0.3709	0.8574	0.9940	0.8821	0.9925	11	0.3433	0.8822	0.9934	0.8763	0.9938
	0.90	0.90	O_{ET}	12	0.3724	0.7290	0.9862	0.7357	0.9857	10	0.3503	0.7582	0.9871	0.7560	0.9873
			O_{MAR}	12	0.3724	0.7290	0.9862	0.7357	0.9857	10	0.3503	0.7582	0.9871	0.7560	0.9873
	0.99	0.90	O_{ET}	9	0.4136	0.9727	0.9834	0.7554	0.9985	10	0.3833	0.9738	0.9852	0.7756	0.9986
			O_{MAR}	8	0.4140	0.9729	0.9853	0.7772	0.9986	10	0.3833	0.9738	0.9852	0.7756	0.9986
	0.90	0.99	O_{ET}	12	0.3315	0.7290	0.9989	0.9727	0.9859	15	0.3052	0.7515	0.9990	0.9749	0.9871
			O_{MAR}	12	0.3336	0.7449	0.9989	0.9730	0.9867	11	0.3076	0.7735	0.9989	0.9747	0.9882
Array w/o master pooling	0.99	0.99	O_{ET}	10	0.3677	0.9705	0.9988	0.9761	0.9984	13	0.3372	0.9703	0.9986	0.9727	0.9985
			O_{MAR}	10	0.3677	0.9705	0.9988	0.9761	0.9984	13	0.3372	0.9703	0.9986	0.9727	0.9985
	0.95	0.95	O_{ET}	10	0.3731	0.8582	0.9933	0.8703	0.9925	13	0.3421	0.8575	0.9924	0.8551	0.9926
			O_{MAR}	10	0.3731	0.8582	0.9933	0.8703	0.9925	13	0.3421	0.8575	0.9924	0.8551	0.9926
	0.90	0.90	O_{ET}	11	0.3784	0.7295	0.9836	0.7004	0.9857	13	0.3484	0.7292	0.9837	0.7000	0.9859
			O_{MAR}	11	0.3784	0.7295	0.9836	0.7004	0.9857	13	0.3484	0.7292	0.9837	0.7000	0.9859
	0.99	0.90	O_{ET}	10	0.4113	0.9704	0.9829	0.7494	0.9984	11	0.3826	0.9703	0.9840	0.7610	0.9984
			O_{MAR}	10	0.4113	0.9704	0.9829	0.7494	0.9984	11	0.3826	0.9703	0.9840	0.7610	0.9984
	0.90	0.99	O_{ET}	11	0.3380	0.7303	0.9988	0.9693	0.9860	13	0.3060	0.7294	0.9988	0.9698	0.9861
			O_{MAR}	11	0.3380	0.7303	0.9988	0.9693	0.9860	13	0.3060	0.7294	0.9988	0.9698	0.9861

Table A.7: OTC summary for $E(P_i) = 0.10$ under informative group testing. Multiple initial group sizes for two-stage hierarchical algorithms are found within a block size of 50, so they are not displayed here. The full OTCs are given in Tables A.13 and A.14. Differences between O_{ET} and O_{MAR} are highlighted.

				$\alpha = 2$						$\alpha = 0.5$					
Algorithm	S_e	S_p	Objective function	Initial group size for OTC	$E(T)/I$	PS_e^W	PS_p^W	$PPPV^W$	$PNPV^W$	Initial group size for OTC	$E(T)/I$	PS_e^W	PS_p^W	$PPPV^W$	$PNPV^W$
Two-stage hierarchical	0.99	0.99	O_{ET}	-	0.5674	0.9801	0.9975	0.9772	0.9978	-	0.4868	0.9833	0.9977	0.9793	0.9981
			O_{MAR}	-	0.5674	0.9801	0.9975	0.9772	0.9978	-	0.4868	0.9833	0.9977	0.9793	0.9981
	0.95	0.95	O_{ET}	-	0.5815	0.9025	0.9863	0.8798	0.9891	-	0.5054	0.9148	0.9873	0.8887	0.9905
			O_{MAR}	-	0.5815	0.9025	0.9863	0.8798	0.9891	-	0.5054	0.9148	0.9873	0.8887	0.9905
	0.90	0.90	O_{ET}	-	0.5973	0.8100	0.9681	0.7382	0.9787	-	0.5271	0.8333	0.9695	0.7522	0.9813
			O_{MAR}	-	0.5973	0.8100	0.9681	0.7382	0.9787	-	0.5271	0.8333	0.9695	0.7522	0.9813
	0.99	0.90	O_{ET}	-	0.6277	0.9807	0.9657	0.7606	0.9978	-	0.5491	0.9839	0.9678	0.7725	0.9982
			O_{MAR}	-	0.6283	0.9801	0.9679	0.7721	0.9977	-	0.5496	0.9833	0.9700	0.7844	0.9981
	0.90	0.99	O_{ET}	-	0.5383	0.8100	0.9977	0.9749	0.9793	-	0.4654	0.8333	0.9979	0.9774	0.9818
			O_{MAR}	-	0.5383	0.8100	0.9977	0.9749	0.9793	-	0.4654	0.8333	0.9979	0.9774	0.9818
Three-stage hierarchical	0.99	0.99	O_{ET}	5	0.5567	0.9764	0.9981	0.9824	0.9974	40	0.5074	0.9733	0.9976	0.9786	0.9970
			O_{MAR}	5	0.5567	0.9764	0.9981	0.9824	0.9974	40	0.5074	0.9733	0.9976	0.9786	0.9970
	0.95	0.95	O_{ET}	8	0.5550	0.8691	0.9891	0.8985	0.9855	40	0.5050	0.8711	0.9876	0.8863	0.9857
			O_{MAR}	8	0.5550	0.8691	0.9891	0.8985	0.9855	40	0.5050	0.8711	0.9876	0.8863	0.9857
	0.90	0.90	O_{ET}	8	0.5461	0.7500	0.9781	0.7916	0.9724	40	0.4994	0.7469	0.9739	0.7604	0.9719
			O_{MAR}	8	0.5461	0.7500	0.9781	0.7916	0.9724	40	0.4994	0.7469	0.9739	0.7604	0.9719
	0.99	0.90	O_{ET}	5	0.6044	0.9764	0.9764	0.8216	0.9973	6	0.5611	0.9774	0.9770	0.8254	0.9974
			O_{MAR}	5	0.6044	0.9764	0.9764	0.8216	0.9973	6	0.5611	0.9774	0.9770	0.8254	0.9974
	0.90	0.99	O_{ET}	8	0.5055	0.7500	0.9982	0.9783	0.9729	40	0.4442	0.7469	0.9980	0.9764	0.9726
			O_{MAR}	6	0.5203	0.7874	0.9978	0.9759	0.9769	40	0.4445	0.7537	0.9980	0.9767	0.9733
Array w/o master pooling	0.99	0.99	O_{ET}	7	0.5585	0.9706	0.9981	0.9823	0.9967	8	0.5092	0.9704	0.9982	0.9837	0.9967
			O_{MAR}	7	0.5585	0.9706	0.9981	0.9823	0.9967	8	0.5092	0.9704	0.9982	0.9837	0.9967
	0.95	0.95	O_{ET}	7	0.5563	0.8587	0.9900	0.9050	0.9844	9	0.5076	0.8576	0.9892	0.8978	0.9843
			O_{MAR}	7	0.5563	0.8587	0.9900	0.9050	0.9844	9	0.5076	0.8576	0.9892	0.8978	0.9843
	0.90	0.90	O_{ET}	8	0.5494	0.7297	0.9757	0.7697	0.9701	9	0.5040	0.7294	0.9777	0.7842	0.9702
			O_{MAR}	8	0.5494	0.7297	0.9757	0.7697	0.9701	9	0.5040	0.7294	0.9777	0.7842	0.9702
	0.99	0.90	O_{ET}	7	0.6024	0.9705	0.9757	0.8161	0.9966	8	0.5558	0.9704	0.9769	0.8236	0.9966
			O_{MAR}	7	0.6024	0.9705	0.9757	0.8161	0.9966	8	0.5558	0.9704	0.9769	0.8236	0.9966
	0.90	0.99	O_{ET}	8	0.5099	0.7302	0.9980	0.9761	0.9708	9	0.4617	0.7298	0.9982	0.9788	0.9708
			O_{MAR}	8	0.5099	0.7302	0.9980	0.9761	0.9708	9	0.4617	0.7298	0.9982	0.9788	0.9708

Table A.8: Largest differences between operating characteristics for OTCs under informative group testing. Values of $E(P_i) = p$ range from 0.005 to 0.150 by 0.005. The frequency column denotes the number of times a different OTC was found among these values of p. Differences between operating characteristics are rounded to four decimal places. Note that the operating characteristic value for O_{ET} is always subtracted from the operating characteristic value for O_{MAR}. Thus, a negative value (indicated with parentheses) means that the value for O_{ET} was larger than the value for O_{MAR}.

					Largest difference				
Algorithm	α	S_e	S_p	Frequency	$E(T)/I$	PS_e^W	PS_p^W	$PPPV^W$	$PNPV^W$
Two-stage hierarchical	2	0.99	0.99	0	-	-	-	-	-
		0.95	0.95	7	0.0006	(0.0023)	0.0011	0.0156	0.0004
		0.90	0.90	12	0.0010	(0.0052)	0.0023	0.0160	0.0007
		0.99	0.90	12	0.0011	(0.0008)	0.0022	0.0182	(0.0001)
		0.90	0.99	2	0.0003	0.0052	0.0000	0.0002	0.0007
	0.5	0.99	0.99	0	-	-	-	-	-
		0.95	0.95	3	0.0003	(0.0035)	0.0011	0.0102	0.0003
		0.90	0.90	15	0.0008	(0.0103)	0.0022	0.0277	0.0007
		0.99	0.90	16	0.0012	(0.0011)	0.0022	0.0194	(0.0001)
		0.90	0.99	11	0.0006	0.0078	(0.0002)	(0.0028)	0.0007
Three-stage hierarchical	2	0.99	0.99	1	0.0000	(0.0019)	0.0002	0.0057	(0.0001)
		0.95	0.95	2	0.0035	0.0219	0.0033	0.0270	0.0034
		0.90	0.90	6	0.0044	0.0152	0.0062	0.0409	0.0023
		0.99	0.90	4	0.0035	0.0006	0.0066	0.0445	0.0001
		0.90	0.99	14	0.0180	0.0500	0.0003	0.0046	0.0064
	0.5	0.99	0.99	1	0.0000	0.0001	0.0001	0.0018	0.0000
		0.95	0.95	0	-	-	-	-	-
		0.90	0.90	3	0.0010	0.0250	0.0033	0.0296	0.0025
		0.99	0.90	5	0.0022	0.0034	0.0070	0.0385	0.0005
		0.90	0.99	9	0.0057	0.0355	0.0003	0.0051	0.0030
Array w/o master pooling	2	0.99	0.99	1	0.0003	0.0004	0.0005	0.0039	0.0001
		0.95	0.95	2	0.0011	0.0012	0.0027	0.0169	0.0002
		0.90	0.90	5	0.0016	0.0012	0.0040	0.0265	0.0003
		0.99	0.90	4	0.0028	0.0003	0.0053	0.0277	0.0001
		0.90	0.99	0	-	-	-	-	-
	0.5	0.99	0.99	0	-	-	-	-	-
		0.95	0.95	4	0.0003	0.0004	0.0015	0.0129	0.0001
		0.90	0.90	14	0.0015	0.0004	0.0032	0.0194	0.0002
		0.99	0.90	8	0.0024	0.0001	0.0041	0.0211	0.0000
		0.90	0.99	1	0.0003	0.0005	0.0003	0.0027	0.0001

Table A.9: Full OTCs for $E(P_i) = 0.01$ under informative two-stage hierarchical group testing. There are no differences in the OTCs for O_{ET} and O_{MAR}.

			Objective	Block		Group sizes				
α	S_e	S_p	function	size	$E(T)/I$	I_{11}	I_{12}	I_{13}	I_{14}	I_{15}
2	0.99	0.99	O_{ET}	50	0.1947	16	11	9	8	6
			O_{MAR}	50	0.1947	16	11	9	8	6
	0.95	0.95	O_{ET}	50	0.2264	18	13	11	8	
			O_{MAR}	50	0.2264	18	13	11	8	
	0.90	0.90	O_{ET}	50	0.2657	18	13	11	8	
			O_{MAR}	50	0.2657	18	13	11	8	
	0.99	0.90	O_{ET}	50	0.2754	18	13	11	8	
			O_{MAR}	50	0.2754	18	13	11	8	
	0.90	0.99	O_{ET}	50	0.1854	18	13	11	8	
			O_{MAR}	50	0.1854	18	13	11	8	
0.5	0.99	0.99	O_{ET}	50	0.1683	25	12	8	5	
			O_{MAR}	50	0.1683	25	12	8	5	
	0.95	0.95	O_{ET}	50	0.2019	25	12	8	5	
			O_{MAR}	50	0.2019	25	12	8	5	
	0.90	0.90	O_{ET}	50	0.2439	25	12	8	5	
			O_{MAR}	50	0.2439	25	12	8	5	
	0.99	0.90	O_{ET}	50	0.2511	25	12	8	5	
			O_{MAR}	50	0.2511	25	12	8	5	
	0.90	0.99	O_{ET}	50	0.1611	25	12	8	5	
			O_{MAR}	50	0.1611	25	12	8	5	

Table A.10: Full OTCs for $E(P_i) = 0.01$ under informative three-stage hierarchical group testing. There are no differences in the OTCs for O_{ET} and O_{MAR}.

			Objective			Group sizes				
α	S_e	S_p	function	I_{11}	$E(T)/I$	I_{21}	I_{22}	I_{23}	I_{24}	I_{25}
2	0.99	0.99	O_{ET}	26	0.1285	9	5	5	4	3
			O_{MAR}	26	0.1285	9	5	5	4	3
	0.95	0.95	O_{ET}	26	0.1375	10	7	5	4	
			O_{MAR}	26	0.1375	10	7	5	4	
	0.90	0.90	O_{ET}	26	0.1497	10	7	5	4	
			O_{MAR}	26	0.1497	10	7	5	4	
	0.99	0.90	O_{ET}	26	0.1638	10	7	5	4	
			O_{MAR}	26	0.1638	10	7	5	4	
	0.90	0.99	O_{ET}	26	0.1168	9	5	5	4	3
			O_{MAR}	26	0.1168	9	5	5	4	3
0.5	0.99	0.99	O_{ET}	33	0.1197	15	6	5	4	3
			O_{MAR}	33	0.1197	15	6	5	4	3
	0.95	0.95	O_{ET}	28	0.1291	13	7	5	3	
			O_{MAR}	28	0.1291	13	7	5	3	
	0.90	0.90	O_{ET}	29	0.1422	14	7	5	3	
			O_{MAR}	29	0.1422	14	7	5	3	
	0.99	0.90	O_{ET}	28	0.1554	13	7	5	3	
			O_{MAR}	28	0.1554	13	7	5	3	
	0.90	0.99	O_{ET}	37	0.1078	17	7	6	4	3
			O_{MAR}	37	0.1078	17	7	6	4	3

Table A.11: Full OTCs for $E(P_i) = 0.05$ under informative two-stage hierarchical group testing. Differences between O_{ET} and O_{MAR} are highlighted.

			Objective	Block		Group sizes								
α	S_e	S_p	function	size	$E(T)/I$	I_{11}	I_{12}	I_{13}	I_{14}	I_{15}	I_{16}	I_{17}	I_{18}	I_{19}
2	0.99	0.99	O_{ET}	50	0.4101	9	7	6	6	5	5	4	4	4
			O_{MAR}	50	0.4101	9	7	6	6	5	5	4	4	4
	0.95	0.95	O_{ET}	50	0.4321	9	7	6	6	5	5	4	4	4
			O_{MAR}	50	0.4321	9	7	6	6	5	5	4	4	4
	0.90	0.90	O_{ET}	50	0.4586	10	8	7	6	5	5	5	4	
			O_{MAR}	50	0.4586	10	8	7	6	5	5	5	4	
	0.99	0.90	O_{ET}	50	0.4798	9	7	6	6	5	5	4	4	4
			O_{MAR}	50	0.4798	9	7	6	6	5	5	4	4	4
	0.90	0.99	O_{ET}	50	0.3898	9	7	6	6	5	5	4	4	4
			O_{MAR}	50	0.3898	9	7	6	6	5	5	4	4	4
0.5	0.99	0.99	O_{ET}	50	0.3584	16	9	6	5	4	4	3	3	
			O_{MAR}	50	0.3584	16	9	6	5	4	4	3	3	
	0.95	0.95	O_{ET}	50	0.3830	16	9	6	5	4	4	3	3	
			O_{MAR}	50	0.3830	16	9	6	5	4	4	3	3	
	0.90	0.90	O_{ET}	50	0.4124	17	9	7	5	5	4	3		
			O_{MAR}	50	0.4124	17	9	7	5	5	4	3		
	0.99	0.90	O_{ET}	50	0.4308	17	9	6	5	5	4	3	1	
			O_{MAR}	50	0.4311	16	9	6	5	4	4	3	3	
	0.90	0.99	O_{ET}	50	0.3411	16	9	6	5	4	4	3	3	
			O_{MAR}	50	0.3411	16	9	6	5	4	4	3	3	

Table A.12: Full OTCs for $E(P_i) = 0.05$ under informative three-stage hierarchical group testing. Differences between O_{ET} and O_{MAR} are highlighted.

			Objective			Group sizes			
α	S_e	S_p	function	I_{11}	$E(T)/I$	I_{21}	I_{22}	I_{23}	I_{24}
2	0.99	0.99	O_{ET}	10	0.3687	4	3	2	1
			O_{MAR}	10	0.3687	4	3	2	1
	0.95	0.95	O_{ET}	11	0.3709	5	3	3	
			O_{MAR}	11	0.3709	5	3	3	
	0.90	0.90	O_{ET}	12	0.3724	5	4	3	
			O_{MAR}	12	0.3724	5	4	3	
	0.99	0.90	O_{ET}	9	0.4136	5	3	1	
			O_{MAR}	8	0.4140	4	3	1	
	0.90	0.99	O_{ET}	12	0.3315	5	4	3	
			O_{MAR}	12	0.3336	5	3	3	1
0.5	0.99	0.99	O_{ET}	11	0.3365	6	3	1	1
			O_{MAR}	11	0.3365	6	3	1	1
	0.95	0.95	O_{ET}	11	0.3433	6	3	1	1
			O_{MAR}	11	0.3433	6	3	1	1
	0.90	0.90	O_{ET}	10	0.3503	6	3	1	
			O_{MAR}	10	0.3503	6	3	1	
	0.99	0.90	O_{ET}	10	0.3833	6	3	1	
			O_{MAR}	10	0.3833	6	3	1	
	0.90	0.99	O_{ET}	15	0.3052	7	4	3	1
			O_{MAR}	11	0.3076	6	3	1	1

Table A.13: Full OTCs for $E(P_i) = 0.10$ under informative two-stage hierarchical group testing. Differences between O_{ET} and O_{MAR} are highlighted.

			Objective	Block		Group sizes												
α	S_e	S_p	function	size	$E(T)/I$	I_{11}	I_{12}	I_{13}	I_{14}	I_{15}	I_{16}	I_{17}	I_{18}	I_{19}	$I_{1,10}$	$I_{1,11}$	$I_{1,12}$	$I_{1,13}$
2	0.99	0.99	O_{ET}	50	0.5674	7	6	5	4	4	4	4	4	3	3	3	3	
			O_{MAR}	50	0.5674	7	6	5	4	4	4	4	4	3	3	3	3	
	0.95	0.95	O_{ET}	50	0.5815	7	6	5	4	4	4	4	4	3	3	3	3	
			O_{MAR}	50	0.5815	7	6	5	4	4	4	4	4	3	3	3	3	
	0.90	0.90	O_{ET}	50	0.5973	8	6	5	5	4	4	4	4	4	3	3		
			O_{MAR}	50	0.5973	8	6	5	5	4	4	4	4	4	3	3		
	0.99	0.90	O_{ET}	50	0.6277	7	6	5	5	4	4	4	4	4	3	3	1	
			O_{MAR}	50	0.6283	7	6	5	4	4	4	4	4	3	3	3	3	
	0.90	0.99	O_{ET}	50	0.5383	7	6	5	4	4	4	4	4	3	3	3	3	
			O_{MAR}	50	0.5383	7	6	5	4	4	4	4	4	3	3	3	3	
0.5	0.99	0.99	O_{ET}	50	0.4868	13	7	5	4	4	4	3	3	3	1	1	1	1
			O_{MAR}	50	0.4868	13	7	5	4	4	4	3	3	3	1	1	1	1
	0.95	0.95	O_{ET}	50	0.5054	13	7	5	5	4	4	3	3	3	1	1	1	
			O_{MAR}	50	0.5054	13	7	5	5	4	4	3	3	3	1	1	1	
	0.90	0.90	O_{ET}	50	0.5271	14	7	6	5	4	4	4	3	1	1	1		
			O_{MAR}	50	0.5271	14	7	6	5	4	4	4	3	1	1	1		
	0.99	0.90	O_{ET}	50	0.5491	13	7	6	5	4	4	3	3	1	1	1	1	1
			O_{MAR}	50	0.5496	13	7	5	4	4	4	3	3	3	1	1	1	1
	0.90	0.99	O_{ET}	50	0.4654	13	7	5	5	4	4	3	3	3	1	1	1	
			O_{MAR}	50	0.4654	13	7	5	5	4	4	3	3	3	1	1	1	

Table A.14: Full OTCs for $E(P_i) = 0.10$ under informative three-stage hierarchical group testing. Differences between O_{ET} and O_{MAR} are highlighted.

			Objective			Group sizes									
α	S_e	S_p	function	I_{11}	$E(T)/I$	I_{21}	I_{22}	I_{23}	I_{24}	I_{25}	I_{26}	I_{27}	I_{28}	I_{29}	$I_{2,10}$
2	0.99	0.99	O_{ET}	5	0.5567	3	1	1							
			O_{MAR}	5	0.5567	3	1	1							
	0.95	0.95	O_{ET}	8	0.5550	4	3	1							
			O_{MAR}	8	0.5550	4	3	1							
	0.90	0.90	O_{ET}	8	0.5461	4	3	1							
			O_{MAR}	8	0.5461	4	3	1							
	0.99	0.90	O_{ET}	5	0.6044	3	1	1							
			O_{MAR}	5	0.6044	3	1	1							
	0.90	0.99	O_{ET}	8	0.5055	4	3	1							
			O_{MAR}	6	0.5203	3	1	1	1						
0.5	0.99	0.99	O_{ET}	40	0.5074	12	6	5	4	4	3	3	1	1	1
			O_{MAR}	40	0.5074	12	6	5	4	4	3	3	1	1	1
	0.95	0.95	O_{ET}	40	0.5050	12	6	5	4	4	3	3	1	1	1
			O_{MAR}	40	0.5050	12	6	5	4	4	3	3	1	1	1
	0.90	0.90	O_{ET}	40	0.4994	12	7	5	4	4	3	3	1	1	
			O_{MAR}	40	0.4994	12	7	5	4	4	3	3	1	1	
	0.99	0.90	O_{ET}	6	0.5611	4	1	1							
			O_{MAR}	6	0.5611	4	1	1							
	0.90	0.99	O_{ET}	40	0.4442	12	7	5	4	4	3	3	1	1	
			O_{MAR}	40	0.4445	12	6	5	4	4	3	3	1	1	1

Table A.15: OTC summary for HIV testing using $S_e = 0.963$, $S_p = 0.9903$, and $p = 0.004$ with non-informative group testing. Equally sized groups are optimal at each stage; thus, an OTC of "24-6-1" means that stage 1 has a group of size 24, stage 2 has four groups of size 6, and stage 3 has twenty-four groups of size 1. There are no differences in the OTCs for O_{ET} and O_{MAR}.

Algorithm	Objective function	OTC	$E(T)/I$	PS_e	PS_p	$PPPV$	$PNPV$
Two-stage	O_{ET}	17-1	0.1313	0.9274	0.9993	0.8478	0.9997
hierarchical	O_{MAR}	17-1	0.1313	0.9274	0.9993	0.8478	0.9997
Three-stage	O_{ET}	49-7-1	0.0732	0.8931	0.9998	0.9402	0.9996
hierarchical	O_{MAR}	49-7-1	0.0732	0.8931	0.9998	0.9402	0.9996
Array w/o	O_{ET}	44-1	0.0749	0.8931	0.9997	0.9348	0.9996
master pooling	O_{MAR}	44-1	0.0749	0.8931	0.9997	0.9348	0.9996
Array w/	O_{ET}	1936-44-1	0.0721	0.8600	0.9998	0.9348	0.9994
master pooling	O_{MAR}	1936-44-1	0.0721	0.8600	0.9998	0.9348	0.9994

Table A.16: OTC summary for chlamydia testing. The test accuracies are $S_e = 0.805$ and $S_p = 0.96$ for females and $S_e = 0.93$ and $S_p = 0.95$ for males. For non-informative group testing, $p = 0.08$ for females, $p = 0.081$ for males, and equally sized groups are optimal at each stage (see Table A.1's caption for how group sizes are denoted). For informative group testing, $P_i \sim \text{beta}(1.1, 11.54)$ for females, $P_i \sim \text{beta}(1.8, 18.20)$ for males, and full OTCs are provided in Tables A.17 and A.18. Results are not displayed for informative array testing with master pooling because no group testing algorithms have been proposed for it. Differences between O_{ET} and O_{MAR} are highlighted.

			Non-informative						Informative					
	Algorithm	Objective function	OTC	$E(T)/I$	PS_e	PS_p	$PPPV$	$PNPV$	Stage 1 size	$E(T)/I$	PS_e^W	PS_p^W	$PPPV^W$	$PNPV^W$
Female	Two-stage hierarchical	O_{ET}	5-1	0.5008	0.6480	0.9897	0.8457	0.9700	-	0.4757	0.6480	0.9901	0.8620	0.9672
		O_{MAR}	5-1	0.5008	0.6480	0.9897	0.8457	0.9700	-	0.4761	0.6480	0.9910	0.8725	0.9673
	Three-stage hierarchical	O_{ET}	12-4-1	0.4099	0.5217	0.9937	0.8789	0.9598	19	0.4102	0.5217	0.9930	0.8773	0.9561
		O_{MAR}	12-4-1	0.4099	0.5217	0.9937	0.8789	0.9598	14	0.4113	0.5479	0.9933	0.8868	0.9584
	Array w/o master pooling	O_{ET}	9-1	0.4327	0.5240	0.9931	0.8687	0.9600	10	0.4187	0.5226	0.9929	0.8742	0.9565
		O_{MAR}	9-1	0.4327	0.5240	0.9931	0.8687	0.9600	10	0.4187	0.5226	0.9929	0.8742	0.9565
	Array w/ master pooling	O_{ET}	81-9-1	0.3485	0.4218	0.9945	0.8687	0.9519	-	-	-	-	-	-
		O_{MAR}	81-9-1	0.3485	0.4218	0.9945	0.8687	0.9519	-	-	-	-	-	-
Male	Two-stage hierarchical	O_{ET}	4-1	0.5523	0.8649	0.9877	0.8606	0.9881	-	0.5462	0.8649	0.9867	0.8652	0.9866
		O_{MAR}	4-1	0.5523	0.8649	0.9877	0.8606	0.9881	-	0.5462	0.8649	0.9867	0.8652	0.9866
	Three-stage hierarchical	O_{ET}	9-3-1	0.4931	0.8044	0.9924	0.9036	0.9829	8	0.5081	0.8206	0.9905	0.8950	0.9824
		O_{MAR}	9-3-1	0.4931	0.8044	0.9924	0.9036	0.9829	8	0.5081	0.8206	0.9905	0.8950	0.9824
	Array w/o master pooling	O_{ET}	8-1	0.5015	0.8056	0.9901	0.8780	0.9830	8	0.5105	0.8053	0.9900	0.8882	0.9809
		O_{MAR}	8-1	0.5015	0.8056	0.9901	0.8780	0.9830	8	0.5105	0.8053	0.9900	0.8882	0.9809
	Array w/ master pooling	O_{ET}	64-8-1	0.4667	0.7492	0.9908	0.8781	0.9782	-	-	-	-	-	-
		O_{MAR}	64-8-1	0.4667	0.7492	0.9908	0.8781	0.9782	-	-	-	-	-	-

Table A.17: Full OTCs for informative two-stage hierarchical testing summarized in Table A.16. Differences between O_{ET} and O_{MAR} are highlighted.

	Objective function	Block size	$E(T)/I$	Group sizes										
				I_{11}	I_{12}	I_{13}	I_{14}	I_{15}	I_{16}	I_{17}	I_{18}	I_{19}	$I_{1,10}$	$I_{1,11}$
Female	O_{ET}	50	0.4757	11	7	6	5	5	4	4	4	4		
	O_{MAR}	50	0.4761	10	7	6	5	4	4	4	4	3	3	
Male	O_{ET}	50	0.5462	8	6	5	5	4	4	4	4	4	3	3
	O_{MAR}	50	0.5462	8	6	5	5	4	4	4	4	4	3	3

Table A.18: Full OTCs for informative three-stage hierarchical testing summarized in Table A.16. Differences between O_{ET} and O_{MAR} are highlighted.

	Objective function			Group sizes			
		I_{11}	$E(T)/I$	I_{21}	I_{22}	I_{23}	I_{24}
Female	O_{ET}	19	0.4102	7	5	4	3
	O_{MAR}	14	0.4113	6	4	3	1
Male	O_{ET}	8	0.5081	4	3	1	
	O_{MAR}	8	0.5081	4	3	1	

Table A.19: OTC summary for O_{GR} with $p = 0.01$ under non-informative two-stage hierarchical testing. An OTC of "11-1" means that stage 1 has a group of size 11 and stage 2 consists of individual testing.

S_e	S_p	D_1	D_2	OTC	$E(T)/I$	PS_e	PS_p	$PPPV$	$PNPV$
0.99	0.99	1	1	11-1	0.2035	0.9801	0.9990	0.9052	0.9998
		1000	1000	3-1	0.3724	0.9801	0.9997	0.9711	0.9998
		1	10	11-1	0.2035	0.9801	0.9990	0.9052	0.9998
		1	100	11-1	0.2035	0.9801	0.9990	0.9052	0.9998
		10	1	10-1	0.2037	0.9801	0.9991	0.9127	0.9998
		100	1	7-1	0.2194	0.9801	0.9993	0.9363	0.9998
0.95	0.95	1	1	11-1	0.2351	0.9025	0.9932	0.5727	0.9990
		1000	1000	3-1	0.4101	0.9025	0.9966	0.7286	0.9990
		1	10	11-1	0.2351	0.9025	0.9932	0.5727	0.9990
		1	100	11-1	0.2351	0.9025	0.9932	0.5727	0.9990
		10	1	9-1	0.2389	0.9025	0.9940	0.6040	0.9990
		100	1	4-1	0.3355	0.9025	0.9962	0.7038	0.9990
0.90	0.90	1	1	11-1	0.2746	0.8100	0.9824	0.3167	0.9981
		1000	1000	3-1	0.4571	0.8100	0.9884	0.4138	0.9981
		1	10	11-1	0.2746	0.8100	0.9824	0.3167	0.9981
		1	100	11-1	0.2746	0.8100	0.9824	0.3167	0.9981
		10	1	8-1	0.2868	0.8100	0.9846	0.3464	0.9981
		100	1	3-1	0.4571	0.8100	0.9884	0.4138	0.9981
0.99	0.90	1	1	11-1	0.2841	0.9801	0.9815	0.3485	0.9998
		1000	1000	3-1	0.4598	0.9801	0.9882	0.4568	0.9998
		1	10	11-1	0.2841	0.9801	0.9815	0.3485	0.9998
		1	100	11-1	0.2841	0.9801	0.9815	0.3485	0.9998
		10	1	8-1	0.2938	0.9801	0.9840	0.3816	0.9998
		100	1	3-1	0.4598	0.9801	0.9882	0.4568	0.9998
0.90	0.99	1	1	11-1	0.1941	0.8100	0.9990	0.8959	0.9981
		1000	1000	3-1	0.3698	0.8100	0.9997	0.9672	0.9981
		1	10	11-1	0.1941	0.8100	0.9990	0.8959	0.9981
		1	100	11-1	0.1941	0.8100	0.9990	0.8959	0.9981
		10	1	11-1	0.1941	0.8100	0.9990	0.8959	0.9981
		100	1	8-1	0.2038	0.8100	0.9993	0.9207	0.9981

Table A.20: OTC summary for O_{GR} with $p = 0.01$ under non-informative three-stage hierarchical testing. When equally sized groups are optimal, we use the same notation as given in other tables (e.g., Table A.1). When unequally sized groups are optimal, we write out each group size for its stage. For example, an OTC of "21-6,5,5,5-1" means that stage 1 has a group of size 21; stage 2 has groups of size 6, 5, 5, and 5; and stage 3 has groups of size 1.

S_e	S_p	D_1	D_2	OTC	$E(T)/I$	PS_e	PS_p	$PPPV$	$PNPV$
0.99	0.99	1	1	25-5-1	0.1354	0.9703	0.9996	0.9604	0.9997
		1000	1000	14-2-1	0.1614	0.9703	0.9999	0.9889	0.9997
		1	10	25-5-1	0.1354	0.9703	0.9996	0.9604	0.9997
		1	100	25-5-1	0.1354	0.9703	0.9996	0.9604	0.9997
		10	1	25-5-1	0.1354	0.9703	0.9996	0.9604	0.9997
		100	1	18-3-1	0.1435	0.9703	0.9998	0.9791	0.9997
0.95	0.95	1	1	25-5-1	0.1444	0.8574	0.9977	0.7907	0.9986
		1000	1000	6-2-1	0.2401	0.8574	0.9993	0.9289	0.9986
		1	10	25-5-1	0.1444	0.8574	0.9977	0.7907	0.9986
		1	100	25-5-1	0.1444	0.8574	0.9977	0.7907	0.9986
		10	1	20-4-1	0.1479	0.8574	0.9982	0.8290	0.9986
		100	1	12-2-1	0.1841	0.8574	0.9992	0.9166	0.9986
0.90	0.90	1	1	24-6-1	0.1562	0.7290	0.9938	0.5437	0.9973
		1000	1000	4-2-1	0.3432	0.7290	0.9980	0.7900	0.9973
		1	10	24-6-1	0.1562	0.7290	0.9938	0.5437	0.9973
		1	100	24-6-1	0.1562	0.7290	0.9938	0.5437	0.9973
		10	1	20-4-1	0.1644	0.7290	0.9955	0.6192	0.9973
		100	1	10-2-1	0.2202	0.7290	0.9976	0.7533	0.9973
0.99	0.90	1	1	21-6,5,5,5-1	0.1714	0.9703	0.9937	0.6074	0.9997
		1000	1000	4-2-1	0.3486	0.9703	0.9979	0.8204	0.9997
		1	10	21-6,5,5,5-1	0.1714	0.9703	0.9937	0.6074	0.9997
		1	100	21-6,5,5,5-1	0.1714	0.9703	0.9937	0.6074	0.9997
		10	1	16-4-1	0.1785	0.9703	0.9951	0.6684	0.9997
		100	1	8-2-1	0.2438	0.9703	0.9975	0.7977	0.9997
0.90	0.99	1	1	25-5-1	0.1229	0.7290	0.9997	0.9564	0.9973
		1000	1000	3-1	0.3698	0.8100	0.9997	0.9672	0.9981
		1	10	25-5-1	0.1229	0.7290	0.9997	0.9564	0.9973
		1	100	11-1	0.1941	0.8100	0.9990	0.8959	0.9981
		10	1	25-5-1	0.1229	0.7290	0.9997	0.9564	0.9973
		100	1	21-3-1	0.1330	0.7290	0.9998	0.9766	0.9973

Appendix B

Operating characteristics for group testing algorithms when testing error is allowed to differ across stages of testing

Chapter 1 provides reasons why the assumption of equal sensitivity and equal specificity across stages of testing might be unrealistic. In this appendix, we present derivations of operating characteristics for hierarchical and array testing algorithms that allow the sensitivity and specificity to each differ across testing stages. Our derivations pertain only to group testing with a single-disease assay because Bilder et al. (2019) and Hou et al. (2020) supplied derivations for the two-disease calculations in the `binGroup2` package. It is important to note that all of our derivations follow those in other papers where the derivations were performed assuming equal sensitivity and equal specificity. Throughout our own work, we purposely use the same notation as these other papers to maintain a consistency in the group testing literature and the associated R functions. Because of the close correspondence to these other papers, we do not provide a multitude of details. Rather, we focus on where unequal sensitivity and specificity values will lead to changes in the operating characteristics.

B.1. Hierarchical testing (excluding informative two-stage hierarchical)

We closely follow the work of Black et al. (2015) to derive the operating characteristics of hierarchical testing, excluding informative two-stage (Dorfman) testing. Derivations for informative two-stage testing are provided in Appendix D. Consider an initial group of size I where the ith individual has probability p_i, $i = 1, ..., I$, of being positive for a disease. Let $G_{s,j} = 1\,(0)$ denote a positive (negative) test result and let $\tilde{G}_{s,j} = 1\,(0)$ denote a positive (negative) true status for the jth group at stage s. There are $I_{s,j}$ individuals screened in the group corresponding to $G_{s,j}$, where $I_{1,1} = I$. If $G_{s,j} = 0$, all members of the corresponding group are declared negative. If $G_{s,j} = 1$, individuals in that group are split into $m_{s,j}$ groups for the next stage of testing. Let c_s be the total possible groups tested at stage s, where $c_1 = 1$ and $c_s = \sum_{j=1}^{c_s - 1} m_{s-1,j}$ for $s = 2, ..., S$.

To help explain this notation, consider the following example based on how HIV testing is performed in Seattle using a three-stage hierarchical testing algorithm (Sherlock et al., 2007). An initial group size of $I = 30$ is used in the first stage of the algorithm. Groups that test positive are split into three groups of size 10 for the second-stage of testing. Individual testing is performed on any sub-group that tests positive. In this context, $I = I_{1,1} = 30$, $c_1 = 1$, and $m_{1,1} = 3$ for the first stage of testing. For the second stage of testing, $I_{2,j} = 10$, $c_2 = 3$, and $m_{2,j} = 10$ for $j = 1, 2, 3$. For the third stage, $I_{3,j} = 1$, $c_3 = 30$, and $m_{3,j} = 0$ for $j = 1, ..., 30$.

B.1.1. Expected number of tests

Define T as the number of tests needed to decode an initial group of size I. Black et al. (2012) showed the expected number of tests for an initial group of size I to be

$$E(T) = 1 + \sum_{s=1}^{S-1} \sum_{j=1}^{c_s} m_{s,j} P\left(\bigcap_{\{(s',j'):G_{s,j}=1\}} \{G_{s',j'} = 1\} \right) \tag{B.1.1}$$

for an S-stage algorithm. The probability in equation (B.1.1) is

$$\begin{aligned} & P\left(\bigcap_{\{(s',j'):G_{s,j}=1\}} \{G_{s',j'} = 1\} \right) \\ &= (1 - S_p)^s \left\{ \prod_{i=1}^{I} (1 - p_i) \right\} + \\ &\quad \sum_{a=1}^{s-1} S_e^a (1 - S_p)^{s-a} \left\{ \prod_{i \in B_{a+1,j'}} (1 - p_i) \right\} \left\{ 1 - \prod_{i \in \bar{B}_{a+1,j'}} (1 - p_i) \right\} + \\ &\quad S_e^s \left\{ 1 - \prod_{i \in B_{s,j}} (1 - p_i) \right\}, \end{aligned} \tag{B.1.2}$$

where $S_e = P\left(G_{s,j} = 1 \,\middle|\, \tilde{G}_{s,j} = 1\right)$ is the test sensitivity and $S_p = P\left(G_{s,j} = 0 \,\middle|\, \tilde{G}_{s,j} = 0\right)$ is the test specificity. Additionally, $i \in B_{s,j}$ represents the individuals who belong to the jth group at stage s, and $i \in \bar{B}_{s,j}$ represents the set of individuals within the parent group of $B_{s,j}$ not including those in $B_{s,j}$ itself (e.g., $i \in \bar{B}_{2,1}$ denotes all individuals within the initial group $B_{1,1}$ who are in stage 2 groups other than $B_{2,1}$).

The probability in equation (B.1.2) represents a series of groups testing positive up to and including $G_{s,j} = 1$ (Black et al., 2015). For example, consider a four-stage hierarchical testing algorithm. One of the probabilities

that needs to be found to calculate $E(T)$ is $P(G_{1,1}=1, G_{2,1}=1, G_{3,1}=1)$, where $s=3$ and $j=1$. The probability $P\left(\bigcap_{\{(s',j'):G_{s,j}=1\}}\{G_{s',j'}=1\}\right)$ can be expressed as the sum of the following:

1. The joint probability that groups at each stage test positive and all are truly negative in status,

2. The joint probabilities that groups at each stage test positive and at least one, but not all, are truly positive in status, and

3. The joint probability that groups at each stage test positive and all are truly positive in status.

It is important to note that this probability is expressed under the assumption of equal sensitivity and equal specificity across all stages of testing. Without this assumption, we show in this section that the probability in equation (B.1.1) becomes

$$
\begin{aligned}
&P\left(\bigcap_{\{(s',j'):G_{s,j}=1\}}\{G_{s',j'}=1\}\right) \\
&\quad= \left\{\prod_{k=1}^{s}(1-S_{p:k})\right\}\left\{\prod_{i=1}^{I}(1-p_i)\right\}+ \\
&\qquad \sum_{a=1}^{s-1}\left\{\prod_{k=1}^{a}S_{e:k}\right\}\left\{\prod_{l=a+1}^{s}(1-S_{p:l})\right\}\times \\
&\qquad \left\{\prod_{i\in B_{a+1,j'}}(1-p_i)\right\}\left\{1-\prod_{i\in B_{a+1,j'}}(1-p_i)\right\}+ \\
&\qquad \left\{\prod_{k=1}^{s}S_{e:k}\right\}\left\{1-\prod_{i\in B_{s,j}}(1-p_i)\right\}, \qquad \text{(B.1.3)}
\end{aligned}
$$

where $S_{e:s} = P\left(G_{s,j}=1 \,\middle|\, \tilde{G}_{s,j}=1\right)$ and $S_{p:s} = P\left(G_{s,j}=0 \,\middle|\, \tilde{G}_{s,j}=0\right)$ are the test sensitivity and test specificity particular to stage $s = 1, ..., S$, respectively. Thus, the only difference between equation (B.1.2) and equation (B.1.3) is the leading sensitivity and specificity products for the three main terms in the expression. To see why these changes occur, suppose $S = 3$. For this situation, we need to find $P(G_{1,1}=1)$ and $P(G_{1,1}=1, G_{2,1}=1), ..., P(G_{1,1}=1, G_{2,c_2}=1)$. The probability $P(G_{1,1}=1)$ is for the initial group testing positive. It is found by taking into account the true group statuses,

$$\begin{aligned} & P(G_{1,1}=1) \\ &= P\left(G_{1,1}=1, \tilde{G}_{1,1}=0\right) + P\left(G_{1,1}=1, \tilde{G}_{1,1}=1\right) \\ &= P\left(G_{1,1}=1 \,\middle|\, \tilde{G}_{1,1}=0\right) P\left(\tilde{G}_{1,1}=0\right) + \\ &\quad P\left(G_{1,1}=1 \,\middle|\, \tilde{G}_{1,1}=1\right) P\left(\tilde{G}_{1,1}=1\right) \\ &= (1-S_{p:1})\left\{\prod_{i=1}^{I}(1-p_i)\right\} + S_{e:1}\left\{1-\prod_{i=1}^{I}(1-p_i)\right\}. \end{aligned} \tag{B.1.4}$$

For the probabilities of the first- and second-stage groups testing positive, consider the derivation for $P(G_{1,1}=1, G_{2,1}=1)$. This probability can be written as the sum of three separate terms:

$$\begin{aligned} & P(G_{1,1}=1, G_{2,1}=1) \\ &= P\left(G_{1,1}=1, G_{2,1}=1, \tilde{G}_{1,1}=0, \tilde{G}_{2,1}=0\right) + \\ &\quad P\left(G_{1,1}=1, G_{2,1}=1, \tilde{G}_{1,1}=1, \tilde{G}_{2,1}=0\right) + \\ &\quad P\left(G_{1,1}=1, G_{2,1}=1, \tilde{G}_{1,1}=1, \tilde{G}_{2,1}=1\right), \end{aligned} \tag{B.1.5}$$

which takes into account the three ways that $\{G_{1,1}=1\} \cap \{G_{2,1}=1\}$ may

occur with respect to the true statuses. The first joint probability is

$$
\begin{aligned}
&P\left(G_{1,1}=1, G_{2,1}=1, \tilde{G}_{1,1}=0, \tilde{G}_{2,1}=0\right) \\
&\quad = P\left(G_{1,1}=1, G_{2,1}=1 \,\middle|\, \tilde{G}_{1,1}=0, \tilde{G}_{2,1}=0\right) \times \\
&\qquad P\left(\tilde{G}_{1,1}=0, \tilde{G}_{2,1}=0\right) \\
&\quad = P\left(G_{1,1}=1 \,\middle|\, \tilde{G}_{1,1}=0\right) P\left(G_{2,1}=1 \,\middle|\, \tilde{G}_{2,1}=0\right) \times \\
&\qquad P\left(\tilde{G}_{1,1}=0, \tilde{G}_{2,1}=0\right) \\
&\quad = (1-S_{p:1})(1-S_{p:2}) P\left(\tilde{G}_{1,1}=0, \tilde{G}_{2,1}=0\right) \\
&\quad = (1-S_{p:1})(1-S_{p:2}) P\left(\tilde{G}_{2,1}=0 \,\middle|\, \tilde{G}_{1,1}=0\right) P\left(\tilde{G}_{1,1}=0\right) \\
&\quad = (1-S_{p:1})(1-S_{p:2}) P\left(\tilde{G}_{1,1}=0\right) \\
&\quad = (1-S_{p:1})(1-S_{p:2}) \left\{\prod_{i=1}^{I}(1-p_i)\right\}, \qquad \text{(B.1.6)}
\end{aligned}
$$

where we make the assumption that the test outcomes are conditionally independent once the true status is known (see Litvak et al. (1994) for more information). Similarly, the second and third joint probabilities can be shown to be

$$
\begin{aligned}
&P\left(G_{1,1}=1, G_{2,1}=1, \tilde{G}_{1,1}=1, \tilde{G}_{2,1}=0\right) \\
&\quad = S_{e:1}(1-S_{p:2}) \left\{\prod_{i\in B_{2,1}}(1-p_i)\right\} \left\{1-\prod_{i\in \bar{B}_{2,1}}(1-p_i)\right\}
\end{aligned}
$$

and

$$
\begin{aligned}
&P\left(G_{1,1}=1, G_{2,1}=1, \tilde{G}_{1,1}=1, \tilde{G}_{2,1}=1\right) \\
&\quad = S_{e:1}S_{e:2} \left\{1-\prod_{i\in B_{2,1}}(1-p_i)\right\}.
\end{aligned}
$$

Combining these results together, the probability in equation (B.1.5) becomes

$$\begin{aligned}
&P\left(G_{1,1}=1, G_{2,1}=1\right) \\
&\quad = \left(1-S_{p:1}\right)\left(1-S_{p:2}\right)\left\{\prod_{i=1}^{I}\left(1-p_i\right)\right\} + \\
&\qquad S_{e:1}\left(1-S_{p:2}\right)\left\{\prod_{i\in B_{2,1}}\left(1-p_i\right)\right\}\left\{1-\prod_{i\in \bar{B}_{2,1}}\left(1-p_i\right)\right\} + \\
&\qquad S_{e:1}S_{e:2}\left\{1-\prod_{i\in B_{2,1}}\left(1-p_i\right)\right\}
\end{aligned}$$

for three-stage hierarchical testing. The patterns shown here for three-stage hierarchical testing are also observed for S-stage hierarchical testing in general. Putting all this information together, one can show that for an S-stage algorithm, the probability in (B.1.1) becomes

$$\begin{aligned}
&P\left(\bigcap_{\{(s',j'):G_{s,j}=1\}}\left\{G_{s',j'}=1\right\}\right) \\
&\qquad = \left\{\prod_{k=1}^{s}\left(1-S_{p:k}\right)\right\}\left\{\prod_{i=1}^{I}\left(1-p_i\right)\right\} + \\
&\qquad\quad \sum_{a=1}^{s-1}\left\{\prod_{k=1}^{a}S_{e:k}\right\}\left\{\prod_{l=a+1}^{s}\left(1-S_{p:l}\right)\right\}\times \\
&\qquad\quad \left\{\prod_{i\in B_{a+1,j'}}\left(1-p_i\right)\right\}\left\{1-\prod_{i\in \bar{B}_{a+1,j'}}\left(1-p_i\right)\right\} + \\
&\qquad\quad \left\{\prod_{k=1}^{s}S_{e:k}\right\}\left\{1-\prod_{i\in B_{s,j}}\left(1-p_i\right)\right\}. \qquad \text{(B.1.7)}
\end{aligned}$$

B.1.2. Accuracy measures

Define $Y_i = 1\,(0)$ as the final positive (negative) test outcome based on the group testing algorithm, and define $\tilde{Y}_i = 1\,(0)$ as the positive (negative) true status of the ith individual, for $i = 1, ..., I$. The pooling sensitivity for the ith individual is the probability of a correct positive diagnosis and is expressed as $PS_e^{(i)} = P\left(Y_i = 1 \,\middle|\, \tilde{Y}_i = 1\right)$. The pooling specificity for the ith individual is the probability of a correct negative diagnosis and is written as $PS_p^{(i)} = P\left(Y_i = 0 \,\middle|\, \tilde{Y}_i = 0\right)$. We also define the pooling positive predictive value and pooling negative predictive value as $PPPV^{(i)} = P\left(\tilde{Y}_i = 1 \,\middle|\, Y_i = 1\right)$ and $PNPV^{(i)} = P\left(\tilde{Y}_i = 0 \,\middle|\, Y_i = 0\right)$, respectively.

B.1.2.1. Pooling sensitivity

For the ith individual to be diagnosed as positive ($Y_i = 1$), the initial group and all later groups containing that individual must test positive. This includes the last group which contains only the ith individual (Black et al., 2015). We define group j^* in stage L ($L \leq S$) as the group in the final stage of testing for individual i (i.e., the stage where individual i could be tested individually with respect to the configuration).

Black et al. (2015) showed the pooling sensitivity for an S-stage algorithm to be

$$\begin{aligned} PS_e^{(i)} &= P\left(Y_i = 1 \,\middle|\, \tilde{Y}_i = 1\right) \\ &= P\left(\bigcap_{\{(s'j'):G_{L,j^*}=1\}} \{G_{s',j'} = 1\} \,\middle|\, \bigcap_{\{(s'j'):G_{L,j^*}=1\}} \left\{\tilde{G}_{s',j'} = 1\right\}\right) \\ &= S_e^L. \end{aligned}$$

This derivation uses the conditional independence assumption previously mentioned in Section B.1.1 to result in the product of the sensitivities at each stage of the algorithm. Allowing for unequal sensitivities across the stages, the pooling sensitivity simply becomes

$$PS_e^{(i)} = \prod_{k=1}^{L} S_{e:k}.$$

Thus, the pooling sensitivity is the same for each individual testing positive within L stages.

B.1.2.2. Pooling specificity

Black et al. (2015) showed the pooling specificity for an S-stage algorithm to be

$$\begin{aligned} PS_p^{(i)} &= P\left(Y_i = 0 \,\middle|\, \tilde{Y}_i = 0\right) \\ &= 1 - P\left(Y_i = 1 \,\middle|\, \tilde{Y}_i = 0\right) \\ &= 1 - \frac{P\left(Y_i = 1\right) - P\left(Y_i = 1 \,\middle|\, \tilde{Y}_i = 1\right) P\left(\tilde{Y}_i = 1\right)}{P\left(\tilde{Y}_i = 0\right)} \\ &= 1 - \frac{P\left(\bigcap_{\{(s'j'):G_{L,j^*}=1\}} \{G_{s',j'} = 1\}\right) - S_e^L p_i}{(1 - p_i)}. \end{aligned}$$

Allowing for unequal sensitivities and unequal specificities across stages of testing, the pooling specificity simply becomes

$$PS_p^{(i)} = 1 - \frac{P\left(\bigcap_{\{(s'j'):G_{L,j^*}=1\}} \{G_{s',j'} = 1\}\right) - PS_e^{(i)} p_i}{(1 - p_i)},$$

where $P\left(\bigcap_{\{(s'j'):G_{L,j^*}=1\}}\{G_{s',j'}=1\}\right)$ is given by equation (B.1.3) and $PS_e^{(i)} = \prod_{k=1}^{L} S_{e:k}$. Notice that the individual pooling specificity is a function of the individual probabilities, unlike what was found for $PS_e^{(i)}$. The pooling positive and negative predictive values, given in Black et al. (2015), are then found through applications of Bayes' rule:

$$PPPV^{(i)} = \frac{p_i PS_e^{(i)}}{p_i PS_e^{(i)} + (1-p_i)\left(1 - PS_p^{(i)}\right)}$$

and

$$PNPV^{(i)} = \frac{(1-p_i)\, PS_p^{(i)}}{(1-p_i)\, PS_p^{(i)} + p_i\left(1 - PS_e^{(i)}\right)}.$$

New expressions for $PPPV^{(i)}$ and $PNPV^{(i)}$ (allowing for unequal sensitivity and unequal specificity across stages of testing) are found by substituting the derived expressions for $PS_e^{(i)}$ and $PS_p^{(i)}$ into the above equations. Overall measures of pooling sensitivity, pooling specificity, and pooling predictive values are provided in Appendix A.

B.2. Array testing without master pooling

We closely follow the work of McMahan et al. (2012b) to derive the operating characteristics for array testing without master pooling. Consider an array with $J > 1$ rows and $K > 1$ columns, and denote the individual assigned to the (j,k) cell as I_{jk}, for $j = 1, ..., J$ and $k = 1, ..., K$. Let $\tilde{Y}_{jk}$ denote the true status of individual I_{jk} based on the group testing algorithm, so that $p_{jk} = P\left(\tilde{Y}_{jk} = 1\right)$ is the probability of being truly positive. We make the same assumption as McMahan et al. (2012b) that the true statuses are independent random variables.

Define $R_j = 1\,(0)$ as the positive (negative) test outcome and $\tilde{R}_j = 1\,(0)$ as the positive (negative) true status of the jth row for $j = 1, ..., J$. Likewise, define C_k as the test outcome and $\tilde{C}_k$ as the true status of the kth column for $k = 1, ..., K$. Note that $\tilde{R}_j = 1$ ($\tilde{C}_k = 1$) if the jth row (kth column) contains at least one truly positive individual. Equivalently, $\tilde{R}_j = I\left(\sum_{k=1}^{K} \tilde{Y}_{jk} > 0\right)$ and $\tilde{C}_k = I\left(\sum_{j=1}^{J} \tilde{Y}_{jk} > 0\right)$, where $I(\cdot)$ represents the indicator function. As in McMahan et al. (2012b), we assume that diagnostic test outcomes are conditionally independent given the true statuses. We also assume that if a group contains at least one positive individual, it will test positive with probability S_e (sensitivity) and if a group consists entirely of negative individuals, it will test negative with probability S_p (specificity). Note that S_e and S_p are allowed to vary across stages of testing (e.g., row/column test, individual test), but we assume that these accuracy measures do not depend on the size of the group.

Using diagnostic tests with perfect accuracy, Phatarfod and Sudbury (1994) classified I_{jk} as negative if $R_j = 0$ or $C_k = 0$. Individual retesting for I_{jk} occurs when $R_j = 1$ and $C_k = 1$. However, diagnostic tests are rarely perfect, so we need to account for the possibility that one or more row (column) tests are positive while all column (row) tests are negative (Kim et al., 2007). As a result of this ambiguity, we use the notation in McMahan et al. (2012b) and partition all individuals in the algorithm into one of two categories:

$$M_+ = \left\{ I_{jk} : \; I\left(R_j = 1, C_k = 1\right) + I\left(R_j = 1, \sum_{k=1}^{K} C_k = 0\right) + I\left(\sum_{j=1}^{J} R_j = 0, C_k = 1\right) = 1 \right\}$$

and $M_- = \overline{M}_+$. Individuals in M_+ are classified as positive or negative after

individual retesting and individuals in $M\_$ are classified as negative without individual testing.

B.2.1. Expected number of tests

Let T denote the total number of tests required to decode one array. McMahan et al. (2012b) showed the expected number of tests is

$$E(T) = J + K + \sum_{j=1}^{J}\sum_{k=1}^{K} E(T_{jk}),$$

where T_{jk} represents the number of tests required to classify I_{jk} after the initial stage of testing (i.e., row and column tests). Using the classification methodology provided by Kim et al. (2007), T_{jk} can be written as

$$T_{jk} = \begin{cases} 1, & \text{if } R_j = 1 \text{ and } C_k = 1 \\ 1, & \text{if } R_j = 1 \text{ and } \sum_{k'=1}^{K} C_{k'} = 0 \\ 1, & \text{if } \sum_{j'=1}^{J} R_{j'} = 0 \text{ and } C_k = 1 \\ 0, & \text{otherwise.} \end{cases}$$

Then, for the two-dimensional array testing algorithm without master pooling,

$$\begin{aligned} E(T_{jk}) &= P(R_j = 1, C_k = 1) + \\ &\quad P\left(R_j = 1, \sum_{k'=1}^{K} C_{k'} = 0\right) + \\ &\quad P\left(\sum_{j'=1}^{J} R_{j'} = 0, C_k = 1\right). \end{aligned} \tag{B.2.1}$$

McMahan et al. (2012b) provided expressions for each of these probabilities. For the first term in equation (B.2.1),

$$
\begin{aligned}
&P\left(R_j = 1, C_k = 1\right) \\
&\quad = \left\{1 - P\left(R_j = 0 \,\middle|\, \tilde{R}_j = 0\right)\right\}\left\{1 - P\left(C_k = 0 \,\middle|\, \tilde{C}_k = 0\right)\right\} \times \\
&\qquad P\left(\tilde{R}_j = 0, \tilde{C}_k = 0\right) + \\
&\qquad P\left(R_j = 1 \,\middle|\, \tilde{R}_j = 1\right)\left\{1 - P\left(C_k = 0 \,\middle|\, \tilde{C}_k = 0\right)\right\} P\left(\tilde{R}_j = 1, \tilde{C}_k = 0\right) + \\
&\qquad \left\{1 - P\left(R_j = 0 \,\middle|\, \tilde{R}_j = 0\right)\right\} P\left(C_k = 1 \,\middle|\, \tilde{C}_k = 1\right) P\left(\tilde{R}_j = 0, \tilde{C}_k = 1\right) + \\
&\qquad P\left(R_j = 1 \,\middle|\, \tilde{R}_j = 1\right) P\left(C_k = 1 \,\middle|\, \tilde{C}_k = 1\right) P\left(\tilde{R}_j = 1, \tilde{C}_k = 1\right).
\end{aligned}
$$

When the sensitivity and specificity are unequal across stages of testing, this becomes

$$
\begin{aligned}
&P\left(R_j = 1, C_k = 1\right) \\
&\qquad = \left(1 - S_{p:R}\right)\left(1 - S_{p:C}\right) P\left(\tilde{R}_j = 0, \tilde{C}_k = 0\right) + \\
&\qquad\quad S_{e:R}\left(1 - S_{p:C}\right) P\left(\tilde{R}_j = 1, \tilde{C}_k = 0\right) + \\
&\qquad\quad \left(1 - S_{p:R}\right) S_{e:C} P\left(\tilde{R}_j = 0, \tilde{C}_k = 1\right) + \\
&\qquad\quad S_{e:R} S_{e:C} P\left(\tilde{R}_j = 1, \tilde{C}_k = 1\right),
\end{aligned}
\tag{B.2.2}
$$

where $S_{e:R}\,(S_{p:R})$ and $S_{e:C}\,(S_{p:C})$ represent the sensitivity (specificity) for the row and column tests, respectively. The jth row and kth column have only individual, I_{jk}, in common. Under the assumption that the individual statuses are independent, $\tilde{R}_j$ and $\tilde{C}_k$ are independent, conditional on $\tilde{Y}_{jk}$ (McMahan et al., 2012b). Expressions for the probabilities in equation (B.2.2) consisting only of $\tilde{R}_j$ and $\tilde{C}_k$ do not change from those already given in McMahan et al.

(2012b). Thus, the first term in equation (B.2.1) can be written as

$$
\begin{aligned}
& P\left(R_j = 1, C_k = 1\right) \\
& \quad = \left(1 - S_{p:R}\right)\left(1 - S_{p:C}\right)\left\{\frac{\pi_R\left(j\right)\pi_C\left(k\right)}{1 - p_{jk}}\right\} + \\
& \qquad S_{e:R}\left(1 - S_{p:C}\right)\left\{\pi_C\left(k\right) - \frac{\pi_R\left(j\right)\pi_C\left(k\right)}{1 - p_{jk}}\right\} + \\
& \qquad \left(1 - S_{p:R}\right)S_{e:C}\left\{\pi_R\left(j\right) - \frac{\pi_R\left(j\right)\pi_C\left(k\right)}{1 - p_{jk}}\right\} + \\
& \qquad S_{e:R}S_{e:C}\left\{1 - \pi_R\left(j\right) - \pi_C\left(k\right) + \frac{\pi_R\left(j\right)\pi_C\left(k\right)}{\left(1 - p_{jk}\right)}\right\}, \qquad \text{(B.2.3)}
\end{aligned}
$$

where $\pi_R\left(j\right) = P\left(\tilde{R}_j = 0\right) = \prod_{k'=1}^{K}\left(1 - p_{jk'}\right)$ and $\pi_C\left(k\right) = P\left(\tilde{C}_k = 0\right) = \prod_{j'=1}^{J}\left(1 - p_{j'k}\right)$. The remaining terms in equation (B.2.1) are much more complicated to derive.

First, we consider the second probability in equation (B.2.1). To find this probability, we must consider each of the 2^K configurations of the true column statuses; i.e., $\left\{\tilde{C}_1 = \tilde{c}_1, \tilde{C}_2 = \tilde{c}_2, ..., \tilde{C}_K = \tilde{c}_K\right\}$,where $\tilde{c}_k \in \{0, 1\}$, for $k = 1, ..., K$ (McMahan et al., 2012b). By the Law of Total Probability,

$$
\begin{aligned}
& P\left(R_j = 1, \sum_{k'=1}^{K} C_{k'} = 0\right) \\
& \quad = \sum_{\tilde{r}=0}^{1}\sum_{\tilde{c}_1=0}^{1}...\sum_{\tilde{c}_K=0}^{1} P\left(R_j = 1, \bigcap_{k=1}^{K}\left\{C_k = 0\right\} \;\middle|\; \tilde{R}_j = \tilde{r}, \bigcap_{k=1}^{K}\left\{\tilde{C}_k = \tilde{c}_k\right\}\right) \times \\
& \qquad P\left(\tilde{R}_j = \tilde{r}, \bigcap_{k=1}^{K}\left\{\tilde{C}_k = \tilde{c}_k\right\}\right).
\end{aligned}
$$

Define $\mathcal{B}_c$, for $c = 1, ..., K$, to be the set of all c-combinations of $\mathcal{K}_0 = \{1, 2, ..., K\}$, and let $\mathcal{B}_0 = \emptyset$, the empty set. For all $\mathcal{B} \in \mathcal{B}_c$, $c = 0, 1, ..., K$,

define the events

$$\tilde{C}(\mathcal{B}) = \bigcap_{k=1}^{K} \left\{ \tilde{C}_k = I\left(k \in \mathcal{B}\right) \right\}$$

$$C(\mathcal{B}) = \bigcap_{k=1}^{K} \left\{ C_k = I\left(k \in \mathcal{B}\right) \right\},$$

where $I(\cdot)$ is the indicator function. For example, suppose $K = 3$. We have $\mathcal{K}_0 = \{1,2,3\}$, $\mathcal{B}_0 = \emptyset$, $\mathcal{B}_1 = \{\{1\},\{2\},\{3\}\}$, $\mathcal{B}_2 = \{\{1,2\},\{1,3\},\{2,3\}\}$, and $\mathcal{B}_3 = \{\{1,2,3\}\}$. The set $\mathcal{B} = \{1,2\} \in \mathcal{B}_2$ corresponds to $\left\{\tilde{C}_1 = 1, \tilde{C}_2 = 1, \tilde{C}_3 = 0\right\}$, the event that columns 1 and 2 are truly positive and column 3 is truly negative. Using this notation from McMahan et al. (2012b), the previous probability can be written as

$$\begin{aligned} & P\left(R_j = 1, \sum_{k'=1}^{K} C_{k'} = 0\right) \\ & \quad = \sum_{\tilde{r}=0}^{1} \sum_{c=0}^{K} \sum_{\mathcal{B} \in \mathcal{B}_c} P\left\{R_j = 1, C(\mathcal{B}_0) \;\middle|\; \tilde{R}_j = \tilde{r}, \tilde{C}(\mathcal{B})\right\} \times \\ & \qquad P\left\{\tilde{R}_j = \tilde{r}, \tilde{C}(\mathcal{B})\right\}. \end{aligned} \tag{B.2.4}$$

Using the assumptions about test sensitivity and specificity and allowing sensitivity and specificity to differ across stages of testing, for all $c \in \{0,1,...,K\}$, we have

$$\begin{aligned} & P\left\{R_j = 1, C(\mathcal{B}_0) \;\middle|\; \tilde{R}_j = 0, \tilde{C}(\mathcal{B}_c)\right\} \\ & \quad = P\left(R_j = 1 \;\middle|\; \tilde{R}_j = 0\right) P\left\{C(\mathcal{B}_0) \;\middle|\; \tilde{C}(\mathcal{B}_c)\right\} \end{aligned}$$

$$
\begin{aligned}
&= \left\{1 - P\left(R_j = 0 \;\middle|\; \tilde{R}_j = 0\right)\right\} \times \\
&\quad P\left[\bigcap_{k=1}^{K} \left\{C_k = I\left(k \in \mathcal{B}_0\right)\right\} \;\middle|\; \bigcap_{k=1}^{K} \tilde{C}_k = I\left(k \in \mathcal{B}_c\right)\right] \\
&= \left(1 - S_{p:R}\right)\left(1 - S_{e:C}\right)^c \left(S_{p:C}\right)^{K-c}
\end{aligned}
$$

and

$$
\begin{aligned}
&P\left\{R_j = 1, C\left(\mathcal{B}_0\right) \;\middle|\; \tilde{R}_j = 1, \tilde{C}\left(\mathcal{B}_c\right)\right\} \\
&\qquad = P\left(R_j = 1 \;\middle|\; \tilde{R}_j = 1\right) P\left\{C(\mathcal{B}_0) \;\middle|\; \tilde{C}(\mathcal{B}_c)\right\} \\
&\qquad = S_{e:R}\left(1 - S_{e:C}\right)^c \left(S_{p:C}\right)^{K-c}.
\end{aligned}
$$

Substituting back into equation (B.2.4) and changing the order of summation, we get

$$
\begin{aligned}
&P\left(R_j = 1, \sum_{k'=1}^{K} C_{k'} = 0\right) \\
&\quad = \sum_{c=0}^{K} \sum_{\mathcal{B} \in \mathcal{B}_c} \left[P\left\{R_j = 1, C(\mathcal{B}_0) \;\middle|\; \tilde{R}_j = 0, \tilde{C}(\mathcal{B})\right\} P\left\{\tilde{R}_j = 0, \tilde{C}(\mathcal{B})\right\} + \right. \\
&\qquad\qquad \left. P\left\{R_j = 1, C(\mathcal{B}_0) \;\middle|\; \tilde{R}_j = 1, \tilde{C}(\mathcal{B})\right\} P\left\{\tilde{R}_j = 1, \tilde{C}(\mathcal{B})\right\}\right] \\
&\quad = \sum_{c=0}^{K} \sum_{\mathcal{B} \in \mathcal{B}_c} \left[\left(1 - S_{p:R}\right)\left(1 - S_{e:C}\right)^c \left(S_{p:C}\right)^{K-c} P\left\{\tilde{R}_j = 0, \tilde{C}(\mathcal{B})\right\} + \right. \\
&\qquad\qquad \left. S_{e:R}\left(1 - S_{e:C}\right)^c \left(S_{p:C}\right)^{K-c} P\left\{\tilde{R}_j = 1, \tilde{C}(\mathcal{B})\right\}\right]. \qquad \text{(B.2.5)}
\end{aligned}
$$

Expressions for the probabilities consisting only of $\tilde{R}_j$ and $\tilde{C}\left(\mathcal{B}\right)$ do not change from those already given in McMahan et al. (2012b). The probability expressions in equation (B.2.5) are

$$
P\left\{\tilde{R}_j = 0, \tilde{C}(\mathcal{B}_c)\right\} \;=\; \pi_R\left(j\right) \lambda_{\mathcal{C}}\left(\mathcal{B}_c \mid \mathcal{K}_0, j\right)
$$

and

$$P\left\{\tilde{R}_j = 1, \tilde{C}(\mathcal{B}_c)\right\} = \lambda_{\mathcal{C}}\left(\mathcal{B}_c \mid \emptyset, j\right) - \pi_R\left(j\right)\lambda_{\mathcal{C}}\left(\mathcal{B}_c \mid \mathcal{K}_0, j\right),$$

where

$$\lambda_{\mathcal{C}}\left(\mathcal{B}_c \mid \mathcal{S}, j\right) = \prod_{k' \in \mathcal{B}_c}\left\{1 - \frac{\pi_C\left(k'\right)}{\left(1 - p_{jk'}\right)^{I(k' \in \mathcal{S})}}\right\}\prod_{k' \in \bar{\mathcal{B}}_c}\frac{\pi_C\left(k'\right)}{\left(1 - p_{jk'}\right)^{I(k' \in \mathcal{S})}},$$

$\bar{\mathcal{B}}_c = \mathcal{K}_0 \setminus \mathcal{B}_c$ and products taken over $\{k' \in \emptyset\}$ are understood to be equal to 1. Therefore, equation (B.2.5) becomes

$$\begin{aligned}
&P\left(R_j = 1, \sum_{k'=1}^{K} C_{k'} = 0\right) \\
&\quad = \sum_{c=0}^{K}\sum_{\mathcal{B} \in \mathcal{B}_c}\Big[\left(1 - S_{p:R}\right)\left(1 - S_{e:C}\right)^c\left(S_{p:C}\right)^{K-c}\pi_R\left(j\right)\lambda_{\mathcal{C}}\left(\mathcal{B} \mid \mathcal{K}_0, j\right) + \\
&\qquad\qquad S_{e:R}\left(1 - S_{e:C}\right)^c\left(S_{p:C}\right)^{K-c}\left\{\lambda_{\mathcal{C}}\left(\mathcal{B} \mid \emptyset, j\right) - \pi_R\left(j\right)\lambda_{\mathcal{C}}\left(\mathcal{B} \mid \mathcal{K}_0, j\right)\right\}\Big] \\
&\quad = \sum_{c=0}^{K}\sum_{\mathcal{B} \in \mathcal{B}_c}\Big\{\left(1 - S_{e:R} - S_{p:R}\right)\left(1 - S_{e:C}\right)^c\left(S_{p:C}\right)^{K-c}\pi_R\left(j\right)\lambda_{\mathcal{C}}\left(\mathcal{B} \mid \mathcal{K}_0, j\right) + \\
&\qquad\qquad S_{e:R}\left(1 - S_{e:C}\right)^c\left(S_{p:C}\right)^{K-c}\lambda_{\mathcal{C}}\left(\mathcal{B} \mid \emptyset, j\right)\Big\} \\
&\quad = \sum_{c=0}^{K}\sum_{\mathcal{B} \in \mathcal{B}_c}\left\{\gamma_0\left(c, K\right)\lambda_{\mathcal{C}}\left(\mathcal{B} \mid \emptyset, j\right) + \gamma_1\left(c, K\right)\pi_R\left(j\right)\lambda_{\mathcal{C}}\left(\mathcal{B} \mid \mathcal{K}_0, j\right)\right\}, \qquad \text{(B.2.6)}
\end{aligned}$$

where $\gamma_0\left(c, K\right) = S_{e:R}\left(1 - S_{e:C}\right)^c\left(S_{p:C}\right)^{K-c}$ and $\gamma_1\left(c, K\right) = \left(1 - S_{e:R} - S_{p:R}\right)\times$ $\left(1 - S_{e:C}\right)^c\left(S_{p:C}\right)^{K-c}$.

The third probability in equation (B.2.1) is found in a similar manner as the second probability. Define $\mathcal{A}_r$, for $r = 1, 2, ..., J$, to be the set of all r-combinations of $\mathcal{J}_0 = \{1, 2, ..., J\}$, and let $\mathcal{A}_0 = \emptyset$. We define

$$\lambda_{\mathcal{R}}\left(\mathcal{A}_r \mid \mathcal{S}, j\right) = \prod_{j' \in \mathcal{A}_r}\left\{1 - \frac{\pi_R\left(j'\right)}{\left(1 - p_{j'k}\right)^{I(j' \in \mathcal{S})}}\right\}\prod_{j' \in \bar{\mathcal{A}}_r}\frac{\pi_R\left(j'\right)}{\left(1 - p_{j'k}\right)^{I(j' \in \mathcal{S})}},$$

where $\bar{\mathcal{A}}_r = \mathcal{J}_0 \setminus \mathcal{A}_r$ and products taken over $\{j' \in \emptyset\}$ are understood to be

equal to 1. Then, the third probability in equation (B.2.1) becomes

$$P\left(\sum_{j'=1}^{J} R_{j'} = 0, C_k = 1\right)$$
$$= \sum_{r=0}^{J} \sum_{\mathcal{A}\in\mathcal{A}_r} \left\{\gamma_2\left(r, J\right)\lambda_{\mathcal{R}}\left(\mathcal{A} \mid \emptyset, k\right) + \gamma_3\left(r, J\right)\lambda_{\mathcal{R}}\left(\mathcal{A} \mid \mathcal{J}_0, k\right)\pi_C\left(k\right)\right\},$$

where $\gamma_2\left(r, J\right) = S_{e:C}\left(1 - S_{e:R}\right)^r \left(S_{p:R}\right)^{J-r}$ and $\gamma_3\left(r, J\right) = \left(1 - S_{e:C} - S_{p:C}\right) \times \left(1 - S_{e:R}\right)^r \left(S_{p:R}\right)^{J-r}$. This follows from equation (B.2.6) by treating the rows as columns and vice versa. This completes the derivation of $E\left(T_{jk}\right)$ in equation (B.2.1).

B.2.2. Accuracy measures

We now present derivations for the pooling sensitivity and pooling specificity. Let $S_{e:I}\left(S_{p:I}\right)$ represent the sensitivity (specificity) for individual testing. Let Y_{jk} denote the test outcome for individual I_{jk} based on individual testing and let $I_{jk}^{+}\left(I_{jk}^{-}\right)$ denote the event that individual I_{jk} is classified as positive (negative) by the group testing algorithm. Define the pooling sensitivity to be $PS_e^{I_{jk}} = P\left(I_{jk}^{+} \,\middle|\, \tilde{Y}_{jk} = 1\right)$ and the pooling specificity to be $PS_p^{I_{jk}} = P\left(I_{jk}^{-} \,\middle|\, \tilde{Y}_{jk} = 0\right)$.

B.2.2.1. Pooling Sensitivity

For array testing without master pooling, individual I_{jk} is classified as positive if its corresponding row and/or column tests are positive and the individual

test is positive. Then, the pooling sensitivity is

$$\begin{aligned} PS_e^{I_{jk}} &= P\left(I_{jk}^{+} \,\middle|\, \tilde{Y}_{jk}=1\right) \\ &= P\left(Y_{jk}=1, R_j=1, C_k=1 \,\middle|\, \tilde{Y}_{jk}=1\right) + \qquad \text{(B.2.7)} \\ &\quad P\left(Y_{jk}=1, R_j=1, \sum_{k'=1}^{K} C_{k'}=0 \,\middle|\, \tilde{Y}_{jk}=1\right) + \qquad \text{(B.2.8)} \\ &\quad P\left(Y_{jk}=1, \sum_{j'=1}^{J} R_{j'}=0, C_k=1 \,\middle|\, \tilde{Y}_{jk}=1\right). \qquad \text{(B.2.9)} \end{aligned}$$

If $\tilde{Y}_{jk}=1$, then $\tilde{R}_j=1$ and $\tilde{C}_k=1$. This fact, together with the conditional independence assumption, implies that equation (B.2.7) is

$$\begin{aligned} &P\left(Y_{jk}=1, R_j=1, C_k=1 \,\middle|\, \tilde{Y}_{jk}=1\right) \\ &\quad = P\left(Y_{jk}=1 \,\middle|\, \tilde{Y}_{jk}=1\right) P\left(R_j=1 \,\middle|\, \tilde{Y}_{jk}=1\right) P\left(C_k=1 \,\middle|\, \tilde{Y}_{jk}=1\right) \\ &\quad = S_{e:I} S_{e:R} S_{e:C}. \end{aligned}$$

Similarly, the probability in equation (B.2.8) can be written as

$$\begin{aligned} &P\left(Y_{jk}=1, R_j=1, \sum_{k'=1}^{K} C_{k'}=0 \,\middle|\, \tilde{Y}_{jk}=1\right) \\ &\qquad = S_{e:I} S_{e:R} P\left(\sum_{k'=1}^{K} C_{k'}=0 \,\middle|\, \tilde{Y}_{jk}=1\right). \end{aligned}$$

McMahan et al. (2012b) showed that the probability on the right-hand side of this expression can be written as

$$P\left(\sum_{k'=1}^{K} C_{k'}=0 \,\middle|\, \tilde{Y}_{jk}=1\right) = (1-S_e) \prod_{k' \neq k} P\left(C_{k'}=0\right),$$

where $P(C_{k'}=0)=1-S_e-(1-S_e-S_p)\,\pi_C(k')$. Allowing the sensitivities and specificities to differ across stages of testing, we can write

$$P\left(\sum_{k'=1}^{K} C_{k'}=0 \;\middle|\; \tilde{Y}_{jk}=1\right) \;=\; (1-S_{e:C})\prod_{k'\neq k} P(C_{k'}=0),$$

where $P(C_{k'}=0)=1-S_{e:C}-(1-S_{e:C}-S_{p:C})\,\pi_C(k')$. Therefore, equation (B.2.8) can be written as

$$P\left(Y_{jk}=1, R_j=1, \sum_{k'=1}^{K} C_{k'}=0 \;\middle|\; \tilde{Y}_{jk}=1\right)$$

$$= \; S_{e:I}S_{e:R}(1-S_{e:C})\prod_{k'\neq k} P(C_{k'}=0). \qquad \text{(B.2.10)}$$

Finally, equation (B.2.9) can be written as

$$P\left(Y_{jk}=1, \sum_{j'=1}^{J} R_{j'}=0, C_k=1 \;\middle|\; \tilde{Y}_{jk}=1\right)$$

$$= \; S_{e:I}S_{e:C}(1-S_{e:R})\prod_{j'\neq j} P(R_{j'}=0),$$

where $P(R_{j'}=0)=1-S_{e:R}-(1-S_{e:R}-S_{p:R})\,\pi_R(j')$. This follows from equation (B.2.10) by treating the rows as columns and vice versa. Combining these results, we obtain

$$PS_e^{I_{jk}} \;=\; S_{e:I}S_{e:R}S_{e:C}+S_{e:I}S_{e:R}(1-S_{e:C})\prod_{k'\neq k} P(C_{k'}=0)+$$
$$S_{e:I}S_{e:C}(1-S_{e:R})\prod_{j'\neq j} P(R_{j'}=0)$$

$$= S_{e:I}\left\{S_{e:R}S_{e:C} + S_{e:R}\left(1 - S_{e:C}\right)\prod_{k' \neq k} P\left(C_{k'} = 0\right) + S_{e:C}\left(1 - S_{e:R}\right)\prod_{j' \neq j} P\left(R_{j'} = 0\right)\right\}.$$

B.2.2.2. Pooling Specificity

We now turn to the derivation of the pooling specificity, $PS_p^{I_{jk}}$. By definition,

$$\begin{aligned} PS_p^{I_{jk}} &= P\left(I_{jk}^- \,\middle|\, \tilde{Y}_{jk} = 0\right) \\ &= 1 - P\left(I_{jk}^+ \,\middle|\, \tilde{Y}_{jk} = 0\right), \end{aligned} \tag{B.2.11}$$

where

$$P\left(I_{jk}^+ \,\middle|\, \tilde{Y}_{jk} = 0\right)$$

$$= P\left(Y_{jk} = 1, R_j = 1, C_k = 1 \,\middle|\, \tilde{Y}_{jk} = 0\right) + \tag{B.2.12}$$

$$P\left(Y_{jk} = 1, R_j = 1, \sum_{k'=1}^{K} C_{k'} = 0 \,\middle|\, \tilde{Y}_{jk} = 0\right) + \tag{B.2.13}$$

$$P\left(Y_{jk} = 1, \sum_{j'=1}^{J} R_{j'} = 0, C_k = 1 \,\middle|\, \tilde{Y}_{jk} = 0\right). \tag{B.2.14}$$

Using the conditional independence assumption, McMahan et al. (2012b) showed that equation (B.2.12) can be written as

$$P\left(Y_{jk} = 1, R_j = 1, C_k = 1 \,\middle|\, \tilde{Y}_{jk} = 0\right)$$

$$\begin{aligned} &= P\left(Y_{jk} = 1 \,\middle|\, \tilde{Y}_{jk} = 0\right) P\left(R_j = 1, C_k = 1 \,\middle|\, \tilde{Y}_{jk} = 0\right) \\ &= \left(1 - S_{p:I}\right) P\left(R_j = 1, C_k = 1 \,\middle|\, \tilde{Y}_{jk} = 0\right). \end{aligned}$$

Using the Law of Total Probability, McMahan et al. (2012b) also showed that

$$\begin{aligned} & P\left(R_j = 1, C_k = 1 \,\middle|\, \tilde{Y}_{jk} = 0\right) \\ & \quad = \sum_{r=0}^{1}\sum_{c=0}^{1}\left\{P\left(\tilde{R}_j = r, \tilde{C}_k = c \,\middle|\, \tilde{Y}_{jk} = 0\right) \times \right. \\ & \qquad \left. P\left(R_j = 1, C_k = 1 \,\middle|\, \tilde{R}_j = r, \tilde{C}_k = c, \tilde{Y}_{jk} = 0\right)\right\}. \end{aligned}$$

Allowing the sensitivities and specificities to differ across stages of testing, it can be shown that

$$\begin{aligned} & P\left(R_j = 1, C_k = 1 \,\middle|\, \tilde{Y}_{jk} = 0\right) \\ & \quad = \left(1 - S_{p:R}\right)\left(1 - S_{p:C}\right) P\left(\tilde{R}_j = 0 \,\middle|\, \tilde{Y}_{jk} = 0\right) P\left(\tilde{C}_k = 0 \,\middle|\, \tilde{Y}_{jk} = 0\right) + \\ & \qquad S_{e:R}\left(1 - S_{p:C}\right) P\left(\tilde{R}_j = 1 \,\middle|\, \tilde{Y}_{jk} = 0\right) P\left(\tilde{C}_k = 0 \,\middle|\, \tilde{Y}_{jk} = 0\right) + \\ & \qquad S_{e:C}\left(1 - S_{p:R}\right) P\left(\tilde{R}_j = 0 \,\middle|\, \tilde{Y}_{jk} = 0\right) P\left(\tilde{C}_k = 1 \,\middle|\, \tilde{Y}_{jk} = 0\right) + \\ & \qquad S_{e:R}S_{e:C} P\left(\tilde{R}_j = 1 \,\middle|\, \tilde{Y}_{jk} = 0\right) P\left(\tilde{C}_k = 1 \,\middle|\, \tilde{Y}_{jk} = 0\right). \end{aligned}$$

Expressions for the probabilities consisting only of $\tilde{R}_j$ and $\tilde{C}_k$ conditioned on $\tilde{Y}_{jk}$ do not change from those already given inMcMahan et al. (2012b). After extensive algebra, equation (B.2.12) becomes

$$P\left(Y_{jk} = 1, R_j = 1, C_k = 1 \,\middle|\, \tilde{Y}_{jk} = 0\right)$$

$$
\begin{aligned}
= \quad & (1 - S_{p:I}) \left[(1 - S_{p:R})(1 - S_{p:C}) \frac{\pi_R(j)\,\pi_C(k)}{(1 - p_{jk})^2} + \right. \\
& S_{e:R}(1 - S_{p:C}) \left\{ \frac{\pi_C(k)}{1 - p_{jk}} - \frac{\pi_R(j)\,\pi_C(k)}{(1 - p_{jk})^2} \right\} + \\
& S_{e:C}(1 - S_{p:R}) \left\{ \frac{\pi_R(j)}{1 - p_{jk}} - \frac{\pi_R(j)\,\pi_C(k)}{(1 - p_{jk})^2} \right\} + \\
& \left. S_{e:R} S_{e:C} \left\{ 1 - \frac{\pi_R(j)}{1 - p_{jk}} - \frac{\pi_C(k)}{1 - p_{jk}} + \frac{\pi_R(j)\,\pi_C(k)}{(1 - p_{jk})^2} \right\} \right].
\end{aligned}
$$

Using conditional independence, we can write equation (B.2.13) as

$$
\begin{aligned}
& P\left(Y_{jk} = 1, R_j = 1, \sum_{k'=1}^{K} C_{k'} = 0 \;\middle|\; \tilde{Y}_{jk} = 0 \right) \\
& \quad = \quad P\left(Y_{jk} = 1 \;\middle|\; \tilde{Y}_{jk} = 0 \right) P\left(R_j = 1, \sum_{k'=1}^{K} C_{k'} = 0 \;\middle|\; \tilde{Y}_{jk} = 0 \right) \\
& \quad = \quad (1 - S_{p:I})\, P\left(R_j = 1, \sum_{k'=1}^{K} C_{k'} = 0 \;\middle|\; \tilde{Y}_{jk} = 0 \right).
\end{aligned}
$$

Conditioning on the true statuses of the rows and columns, McMahan et al. (2012b) showed that

$$
\begin{aligned}
& P\left(R_j = 1, \sum_{k'=1}^{K} C_{k'} = 0 \;\middle|\; \tilde{Y}_{jk} = 0 \right) \\
& \quad = \quad \sum_{\tilde{r}=0}^{1} \sum_{c=0}^{K} \sum_{\mathcal{B} \in \mathcal{B}_c} P\left\{ R_j = 1, C(\mathcal{B}_0) \;\middle|\; \tilde{R}_j = \tilde{r}, \tilde{C}(\mathcal{B}), \tilde{Y}_{jk} = 0 \right\} \times \\
& \qquad P\left\{ \tilde{R}_j = \tilde{r}, \tilde{C}(\mathcal{B}) \;\middle|\; \tilde{Y}_{jk} = 0 \right\}.
\end{aligned}
$$

Allowing the sensitivities and specificities to differ across stages of testing, it can be shown that

$$
P\left\{ R_j = 1, C(\mathcal{B}_0) \;\middle|\; \tilde{R}_j = 0, \tilde{C}(\mathcal{B}_c), \tilde{Y}_{jk} = 0 \right\}
$$

$$= \quad (1 - S_{p:R})(1 - S_{e:C})^c (S_{p:C})^{K-c}$$

and

$$P\left\{R_j = 1, C(\mathcal{B}_0) \;\middle|\; \tilde{R}_j = 1, \tilde{C}(\mathcal{B}_c), \tilde{Y}_{jk} = 0\right\}$$
$$= \quad S_{e:R}(1 - S_{e:C})^c (S_{p:C})^{K-c}.$$

This allows us to rewrite

$$P\left(R_j = 1, \sum_{k'=1}^{K} C_{k'} = 0 \;\middle|\; \tilde{Y}_{jk} = 0\right)$$
$$= \quad \sum_{c=0}^{K} \sum_{\mathcal{B} \in \mathcal{B}_c} \left[(1 - S_{p:R})(1 - S_{e:C})^c (S_{p:C})^{K-c} P\left\{\tilde{R}_j = 0, \tilde{C}(\mathcal{B}) \;\middle|\; \tilde{Y}_{jk} = 0\right\} + \right.$$
$$\left. S_{e:R}(1 - S_{e:C})^c (S_{p:C})^{K-c} P\left\{\tilde{R}_j = 1, \tilde{C}(\mathcal{B}) \;\middle|\; \tilde{Y}_{jk} = 0\right\}\right].$$

Expressions for the probabilities consisting only of $\tilde{R}_j$ and $\tilde{C}(\mathcal{B})$ conditioned on $\tilde{Y}_{jk}$ do not change from those already given in McMahan et al. (2012b). Therefore, after extensive algebra, we can write equation (B.2.13) as

$$P\left(Y_{jk} = 1, R_j = 1, \sum_{k'=1}^{K} C_{k'} = 0 \;\middle|\; \tilde{Y}_{jk} = 0\right)$$
$$= \quad (1 - S_{p:I}) \sum_{c=0}^{K} \sum_{\mathcal{B} \in \mathcal{B}_c} \left\{\gamma_0(c, K)\, \lambda_{\mathcal{C}}(\mathcal{B} \mid \{k\}, j) + \right.$$
$$\left. \gamma_1(c, K) \frac{\pi_R(j)}{1 - p_{jk}} \lambda_{\mathcal{C}}(\mathcal{B} \mid \mathcal{K}_0, j)\right\}, \tag{B.2.15}$$

where $\gamma_0(c, K)$ and $\gamma_1(c, K)$ were previously defined in the derivations for $E(T_{jk})$. In a similar manner, equation (B.2.14) can be expressed as

$$P\left(Y_{jk} = 1, \sum_{j'=1}^{J} R_{j'} = 0, C_k = 1 \;\middle|\; \tilde{Y}_{jk} = 0\right)$$

$$= (1 - S_{p:I}) \sum_{r=0}^{J} \sum_{\mathcal{A} \in \mathcal{A}_r} \left\{ \gamma_2 (r, J) \lambda_{\mathcal{R}} (\mathcal{A} \mid \{j\}, k) + \gamma_3 (r, J) \lambda_{\mathcal{R}} (\mathcal{A} \mid \mathcal{J}_0, k) \frac{\pi_C (k)}{1 - p_{jk}} \right\},$$

where $\gamma_2 (r, J)$ and $\gamma_3 (r, J)$ were previously defined in the derivations for $E (T_{jk})$. This follows from equation (B.2.15) by treating the rows as columns and vice versa. Combining the expressions for (B.2.12), (B.2.13), and (B.2.14), we have

$$\begin{aligned}
& P \left(I_{jk}^{+} \,\middle|\, \tilde{Y}_{jk} = 0 \right) \\
&= (1 - S_{p:I}) \left[(1 - S_{p:R}) (1 - S_{p:C}) \frac{\pi_R (j) \pi_C (k)}{(1 - p_{jk})^2} + \right. \\
&\quad S_{e:R} (1 - S_{p:C}) \left\{ \frac{\pi_C (k)}{1 - p_{jk}} - \frac{\pi_R (j) \pi_C (k)}{(1 - p_{jk})^2} \right\} + \\
&\quad S_{e:C} (1 - S_{p:R}) \left\{ \frac{\pi_R (j)}{1 - p_{jk}} - \frac{\pi_R (j) \pi_C (k)}{(1 - p_{jk})^2} \right\} + \\
&\quad S_{e:R} S_{e:C} \left\{ 1 - \frac{\pi_R (j)}{1 - p_{jk}} - \frac{\pi_C (k)}{1 - p_{jk}} + \frac{\pi_R (j) \pi_C (k)}{(1 - p_{jk})^2} \right\} + \\
&\quad \sum_{c=0}^{K} \sum_{\mathcal{B} \in \mathcal{B}_c} \left\{ \frac{\gamma_1 (c, K) \pi_R (j) \lambda_{\mathcal{C}} (\mathcal{B} \mid \mathcal{K}_0, j)}{1 - p_{jk}} + \right. \\
&\quad \left. \gamma_0 (c, K) \lambda_{\mathcal{C}} (\mathcal{B} \mid \{k\}, j) \right\} + \\
&\quad \sum_{r=0}^{J} \sum_{\mathcal{A} \in \mathcal{A}_r} \left\{ \frac{\gamma_3 (r, J) \lambda_{\mathcal{R}} (\mathcal{A} \mid \mathcal{J}_0, k) \pi_C (k)}{1 - p_{jk}} + \right. \\
&\quad \left. \left. \gamma_2 (r, J) \lambda_{\mathcal{R}} (\mathcal{A} \mid \{j\}, k) \right\} \right]. \qquad \text{(B.2.16)}
\end{aligned}$$

The final expression for $PS_p^{I_{jk}}$ follows from equation (B.2.16) substituted into equation (B.2.11).

The pooling positive and negative predictive values are defined as $PPPV^{I_{jk}} = P\left(\tilde{Y}_{jk} = 1 \,\middle|\, I_{jk}^{+}\right)$ and $PNPV^{I_{jk}} = P\left(\tilde{Y}_{jk} = 0 \,\middle|\, I_{jk}^{-}\right)$, respectively. Expressions for the pooling predictive values follow directly from McMahan et al. (2012b, p. 798).

B.3. Array testing with master pooling

We closely follow the work of Kim et al. (2007) to derive the operating characteristics for array testing with master pooling. Their derivations assume that $p_i = p$ for all individuals in the algorithm and that diagnostic accuracy is the same across stages of testing. We generalize their derivations to allow for heterogeneous risk probabilities p_i, for individuals $i = 1, ..., I$, and to allow for unequal sensitivity and unequal specificity values across stages of testing. For consistency in the group testing literature, we adhere to the notation in McMahan et al. (2012b) whenever possible.

Consider an array with $J > 1$ rows and $K > 1$ columns, and denote the individual assigned to the (j, k) cell as I_{jk}, for $j = 1, ..., J$ and $k = 1, ..., K$. In array testing with master pooling, we first test a master group of size $J \times K$ before testing rows and columns, and finally proceed to individual testing. Recall that Y_{jk}, R_j, and C_k denote the test outcome of the individual test, row test, and column test corresponding to individual I_{jk}, respectively. Similarly, $\tilde{Y}_{jk}$, $\tilde{R}_j$, and $\tilde{C}_k$ denote the true status of individual I_{jk}, the jth row, and the kth column. We again make the assumption that test outcomes are conditionally independent given the true statuses. Also recall that $S_{e:R}\,(S_{p:R})$, $S_{e:C}\,(S_{p:C})$, and $S_{e:I}\,(S_{p:I})$ represent the sensitivity (specificity) for row, column, and individual testing, respectively.

B.3.1. Expected number of tests

Let the number of tests required to decode the full array be $T = T_0 + T_1 + T_2$, where $T_0 = 1$ corresponds to testing the master group, T_1 corresponds to possible row and column testing, and T_2 corresponds to possible individual testing. Let X_0 be a random variable that equals 1 if the master group tests positive and 0 otherwise, such that $T_1 = JKX_0$ and $E(T_1) = JK \times P(X_0 = 1)$. Define $\tilde{X}_0$ as the true status of the master group. Let $S_{e:M}$ ($S_{p:M}$) represent the sensitivity (specificity) for the master group. The probability that the master group tests positive can be written as

$$\begin{aligned} P(X_0 = 1) &= P\left(X_0 = 1, \tilde{X}_0 = 0\right) + P\left(X_0 = 1, \tilde{X}_0 = 1\right) \\ &= P\left(X_0 = 1 \,\middle|\, \tilde{X}_0 = 0\right) P\left(\tilde{X}_0 = 0\right) + \\ &\quad P\left(X_0 = 1 \,\middle|\, \tilde{X}_0 = 1\right) P\left(\tilde{X}_0 = 1\right) \\ &= (1 - S_{p:M}) \prod_{j,k} (1 - p_{jk}) + S_{e:M} \left\{ 1 - \prod_{j,k} (1 - p_{jk}) \right\}. \end{aligned}$$

Thus, the expected number of tests corresponding to possible row and column testing is

$$E(T_1) = JK \left[(1 - S_{p:M}) \prod_{j,k} (1 - p_{jk}) + S_{e:M} \left\{ 1 - \prod_{j,k} (1 - p_{jk}) \right\} \right].$$

This leads to an expression for the expected number of tests for the whole array,

$$E(T) = 1 + (J+K)\left[(1-S_{p:M})\prod_{j,k}(1-p_{jk}) + S_{e:M}\left\{1-\prod_{j,k}(1-p_{jk})\right\}\right] + \sum_{j,k}E(T_{2jk}),$$

where T_{2jk} represents the number of tests required to classify I_{jk} after the initial stage of testing (i.e., master group test) and the second stage of testing (i.e., row and column tests). Using the classification methodology provided by Kim et al. (2007), T_{2jk} can be written as

$$T_{2jk} = \begin{cases} 1 & \text{if } X_0 = 1, R_j = 1 \text{ and } C_k = 1 \\ 1 & \text{if } X_0 = 1, R_j = 1 \text{ and } \sum C_k = 0 \\ 1 & \text{if } X_0 = 1, \sum R_j = 0 \text{ and } C_k = 1 \\ 0 & \text{otherwise.} \end{cases}$$

Then, for the two-dimensional array testing algorithm with master pooling,

$$\begin{aligned} E(T_{2jk}) = {} & P\left(X_0 = 1, R_j = 1, C_k = 1\right) + \\ & P\left(X_0 = 1, R_j = 1, \sum_k C_k = 0\right) + \\ & P\left(X_0 = 1, \sum_j R_j = 0, C_k = 1\right). \end{aligned} \tag{B.3.1}$$

The first probability in equation (B.3.1) can be written as

$P\left(X_0 = 1, R_j = 1, C_k = 1\right)$

$$
\begin{aligned}
&= P\left(X_0=1,R_j=1,C_k=1,\tilde{X}_0=0,\tilde{R}_j=0,\tilde{C}_k=0\right)+\\
&\quad P\left(X_0=1,R_j=1,C_k=1,\tilde{X}_0=1,\tilde{R}_j=0,\tilde{C}_k=0\right)+\\
&\quad P\left(X_0=1,R_j=1,C_k=1,\tilde{X}_0=1,\tilde{R}_j=1,\tilde{C}_k=0\right)+\\
&\quad P\left(X_0=1,R_j=1,C_k=1,\tilde{X}_0=1,\tilde{R}_j=0,\tilde{C}_k=1\right)+\\
&\quad P\left(X_0=1,R_j=1,C_k=1,\tilde{X}_0=1,\tilde{R}_j=1,\tilde{C}_k=1\right)\\
&= P\left(X_0=1,R_j=1,C_k=1\,\middle|\,\tilde{X}_0=0,\tilde{R}_j=0,\tilde{C}_k=0\right)\times\\
&\quad P\left(\tilde{X}_0=0,\tilde{R}_j=0,\tilde{C}_k=0\right)+\\
&\quad P\left(X_0=1,R_j=1,C_k=1\,\middle|\,\tilde{X}_0=1,\tilde{R}_j=0,\tilde{C}_k=0\right)\times\\
&\quad P\left(\tilde{X}_0=1,\tilde{R}_j=0,\tilde{C}_k=0\right)+\\
&\quad P\left(X_0=1,R_j=1,C_k=1\,\middle|\,\tilde{X}_0=1,\tilde{R}_j=1,\tilde{C}_k=0\right)\times\\
&\quad P\left(\tilde{X}_0=1,\tilde{R}_j=1,\tilde{C}_k=0\right)+\\
&\quad P\left(X_0=1,R_j=1,C_k=1\,\middle|\,\tilde{X}_0=1,\tilde{R}_j=0,\tilde{C}_k=1\right)\times\\
&\quad P\left(\tilde{X}_0=1,\tilde{R}_j=0,\tilde{C}_k=1\right)+\\
&\quad P\left(X_0=1,R_j=1,C_k=1\,\middle|\,\tilde{X}_0=1,\tilde{R}_j=1,\tilde{C}_k=1\right)\times\\
&\quad P\left(\tilde{X}_0=1,\tilde{R}_j=1,\tilde{C}_k=1\right)
\end{aligned}
$$

Using the conditional independence assumption, we can write

$$
\begin{aligned}
&P\left(X_0=1,R_j=1,C_k=1\right)\\
&\quad= \left(1-S_{p:M}\right)\left(1-S_{p:R}\right)\left(1-S_{p:C}\right)\times\\
&\qquad P\left(\tilde{X}_0=0,\tilde{R}_j=0,\tilde{C}_k=0\right)+ &\text{(B.3.2)}\\
&\qquad S_{e:M}\left(1-S_{p:R}\right)\left(1-S_{p:C}\right)P\left(\tilde{X}_0=1,\tilde{R}_j=0,\tilde{C}_k=0\right)+ &\text{(B.3.3)}\\
&\qquad S_{e:M}S_{e:R}\left(1-S_{p:C}\right)P\left(\tilde{X}_0=1,\tilde{R}_j=1,\tilde{C}_k=0\right)+ &\text{(B.3.4)}\\
&\qquad S_{e:M}\left(1-S_{p:R}\right)S_{e:C}P\left(\tilde{X}_0=1,\tilde{R}_j=0,\tilde{C}_k=1\right)+ &\text{(B.3.5)}\\
&\qquad S_{e:M}S_{e:R}S_{e:C}P\left(\tilde{X}_0=1,\tilde{R}_j=1,\tilde{C}_k=1\right) &\text{(B.3.6)}
\end{aligned}
$$

The jth row and kth column have only individual I_{jk} in common. Under the assumption that the individual statuses are independent, $\tilde{R}_j$ and $\tilde{C}_k$ are also independent, conditional on $\tilde{Y}_{jk}$. Recall that if any row or column tests positive, $\tilde{X}_0 = 1$. Then, we can write

$$P\left(\tilde{X}_0 = 1, \tilde{R}_j = 1, \tilde{C}_k = 1\right) = P\left(\tilde{R}_j = 1, \tilde{C}_k = 1\right),$$

$$P\left(\tilde{X}_0 = 1, \tilde{R}_j = 0, \tilde{C}_k = 1\right) = P\left(\tilde{R}_j = 0, \tilde{C}_k = 1\right),$$

and

$$P\left(\tilde{X}_0 = 1, \tilde{R}_j = 1, \tilde{C}_k = 0\right) = P\left(\tilde{R}_j = 1, \tilde{C}_k = 0\right).$$

Each of these three probabilities were derived in McMahan et al. (2012b, p. 2 in the web appendix). Next, we'll consider the probability expression in equation (B.3.3). In order to have $\tilde{X}_0 = 1$, at least one individual in the array must be truly positive. In this probability expression, we have the added requirement that the truly positive individual(s) is not located in the jth row or the kth column. Then, we can write equation (B.3.3) as

$$\begin{aligned} & P\left(\tilde{X}_0 = 1, \tilde{R}_j = 0, \tilde{C}_k = 0\right) \\ &= P\left(\tilde{X}_0 = 1\right) - \left\{P\left(\tilde{R}_j = 1, \tilde{C}_k = 0\right) + \right. \\ &\quad \left. P\left(\tilde{R}_j = 0, \tilde{C}_k = 1\right) + P\left(\tilde{R}_j = 1, \tilde{C}_k = 1\right)\right\} \\ &= \left\{1 - P\left(\tilde{X}_0 = 0\right)\right\} - P\left(\tilde{R}_j = 1, \tilde{C}_k = 0\right) - \\ &\quad P\left(\tilde{R}_j = 0, \tilde{C}_k = 1\right) - P\left(\tilde{R}_j = 1, \tilde{C}_k = 1\right). \end{aligned}$$

Define $\pi_M = P\left(\tilde{X}_0 = 0\right) = \prod_{j'=1}^{J} \prod_{k'=1}^{K} (1 - p_{j'k'})$. Then, using substitution and some algebraic manipulation, we can write

$$P\left(\tilde{X}_0 = 1, \tilde{R}_j = 0, \tilde{C}_k = 0\right) = \frac{\pi_R(j)\,\pi_C(k)}{1 - p_{jk}} - \pi_M.$$

Note that $\tilde{X}_0 = 0$ if all individuals in the array are truly negative and, hence, all rows and columns are truly negative. Then, the probability expression in equation (B.3.2) is

$$P\left(\tilde{X}_0 = 0, \tilde{R}_j = 0, \tilde{C}_k = 0\right) = P\left(\tilde{X}_0 = 0\right) = \pi_M.$$

Combining these five probability expressions, we get

$$\begin{aligned}
&P(X_0 = 1, R_j = 1, C_k = 1) \\
&= (1 - S_{p:M})(1 - S_{p:R})(1 - S_{p:C})\,\pi_M + \\
&\quad S_{e:M}(1 - S_{p:R})(1 - S_{p:C})\left\{\frac{\pi_R(j)\,\pi_C(k)}{1 - p_{jk}} - \pi_M\right\} + \\
&\quad S_{e:M}S_{e:R}(1 - S_{p:C})\left\{\pi_C(k) - \frac{\pi_R(j)\,\pi_C(k)}{1 - p_{jk}}\right\} + \\
&\quad S_{e:M}(1 - S_{p:R})\,S_{e:C}\left\{\pi_R(j) - \frac{\pi_R(j)\,\pi_C(k)}{1 - p_{jk}}\right\} + \\
&\quad S_{e:M}S_{e:R}S_{e:C}\left\{1 - \pi_R(j) - \pi_C(k) + \frac{\pi_R(j)\,\pi_C(k)}{1 - p_{jk}}\right\}.
\end{aligned}$$

Next, we consider the second probability in equation (B.3.1). To find this probability, recall that the master group must test positive ($X_0 = 1$) in order for row/column testing to occur. Using the Law of Total Probability and the notation presented in Section B.2, we can write

$$P\left(X_0 = 1, R_j = 1, \sum_{k'=1}^{K} C_{k'} = 0\right)$$

$$= \sum_{\tilde{r}=0}^{1}\sum_{\tilde{c}_1=0}^{1}\cdots\sum_{\tilde{c}_K=0}^{1}\left\{P\left(R_j=1,\bigcap_{k=1}^{K}\{C_k=0\}\,\middle|\,X_0=1,\tilde{R}_j=\tilde{r},\bigcap_{k=1}^{K}\left\{\tilde{C}_k=\tilde{c}_k\right\}\right)\times P\left(X_0=1\,\middle|\,\tilde{R}_j=\tilde{r},\bigcap_{k=1}^{K}\left\{\tilde{C}_k=\tilde{c}_k\right\}\right)P\left(\tilde{R}_j=\tilde{r},\bigcap_{k=1}^{K}\left\{\tilde{C}_k=\tilde{c}_k\right\}\right)\right\}.$$

Using the definitions of $\tilde{C}(\mathcal{B})$ and $C(\mathcal{B})$ in Section B.2, this probability can be written as

$$P\left(X_0=1,R_j=1,\sum_{k'=1}^{K}C_{k'}=0\right)$$
$$= \sum_{\tilde{r}=0}^{1}\sum_{c=0}^{K}\sum_{\mathcal{B}\in\mathcal{B}_c}\left[P\left\{R_j=1,C(\mathcal{B}_0)\,\middle|\,X_0=1,\tilde{R}_j=\tilde{r},\tilde{C}(\mathcal{B})\right\}\times P\left\{X_0=1\,\middle|\,\tilde{R}_j=\tilde{r},\tilde{C}(\mathcal{B})\right\}P\left\{\tilde{R}_j=\tilde{r},\tilde{C}(\mathcal{B})\right\}\right]. \quad \text{(B.3.7)}$$

Since a master group must test positive for a row to be tested, we have

$$P\left\{R_j=1,C(\mathcal{B}_0)\,\middle|\,X_0=1,\tilde{R}_j=0,\tilde{C}(\mathcal{B}_c)\right\} = P\left\{R_j=1,C(\mathcal{B}_0)\,\middle|\,\tilde{R}_j=0,\tilde{C}(\mathcal{B}_c)\right\}$$

and

$$P\left\{R_j=1,C(\mathcal{B}_0)\,\middle|\,X_0=1,\tilde{R}_j=1,\tilde{C}(\mathcal{B}_c)\right\} = P\left\{R_j=1,C(\mathcal{B}_0)\,\middle|\,\tilde{R}_j=1,\tilde{C}(\mathcal{B}_c)\right\}.$$

Derivations for both of these resulting probability expressions were shown in Section B.2.1. Note that $P\left\{R_j=1,C(\mathcal{B}_0)\,\middle|\,\tilde{R}_j=1,\tilde{C}(\mathcal{B}_0)\right\}=0$ because we cannot have a truly positive row when all columns are truly negative.

From Section B.2, we recall that

$$P\left\{\tilde{R}_j=0,\tilde{C}(\mathcal{B}_c)\right\} = \pi_R(j)\,\lambda_{\mathcal{C}}(\mathcal{B}_c\mid\mathcal{K}_0,j)$$

and

$$P\left\{\tilde{R}_j = 1, \tilde{C}\left(\mathcal{B}_c\right)\right\} = \lambda_{\mathcal{C}}\left(\mathcal{B}_c \mid \emptyset, j\right) - \pi_R\left(j\right)\lambda_{\mathcal{C}}\left(\mathcal{B}_c \mid \mathcal{K}_0, j\right). \tag{B.3.8}$$

Now, we consider the "middle" probability expression in equation (B.3.7). We have two cases:

1. When $c = 0$,

$$\begin{aligned} P\left\{X_0 = 1 \;\middle|\; \tilde{R}_j = 0, \tilde{C}\left(\mathcal{B}_0\right)\right\} &= P\left(X_0 = 1 \;\middle|\; \tilde{X}_0 = 0\right) \\ &= 1 - P\left(X_0 = 0 \;\middle|\; \tilde{X}_0 = 0\right) \\ &= 1 - S_{p:M} \end{aligned}$$

   and

$$P\left\{X_0 = 1 \;\middle|\; \tilde{R}_j = 1, \tilde{C}\left(\mathcal{B}_0\right)\right\} = 0.$$

2. When $c = 1, ..., K$ (i.e., at least one column in the array is truly positive), we see that

$$P\left\{X_0 = 1 \;\middle|\; \tilde{R}_j = 0, \tilde{C}\left(\mathcal{B}_c\right)\right\} = P\left(X_0 = 1 \;\middle|\; \tilde{X}_0 = 1\right) = S_{e:M}$$

   and

$$P\left\{X_0 = 1 \;\middle|\; \tilde{R}_j = 1, \tilde{C}\left(\mathcal{B}_c\right)\right\} = P\left(X_0 = 1 \;\middle|\; \tilde{X}_0 = 1\right) = S_{e:M}.$$

Substituting back into equation (B.3.7), we get

$$P\left(X_0 = 1, R_j = 1, \sum_{k'=1}^{K} C_{k'} = 0\right)$$

$$
\begin{aligned}
= & \sum_{c=0}^{K} \sum_{\mathcal{B} \in \mathcal{B}_c} \Big[\left(1 - S_{p:R}\right) \left(1 - S_{e:C}\right)^c \left(S_{p:C}\right)^{K-c} \times \\
& P\left\{ X_0 = 1 \;\middle|\; \tilde{R}_j = 0, \tilde{C}\left(\mathcal{B}\right) \right\} \pi_R\left(j\right) \lambda_{\mathcal{C}}\left(\mathcal{B} \mid \mathcal{K}_0, j\right) + \\
& S_{e:R} \left(1 - S_{e:C}\right)^c \left(S_{p:C}\right)^{K-c} P\left\{ X_0 = 1 \;\middle|\; \tilde{R}_j = 1, \tilde{C}\left(\mathcal{B}\right) \right\} \times \\
& \left\{ \lambda_{\mathcal{C}}\left(\mathcal{B} \mid \emptyset, j\right) - \pi_R\left(j\right) \lambda_{\mathcal{C}}\left(\mathcal{B} \mid \mathcal{K}_0, j\right) \right\} \Big] .
\end{aligned}
$$

After extensive algebra, we can write the second probability in equation (B.3.1) as

$$
\begin{aligned}
& P\left(X_0 = 1, R_j = 1, \sum_{k'=1}^{K} C_{k'} = 0 \right) \\
& \quad = \left(1 - S_{p:M} - S_{e:M}\right) \left(1 - S_{p:R}\right) \left(S_{p:C}\right)^K \pi_R\left(j\right) \lambda_{\mathcal{C}}\left(\mathcal{B}_0 \mid \mathcal{K}_0, j\right) + \\
& \quad\quad S_{e:M} \sum_{c=0}^{K} \sum_{\mathcal{B} \in \mathcal{B}_c} \left\{ \gamma_0\left(c, K\right) \lambda_{\mathcal{C}}\left(\mathcal{B} \mid \emptyset, j\right) + \right. \\
& \quad\quad \left. \gamma_1\left(c, K\right) \pi_R\left(j\right) \lambda_{\mathcal{C}}\left(\mathcal{B} \mid \mathcal{K}_0, j\right) \right\} , \qquad \text{(B.3.9)}
\end{aligned}
$$

where $\gamma_0\left(c, K\right)$ and $\gamma_1\left(c, K\right)$ are as defined in Section B.2.

In a similar manner, the third probability in equation (B.3.1) can be expressed as

$$
\begin{aligned}
& P\left(X_0 = 1, \sum_{j'=1}^{J} R_{j'} = 0, C_k = 1 \right) \\
& \qquad = S_{e:M} \sum_{r=0}^{J} \sum_{\mathcal{A} \in \mathcal{A}_r} \left\{ \gamma_2\left(r, J\right) \lambda_R\left(\mathcal{A} \mid \emptyset, k\right) + \right. \\
& \qquad\quad \left. \gamma_3\left(r, J\right) \pi_C\left(k\right) \lambda_R\left(\mathcal{A} \mid \mathcal{J}_0, k\right) \right\} ,
\end{aligned}
$$

where $\gamma_2\left(r, J\right)$ and $\gamma_3\left(r, J\right)$ are as defined in Section B.2. This follows from equation (B.3.9) by treating the rows as columns and vice versa. This completes the derivation of $E\left(T_{2jk}\right)$ in equation (B.3.1).

B.3.2. Accuracy measures

Recall from Section B.2 that $I_{jk}^{+}\left(I_{jk}^{-}\right)$ denotes the event that individual I_{jk} is classified as positive (negative) by the group testing algorithm.

B.3.2.1. Pooling sensitivity

For array testing with master pooling, individual I_{jk} is classified as positive if the master group test is positive, the corresponding row and/or column tests are positive, and the individual test is positive. Then, the pooling sensitivity is

$$\begin{aligned} PS_e^{I_{jk}} &= P\left(I_{jk}^{+} \,\middle|\, \tilde{Y}_{jk}=1\right) \\ &= P\left(X_0=1,R_j=1,C_k=1,Y_{jk}=1 \,\middle|\, \tilde{Y}_{jk}=1\right)+ \qquad \text{(B.3.10)} \\ &\quad P\left(X_0=1,R_j=1,\sum_{k'=1}^{K}C_k=0,Y_{jk}=1 \,\middle|\, \tilde{Y}_{jk}=1\right)+ \qquad \text{(B.3.11)} \\ &\quad P\left(X_0=1,\sum_{j'=1}^{J}R_j=0,C_k=1,Y_{jk}=1 \,\middle|\, \tilde{Y}_{jk}=1\right). \qquad \text{(B.3.12)} \end{aligned}$$

If $\tilde{Y}_{jk}=1$, then $\tilde{R}_j=1$, $\tilde{C}_k=1$, and $\tilde{X}_0=1$. This fact, together with the conditional independence assumption, implies that equation (B.3.10) is

$$\begin{aligned} &P\left(X_0=1,R_j=1,C_k=1,Y_{jk}=1 \,\middle|\, \tilde{Y}_{jk}=1\right) \\ &\quad = P\left(X_0=1 \,\middle|\, \tilde{Y}_{jk}=1\right)P\left(R_j=1 \,\middle|\, \tilde{Y}_{jk}=1\right)\times \\ &\qquad P\left(C_k=1 \,\middle|\, \tilde{Y}_{jk}=1\right)P\left(Y_{jk}=1 \,\middle|\, \tilde{Y}_{jk}=1\right) \\ &\quad = S_{e:M}S_{e:R}S_{e:C}S_{e:I}. \end{aligned}$$

Similarly, equation (B.3.11) can be written as

$$\begin{aligned} & P\left(X_0 = 1, R_j = 1, \sum_{k'=1}^{K} C_k = 0, Y_{jk} = 1 \;\middle|\; \tilde{Y}_{jk} = 1\right) \\ &= P\left(X_0 = 1 \;\middle|\; \tilde{Y}_{jk} = 1\right) P\left(R_j = 1 \;\middle|\; \tilde{Y}_{jk} = 1\right) \times \\ &\quad P\left(\sum_{k'=1}^{K} C_k = 0 \;\middle|\; \tilde{Y}_{jk} = 1\right) P\left(Y_{jk} = 1 \;\middle|\; \tilde{Y}_{jk} = 1\right) \\ &= S_{e:M} S_{e:R} S_{e:I} P\left(\sum_{k'=1}^{K} C_k = 0 \;\middle|\; \tilde{C}_k = 1\right). \end{aligned}$$

Because C_k is independent of $C_{k'}$ for all $k \neq k'$, we can write

$$\begin{aligned} P\left(\sum_{k'=1}^{K} C_k = 0 \;\middle|\; \tilde{C}_k = 1\right) &= P\left(\bigcap_{k=1}^{K} \{C_k = 0\} \;\middle|\; \tilde{C}_k = 1\right) \\ &= P\left(C_k = 0 \;\middle|\; \tilde{C}_k = 1\right) P\left(\bigcap_{k' \neq k} \{C_k = 0\}\right) \\ &= (1 - S_{e:C}) \prod_{k' \neq k} P\left(C_{k'} = 0\right), \end{aligned}$$

where $P\left(C_{k'} = 0\right)$ was derived in Section B.2 to allow for unequal sensitivities and unequal specificities across stages of testing. Therefore, equation (B.3.11) can be written as

$$\begin{aligned} & P\left(X_0 = 1, R_j = 1, \sum_{k'=1}^{K} C_k = 0, Y_{jk} = 1 \;\middle|\; \tilde{Y}_{jk} = 1\right) \\ &\quad = S_{e:M} S_{e:R} S_{e:I} \left(1 - S_{e:C}\right) \prod_{k' \neq k} P\left(C_{k'} = 0\right). \end{aligned} \tag{B.3.13}$$

Finally, equation (B.3.12) can be written as

$$P\left(X_0 = 1, \sum_{j'=1}^{J} R_j = 0, C_k = 1, Y_{jk} = 1 \;\middle|\; \tilde{Y}_{jk} = 1\right)$$

$$= \quad S_{e:M}S_{e:C}S_{e:I}\left(1-S_{e:R}\right)\prod_{j'\neq j}P\left(R_{j'}=0\right).$$

This follows from equation (B.3.13) by treating the rows as columns and vice versa. Combining these results, we obtain

$$\begin{aligned}
PS_e^{I_{jk}} &= S_{e:M}S_{e:R}S_{e:C}S_{e:I}+S_{e:M}S_{e:R}S_{e:I}\left(1-S_{e:C}\right)\prod_{k'\neq k}P\left(C_{k'}=0\right)+ \\
&\quad S_{e:M}S_{e:C}S_{e:I}\left(1-S_{e:R}\right)\prod_{j'\neq j}P\left(R_{j'}=0\right) \\
&= S_{e:M}S_{e:I}\left\{S_{e:R}S_{e:C}+S_{e:R}\left(1-S_{e:C}\right)\prod_{k'\neq k}P\left(C_{k'}=0\right)+\right. \\
&\quad \left. S_{e:C}\left(1-S_{e:R}\right)\prod_{j'\neq j}P\left(R_{j'}=0\right)\right\}.
\end{aligned}$$

B.3.2.2. Pooling specificity

We now turn to the derivation of the pooling specificity, $PS_p^{I_{jk}}$. By definition,

$$\begin{aligned}
PS_p^{I_{jk}} &= P\left(I_{jk}^{-}\,\middle|\,\tilde{Y}_{jk}=0\right) \\
&= 1-P\left(I_{jk}^{+}\,\middle|\,\tilde{Y}_{jk}=0\right),
\end{aligned}$$

where

$$\begin{aligned}
&P\left(I_{jk}^{+}\,\middle|\,\tilde{Y}_{jk}=0\right) \\
&\quad = P\left(X_0=1,R_j=1,C_k=1,Y_{jk}=1\,\middle|\,\tilde{Y}_{jk}=0\right)+ && \text{(B.3.14)} \\
&\quad P\left(X_0=1,R_j=1,\sum_{k'=1}^{K}C_k=0,Y_{jk}=1\,\middle|\,\tilde{Y}_{jk}=0\right)+ && \text{(B.3.15)} \\
&\quad P\left(X_0=1,\sum_{j'=1}^{J}R_j=0,C_k=1,Y_{jk}=1\,\middle|\,\tilde{Y}_{jk}=0\right). && \text{(B.3.16)}
\end{aligned}$$

Using the conditional independence assumption, equation (B.3.14) can be written as

$$\begin{aligned} & P\left(X_0 = 1, R_j = 1, C_k = 1, Y_{jk} = 1 \;\middle|\; \tilde{Y}_{jk} = 0\right) \\ &\quad = P\left(Y_{jk} = 1 \;\middle|\; \tilde{Y}_{jk} = 0\right) P\left(X_0 = 1, R_j = 1, C_k = 1 \;\middle|\; \tilde{Y}_{jk} = 0\right) \\ &\quad = (1 - S_{p:I})\, P\left(X_0 = 1, R_j = 1, C_k = 1 \;\middle|\; \tilde{Y}_{jk} = 0\right). \end{aligned} \tag{B.3.17}$$

Using the Law of Total Probability, the probability expression on the right-hand side of equation (B.3.17) can be written as

$$\begin{aligned} & P\left(X_0 = 1, R_j = 1, C_k = 1 \;\middle|\; \tilde{Y}_{jk} = 0\right) \\ &\quad = P\left(X_0 = 1, R_j = 1, C_k = 1 \;\middle|\; \tilde{X}_0 = 0, \tilde{R}_j = 0, \tilde{C}_k = 0, \tilde{Y}_{jk} = 0\right) \times \\ &\qquad P\left(\tilde{X}_0 = 0, \tilde{R}_j = 0, \tilde{C}_k = 0 \;\middle|\; \tilde{Y}_{jk} = 0\right) + \\ &\qquad P\left(X_0 = 1, R_j = 1, C_k = 1 \;\middle|\; \tilde{X}_0 = 1, \tilde{R}_j = 0, \tilde{C}_k = 0, \tilde{Y}_{jk} = 0\right) \times \\ &\qquad P\left(\tilde{X}_0 = 1, \tilde{R}_j = 0, \tilde{C}_k = 0 \;\middle|\; \tilde{Y}_{jk} = 0\right) + \\ &\qquad P\left(X_0 = 1, R_j = 1, C_k = 1 \;\middle|\; \tilde{X}_0 = 1, \tilde{R}_j = 1, \tilde{C}_k = 0, \tilde{Y}_{jk} = 0\right) \times \\ &\qquad P\left(\tilde{X}_0 = 1, \tilde{R}_j = 1, \tilde{C}_k = 0 \;\middle|\; \tilde{Y}_{jk} = 0\right) + \\ &\qquad P\left(X_0 = 1, R_j = 1, C_k = 1 \;\middle|\; \tilde{X}_0 = 1, \tilde{R}_j = 0, \tilde{C}_k = 1, \tilde{Y}_{jk} = 0\right) \times \\ &\qquad P\left(\tilde{X}_0 = 1, \tilde{R}_j = 0, \tilde{C}_k = 1 \;\middle|\; \tilde{Y}_{jk} = 0\right) + \\ &\qquad P\left(X_0 = 1, R_j = 1, C_k = 1 \;\middle|\; \tilde{X}_0 = 1, \tilde{R}_j = 1, \tilde{C}_k = 1, \tilde{Y}_{jk} = 0\right) \times \\ &\qquad P\left(\tilde{X}_0 = 1, \tilde{R}_j = 1, \tilde{C}_k = 1 \;\middle|\; \tilde{Y}_{jk} = 0\right). \end{aligned}$$

Recall that $\tilde{X}_0 = 0$ if all individuals in the array are truly negative and, hence, all rows and columns are truly negative. Additionally, $\tilde{X}_0 = 1$ if at least one

row or at least one column is truly positive. Then, we have

$$
\begin{aligned}
& P\left(X_0 = 1, R_j = 1, C_k = 1 \,\middle|\, \tilde{Y}_{jk} = 0\right) \\
&= P\left(X_0 = 1 \,\middle|\, \tilde{X}_0 = 0\right) P\left(R_j = 1 \,\middle|\, \tilde{R}_j = 0\right) \times \\
&\quad P\left(C_k = 1 \,\middle|\, \tilde{C}_k = 0\right) P\left(\tilde{X}_0 = 0 \,\middle|\, \tilde{Y}_{jk} = 0\right) + \\
&\quad P\left(X_0 = 1 \,\middle|\, \tilde{X}_0 = 1\right) P\left(R_j = 1 \,\middle|\, \tilde{R}_j = 0\right) P\left(C_k = 1 \,\middle|\, \tilde{C}_k = 0\right) \times \\
&\quad P\left(\tilde{X}_0 = 1, \tilde{R}_j = 0, \tilde{C}_k = 0 \,\middle|\, \tilde{Y}_{jk} = 0\right) + \\
&\quad P\left(X_0 = 1 \,\middle|\, \tilde{X}_0 = 1\right) P\left(R_j = 1 \,\middle|\, \tilde{R}_j = 1\right) P\left(C_k = 1 \,\middle|\, \tilde{C}_k = 0\right) \times \\
&\quad P\left(\tilde{R}_j = 1 \,\middle|\, \tilde{Y}_{jk} = 0\right) P\left(\tilde{C}_k = 0 \,\middle|\, \tilde{Y}_{jk} = 0\right) + \\
&\quad P\left(X_0 = 1 \,\middle|\, \tilde{X}_0 = 1\right) P\left(R_j = 1 \,\middle|\, \tilde{R}_j = 0\right) P\left(C_k = 1 \,\middle|\, \tilde{C}_k = 1\right) \times \\
&\quad P\left(\tilde{R}_j = 0 \,\middle|\, \tilde{Y}_{jk} = 0\right) P\left(\tilde{C}_k = 1 \,\middle|\, \tilde{Y}_{jk} = 0\right) + \\
&\quad P\left(X_0 = 1 \,\middle|\, \tilde{X}_0 = 1\right) P\left(R_j = 1 \,\middle|\, \tilde{R}_j = 1\right) P\left(C_k = 1 \,\middle|\, \tilde{C}_k = 1\right) \times \\
&\quad P\left(\tilde{R}_j = 1 \,\middle|\, \tilde{Y}_{jk} = 0\right) P\left(\tilde{C}_k = 1 \,\middle|\, \tilde{Y}_{jk} = 0\right) \\
&= (1 - S_{p:M})(1 - S_{p:R})(1 - S_{p:C}) P\left(\tilde{X}_0 = 0 \,\middle|\, \tilde{Y}_{jk} = 0\right) + \\
&\quad S_{e:M}(1 - S_{p:R})(1 - S_{p:C}) P\left(\tilde{X}_0 = 1, \tilde{R}_j = 0, \tilde{C}_k = 0 \,\middle|\, \tilde{Y}_{jk} = 0\right) + \\
&\quad S_{e:M} S_{e:R}(1 - S_{p:C}) \left\{1 - P\left(\tilde{R}_j = 0 \,\middle|\, \tilde{Y}_{jk} = 0\right)\right\} \times \\
&\quad P\left(\tilde{C}_k = 0 \,\middle|\, \tilde{Y}_{jk} = 0\right) + \\
&\quad S_{e:M}(1 - S_{p:R}) S_{e:C} P\left(\tilde{R}_j = 0 \,\middle|\, \tilde{Y}_{jk} = 0\right) \times \\
&\quad \left\{1 - P\left(\tilde{C}_k = 0 \,\middle|\, \tilde{Y}_{jk} = 0\right)\right\} + \\
&\quad S_{e:M} S_{e:R} S_{e:C} \left\{1 - P\left(\tilde{R}_j = 0 \,\middle|\, \tilde{Y}_{jk} = 0\right)\right\} \times \\
&\quad \left\{1 - P\left(\tilde{C}_k = 0 \,\middle|\, \tilde{Y}_{jk} = 0\right)\right\}. \qquad \text{(B.3.18)}
\end{aligned}
$$

Expressions for $P\left(\tilde{R}_j = 0 \,\middle|\, \tilde{Y}_{jk} = 0\right)$ and $P\left(\tilde{C}_k = 0 \,\middle|\, \tilde{Y}_{jk} = 0\right)$ were presented by McMahan et al. (2012b) and utilized in derivations presented in

Section B.2. Recall that $\pi_M = P\left(\tilde{X}_0 = 0\right) = \prod_{j',k'} (1 - p_{j'k'})$. Then, we have

$$P\left(\tilde{X}_0 = 0 \,\middle|\, \tilde{Y}_{jk} = 0\right) = \frac{P\left(\tilde{X}_0 = 0, \tilde{Y}_{jk} = 0\right)}{P\left(\tilde{Y}_{jk} = 0\right)} = \frac{P\left(\tilde{X}_0 = 0\right)}{P\left(\tilde{Y}_{jk} = 0\right)} = \frac{\pi_M}{1 - p_{jk}}.$$

For the remaining probability expression in equation (B.3.18), Bayes' theorem allows us to write

$$\begin{aligned} & P\left(\tilde{X}_0 = 1, \tilde{R}_j = 0, \tilde{C}_k = 0 \,\middle|\, \tilde{Y}_{jk} = 0\right) \\ &= \frac{P\left(\tilde{X}_0 = 1, \tilde{R}_j = 0, \tilde{C}_k = 0\right) P\left(\tilde{Y}_{jk} = 0 \,\middle|\, \tilde{X}_0 = 1, \tilde{R}_j = 0, \tilde{C}_k = 0\right)}{P\left(\tilde{Y}_{jk} = 0\right)} \\ &= \frac{P\left(\tilde{X}_0 = 1, \tilde{R}_j = 0, \tilde{C}_k = 0\right)}{P\left(\tilde{Y}_{jk} = 0\right)}. \end{aligned}$$

From Section B.2, we have

$$P\left(\tilde{X}_0 = 1, \tilde{R}_j = 0, \tilde{C}_k = 0\right) = \frac{\pi_R(j)\,\pi_C(k)}{1 - p_{jk}} - \pi_M.$$

Thus, we can write

$$\begin{aligned} & P\left(\tilde{X}_0 = 1, \tilde{R}_j = 0, \tilde{C}_k = 0 \,\middle|\, \tilde{Y}_{jk} = 0\right) \\ &= \frac{P\left(\tilde{X}_0 = 1, \tilde{R}_j = 0, \tilde{C}_k = 0\right)}{P\left(\tilde{Y}_{jk} = 0\right)} \\ &= \frac{1}{1 - p_{jk}} \left\{ \frac{\pi_R(j)\,\pi_C(k)}{1 - p_{jk}} - \pi_M \right\} \\ &= \frac{\pi_R(j)\,\pi_C(k)}{(1 - p_{jk})^2} - \frac{\pi_M}{1 - p_{jk}}. \end{aligned}$$

Substituting back into the probability expression on the right-hand side of equation (B.3.17), we get

$$
\begin{aligned}
&P\left(X_0 = 1, R_j = 1, C_k = 1 \;\middle|\; \tilde{Y}_{jk} = 0\right) \\
&\quad = \left(1 - S_{p:M}\right)\left(1 - S_{p:R}\right)\left(1 - S_{p:C}\right)\frac{\pi_M}{1 - p_{jk}} + \\
&\qquad S_{e:M}\left(1 - S_{p:R}\right)\left(1 - S_{p:C}\right)\left\{\frac{\pi_R(j)\,\pi_C(k)}{\left(1 - p_{jk}\right)^2} - \frac{\pi_M}{1 - p_{jk}}\right\} + \\
&\qquad S_{e:M}S_{e:R}\left(1 - S_{p:C}\right)\left\{\frac{\pi_C(k)}{1 - p_{jk}} - \frac{\pi_R(j)\,\pi_C(k)}{\left(1 - p_{jk}\right)^2}\right\} + \\
&\qquad S_{e:M}\left(1 - S_{p:R}\right)S_{e:C}\left\{\frac{\pi_R(j)}{1 - p_{jk}} - \frac{\pi_R(j)\,\pi_C(k)}{\left(1 - p_{jk}\right)^2}\right\} + \\
&\qquad S_{e:M}S_{e:R}S_{e:C}\left\{1 - \frac{\pi_R(j)}{1 - p_{jk}} - \frac{\pi_C(k)}{1 - p_{jk}} + \frac{\pi_R(j)\,\pi_C(k)}{\left(1 - p_{jk}\right)^2}\right\}.
\end{aligned}
$$

Thus, equation (B.3.14) can be written as

$$
\begin{aligned}
&P\left(X_0 = 1, R_j = 1, C_k = 1, Y_{jk} = 1 \;\middle|\; \tilde{Y}_{jk} = 0\right) \\
&\quad = \left(1 - S_{p:I}\right)\left[\left(1 - S_{p:M}\right)\left(1 - S_{p:R}\right)\left(1 - S_{p:C}\right)\frac{\pi_M}{1 - p_{jk}} + \right. \\
&\qquad S_{e:M}\left(1 - S_{p:R}\right)\left(1 - S_{p:C}\right)\left\{\frac{\pi_R(j)\,\pi_C(k)}{\left(1 - p_{jk}\right)^2} - \frac{\pi_M}{1 - p_{jk}}\right\} + \\
&\qquad S_{e:M}S_{e:R}\left(1 - S_{p:C}\right)\left\{\frac{\pi_C(k)}{1 - p_{jk}} - \frac{\pi_R(j)\,\pi_C(k)}{\left(1 - p_{jk}\right)^2}\right\} + \\
&\qquad S_{e:M}\left(1 - S_{p:R}\right)S_{e:C}\left\{\frac{\pi_R(j)}{1 - p_{jk}} - \frac{\pi_R(j)\,\pi_C(k)}{\left(1 - p_{jk}\right)^2}\right\} + \\
&\qquad \left. S_{e:M}S_{e:R}S_{e:C}\left\{1 - \frac{\pi_R(j)}{1 - p_{jk}} - \frac{\pi_C(k)}{1 - p_{jk}} + \frac{\pi_R(j)\,\pi_C(k)}{\left(1 - p_{jk}\right)^2}\right\}\right].
\end{aligned}
$$

Using conditional independence, we can write equation (B.3.15) as

$$
\begin{aligned}
&P\left(X_0 = 1, R_j = 1, \sum_{k'=1}^{K} C_{k'} = 0, Y_{jk} = 1 \;\middle|\; \tilde{Y}_{jk} = 0\right) \\
&\quad = \; P\left(Y_{jk} = 1 \;\middle|\; \tilde{Y}_{jk} = 0\right) P\left(X_0 = 1, R_j = 1, \sum_{k'=1}^{K} C_{k'} = 0 \;\middle|\; \tilde{Y}_{jk} = 0\right) \\
&\quad = \; (1 - S_{p:I})\, P\left(X_0 = 1, R_j = 1, \sum_{k'=1}^{K} C_{k'} = 0 \;\middle|\; \tilde{Y}_{jk} = 0\right).
\end{aligned}
$$

Conditioning on the true statuses of the rows and columns, the probability expression on the right-hand side becomes

$$
\begin{aligned}
&P\left(X_0 = 1, R_j = 1, \sum_{k'=1}^{K} C_{k'} = 0 \;\middle|\; \tilde{Y}_{jk} = 0\right) \\
&\quad = \; \sum_{\tilde{r}=0}^{1} \sum_{c=0}^{K} \sum_{\mathcal{B} \in \mathcal{B}_c} P\left\{R_j = 1, C(\mathcal{B}_0) \;\middle|\; X_0 = 1, \tilde{R}_j = \tilde{r}, \tilde{C}(\mathcal{B}), \tilde{Y}_{jk} = 0\right\} \times \\
&\qquad P\left(X_0 = 1 \;\middle|\; \tilde{R}_j = \tilde{r}, \tilde{C}(\mathcal{B}), \tilde{Y}_{jk} = 0\right) \times \\
&\qquad P\left(\tilde{R}_j = \tilde{r}, \tilde{C}(\mathcal{B}) \;\middle|\; \tilde{Y}_{jk} = 0\right). \qquad \text{(B.3.19)}
\end{aligned}
$$

Note that

$$
\begin{aligned}
&P\left\{R_j = 1, C(\mathcal{B}_0) \;\middle|\; X_0 = 1, \tilde{R}_j = 0, \tilde{C}(\mathcal{B}_c), \tilde{Y}_{jk} = 0\right\} \\
&\qquad = \; P\left\{R_j = 1, C(\mathcal{B}_0) \;\middle|\; \tilde{R}_j = 0, \tilde{C}(\mathcal{B}_c), \tilde{Y}_{jk} = 0\right\}
\end{aligned}
$$

and

$$
\begin{aligned}
&P\left\{R_j = 1, C(\mathcal{B}_0) \;\middle|\; X_0 = 1, \tilde{R}_j = 1, \tilde{C}(\mathcal{B}_c), \tilde{Y}_{jk} = 0\right\} \\
&\qquad = \; P\left\{R_j = 1, C(\mathcal{B}_0) \;\middle|\; \tilde{R}_j = 1, \tilde{C}(\mathcal{B}_c), \tilde{Y}_{jk} = 0\right\}.
\end{aligned}
$$

Expressions for both of these probabilities were provided in Section B.2. Expressions for the probabilities consisting only of $\tilde{R}_j$ and $\tilde{C}(\mathcal{B})$ conditioned on $\tilde{Y}_{jk}$ are the same as those given by McMahan et al. (2012b). Now, we consider the "middle" probability expression in equation (B.3.19). We have two different cases:

1. When $c = 0$, we have

$$\begin{aligned} P\left\{X_0 = 1 \,\middle|\, \tilde{R}_j = 0, \tilde{C}(\mathcal{B}_0), \tilde{Y}_{jk} = 0\right\} &= P\left(X_0 = 1 \,\middle|\, \tilde{X}_0 = 0\right) \\ &= 1 - S_{p:M} \end{aligned}$$

and

$$P\left\{X_0 = 1 \,\middle|\, \tilde{R}_j = 1, \tilde{C}(\mathcal{B}_0), \tilde{Y}_{jk} = 0\right\} = 0.$$

2. When $c = 1, ..., K$ (i.e., at least one column in the array is truly positive), we see that

$$\begin{aligned} P\left(X_0 = 1 \,\middle|\, \tilde{R}_j = 0, \tilde{C}(\mathcal{B}_c), \tilde{Y}_{jk} = 0\right) &= P\left(X_0 = 1 \,\middle|\, \tilde{X}_0 = 1\right) \\ &= S_{e:M} \end{aligned}$$

and

$$\begin{aligned} P\left(X_0 = 1 \,\middle|\, \tilde{R}_j = 1, \tilde{C}(\mathcal{B}_c), \tilde{Y}_{jk} = 0\right) &= P\left(X_0 = 1 \,\middle|\, \tilde{X}_0 = 1\right) \\ &= S_{e:M}. \end{aligned}$$

Substituting back into equation (B.3.19) and performing a significant amount of algebraic manipulation gives us

$$
\begin{aligned}
&P\left(X_0 = 1, R_j = 1, \sum_{k'=1}^{K} C_{k'} = 0 \;\middle|\; \tilde{Y}_{jk} = 0\right) \\
&\quad = \left(1 - S_{p:M}\right)\left(1 - S_{p:R}\right)\left(S_{p:C}\right)^K \frac{\pi_R(j)}{1 - p_{jk}} \lambda_C\left(\mathcal{B}_0 \mid \mathcal{K}_0, j\right) + \\
&\qquad S_{e:M} \sum_{c=1}^{K} \sum_{\mathcal{B} \in \mathcal{B}_c} \left\{ \gamma_0(c, K)\, \lambda_{\mathcal{C}}\left(\mathcal{B} \mid \{k\}, j\right) + \right. \\
&\qquad \left. \gamma_1(c, K) \frac{\pi_R(j)}{1 - p_{jk}} \lambda_C\left(\mathcal{B} \mid \mathcal{K}_0, j\right) \right\},
\end{aligned}
$$

where $\gamma_0(c, K)$ and $\gamma_1(c, K)$ are as defined previously. Then, the expression for equation (B.3.15) is

$$
\begin{aligned}
&P\left(X_0 = 1, R_j = 1, \sum_{k'=1}^{K} C_{k'} = 0, Y_{jk} = 1 \;\middle|\; \tilde{Y}_{jk} = 0\right) \\
&\quad = \left(1 - S_{p:I}\right) \left[\left(1 - S_{p:M}\right)\left(1 - S_{p:R}\right)\left(S_{p:C}\right)^K \frac{\pi_R(j)}{1 - p_{jk}} \lambda_C\left(\mathcal{B}_0 \mid \mathcal{K}_0, j\right) + \right. \\
&\qquad S_{e:M} \sum_{c=1}^{K} \sum_{\mathcal{B} \in \mathcal{B}_c} \left\{ \gamma_0(c, K)\, \lambda_{\mathcal{C}}\left(\mathcal{B} \mid \{k\}, j\right) + \right. \\
&\qquad \left. \left. \gamma_1(c, K) \frac{\pi_R(j)}{1 - p_{jk}} \lambda_C\left(\mathcal{B} \mid \mathcal{K}_0, j\right) \right\} \right]. \qquad \text{(B.3.20)}
\end{aligned}
$$

In a similar manner, equation (B.3.16) can be written as

$$
P\left(X_0 = 1, \sum_{j'=1}^{J} R_{j'} = 0, C_k = 1, Y_{jk} = 1 \;\middle|\; \tilde{Y}_{jk} = 0\right)
$$

$$
\begin{aligned}
= \quad & (1 - S_{p:I}) \left[(1 - S_{p:M}) \left(S_{p:R} \right)^J (1 - S_{p:C}) \, \lambda_{\mathcal{R}} \left(\mathcal{A}_0 \mid \mathcal{J}_0, k \right) \frac{\pi_C (k)}{1 - p_{jk}} + \right. \\
& S_{e:M} \sum_{r=1}^{J} \sum_{\mathcal{A} \in \mathcal{A}_r} \left\{ \gamma_2 (r, J) \, \lambda_{\mathcal{R}} \left(\mathcal{A} \mid \{j\}, k \right) + \right. \\
& \left. \left. \gamma_3 (r, J) \, \lambda_{\mathcal{R}} \left(\mathcal{A} \mid \mathcal{J}_0, k \right) \frac{\pi_C (k)}{1 - p_{jk}} \right\} \right],
\end{aligned}
$$

where $\gamma_2 (r, J)$ and $\gamma_3 (r, J)$ are as previously defined. This follows from equation (B.3.20) by treating the rows as columns and vice versa. This concludes the derivations for $PS_p^{I_{jk}}$.

Appendix C

R function documentation

This appendix contains important parts of the documentation for the `binGroup2` package. Included are 1) the help page for the package as a whole, 2) the index of available functions, and 3) the help documents for the `opChar1()`, `opChar2()`, `OTC1()`, and `OTC2()` functions.

binGroup2: Identification and Estimation using Group Testing

Description

Methods for the group testing identification and estimation problems.

Details

Methods for identification of positive items in group testing designs: Operating characteristics (e.g., expected number of tests) are calculated for commonly used hierarchical and array-based algorithms. Optimal testing configurations for an algorithm can be found as well. Please see Hitt et al. (2019) for specific details.

Methods for estimation and inference for proportions in group testing designs: For estimating one proportion or the difference of proportions, confidence interval methods are included that account for different pool sizes. Functions for hypothesis testing of proportions, calculation of power, and calculation of the expected width of confidence intervals are also included. Furthermore, regression methods and simulation of group testing data are implemented for simple pooling, halving, and array testing designs.

The `binGroup2` package is based upon the `binGroup` package that was originally designed for the group testing estimation problem. Over time, additional functions for estimation and for the group testing identification problem were included. Due to the diverse styles resulting from these additions, we have created `binGroup2` as a way to unify functions in a coherent structure and incorporate additional functions for identification. The name “binGroup” originates from the assumption in basic estimation for group testing that the number of positive groups has a binomial distribution. While more advanced estimation methods no longer make this assumption, we continue with the `binGroup` name for consistency.

Bilder (2019a,b) provide introductions to group testing. These papers and additional details about group testing are available at http://chrisbilder.com/grouptesting.

This research was supported by the National Institutes of Health under grant R01 AI121351.

Identification

The binGroup2 package focuses on the group testing identification problem using hierarchical and array-based group testing algorithms.

The OTC1 function implements a number of group testing algorithms, described in Hitt et al. (2019), which calculate the operating characteristics and find the optimal testing configuration over a range of possible initial group sizes and/or testing configurations (sets of subsequent group sizes).
The OTC2 function does the same with a multiplex assay that tests for two diseases.

The operatingCharacteristics1 (opChar1) and operatingCharacteristics2(opChar2) functions calculate operating characteristics for a specified testing configuration with assays that test for one and two diseases, respectively.

These functions allow the sensitivity and specificity to differ across stages of testing. This means that the accuracy of the diagnostic test can differ for stages in a hierarchical testing algorithm or between row/column testing and individual testing in an array testing algorithm.

Estimation

The binGroup2 package also provides functions for estimation and inference for proportions in group testing designs.

The propCI function calculates the point estimate and confidence intervals for a single proportion from group testing data. The propDiffCI function does the same for the difference of proportions. A number of confidence interval methods are available for groups of equal or different sizes.

The gtWidth function calculates the expected width of confidence intervals in group testing.
The gtTest function calculates p-values for hypothesis tests of single proportions. ThegtPower function calculates power to reject a hypothesis.

The designPower function iterates either the number of groups or group size in a one-parameter group testing design until a pre-specified power level is achieved. The designEstfunction finds the optimal group size corresponding to the minimal mean-squared error of the point estimator.

The gtReg function implements regression methods and the gtSim function simulates group testing data for simple pooling, halving, and array testing designs.

Author(s)

Maintainer: Brianna Hitt brianna.hitt@huskers.unl.edu (ORCID)

Authors:

- Christopher Bilder (ORCID)
- Frank Schaarschmidt (ORCID)
- Brad Biggerstaff (ORCID)
- Christopher McMahan (ORCID)
- Joshua Tebbs (ORCID)

Other contributors:

- Boan Zhang [contributor]
- Michael Black [contributor]
- Peijie Hou [contributor]
- Peng Chen [contributor]

References

Altman, D., Bland, J. (1994). “Diagnostic tests 1: Sensitivity and specificity.” *BMJ*, **308**, 1552.

Altman, D., Bland, J. (1994). “Diagnostic tests 2: Predictive values.” *BMJ*, **309**, 102.

Biggerstaff, B. (2008). “Confidence intervals for the difference of proportions estimated from pooled samples.” *Journal of Agricultural, Biological, and Environmental Statistics*, **13**, 478–496. doi: 10.1198/108571108X379055.

Bilder, C., Tebbs, J., Chen, P. (2010). “Informative retesting.” *Journal of the American Statistical Association*, **105**, 942–955. doi: 10.1198/jasa.2010.ap09231.

Bilder, C., Tebbs, J., McMahan, C. (2019). “Informative group testing for multiplex assays.”*Biometrics*, **75**, 278–288. doi: 10.1111/biom.12988.

Bilder, C. (2019a). “Group Testing for Estimation.” *Wiley StatsRef: Statistics Reference Online*. doi: 10.1002/9781118445112.stat08231.

Bilder, C. (2019b). “Group Testing for Identification.” *Wiley StatsRef: Statistics Reference Online*. doi: 10.1002/9781118445112.stat08227.

Black, M., Bilder, C., Tebbs, J. (2012). “Group testing in heterogeneous populations by using halving algorithms.” *Journal of the Royal Statistical Society. Series C: Applied Statistics*, **61**, 277–290. doi: 10.1111/j.1467-9876.2011.01008.x.

Black, M., Bilder, C., Tebbs, J. (2015). “Optimal retesting configurations for hierarchical group testing.” *Journal of the Royal Statistical Society. Series C: Applied Statistics*, **64**, 693–710. doi: 10.1111/rssc.12097.

Graff, L., Roeloffs, R. (1972). “Group testing in the presence of test error; an extension of the Dorfman procedure.” *Technometrics*, **14**, 113–122. doi: 10.1080/00401706.1972.10488888.

Hepworth, G. (1996). “Exact confidence intervals for proportions estimated by group testing.”*Biometrics*, **52**, 1134–1146.

Hepworth, G., Biggerstaff, B. (2017). “Bias correction in estimating proportions by pooled testing.” *Journal of Agricultural, Biological, and Environmental Statistics*, **22**, 602–614. doi: 10.1007/s13253-017-0297-2.

Hitt, B., Bilder, C., Tebbs, J., McMahan, C. (2019). “The objective function controversy for group testing: Much ado about nothing?” *Statistics in Medicine*, **38**, 4912–4923. doi: 10.1002/sim.8341.

Hou, P., Tebbs, J., Wang, D., McMahan, C., Bilder, C. (2020). “Array testing with multiplex assays.” To appear in *Biostatistics*.

Malinovsky, Y., Albert, P., Roy, A. (2016). “Reader reaction: A note on the evaluation of group testing algorithms in the presence of misclassification.” *Biometrics*, **72**, 299–302. doi: 10.1111/biom.12385.

McMahan, C., Tebbs, J., Bilder, C. (2012a). “Informative Dorfman Screening.” *Biometrics*, **68**, 287–296. doi: 10.1111/j.1541-0420.2011.01644.x.

McMahan, C., Tebbs, J., Bilder, C. (2012b). “Two-Dimensional Informative Array Testing.”*Biometrics*, **68**, 793–804. doi: 10.1111/j.1541-0420.2011.01726.x.

Schaarschmidt, F. (2007). “Experimental design for one-sided confidence intervals or hypothesis tests in binomial group testing.” *Communications in Biometry and Crop Science*, **2**, 32–40. ISSN 1896-0782.

Swallow, W. (1985). “Group testing for estimating infection rates and probabilities of disease transmission.” *Phytopathology*, **75**, 882–889. doi: 10.1094/Phyto-75-882.

Tebbs, J., Bilder, C. (2004). “Confidence interval procedures for the probability of disease transmission in multiple-vector-transfer designs.” *Journal of Agricultural, Biological, and Environmental Statistics*, **9**, 75–90. doi: 10.1198/1085711043127.

Vansteelandt, S., Goetghebeur, E., Verstraeten, T. (2000). “Regression models for disease prevalence with diagnostic tests on pools of serum samples.” *Biometrics*, **56**, 1126–1133. doi: 10.1111/j.0006-341x.2000.01126.x.

Xie, M. (2001). “Regression analysis of group testing samples.” *Statistics in Medicine*, **20**, 1957–1969. doi: 10.1002/sim.817.

Examples

```
# Estimated running time for all examples was calculated
#   using a computer with 16 GB of RAM and one core of
#   an Intel i7-6500U processor. Please take this into
#   account when interpreting the run times given.

# 1) Identification using hierarchical and array-based group testing
#   algorithms with an assay that tests for one disease.

# 1.1) Find the optimal testing configuration over a range of initial
#   group sizes, using informative three-stage hierarchical testing, where
#   p denotes the overall prevalence of disease;
#   Se denotes the sensitivity of the diagnostic test;
#   Sp denotes the specificity of the diagnostic test;
#   group.sz denotes the range of initial pool sizes for consideration; and
#   obj.fn specifies the objective functions for which to find results.

# This example takes approximately 25 seconds to run.

set.seed(1002)
results1 <- OTC1(algorithm="ID3", p=0.01, Se=0.95, Sp=0.95,
                 group.sz=3:30, obj.fn=c("ET", "MAR"), alpha=2)
summary(results1)

# 1.2) Find the optimal testing configuration using non-informative
# array testing without master pooling.
# The sensitivity and specificity differ for row/column testing and
#   individual testing.

# This example takes approximately 15 seconds to run.

results2 <- OTC1(algorithm="A2", p=0.05, Se=c(0.95, 0.99),
```

```
                  Sp=c(0.95, 0.98), group.sz=3:20, obj.fn=c("ET", "MAR"))
summary(results2)

# 1.3) Calculate the operating characteristics using informative
#   two-stage hierarchical (Dorfman) testing, implemented via the
#   pool-specific optimal Dorfman (PSOD) method described in
#   McMahan et al. (2012a).
# Hierarchical testing configurations are specified by a matrix
#   in the hier.config argument. The rows of the matrix correspond
#   to the stages of the hierarchical testing algorithm, the columns
#   correspond to the individuals to be tested, and the cell values
#   correspond to the group number of each individual at each stage.
config.mat <- matrix(data=c(rep(1, 5), rep(2, 4), 3, 1:10),
                     nrow=2, ncol=10, byrow=TRUE)
set.seed(8791)
results3 <- opChar1(algorithm="ID2", p=0.02, Se=0.95, Sp=0.99,
                    hier.config=config.mat, alpha=0.5)
summary(results3)

# 1.4) Calculate the operating characteristics using non-informative
#   four-stage hierarchical testing.
config.mat <- matrix(data=c(rep(1, 15), rep(c(1, 2, 3), each=5),
                            rep(1, 3), rep(2, 2), rep(3, 3), rep(4, 2),
                            rep(5, 4), 6, 1:15),
                     nrow=4, ncol=15, byrow=TRUE)
results4 <- opChar1(algorithm="D4", p=0.008, Se=0.96, Sp=0.98,
                    hier.config=config.mat, a=c(1, 4, 6, 9, 11, 15))
summary(results4)


# 2) Identification using hierarchical and array-based group testing
#   algorithms with a multiplex assay that tests for two diseases.

# 2.1) Find the optimal testing configuration using non-informative
#   two-stage hierarchical testing, given
#   p.vec, a vector of overall joint probabilities of disease;
#   Se, a vector of sensitivity values for each disease; and
#   Sp, a vector of specificity values for each disease.
# Se and Sp can also be specified as a matrix, where one value
#   is specified for each disease at each stage of testing.
results5 <- OTC2(algorithm="D2", p.vec=c(0.90, 0.04, 0.04, 0.02),
                 Se=c(0.99, 0.99), Sp=c(0.99, 0.99), group.sz=3:50)
summary(results5)

# 2.2) Calculate the operating characteristics for informative
#   five-stage hierarchical testing, given
#   alpha.vec, a vector of shape parameters for the Dirichlet distribution;
#   Se, a matrix of sensitivity values; and
#   Sp, a matrix of specificity values.
Se <- matrix(data=rep(0.95, 10), nrow=2, ncol=5, byrow=TRUE)
Sp <- matrix(data=rep(0.99, 10), nrow=2, ncol=5, byrow=TRUE)
config.mat <- matrix(data=c(rep(1, 24), rep(1, 18), rep(2, 6),
                            rep(1, 9), rep(2, 9), rep(3, 4), 4, 5,
                            rep(1, 6), rep(2, 3), rep(3, 5), rep(4, 4),
                            rep(5, 3), 6, rep(NA, 2), 1:21, rep(NA, 3)),
                     nrow=5, ncol=24, byrow=TRUE)
results6 <- opChar2(algorithm="ID5", alpha=c(18.25, 0.75, 0.75, 0.25),
```

```
                    Se=Se, Sp=Sp, hier.config=config.mat)
summary(results6)

# 3) Estimation of the overall disease prevalence and calculation
#   of confidence intervals.

# 3.1) Suppose 3 groups out of 24 test positively.
#   Each group has a size of 7.
propCI(x=3, m=7, n=24, ci.method="CP")
propCI(x=3, m=7, n=24, ci.method="Blaker")
propCI(x=3, m=7, n=24, ci.method="score")
propCI(x=3, m=7, n=24, ci.method="soc")

# 3.2) Consider the following situation:
#   0 out of 5 groups test positively with groups
#   of size 1 (individual testing),
#   0 out of 5 groups test positively with groups of size 5,
#   1 out of 5 groups test positively with groups of size 10,
#   2 out of 5 groups test positively with groups of size 50
propCI(x=c(0,0,1,2), m=c(1,5,10,50), n=c(5,5,5,5),
       pt.method="Gart", ci.method="skew-score")

# 4) Estimate a group testing regression model.

# 4.1) Fit a group testing regression model with
#   simple pooling using the "hivsurv" dataset.
data(hivsurv)
fit1 <- gtReg(type="sp", formula = groupres ~ AGE + EDUC.,
              data = hivsurv, groupn = gnum, sens = 0.9,
              spec = 0.9, method = "Xie")
summary(fit1)

# 4.2) Simulate data for the halving protocol, and
#   fit a group testing regression model.
set.seed(46)
gt.data <- gtSim(type="halving", par=c(-6, 0.1),
                 gshape=17, gscale=1.4, size1=1000,
                 size2=5, sens=0.95, spec=0.95)
fit2 <- gtReg(type="halving", formula=gres~x,
              data=gt.data, groupn=groupn, subg=subgroup,
              retest=retest, sens=0.95, spec=0.95,
              start=c(-6, 0.1), trace=TRUE)
summary(fit2)

# This example takes approximately 20 seconds to run.
# 4.3) Simulate data in 5x6 array testing form, and
#   fit a group testing regression model.
set.seed(9128)
array.sim <- gtSim(type="array", par=c(-7, 0.1),
                   size1=c(5,6), size2=c(4,5), sens=0.95, spec=0.95)
set1 <- array.sim$dframe

fit3 <- gtReg(type="array",
              formula=cbind(col.resp, row.resp)~x,
              data=set1, coln=coln, rown=rown,
              arrayn=arrayn, sens=0.95, spec=0.95,
              tol=0.005, n.gibbs=2000, trace=TRUE)
```

```
summary(fit3)
```

Identification and Estimation using Group Testing

Documentation for package ‘binGroup2’ version 1.0.2

- DESCRIPTION file.

Help Pages

binGroup2-package	binGroup2: Identification and Estimation using Group Testing
binGroup2	binGroup2: Identification and Estimation using Group Testing
designEst	Optimal group size determination based on minimal MSE when estimating an overall prevalence
designPower	Number of groups or group size needed to achieve a power level in one parameter group testing
expectOrderBeta	Determine a vector of probabilities for informative group testing algorithms
gtPower	Power to reject a hypothesis for one proportion in group testing
gtReg	Fitting group testing regression models
gtRegControl	Auxiliary for controlling group testing regression
gtSim	Simulation function for group testing data
gtTest	Hypothesis test for one proportion in group testing
gtWidth	Expected width of confidence intervals in group testing
halving	Probability mass function for halving
hivsurv	Data from an HIV surveillance project
informativeArrayProb	Arrange a matrix of probabilities for informative array testing
opChar1	Calculate operating characteristics for group testing algorithms that use a single-disease assay
opChar2	Calculate operating characteristics for group testing algorithms that use a multiplex assay for two diseases
operatingCharacteristics1	Calculate operating characteristics for group testing algorithms that use a single-disease assay
operatingCharacteristics2	Calculate operating characteristics for group testing algorithms that use a multiplex assay for two diseases
OTC1	Find the optimal testing configuration for group testing algorithms that use a single-disease assay

OTC2	Find the optimal testing configuration for group testing algorithms that use a multiplex assay for two diseases
predict.gtReg	Predict method for group testing regression model fits
print.designPower	Print method for objects of class "designPower"
print.gtTest	Print method for objects of class "gtTest"
print.propCI	Print method for objects of class "propCI"
print.propDiffCI	Print method for objects of class "propDiffCI"
print.summary.gtReg	Print method for 'summary.gtReg'
propCI	Confidence intervals for one proportion in group testing
propDiffCI	Confidence intervals for the difference of proportions in group testing
Sterrett	Summary measures for Sterrett algorithms
summary.gtReg	Summary method for group testing regression model fits
summary.opChar	Summary method for operating characteristics results
summary.OTC	Summary method for optimal testing configuration results

Calculate operating characteristics for group testing algorithms that use a single-disease assay

Description

Calculate operating characteristics, such as the expected number of tests, for a specified testing configuration using non-informative and informative hierarchical and array-based group testing algorithms. Single-disease assays are used at each stage of the algorithms.

Usage

```
operatingCharacteristics1(
  algorithm,
  p = NULL,
  probabilities = NULL,
  Se = 0.99,
  Sp = 0.99,
  hier.config = NULL,
  rowcol.sz = NULL,
  alpha = 2,
  a = NULL,
  print.time = TRUE,
  ...
)

opChar1(
  algorithm,
  p = NULL,
  probabilities = NULL,
  Se = 0.99,
  Sp = 0.99,
  hier.config = NULL,
  rowcol.sz = NULL,
  alpha = 2,
  a = NULL,
  print.time = TRUE,
  ...
)
```

Arguments

`algorithm`	character string defining the group testing algorithm to be used. Non-informative testing options include two-stage hierarchical ("`D2`"), three-stage hierarchical ("`D3`"), four-stage hierarchical ("`D4`"), square array testing without master pooling ("`A2`"), and square array testing with master pooling ("`A2M`"). Informative testing options include

	two-stage hierarchical ("ID2"), three-stage hierarchical ("ID3"), four-stage hierarchical ("ID4"), and square array testing without master pooling ("IA2").
p	overall probability of disease that will be used to generate a vector/matrix of individual probabilities. For non-informative algorithms, a homogeneous set of probabilities will be used. For informative algorithms, the expectOrderBeta function will be used to generate a heterogeneous set of probabilities. Further details are given under 'Details'. Either p or probabilities should be specified, but not both.
probabilities	a vector of individual probabilities, which is homogeneous for non-informative testing algorithms and heterogeneous for informative testing algorithms. Either p or probabilities should be specified, but not both.
Se	a vector of sensitivity values, where one value is given for each stage of testing (in order). If a single value is provided, sensitivity values are assumed to be equal to this value for all stages of testing. Further details are given under 'Details'.
Sp	a vector of specificity values, where one value is given for each stage of testing (in order). If a single value is provided, specificity values are assumed to be equal to this value for all stages of testing. Further details are given under 'Details'.
hier.config	a matrix specifying the configuration for a hierarchical testing algorithm. The rows correspond to the stages of testing, the columns correspond to each individual to be tested, and the cell values specify the group number of each individual at each stage. Further details are given under 'Details'. For array testing algorithms, this argument will be ignored.
rowcol.sz	the row/column size for array testing algorithms. For hierarchical testing algorithms, this argument will be ignored.
alpha	a shape parameter for the beta distribution that specifies the degree of heterogeneity for the generated probability vector (for informative testing only).
a	a vector containing indices indicating which individuals to calculate individual accuracy measures for. If NULL, individual accuracy measures will be displayed for all individuals in the algorithm.
print.time	a logical value indicating whether the length of time for calculations should be printed. The default is TRUE.
...	arguments to be passed to the expectOrderBeta function, which generates a vector of probabilities for informative testing algorithms. Further details are given under 'Details'.

Details

This function computes the operating characteristics for group testing algorithms with an assay that tests for one disease, as described in Hitt et al. (2019).

Available algorithms include two-, three-, and four-stage hierarchical testing and array testing with and without master pooling. Both non-informative and informative group testing settings are allowed for each algorithm, except informative array testing with master pooling is unavailable because this method has not appeared in the group testing literature. Operating characteristics calculated are expected number of tests, pooling sensitivity, pooling specificity, pooling positive predictive value, and pooling negative predictive value for each individual.

For informative algorithms where the `p` argument is specified, the expected value of order statistics from a beta distribution are found. These values are used to represent disease risk probabilities for each individual to be tested. The beta distribution has two parameters: a mean parameter `p` (overall disease prevalence) and a shape parameter `alpha`(heterogeneity level). Depending on the specified `p`, `alpha`, and overall group size, simulation may be necessary to generate the vector of individual probabilities. This is done using `expectOrderBeta` and requires the user to set a seed to reproduce results.

Informative two-stage hierarchical (Dorfman) testing is implemented via the pool-specific optimal Dorfman (PSOD) method described in McMahan et al. (2012a), where the greedy algorithm proposed for PSOD is replaced by considering all possible testing configurations. Informative array testing is implemented via the gradient method (the most efficient array design), where higher-risk individuals are grouped in the left-most columns of the array. For additional details on the gradient arrangement method for informative array testing, see McMahan et al. (2012b).

The sensitivity/specificity values are allowed to vary across stages of testing. For hierarchical testing, a different sensitivity/specificity value may be used for each stage of testing. For array testing, a different sensitivity/specificity value may be used for master pool testing (if included), row/column testing, and individual testing. The values must be specified in order of the testing performed. For example, values are specified as (stage 1, stage 2, stage 3) for three-stage hierarchical testing or (master pool testing, row/column testing, individual testing) for array testing with master pooling. A single sensitivity/specificity value may be specified instead. In this situation, sensitivity/specificity values for all stages are assumed to be equal.

The matrix specified by `hier.config` defines the hierarchical group testing algorithm for *I*individuals. The rows of the matrix correspond to the stages *s=1,...,S* in the testing algorithm, and the columns correspond to individuals *i=1,...I*. The cell values within the matrix represent the group number of individual *i* at stage *s*. For three-stage, four-stage, and non-informative two-stage hierarchical testing, the first row of the matrix consists of all ones. This indicates that all individuals in the algorithm are tested together in a single group in the first stage of testing. For informative two-stage hierarchical testing, the initial group (block) is not tested. Thus, the first row of the matrix consists of the group numbers for each individual in the first stage of testing. For all hierarchical algorithms, the final row of the matrix denotes individual testing. Individuals who are not tested in a particular stage are represented by "NA" (e.g., an individual tested in a group of size 1 in the second stage of testing would not be tested again in a third stage of testing). It is important to note that this matrix represents the testing that could be performed if each group tests positively at each stage prior to the last. For more details on this matrix (called a group membership matrix), see Bilder et al. (2019).

For array testing without master pooling, the `rowcol.sz` specified represents the row/column size for initial (stage 1) testing. For array testing with master pooling, the `rowcol.sz` specified represents the row/column size for stage 2 testing. This is because the master pool size is the overall array size, given by the square of the row/column size.

The displayed overall pooling sensitivity, pooling specificity, pooling positive predictive value, and pooling negative predictive value are weighted averages of the corresponding individual accuracy measures for all individuals within the initial group (or block) for a hierarchical algorithm, or within the entire array for an array-based algorithm. Expressions for these averages are provided in the Supplementary Material for Hitt et al. (2019). These expressions are based on accuracy definitions given by Altman and Bland (1994a, 1994b).

The `operatingCharacteristics1` function accepts additional arguments, namely `num.sim`, to be passed to the `expectOrderBeta` function, which generates a vector of probabilities for informative group testing algorithms. The `num.sim` argument specifies the number of simulations from the beta distribution when simulation is used. By default, 10,000 simulations are used.

Value

A list containing:

`algorithm` the group testing algorithm used for calculations.

`prob` the probability of disease or the vector of individual probabilities, as specified by the user.

`alpha` level of heterogeneity for the generated probability vector (for informative testing only).

`Se` the vector of sensitivity values for each stage of testing.

`Sp` the vector of specificity values for each stage of testing.

`Config` a list specifying elements of the specified testing configuration, which may include:

- Stage1
  group size for the first stage of hierarchical testing, if applicable.
- Stage2
  group sizes for the second stage of hierarchical testing, if applicable.
- Stage3
  group sizes for the third stage of hierarchical testing, if applicable.
- Block.sz
  the block size/initial group size for informative Dorfman testing, which is not tested.
- pool.szs
  group sizes for the first stage of testing for informative Dorfman testing.
- Array.dim
  the row/column size for array testing.
- Array.sz
  the overall array size for array testing (the square of the row/column size).

`p.vec` the sorted vector of individual probabilities, if applicable.

`p.mat` the sorted matrix of individual probabilities in gradient arrangement, if applicable. Further details are given under 'Details'.

`ET` the expected testing expenditure to decode all individuals in the algorithm; this includes all individuals in all groups for hierarchical algorithms or in the entire array for array testing.

`value` the value of the expected number of tests per individual.

`Accuracy` a list containing:

- Individual
  a matrix of accuracy measures for each individual specified in `a`. The rows correspond to each unique set of accuracy measures in the algorithm. Individuals with the same set of accuracy measures are displayed together in a single row of the matrix. The columns correspond to the pooling sensitivity, pooling specificity, pooling positive predictive value, pooling negative predictive value, and the indices for the individuals in each row of the matrix.
- Overall
  a matrix of overall accuracy measures for the algorithm. The columns correspond to the pooling sensitivity, pooling specificity, pooling positive predictive value, and pooling negative predictive value for the overall algorithm. Further details are given under 'Details'.

Note

This function returns the pooling positive and negative predictive values for all individuals even though these measures are diagnostic specific; e.g., the pooling positive predictive value should only be considered for those individuals who have tested positive.

Additionally, only stage dependent sensitivity and specificity values are allowed within the program (no group within stage dependent values are allowed). See Bilder et al. (2019) for additional information.

Author(s)

Brianna D. Hitt

References

Altman, D., Bland, J. (1994). "Diagnostic tests 1: Sensitivity and specificity." *BMJ*, **308**, 1552.

Altman, D., Bland, J. (1994). "Diagnostic tests 2: Predictive values." *BMJ*, **309**, 102.

Bilder, C., Tebbs, J., McMahan, C. (2019). "Informative group testing for multiplex assays."*Biometrics*, **75**, 278–288. doi: 10.1111/biom.12988.

Hitt, B., Bilder, C., Tebbs, J., McMahan, C. (2019). "The objective function controversy for group testing: Much ado about nothing?" *Statistics in Medicine*, **38**, 4912–4923. doi: 10.1002/sim.8341.

McMahan, C., Tebbs, J., Bilder, C. (2012a). "Informative Dorfman Screening." *Biometrics*, **68**, 287–296. doi: 10.1111/j.1541-0420.2011.01644.x.

McMahan, C., Tebbs, J., Bilder, C. (2012b). "Two-Dimensional Informative Array Testing."*Biometrics*, **68**, 793–804. doi: 10.1111/j.1541-0420.2011.01726.x.

See Also

Other operating characteristic functions: `Sterrett()`, `halving()`, `operatingCharacteristics2()`

Examples

```
# Calculate the operating characteristics for non-informative
#   two-stage hierarchical (Dorfman) testing.
config.mat <- matrix(data = c(rep(1, 10), 1:10),
                     nrow = 2, ncol = 10, byrow = TRUE)
opChar1(algorithm="D2", p=0.05, Se=0.99, Sp=0.99,
        hier.config=config.mat)
opChar1(algorithm="D2", p=0.05, Se=0.99, Sp=0.99,
        hier.config=config.mat, a=c(1,4), print.time=FALSE)

# Calculate the operating characteristics for informative
#   two-stage hierarchical (Dorfman) testing.
# A vector of individual probabilities is generated using
#   the expected value of order statistics from a beta
#   distribution with p = 0.01 and a heterogeneity level
```

```
#   of alpha = 0.5.
config.mat <- matrix(data = c(rep(1:3, each = 10), 1:30),
                     nrow = 2, ncol = 30, byrow = TRUE)
set.seed(52613)
opChar1(algorithm="ID2", p=0.01, Se=0.95, Sp=0.95,
        hier.config=config.mat, alpha=0.5, num.sim=10000)
# Equivalent code using a heterogeneous vector of
#   probabilities
set.seed(52613)
probs <- expectOrderBeta(p=0.01, alpha=0.5, grp.sz=30)
opChar1(algorithm="ID2", probabilities=probs, Se=0.95, Sp=0.95,
        hier.config=config.mat)

# Calculate the operating characteristics for
#   non-informative three-stage hierarchical testing.
config.mat <- matrix(data = c(rep(1, 18), rep(1:3, each = 5),
                              rep(4, 3), 1:18),
                     nrow = 3, ncol = 18, byrow = TRUE)
opChar1(algorithm="D3", p=0.001, Se=0.95, Sp=0.95,
        hier.config=config.mat)
opChar1(algorithm="D3", p=0.001, Se=c(0.95, 0.95, 0.99),
        Sp=c(0.96, 0.96, 0.98), hier.config=config.mat)

# Calculate the operating characteristics for
#   informative three-stage hierarchical testing,
#   given a heterogeneous vector of probabilities.
config.mat <- matrix(data = c(rep(1, 6), rep(1:2, each = 3),
                              1:6), nrow = 3, ncol = 6,
                     byrow = TRUE)
set.seed(52613)
opChar1(algorithm="ID3",
         probabilities=c(0.012, 0.014, 0.011, 0.012, 0.010, 0.015),
         Se=0.99, Sp=0.99, hier.config=config.mat,
         alpha=0.5, num.sim=5000)

# Calculate the operating characteristics for
#   non-informative four-stage hierarchical testing.
config.mat <- matrix(data = c(rep(1, 12), rep(1, 8),
                              rep(2, 2), 3, 4, rep(1, 5),
                              rep(2, 3), 3, 4, rep(NA, 2),
                              1:8, rep(NA, 4)), nrow = 4,
                     ncol = 12, byrow = TRUE)
opChar1(algorithm="D4", p=0.041, Se=0.99, Sp=0.90,
        hier.config=config.mat)

# Calculate the operating characteristics for
#   informative four-stage hierarchical testing.
# A vector of individual probabilities is generated using
#   the expected value of order statistics from a beta
#   distribution with p = 0.041 and a heterogeneity level
#   of alpha = 0.5.
config.mat <- matrix(data = c(rep(1, 12), rep(1, 8),
                              rep(2, 2), 3, 4, rep(1, 5),
                              rep(2, 3), 3, 4, rep(NA, 2),
                              1:8, rep(NA, 4)), nrow = 4,
                     ncol = 12, byrow = TRUE)
set.seed(5678)
```

```
opChar1(algorithm="ID4", p=0.041, Se=0.99, Sp=0.90,
        hier.config=config.mat, alpha=0.5)

# Calculate the operating characteristics for
#   non-informative array testing without master pooling.
opChar1(algorithm="A2", p=0.005, Se=c(0.95, 0.99),
        Sp=c(0.95, 0.99), rowcol.sz=8, a=1)

# Calculate the operating characteristics for
#   informative array testing without master pooling.
# A vector of individual probabilities is generated using
#   the expected value of order statistics from a beta
#   distribution with p = 0.03 and a heterogeneity level
#   of alpha = 2.
set.seed(1002)
opChar1(algorithm="IA2", p=0.03, Se=0.95, Sp=0.95,
         rowcol.sz=8, alpha=2, a=1:10)

# Calculate the operating characteristics for
#   non-informative array testing with master pooling.
opChar1(algorithm="A2M", p=0.02, Se=c(0.95,0.95,0.99),
        Sp=c(0.98,0.98,0.99), rowcol.sz=5)
```

Calculate operating characteristics for group testing algorithms that use a multiplex assay for two diseases

Description

Calculate operating characteristics, such as the expected number of tests, for a specified testing configuration using non-informative and informative hierarchical and array-based group testing algorithms. Multiplex assays for two diseases are used at each stage of the algorithms.

Usage

```
operatingCharacteristics2(
  algorithm,
  p.vec = NULL,
  probabilities = NULL,
  alpha = NULL,
  Se,
  Sp,
  hier.config = NULL,
  rowcol.sz = NULL,
  ordering = matrix(data = c(0, 1, 0, 1, 0, 0, 1, 1), nrow = 4, ncol = 2),
  a = NULL,
  print.time = TRUE,
  ...
)

opChar2(
  algorithm,
  p.vec = NULL,
  probabilities = NULL,
  alpha = NULL,
  Se,
  Sp,
  hier.config = NULL,
  rowcol.sz = NULL,
  ordering = matrix(data = c(0, 1, 0, 1, 0, 0, 1, 1), nrow = 4, ncol = 2),
  a = NULL,
  print.time = TRUE,
  ...
)
```

Arguments

`algorithm`	character string defining the group testing algorithm to be used. Non-informative testing options include two-stage hierarchical ("`D2`"), three-stage hierarchical ("`D3`"), four-stage hierarchical ("`D4`"), five-stage hierarchical ("`D5`"), square array testing

without master pooling ("A2"), and square array testing with master pooling ("A2M"). Informative testing options include two-stage hierarchical ("ID2"), three-stage hierarchical ("ID3"), four-stage hierarchical ("ID4"), and five-stage hierarchical ("ID5") testing.

`p.vec` vector of overall joint probabilities. The joint probabilities are assumed to be equal for all individuals in the algorithm (non-informative testing only). There are four joint probabilities to consider: *p_00*, the probability that an individual tests negative for both diseases; *p_10*, the probability that an individual tests positive only for the first disease; *p_01*, the probability that an individual tests positive only for the second disease; and *p_11*, the probability that an individual tests positive for both diseases. The joint probabilities must sum to 1. Only one of `p.vec`, `probabilities`, or `alpha` should be specified.

`probabilities` matrix of joint probabilities for each individual, where rows correspond to the four joint probabilities and columns correspond to each individual in the algorithm. Only one of `p.vec`, `probabilities`, or `alpha` should be specified.

`alpha` a vector containing positive shape parameters of the Dirichlet distribution (for informative testing only). The vector will be used to generate a heterogeneous matrix of joint probabilities for each individual. The vector must have length 4. Further details are given under 'Details'. Only one of `p.vec`, `probabilities`, or `alpha` should be specified.

`Se` matrix of sensitivity values, where one value is given for each disease (or infection) at each stage of testing. The rows of the matrix correspond to each disease *k=1,...,K*, and the columns of the matrix correspond to each stage of testing *s=1,...,S*. If a vector of *K* values is provided, the sensitivity values associated with disease *k* are assumed to be equal to the *k*th value in the vector for all stages of testing. Further details are given under 'Details'.

`Sp` a matrix of specificity values, where one value is given for each disease (or infection) at each stage of testing. The rows of the matrix correspond to each disease *k=1,...,K*, and the columns of the matrix correspond to each stage of testing *s=1,...,S*. If a vector of *K* values is provided, the specificity values associated with disease *k* are assumed to be equal to the *k*th value in the vector for all stages of testing. Further details are given under 'Details'.

`hier.config` a matrix specifying the configuration for a hierarchical testing algorithm. The rows correspond to the stages of testing, the columns correspond to each individual to be tested, and the cell values specify the group number of each individual at each stage. Further details are given under 'Details'. For array testing algorithms, this argument will be ignored.

`rowcol.sz` the row/column size for array testing algorithms. For hierarchical testing algorithms, this argument will be ignored.

`ordering` a matrix detailing the ordering for the binary responses of the diseases. The columns of the matrix correspond to each disease and the rows of the matrix correspond to each of the 4 sets of binary responses for two diseases. This ordering is used with the joint probabilities. The default ordering is (p_00, p_10, p_01, p_11).

`a` a vector containing indices indicating which individuals to calculate individual accuracy measures for. If `NULL`, individual accuracy measures will be displayed for all individuals in the algorithm.

`print.time` a logical value indicating whether the length of time for calculations should be printed. The default is `TRUE`.

`...`	additional arguments to be passed to functions for hierarchical testing with multiplex assays for two diseases.

Details

This function computes the operating characteristics for standard group testing algorithms with a multiplex assay that tests for two diseases. Calculations for hierarchical group testing algorithms are performed as described in Bilder et al. (2019) and calculations for array-based group testing algorithms are performed as described in Hou et al. (2019).

Available algorithms include two-, three-, four-, and five-stage hierarchical testing and array testing with and without master pooling. Both non-informative and informative group testing settings are allowed for hierarchical algorithms. Only non-informative group testing settings are allowed for array testing algorithms. Operating characteristics calculated are expected number of tests, pooling sensitivity, pooling specificity, pooling positive predictive value, and pooling negative predictive value for each individual.

For informative algorithms where the `alpha` argument is specified, a heterogeneous matrix of joint probabilities for each individual is generated using the Dirichlet distribution. This is done using `rBeta2009::rdirichlet` and requires the user to set a seed to reproduce results. See Bilder et al. (2019) for additional details on the use of the Dirichlet distribution for this purpose.

The sensitivity/specificity values are allowed to vary across stages of testing. For hierarchical testing, a different sensitivity/specificity value may be used for each stage of testing. For array testing, a different sensitivity/specificity value may be used for master pool testing (if included), row/column testing, and individual testing. The values must be specified in the order of the testing performed. For example, values are specified as (stage 1, stage 2, stage 3) for three-stage hierarchical testing or (master pool testing, row/column testing, individual testing) for array testing with master pooling. A vector of *K* sensitivity/specificity values may be specified, and sensitivity/specificity values for all stages of testing are assumed to be equal. The first value in the vector will be used at each stage of testing for the first disease, and the second value in the vector will be used at each stage of testing for the second disease.

The matrix specified by `hier.config` defines the hierarchical group testing algorithm for *I*individuals. The rows of the matrix correspond to the stages *s=1,...,S* in the testing algorithm, and the columns correspond to individuals *i=1,...I*. The cell values within the matrix represent the group number of individual *i* at stage *s*. For three-stage, four-stage, five-stage, and non-informative two-stage hierarchical testing, the first row of the matrix consists of all ones. This indicates that all individuals in the algorithm are tested together in a single group in the first stage of testing. For informative two-stage hierarchical testing, the initial group (block) is not tested. Thus, the first row of the matrix consists of the group numbers for each individual in the first stage of testing. For all hierarchical algorithms, the final row of the matrix denotes individual testing. Individuals who are not tested in a particular stage are represented by "NA" (e.g., an individual tested in a group of size 1 in the second stage of testing would not be tested again in a third stage of testing). It is important to note that this matrix represents the testing that could be performed if each group tests positively at each stage prior to the last. For more details on this matrix (called a group membership matrix), see Bilder et al. (2019).

For array testing without master pooling, the `rowcol.sz` specified represents the row/column size for initial (stage 1) testing. For array testing with master pooling, the `rowcol.sz` specified represents the row/column size for stage 2 testing. This is because the master pool size is the overall array size, given by the square of the row/column size.

The displayed overall pooling sensitivity, pooling specificity, pooling positive predictive value, and pooling negative predictive value are weighted averages of the corresponding individual accuracy measures for

all individuals within the initial group (or block) for a hierarchical algorithm, or within the entire array for an array-based algorithm. Expressions for these averages are provided in the Supplementary Material for Hitt et al. (2019). These expressions are based on accuracy definitions given by Altman and Bland (1994a, 1994b).

Value

A list containing:

`algorithm` the group testing algorithm used for calculations.

`prob.vec` the vector of joint probabilities provided by the user, if applicable (for non-informative algorithms only).

`joint.p` the matrix of joint probabilities for each individual provided by the user, if applicable.

`alpha.vec` the alpha vector provided by the user, if applicable (for informative algorithms only).

`Se` the matrix of sensitivity values for each disease at each stage of testing.

`Sp` the matrix of specificity values for each disease at each stage of testing.

`Config` a list specifying elements of the specified testing configuration, which may include:

Stage1
: group size for the first stage of hierarchical testing, if applicable.

Stage2
: group sizes for the second stage of hierarchical testing, if applicable.

Stage3
: group sizes for the third stage of hierarchical testing, if applicable.

Stage4
: group sizes for the fourth stage of hierarchical testing, if applicable.

Block.sz
: the block size/initial group size for informative Dorfman testing, which is not tested.

pool.szs
: group sizes for the first stage of testing for informative Dorfman testing.

Array.dim
: the row/column size for array testing.

Array.sz
: the overall array size for array testing (the square of the row/column size).

`p.mat` the matrix of joint probabilities for each individual in the algorithm. Each row corresponds to one of the four joint probabilities. Each column corresponds to an individual in the testing algorithm.

`ET` the expected testing expenditure for the OTC.

`value` the value of the expected number of tests per individual.

`Accuracy` a list containing:

Disease 1 Individual
: a matrix of accuracy measures, pertaining to the first disease, for each individual specified in `a`. The rows correspond to each unique set of accuracy measures in the algorithm. Individuals with the same set of accuracy measures are displayed together in a single row of the matrix. The columns correspond to the pooling

sensitivity, pooling specificity, pooling positive predictive value, pooling negative predictive value, and the indices for the individuals in each row of the matrix. Individual accuracy measures are not displayed for array testing algorithms.

Disease 2 Individual
: a matrix of accuracy measures, pertaining to the second disease, for each individual specified in `a`. The rows correspond to each unique set of accuracy measures in the algorithm. Individuals with the same set of accuracy measures are displayed together in a single row of the matrix. The columns correspond to the pooling sensitivity, pooling specificity, pooling positive predictive value, pooling negative predictive value, and the indices for the individuals in each row of the matrix. Individual accuracy measures are not displayed for array testing algorithms.

Overall
: a matrix of overall accuracy measures for the algorithm. The rows correspond to each disease. The columns correspond to the pooling sensitivity, pooling specificity, pooling positive predictive value, and pooling negative predictive value for the overall algorithm. Further details are given under 'Details'.

Note

This function returns the pooling positive and negative predictive values for all individuals even though these measures are diagnostic specific; e.g., the pooling positive predictive value should only be considered for those individuals who have tested positive.

Additionally, only stage dependent sensitivity and specificity values are allowed within the program (no group within stage dependent values are allowed). See Bilder et al. (2019) for additional information.

Author(s)

This function was written by Brianna D. Hitt. It calls `ET.all.stages.new` and `PSePSpAllStages`, which were originally written by Christopher Bilder for Bilder et al. (2019), and `ARRAY`, which was originally written by Peijie Hou for Hou et al. (2020). The functions `ET.all.stages.new`, `PSePSpAllStages`, and `ARRAY` were obtained from http://chrisbilder.com/grouptesting. Minor modifications were made to the functions for inclusion in the binGroup2 package.

References

Altman, D., Bland, J. (1994). “Diagnostic tests 1: Sensitivity and specificity.” *BMJ*, **308**, 1552.

Altman, D., Bland, J. (1994). “Diagnostic tests 2: Predictive values.” *BMJ*, **309**, 102.

Bilder, C., Tebbs, J., McMahan, C. (2019). “Informative group testing for multiplex assays.”*Biometrics*, **75**, 278–288. doi: 10.1111/biom.12988.

Hitt, B., Bilder, C., Tebbs, J., McMahan, C. (2019). “The objective function controversy for group testing: Much ado about nothing?” *Statistics in Medicine*, **38**, 4912–4923. doi: 10.1002/sim.8341.

Hou, P., Tebbs, J., Wang, D., McMahan, C., Bilder, C. (2020). “Array testing with multiplex assays.” To appear in *Biostatistics*.

McMahan, C., Tebbs, J., Bilder, C. (2012a). “Informative Dorfman Screening.” *Biometrics*, **68**, 287–296. doi: 10.1111/j.1541-0420.2011.01644.x.

See Also

Other operating characteristic functions: `Sterrett()`, `halving()`, `operatingCharacteristics1()`

Other multiplex testing functions: `OTC2()`

Examples

```
# Calculate the operating characteristics for
#   non-informative two-stage hierarchical
#   (Dorfman) testing.
config.mat <- matrix(data = c(rep(1, 24), 1:24),
                     nrow = 2, ncol = 24, byrow = TRUE)
Se <- matrix(data=c(0.95, 0.95, 0.95, 0.95),
             nrow=2, ncol=2,
             dimnames=list(Infection=1:2, Stage=1:2))
Sp <- matrix(data=c(0.99, 0.99, 0.99, 0.99),
             nrow=2, ncol=2,
             dimnames=list(Infection=1:2, Stage=1:2))
opChar2(algorithm="D2", p.vec=c(0.90, 0.04, 0.04, 0.02),
         Se=Se, Sp=Sp, hier.config=config.mat)
opChar2(algorithm="D2", p.vec=c(0.90, 0.04, 0.04, 0.02),
         Se=Se, Sp=Sp, hier.config=config.mat, a=c(1, 13, 24),
         print.time = FALSE)

# Calculate the operating characteristics for informative
#   two-stage hierarchical (Dorfman) testing.
# A matrix of joint probabilities for each individual is
#   generated using the Dirichlet distribution.
config.mat <- matrix(data = c(rep(1, 5), rep(2, 4), 3, 1:9, NA),
                     nrow = 2, ncol = 10, byrow = TRUE)
Se <- matrix(data=c(0.95, 0.95, 0.99, 0.99),
             nrow=2, ncol=2,
             dimnames=list(Infection=1:2, Stage=1:2))
Sp <- matrix(data=c(0.96, 0.96, 0.98, 0.98),
             nrow=2, ncol=2,
             dimnames=list(Infection=1:2, Stage=1:2))
set.seed(8791)
opChar2(algorithm="ID2", alpha=c(18.25, 0.75, 0.75, 0.25),
         Se=Se, Sp=Sp, hier.config=config.mat)
# Equivalent code using a heterogeneous matrix of joint
#   probabilities for each individual
set.seed(8791)
p.unordered <- t(rBeta2009::rdirichlet(n = 10,
                            shape = c(18.25, 0.75, 0.75, 0.25)))
p.ordered <- p.unordered[, order(1 - p.unordered[1,])]
opChar2(algorithm="ID2", probabilities=p.ordered,
        Se=Se, Sp=Sp, hier.config=config.mat)

# Calculate the operating characteristics for
#   non-informative three-stage hierarchical testing.
config.mat <- matrix(data = c(rep(1, 10), rep(1, 5),
```

```
                              rep(2, 4), 3, 1:9, NA),
                      nrow = 3, ncol = 10, byrow = TRUE)
Se <- matrix(data=rep(0.95, 6), nrow=2, ncol=3,
             dimnames=list(Infection=1:2, Stage=1:3))
Sp <- matrix(data=rep(0.99, 6), nrow=2, ncol=3,
             dimnames=list(Infection=1:2, Stage=1:3))
opChar2(algorithm="D3", p.vec=c(0.95, 0.02, 0.02, 0.01),
         Se=Se, Sp=Sp, hier.config=config.mat)
opChar2(algorithm="D3", p.vec=c(0.95, 0.02, 0.02, 0.01),
         Se=Se, Sp=Sp, hier.config=config.mat, a=c(1, 6, 10))

# Calculate the operating characteristics for informative
#   three-stage hierarchical testing.
# A matrix of joint probabilities for each individual is
#   generated using the Dirichlet distribution.
config.mat <- matrix(data = c(rep(1, 15),
                              rep(c(1, 2, 3), each = 5), 1:15),
                      nrow = 3, ncol = 15, byrow = TRUE)
Se <- matrix(data=rep(0.95, 6), nrow=2, ncol=3,
             dimnames=list(Infection=1:2, Stage=1:3))
Sp <- matrix(data=rep(0.99, 6), nrow=2, ncol=3,
             dimnames=list(Infection=1:2, Stage=1:3))
opChar2(algorithm="ID3", alpha=c(18.25, 0.75, 0.75, 0.25),
         Se=Se, Sp=Sp, hier.config=config.mat)

# Calculate the operating characteristics for
#   non-informative four-stage hierarchical testing.
config.mat <- matrix(data = c(rep(1, 12), rep(1, 6), rep(2, 6),
                              rep(1, 4), rep(2, 2), rep(3, 3),
                              rep(4, 3), 1:12),
                      nrow = 4, ncol = 12, byrow = TRUE)
Se <- matrix(data=rep(0.95, 8), nrow=2, ncol=4,
             dimnames=list(Infection=1:2, Stage=1:4))
Sp <- matrix(data=rep(0.99, 8), nrow=2, ncol=4,
             dimnames=list(Infection=1:2, Stage=1:4))
opChar2(algorithm="D4", p.vec=c(0.92, 0.05, 0.02, 0.01),
         Se=Se, Sp=Sp, hier.config=config.mat)

# Calculate the operating characteristics for informative
#   five-stage hierarchical testing.
# A matrix of joint probabilities for each individual is
#   generated using the Dirichlet distribution.
config.mat <- matrix(data = c(rep(1, 20), rep(1, 10), rep(2, 10),
                              rep(c(1, 2, 3, 4), each = 5),
                              rep(1, 3), rep(2, 2), rep(3, 3),
                              rep(4, 2), rep(5, 3), rep(6, 2),
                              rep(7, 3), rep(8, 2), 1:20),
                      nrow = 5, ncol = 20, byrow = TRUE)
Se <- matrix(data=rep(0.95, 10), nrow=2, ncol=5,
             dimnames=list(Infection=1:2, Stage=1:5))
Sp <- matrix(data=rep(0.99, 10), nrow=2, ncol=5,
             dimnames=list(Infection=1:2, Stage=1:5))
opChar2(algorithm="ID5", alpha=c(18.25, 0.75, 0.75, 0.25),
        Se=Se, Sp=Sp, hier.config=config.mat)

# Calculate the operating characteristics for
#   non-informative array testing without master pooling.
```

```
Se <- matrix(data=rep(0.95, 4), nrow=2, ncol=2,
             dimnames=list(Infection=1:2, Stage=1:2))
Sp <- matrix(data=rep(0.99, 4), nrow=2, ncol=2,
             dimnames=list(Infection=1:2, Stage=1:2))
opChar2(algorithm="A2", p.vec=c(0.90, 0.04, 0.04, 0.02),
         Se=Se, Sp=Sp, rowcol.sz=12)

# Calculate the operating characteristics for
#   non-informative array testing with master pooling.
Se <- matrix(data=rep(0.95, 6), nrow=2, ncol=3,
             dimnames=list(Infection=1:2, Stage=1:3))
Sp <- matrix(data=rep(0.99, 6), nrow=2, ncol=3,
             dimnames=list(Infection=1:2, Stage=1:3))
opChar2(algorithm="A2M", p.vec=c(0.90, 0.04, 0.04, 0.02),
         Se=Se, Sp=Sp, rowcol.sz=10)
```

Find the optimal testing configuration for group testing algorithms that use a single-disease assay

Description

Find the optimal testing configuration (OTC) using non-informative and informative hierarchical and array-based group testing algorithms. Single-disease assays are used at each stage of the algorithms.

Usage

```
OTC1(
  algorithm,
  p = NULL,
  probabilities = NULL,
  Se = 0.99,
  Sp = 0.99,
  group.sz,
  obj.fn = c("ET", "MAR"),
  weights = NULL,
  alpha = 2,
  trace = TRUE,
  print.time = TRUE,
  ...
)
```

Arguments

`algorithm`	character string defining the group testing algorithm to be used. Non-informative testing options include two-stage hierarchical ("`D2`"), three-stage hierarchical ("`D3`"), square array testing without master pooling ("`A2`"), and square array testing with master pooling ("`A2M`"). Informative testing options include two-stage hierarchical ("`ID2`"), three-stage hierarchical ("`ID3`"), and square array testing without master pooling ("`IA2`").
`p`	overall probability of disease that will be used to generate a vector/matrix of individual probabilities. For non-informative algorithms, a homogeneous set of probabilities will be used. For informative algorithms, the `expectOrderBeta` function will be used to generate a heterogeneous set of probabilities. Further details are given under 'Details'. Either `p` or `probabilities` should be specified, but not both.
`probabilities`	a vector of individual probabilities, which is homogeneous for non-informative testing algorithms and heterogeneous for informative testing algorithms. Either `p` or `probabilities` should be specified, but not both.
`Se`	a vector of sensitivity values, where one value is given for each stage of testing (in order). If a single value is provided, sensitivity values are assumed to be equal to this

value for all stages of testing. Further details are given under 'Details'.

`Sp` a vector of specificity values, where one value is given for each stage of testing (in order). If a single value is provided, specificity values are assumed to be equal to this value for all stages of testing. Further details are given under 'Details'.

`group.sz` a single group size or range of group sizes for which to calculate operating characteristics and/or find the OTC. The details of group size specification are given under 'Details'.

`obj.fn` a list of objective functions which are minimized to find the OTC. The expected number of tests per individual, "`ET`", will always be calculated. Additional options include "`MAR`" (the expected number of tests divided by the expected number of correct classifications, described in Malinovsky et al. (2016)), and "`GR`" (a linear combination of the expected number of tests, the number of misclassified negatives, and the number of misclassified positives, described in Graff & Roeloffs (1972)). See Hitt et al. (2019) for additional details. The first objective function specified in this list will be used to determine the results for the top configurations. Further details are given under 'Details'.

`weights` a matrix of up to six sets of weights for the GR function. Each set of weights is specified by a row of the matrix.

`alpha` a shape parameter for the beta distribution that specifies the degree of heterogeneity for the generated probability vector (for informative testing only).

`trace` a logical value indicating whether the progress of calculations should be printed for each initial group size provided by the user. The default is `TRUE`.

`print.time` a logical value indicating whether the length of time for calculations should be printed. The default is `TRUE`.

`...` arguments to be passed to the `expectOrderBeta` function, which generates a vector of probabilities for informative testing algorithms. Further details are given under 'Details'.

Details

This function finds the OTC for group testing algorithms with an assay that tests for one disease and computes the associated operating characteristics, as described in Hitt et al. (2019).

Available algorithms include two- and three-stage hierarchical testing and array testing with and without master pooling. Both non-informative and informative group testing settings are allowed for each algorithm, except informative array testing with master pooling is unavailable because this method has not appeared in the group testing literature. Operating characteristics calculated are expected number of tests, pooling sensitivity, pooling specificity, pooling positive predictive value, and pooling negative predictive value for each individual.

For informative algorithms where the `p` argument is specified, the expected value of order statistics from a beta distribution are found. These values are used to represent disease risk probabilities for each individual to be tested. The beta distribution has two parameters: a mean parameter `p` (overall disease prevalence) and a shape parameter `alpha`(heterogeneity level). Depending on the specified `p`, `alpha`, and overall group size, simulation may be necessary to generate the vector of individual probabilities. This is done using `expectOrderBeta` and requires the user to set a seed to reproduce results.

Informative two-stage hierarchical (Dorfman) testing is implemented via the pool-specific optimal Dorfman (PSOD) method described in McMahan et al. (2012a), where the greedy algorithm proposed for PSOD is replaced by considering all possible testing configurations. Informative array testing is implemented via the gradient method (the most efficient array design), where higher-risk individuals are grouped in the left-most columns of the array. For additional details on the gradient arrangement method for informative array testing, see McMahan et al. (2012b).

The sensitivity/specificity values are allowed to vary across stages of testing. For hierarchical testing, a different sensitivity/specificity value may be used for each stage of testing. For array testing, a different sensitivity/specificity value may be used for master pool testing (if included), row/column testing, and individual testing. The values must be specified in order of the testing performed. For example, values are specified as (stage 1, stage 2, stage 3) for three-stage hierarchical testing or (master pool testing, row/column testing, individual testing) for array testing with master pooling. A single sensitivity/specificity value may be specified instead. In this situation, sensitivity/specificity values for all stages are assumed to be equal.

The value(s) specified by `group.sz` represent the initial (stage 1) group size for hierarchical testing and the row/column size for array testing. For informative two-stage hierarchical testing, the `group.sz` specified represents the block size used in the pool-specific optimal Dorfman (PSOD) method, where the initial group (block) is not tested. For more details on informative two-stage hierarchical testing implemented via the PSOD method, see Hitt et al. (2019) and McMahan et al. (2012a).

If a single value is provided for `group.sz` with array testing or non-informative two-stage hierarchical testing, operating characteristics will be calculated and no optimization will be performed. If a single value is provided for `group.sz` with three-stage hierarchical or informative two-stage hierarchical, the OTC will be found over all possible configurations. If a range of group sizes is specified, the OTC will be found over all group sizes.

In addition to the OTC, operating characteristics for some of the other configurations corresponding to each initial group size provided by the user will be displayed. These additional configurations are only determined for whichever objective function ("ET", "MAR", or "GR") is specified first in the function call. If "GR" is the objective function listed first, the first set of corresponding weights will be used. For algorithms where there is only one configuration for each initial group size (non-informative two-stage hierarchical and all array testing algorithms), results for each initial group size are provided. For algorithms where there is more than one possible configuration for each initial group size (informative two-stage hierarchical and all three-stage hierarchical algorithms), two sets of configurations are provided: 1) the best configuration for each initial group size, and 2) the top 10 configurations for each initial group size provided by the user. If a single value is provided for `group.sz`with array testing or non-informative two-stage hierarchical testing, operating characteristics will not be provided for configurations other than that specified by the user. Results are sorted by the value of the objective function per individual, `value`.

The displayed overall pooling sensitivity, pooling specificity, pooling positive predictive value, and pooling negative predictive value are weighted averages of the corresponding individual accuracy measures for all individuals within the initial group (or block) for a hierarchical algorithm, or within the entire array for an array-based algorithm. Expressions for these averages are provided in the Supplementary Material for Hitt et al. (2019). These expressions are based on accuracy definitions given by Altman and Bland (1994a, 1994b). Individual accuracy measures can be calculated using the `operatingCharacteristics1`(`opChar1`) function.

The `OTC1` function accepts additional arguments, namely `num.sim`, to be passed to the `expectOrderBeta` function, which generates a vector of probabilities for informative group testing algorithms. The `num.sim` argument specifies the number of simulations from the beta distribution when simulation is used. By default, 10,000 simulations are used.

Value

A list containing:

`algorithm` the group testing algorithm used for calculations.

`prob` the probability of disease or the vector of individual probabilities, as specified by the user.

`alpha` level of heterogeneity for the generated probability vector (for informative testing only).

`Se` the vector of sensitivity values for each stage of testing.

`Sp` the vector of specificity values for each stage of testing.

`opt.ET`, `opt.MAR`, `opt.GR` a list of results for each objective function specified by the user, containing:

OTC
: a list specifying elements of the optimal testing configuration, which may include:

  Stage1
  : group size for the first stage of hierarchical testing, if applicable.

  Stage2
  : group sizes for the second stage of hierarchical testing, if applicable.

  Block.sz
  : the block size/initial group size for informative Dorfman testing, which is not tested.

  pool.szs
  : group sizes for the first stage of testing for informative Dorfman testing.

  Array.dim
  : the row/column size for array testing.

  Array.sz
  : the overall array size for array testing (the square of the row/column size).

p.vec
: the sorted vector of individual probabilities, if applicable.

p.mat
: the sorted matrix of individual probabilities in gradient arrangement, if applicable. Further details are given under 'Details'.

ET
: the expected testing expenditure to decode all individuals in the algorithm; this includes all individuals in all groups for hierarchical algorithms or in the entire array for array testing.

value
: the value of the objective function per individual.

Accuracy
: a matrix of overall accuracy measures for the algorithm. The columns correspond to the pooling sensitivity, pooling specificity, pooling positive predictive value, and pooling negative predictive value for the overall algorithm. Further details are given under 'Details'.

`Configs` a data frame containing results for the best configuration for each initial group size provided by the user. The columns correspond to the initial group size, configuration (if applicable), overall array size (if applicable), expected number of tests, value of the

objective function per individual, pooling sensitivity, pooling specificity, pooling positive predictive value, and pooling negative predictive value. No results are displayed if a single `group.sz` is provided. Further details are given under 'Details'.

`Top.Configs` a data frame containing results for some of the top configurations for each initial group size provided by the user. The columns correspond to the initial group size, configuration, expected number of tests, value of the objective function per individual, pooling sensitivity, pooling specificity, pooling positive predictive value, and pooling negative predictive value. No results are displayed for non-informative two-stage hierarchical testing or for array testing algorithms. Further details are given under 'Details'.

Note

This function returns the pooling positive and negative predictive values for all individuals even though these measures are diagnostic specific; e.g., the pooling positive predictive value should only be considered for those individuals who have tested positive.

Additionally, only stage dependent sensitivity and specificity values are allowed within the program (no group within stage dependent values are allowed). See Bilder et al. (2019) for additional information.

Author(s)

Brianna D. Hitt

References

Altman, D., Bland, J. (1994). “Diagnostic tests 1: Sensitivity and specificity.” *BMJ*, **308**, 1552.

Altman, D., Bland, J. (1994). “Diagnostic tests 2: Predictive values.” *BMJ*, **309**, 102.

Bilder, C., Tebbs, J., McMahan, C. (2019). “Informative group testing for multiplex assays.”*Biometrics*, **75**, 278–288. doi: 10.1111/biom.12988.

Graff, L., Roeloffs, R. (1972). “Group testing in the presence of test error; an extension of the Dorfman procedure.” *Technometrics*, **14**, 113–122. doi: 10.1080/00401706.1972.10488888.

Hitt, B., Bilder, C., Tebbs, J., McMahan, C. (2019). “The objective function controversy for group testing: Much ado about nothing?” *Statistics in Medicine*, **38**, 4912–4923. doi: 10.1002/sim.8341.

Malinovsky, Y., Albert, P., Roy, A. (2016). “Reader reaction: A note on the evaluation of group testing algorithms in the presence of misclassification.” *Biometrics*, **72**, 299–302. doi: 10.1111/biom.12385.

McMahan, C., Tebbs, J., Bilder, C. (2012a). “Informative Dorfman Screening.” *Biometrics*, **68**, 287–296. doi: 10.1111/j.1541-0420.2011.01644.x.

McMahan, C., Tebbs, J., Bilder, C. (2012b). “Two-Dimensional Informative Array Testing.”*Biometrics*, **68**, 793–804. doi: 10.1111/j.1541-0420.2011.01726.x.

See Also

Other OTC functions: OTC2()

Examples

```
# Estimated running time for all examples was calculated
#   using a computer with 16 GB of RAM and one core of
#   an Intel i7-6500U processor. Please take this into
#   account when interpreting the run times given.

# Find the OTC for non-informative
#   two-stage hierarchical (Dorfman) testing.
OTC1(algorithm="D2", p=0.05, Se=0.99, Sp=0.99,
     group.sz=3:100, obj.fn=c("ET", "MAR"),
     trace=TRUE, print.time=TRUE)

# Find the OTC for informative two-stage hierarchical
#   (Dorfman) testing.
# A vector of individual probabilities is generated using
#   the expected value of order statistics from a beta
#   distribution with p = 0.01 and a heterogeneity level
#   of alpha = 0.5.
# This example takes approximately 2.5 minutes to run.

set.seed(52613)
OTC1(algorithm="ID2", p=0.01, Se=0.95, Sp=0.95, group.sz=50,
     obj.fn=c("ET", "MAR", "GR"),
     weights=matrix(data=c(1, 1, 10, 10, 0.5, 0.5),
     nrow=3, ncol=2, byrow=TRUE), alpha=0.5,
     trace=FALSE, print.time=TRUE, num.sim=10000)

# Find the OTC over all possible testing configurations
#   for non-informative three-stage hierarchical testing
#   with a specified group size.
OTC1(algorithm="D3", p=0.001, Se=0.95, Sp=0.95, group.sz=18,
     obj.fn=c("ET", "MAR", "GR"),
     weights=matrix(data=c(1, 1), nrow=1, ncol=2, byrow=TRUE),
     trace=FALSE, print.time=FALSE)

# Find the OTC for non-informative three-stage
#   hierarchical testing.
# This example takes approximately 20 seconds to run.

OTC1(algorithm="D3", p=0.06, Se=0.90, Sp=0.90,
     group.sz=3:30, obj.fn=c("ET", "MAR", "GR"),
     weights=matrix(data=c(1, 1, 10, 10, 100, 100),
     nrow=3, ncol=2, byrow=TRUE))

# Find the OTC over all possible configurations
#   for informative three-stage hierarchical testing
#   with a specified group size and a heterogeneous
#   vector of probabilities.
set.seed(1234)
OTC1(algorithm="ID3",
     probabilities=c(0.012, 0.014, 0.011, 0.012, 0.010, 0.015),
     Se=0.99, Sp=0.99, group.sz=6, obj.fn=c("ET","MAR","GR"),
     weights=matrix(data=c(1, 1), nrow=1, ncol=2, byrow=TRUE),
```

```
     alpha=0.5, num.sim=5000, trace=FALSE)

# Calculate the operating characteristics for
#   non-informative array testing without master pooling
#   with a specified array size.
OTC1(algorithm="A2", p=0.005, Se=0.95, Sp=0.95, group.sz=8,
     obj.fn=c("ET", "MAR"), trace=FALSE)

# Find the OTC for informative array testing without
#   master pooling.
# A vector of individual probabilities is generated using
#   the expected value of order statistics from a beta
#   distribution with p = 0.03 and a heterogeneity level
#   of alpha = 2. The probabilities are then arranged in
#   a matrix using the gradient method.
# This example takes approximately 30 seconds to run.

set.seed(1002)
OTC1(algorithm="IA2", p=0.03, Se=0.95, Sp=0.95,
     group.sz=3:20, obj.fn=c("ET", "MAR", "GR"),
     weights=matrix(data=c(1, 1, 10, 10, 100, 100),
                    nrow=3, ncol=2, byrow=TRUE), alpha=2)

# Find the OTC for non-informative array testing
#   with master pooling.
# This example takes approximately 20 seconds to run.

OTC1(algorithm="A2M", p=0.02, Se=0.90, Sp=0.90,
     group.sz=3:20, obj.fn=c("ET", "MAR", "GR"),
     weights=matrix(data=c(1, 1, 10, 10, 0.5, 0.5, 2, 2, 100, 100,
                           10, 100), nrow=6, ncol=2, byrow=TRUE))
```

Find the optimal testing configuration for group testing algorithms that use a multiplex assay for two diseases

Description

Find the optimal testing configuration (OTC) using non-informative and informative hierarchical and array-based group testing algorithms. Multiplex assays for two diseases are used at each stage of the algorithms.

Usage

```
OTC2(
  algorithm,
  p.vec = NULL,
  probabilities = NULL,
  alpha = NULL,
  Se,
  Sp,
  ordering = matrix(data = c(0, 1, 0, 1, 0, 0, 1, 1), nrow = 4, ncol = 2),
  group.sz,
  trace = TRUE,
  print.time = TRUE,
  ...
)
```

Arguments

`algorithm`	character string defining the group testing algorithm to be used. Non-informative testing options include two-stage hierarchical ("`D2`"), three-stage hierarchical ("`D3`"), square array testing without master pooling ("`A2`"), and square array testing with master pooling ("`A2M`"). Informative testing options include two-stage hierarchical ("`ID2`") and three-stage hierarchical ("`ID3`") testing.
`p.vec`	vector of overall joint probabilities. The joint probabilities are assumed to be equal for all individuals in the algorithm (non-informative testing only). There are four joint probabilities to consider: *p_00*, the probability that an individual tests negative for both diseases; *p_10*, the probability that an individual tests positive only for the first disease; *p_01*, the probability that an individual tests positive only for the second disease; and *p_11*, the probability that an individual tests positive for both diseases. The joint probabilities must sum to 1. Only one of `p.vec`, `probabilities`, or `alpha` should be specified.
`probabilities`	matrix of joint probabilities for each individual, where rows correspond to the four joint probabilities and columns correspond to each individual in the algorithm. Only one of `p.vec`, `probabilities`, or `alpha` should be specified.

`alpha` vector containing positive shape parameters of the Dirichlet distribution (for informative testing only). The vector will be used to generate a heterogeneous matrix of joint probabilities for each individual. The vector must have length 4. Further details are given under 'Details'. Only one of `p.vec`, `probabilities`, or `alpha` should be specified.

`Se` matrix of sensitivity values, where one value is given for each disease (or infection) at each stage of testing. The rows of the matrix correspond to each disease *k=1,...,K*, and the columns of the matrix correspond to each stage of testing *s=1,...,S*. If a vector of *K* values is provided, the sensitivity values associated with disease *k* are assumed to be equal to the *k*th value in the vector for all stages of testing. Further details are given under 'Details'.

`Sp` matrix of specificity values, where one value is given for each disease (or infection) at each stage of testing. The rows of the matrix correspond to each disease *k=1,...,K*, and the columns of the matrix correspond to each stage of testing *s=1,...,S*. If a vector of *K* values is provided, the specificity values associated with disease *k* are assumed to be equal to the *k*th value in the vector for all stages of testing. Further details are given under 'Details'.

`ordering` matrix detailing the ordering for the binary responses of the diseases. The columns of the matrix correspond to each disease and the rows of the matrix correspond to each of the 4 sets of binary responses for two diseases. This ordering is used with the joint probabilities. The default ordering is (p_00, p_10, p_01, p_11).

`group.sz` single group size or range of group sizes for which to calculate operating characteristics and/or find the OTC. The details of group size specification are given under 'Details'.

`trace` a logical value indicating whether the progress of calculations should be printed for each initial group size provided by the user. The default is `TRUE`.

`print.time` a logical value indicating whether the length of time for calculations should be printed. The default is `TRUE`.

`...` additional arguments to be passed to functions for hierarchical testing with multiplex assays for two diseases.

Details

This function finds the OTC for standard group testing algorithms with a multiplex assay that tests for two diseases and computes the associated operating characteristics. Calculations for hierarchical group testing algorithms are performed as described in Bilder et al. (2019) and calculations for array-based group testing algorithms are performed as described in Hou et al. (2019).

Available algorithms include two- and three-stage hierarchical testing and array testing with and without master pooling. Both non-informative and informative group testing settings are allowed for hierarchical algorithms. Only non-informative group testing settings are allowed for array testing algorithms. Operating characteristics calculated are expected number of tests, pooling sensitivity, pooling specificity, pooling positive predictive value, and pooling negative predictive value for each individual.

For informative algorithms where the `alpha` argument is specified, a heterogeneous matrix of joint probabilities for each individual is generated using the Dirichlet distribution. This is done using `rBeta2009::rdirichlet` and requires the user to set a seed to reproduce results. See Bilder et al. (2019) for additional details on the use of the Dirichlet distribution for this purpose.

The sensitivity/specificity values are allowed to vary across stages of testing. For hierarchical testing, a different sensitivity/specificity value may be used for each stage of testing. For array testing, a different sensitivity/specificity value may be used for master pool testing (if included), row/column testing, and individual testing. The values must be specified in the order of the testing performed. For example, values are specified as (stage 1, stage 2, stage 3) for three-stage hierarchical testing or (master pool testing, row/column testing, individual testing) for array testing with master pooling. A vector of *K* sensitivity/specificity values may be specified, and sensitivity/specificity values for all stages of testing are assumed to be equal. The first value in the vector will be used at each stage of testing for the first disease, and the second value in the vector will be used at each stage of testing for the second disease.

The value(s) specified by `group.sz` represent the initial (stage 1) group size for hierarchical testing and the row/column size for array testing. If a single value is provided for `group.sz`with two-stage hierarchical or array testing, operating characteristics will be calculated and no optimization will be performed. If a single value is provided for `group.sz` with three-stage hierarchical, the OTC will be found over all possible configurations with this initial group size. If a range of group sizes is specified, the OTC will be found over all group sizes.

In addition to the OTC, operating characteristics for some of the other configurations corresponding to each initial group size provided by the user are displayed. For algorithms where there is only one configuration for each initial group size (non-informative two-stage hierarchical and all array testing algorithms), results for each initial group size are provided. For algorithms where there is more than one possible configuration for each initial group size (informative two-stage hierarchical and all three-stage hierarchical algorithms), two sets of configurations are provided: 1) the best configuration for each initial group size, and 2) the top 10 configurations for each initial group size provided by the user. If a single value is provided for `group.sz` with array testing or non-informative two-stage hierarchical testing, operating characteristics will not be provided for configurations other than that specified by the user. Results are sorted by the value of the objective function per individual, `value`.

The displayed overall pooling sensitivity, pooling specificity, pooling positive predictive value, and pooling negative predictive value are weighted averages of the corresponding individual accuracy measures for all individuals within the initial group (or block) for a hierarchical algorithm, or within the entire array for an array-based algorithm. Expressions for these averages are provided in the Supplementary Material for Hitt et al. (2019). These expressions are based on accuracy definitions given by Altman and Bland (1994a, 1994b). Individual accuracy measures can be calculated using the `operatingCharacteristics2`(`opChar2`) function.

Value

A list containing:

`algorithm`	the group testing algorithm used for calculations.
`prob.vec`	the vector of joint probabilities provided by the user, if applicable (for non-informative algorithms only).
`joint.p`	the matrix of joint probabilities for each individual provided by the user, if applicable.
`alpha.vec`	the alpha vector provided by the user, if applicable (for informative algorithms only).
`Se`	the matrix of sensitivity values for each disease at each stage of testing.
`Sp`	the matrix of specificity values for each disease at each stage of testing.
`opt.ET`	a list containing:

OTC
: a list specifying elements of the optimal testing configuration, which may include:

  Stage1
  : group size for the first stage of hierarchical testing, if applicable.

  Stage2
  : group sizes for the second stage of hierarchical testing, if applicable.

  Block.sz
  : the block size/initial group size for informative Dorfman testing, which is not tested.

  pool.szs
  : group sizes for the first stage of testing for informative Dorfman testing.

  Array.dim
  : the row/column size for array testing.

  Array.sz
  : the overall array size for array testing (the square of the row/column size).

p.mat
: the matrix of joint probabilities for each individual in the algorithm. Each row corresponds to one of the four joint probabilities. Each column corresponds to an individual in the testing algorithm.

ET
: the expected testing expenditure for the OTC.

value
: the value of the expected number of tests per individual.

Accuracy
: the matrix of overall accuracy measures for the algorithm. The rows correspond to each disease. The columns correspond to the pooling sensitivity, pooling specificity, pooling positive predictive value, and pooling negative predictive value for the overall algorithm. Further details are given under 'Details'.

`Configs`
: a data frame containing results for the best configuration for each initial group size provided by the user. The columns correspond to the initial group size, configuration (if applicable), overall array size (if applicable), expected number of tests, value of the objective function per individual, and accuracy measures for each disease. Accuracy measures include the pooling sensitivity, pooling specificity, pooling positive predictive value, and pooling negative predictive value. No results are displayed if a single `group.sz` is provided. Further details are given under 'Details'.

`Top.Configs`
: a data frame containing results for some of the top configurations for each initial group size provided by the user. The columns correspond to the initial group size, configuration, expected number of tests, value of the objective function per individual, and accuracy measures for each disease. Accuracy measures include the pooling sensitivity, pooling specificity, pooling positive predictive value, and pooling negative predictive value. No results are displayed for non-informative two-stage hierarchical testing or for array testing algorithms. Further details are given under 'Details'.

Note

This function returns the pooling positive and negative predictive values for all individuals even though these measures are diagnostic specific; e.g., the pooling positive predictive value should only be considered for those individuals who have tested positive.

Additionally, only stage dependent sensitivity and specificity values are allowed within the program (no group within stage dependent values are allowed). See Bilder et al. (2019) for additional information.

Author(s)

This function was written by Brianna D. Hitt. It calls `ET.all.stages.new` and `PSePSpAllStages`, which were originally written by Christopher Bilder for Bilder et al. (2019), and `ARRAY`, which was originally written by Peijie Hou for Hou et al. (2020). The functions `ET.all.stages.new`, `PSePSpAllStages`, and `ARRAY` were obtained from http://chrisbilder.com/grouptesting. Minor modifications were made to the functions for inclusion in the binGroup2 package.

References

Altman, D., Bland, J. (1994). "Diagnostic tests 1: Sensitivity and specificity." *BMJ*, **308**, 1552.

Altman, D., Bland, J. (1994). "Diagnostic tests 2: Predictive values." *BMJ*, **309**, 102.

Bilder, C., Tebbs, J., McMahan, C. (2019). "Informative group testing for multiplex assays."*Biometrics*, **75**, 278–288. doi: 10.1111/biom.12988.

Hitt, B., Bilder, C., Tebbs, J., McMahan, C. (2019). "The objective function controversy for group testing: Much ado about nothing?" *Statistics in Medicine*, **38**, 4912–4923. doi: 10.1002/sim.8341.

Hou, P., Tebbs, J., Wang, D., McMahan, C., Bilder, C. (2020). "Array testing with multiplex assays." To appear in *Biostatistics*.

McMahan, C., Tebbs, J., Bilder, C. (2012a). "Informative Dorfman Screening." *Biometrics*, **68**, 287–296. doi: 10.1111/j.1541-0420.2011.01644.x.

See Also

Other OTC functions: `OTC1()`

Other multiplex testing functions: `operatingCharacteristics2()`

Examples

```
# Estimated running time for all examples was calculated
#   using a computer with 16 GB of RAM and one core of
#   an Intel i7-6500U processor. Please take this into
#   account when interpreting the run times given.

# Find the OTC for non-informative two-stage
#   hierarchical (Dorfman) testing
Se <- matrix(data = c(0.95, 0.95, 0.99, 0.99), nrow = 2, ncol = 2,
             dimnames = list(Infection = 1:2, Stage = 1:2))
Sp <- matrix(data = c(0.96, 0.96, 0.98, 0.98), nrow = 2, ncol = 2,
             dimnames = list(Infection = 1:2, Stage = 1:2))
OTC2(algorithm = "D2", p.vec=c(0.90, 0.04, 0.04, 0.02),
     Se = Se, Sp = Sp, group.sz = 3:30)
```

```
# Find the OTC over all possible testing configurations
#   for informative two-stage hierarchical (Dorfman)
#   testing with a specified group size.
# A matrix of joint probabilities for each individual is
#   generated using the Dirichlet distribution.
# This examples takes approximately 25 seconds to run.
Se <- matrix(data = rep(0.95, 4), nrow = 2, ncol = 2,
             dimnames = list(Infection = 1:2, Stage = 1:2))
Sp <- matrix(data = rep(0.99, 4), nrow = 2, ncol = 2,
             dimnames = list(Infection = 1:2, Stage = 1:2))

set.seed(1002)
OTC2(algorithm = "ID2", alpha=c(18.25, 0.75, 0.75, 0.25),
     Se = Se, Sp = Sp, group.sz = 10:20)

# Find the OTC for non-informative three-stage
#   hierarchical testing.
# This example takes approximately 1 minute to run.
Se <- matrix(data = rep(0.95, 6), nrow = 2, ncol = 3,
             dimnames = list(Infection = 1:2, Stage = 1:3))
Sp <- matrix(data = rep(0.99, 6), nrow = 2, ncol = 3,
             dimnames = list(Infection = 1:2, Stage = 1:3))

OTC2(algorithm = "D3", p.vec=c(0.95, 0.02, 0.02, 0.01),
     Se = Se, Sp = Sp, group.sz = 3:20)

# Find the OTC over all possible configurations
#   for informative three-stage hierarchical
#   testing with a specified group size
#   and a heterogeneous matrix of joint
#   probabilities for each individual.
set.seed(8791)
Se <- matrix(data = rep(0.95, 6), nrow = 2, ncol = 3,
             dimnames = list(Infection = 1:2, Stage = 1:3))
Sp <- matrix(data = rep(0.99, 6), nrow = 2, ncol = 3,
             dimnames = list(Infection = 1:2, Stage = 1:3))
p.unordered <- t(rBeta2009::rdirichlet(n = 12,
                            shape = c(18.25, 0.75, 0.75, 0.25)))
p.ordered <- p.unordered[, order(1 - p.unordered[1,])]
OTC2(algorithm="ID3", probabilities = p.ordered,
         Se=Se, Sp=Sp, group.sz = 12,
         trace=FALSE, print.time=FALSE)

# Find the OTC for non-informative array testing
#   without master pooling.
Se <- matrix(data = rep(0.95, 4), nrow = 2, ncol = 2,
             dimnames = list(Infection = 1:2, Stage = 1:2))
Sp <- matrix(data = rep(0.99, 4), nrow = 2, ncol = 2,
             dimnames = list(Infection = 1:2, Stage = 1:2))
OTC2(algorithm = "A2", p.vec=c(0.90, 0.04, 0.04, 0.02),
     Se = Se, Sp = Sp, group.sz = 3:12)

# Find the OTC for non-informative array testing
#   with master pooling.
Se <- matrix(data = rep(0.95, 6), nrow = 2, ncol = 3,
             dimnames = list(Infection = 1:2, Stage = 1:3))
```

```
Sp <- matrix(data = rep(0.99, 6), nrow = 2, ncol = 3,
             dimnames = list(Infection = 1:2, Stage = 1:3))
OTC2(algorithm = "A2M", p.vec=c(0.90, 0.04, 0.04, 0.02),
     Se = Se, Sp = Sp, group.sz = 10,
     trace=FALSE, print.time=FALSE)
```

Appendix D

Operating characteristics for informative two-stage hierarchical testing

In Appendix B, we presented derivations of operating characteristics for S-stage hierarchical testing with the exception of informative two-stage hierarchical testing. In this section, we provide derivations for the expected number of tests and accuracy measures for informative two-stage hierarchical testing. While this dissertation focuses on the implementation of this algorithm via the pool-specific optimal Dorfman (PSOD) method described in McMahan et al. (2012a), the derivations presented here are easily generalizable to other informative two-stage hierarchical algorithms presented in McMahan et al. (2012a).

McMahan et al. (2012a) presented derivations of operating characteristics for informative two-stage hierarchical testing. The authors make the assumption that the sensitivity and specificity are not dependent on group size and use the same diagnostic accuracy for both stages of the testing algorithm. In the derivations presented here, we allow unequal sensitivity and unequal specificity values across stages of testing. For consistency in the group testing literature and associated R functions, we use the same notation and derivation path as McMahan et al. (2012a) whenever possible.

D.1. Expected number of tests

Consider an initial block of size N individuals to be tested for a disease. As in McMahan et al. (2012a), we initially assume that the true probabilities for each individual are known. The individuals are ordered from lowest to highest probability of disease and divided into groups of size c_j, $j = 1, ..., J$, so that group $\mathcal{P}_j$ contains the c_j lowest risk individuals which remain after constructing the first $j-1$ groups. Let $I_{j(k)}$ denote the kth ordered individual in the jth group, for $j = 1, ..., J$ and $k = 1, ..., c_j$, with corresponding probabilities $p_{j(k)}$.

Further, let $G_j = 1\,(0)$ denote the positive (negative) test result and let $\tilde{G}_j = 1(0)$ denote the positive (negative) true status of the jth group. If $G_j = 0$, all individuals in the corresponding group are declared negative. If $G_j = 1$, individuals in the corresponding group are individually retested in the second and final stage of testing. Define $S_{e:1} = P\left(G_j = 1 \,\middle|\, \tilde{G}_j = 1\right)$ and $S_{p:1} = P\left(G_j = 0 \,\middle|\, \tilde{G}_j = 0\right)$ as the test sensitivity and the test specificity corresponding to the group tests performed in the first stage of the algorithm. We still assume that the sensitivity and specificity are not dependent on c_j.

The probability the jth group is truly positive is

$$P\left(\tilde{G}_j = 1\right) = 1 - \prod_{k=1}^{c_j} \left(1 - p_{j(k)}\right).$$

Using the Law of Total Probability, McMahan et al. (2012a) expressed the

probability of the jth group testing positive as

$$\begin{aligned} P\left(G_j = 1\right) &= P\left(G_j = 1 \,\middle|\, \tilde{G}_j = 0\right) P\left(\tilde{G}_j = 0\right) + \\ &\quad P\left(G_j = 1 \,\middle|\, \tilde{G}_j = 1\right) P\left(\tilde{G}_j = 1\right) \\ &= S_e + \left(1 - S_e - S_p\right) \prod_{k=1}^{c_j} \left(1 - p_{j(k)}\right). \end{aligned}$$

Substituting $S_{e:1}$ ($S_{p:1}$) for S_e (S_p) in the above expression, we can write

$$P\left(G_j = 1\right) = S_{e:1} + \left(1 - S_{e:1} - S_{p:1}\right) \prod_{k=1}^{c_j} \left(1 - p_{j(k)}\right). \tag{D.1.1}$$

Let $T_{\mathcal{P}_j}$ denote the number of tests needed to identify all positive individuals in group $\mathcal{P}_j$. If $c_j > 1$, then $P\left(T_{\mathcal{P}_j} = 1\right) = P\left(G_j = 0\right)$ and $P\left(T_{\mathcal{P}_j} = c_j + 1\right) = P\left(G_j = 1\right)$. If $c_j = 1$, then $P\left(T_{\mathcal{P}_j} = 1\right) = 1$. Using these facts, McMahan et al. (2012a) expressed the expected number of tests for a block of size N as

$$\begin{aligned} E\left(T\right) &= \sum_{j=1}^{J} E\left(T_{\mathcal{P}_j}\right) \\ &= J + \sum_{j=1}^{J} c_j I\left(c_j > 1\right) P\left(G_j = 1\right), \end{aligned} \tag{D.1.2}$$

where $I\left(\cdot\right)$ represents the indicator function. Substituting equation (D.1.1) into equation (D.1.2) gives

$$E\left(T\right) = J + \sum_{j=1}^{J} c_j I\left(c_j > 1\right) \left\{ S_{e:1} + \left(1 - S_{e:1} - S_{p:1}\right) \prod_{k=1}^{c_j} \left(1 - p_{j(k)}\right) \right\}.$$

D.2. Accuracy measures

Let $Y_{j(k)}$ denote the second-stage test outcome and $\tilde{Y}_{j(k)}$ denote the true status of individual $I_{j(k)}$, so that $P\left(\tilde{Y}_{j(k)}=1\right) = p_{j(k)}$. As in McMahan et al. (2012a), we adopt the assumption that true individual statuses are independent random variables. Define $S_{e:2} = P\left(Y_{j(k)}=1 \,\middle|\, \tilde{Y}_{j(k)}=1\right)$ and $S_{p:2} = P\left(Y_{j(k)}=0 \,\middle|\, \tilde{Y}_{j(k)}=0\right)$ as the test sensitivity and the test specificity corresponding to the individual testing conducted in the second stage of the algorithm. Let $I^{+}_{j(k)}$ $\left(I^{-}_{j(k)}\right)$ denote the event that individual $I_{j(k)}$ is classified as positive (negative) by the PSOD algorithm. The pooling sensitivity for individual $I_{j(k)}$ is the probability of a correct positive diagnosis, $PS_e^{I_{j(k)}} = P\left(I^{+}_{j(k)} \,\middle|\, \tilde{Y}_{j(k)}=1\right)$. The pooling specificity for individual $I_{j(k)}$ is the probability of a correct negative diagnosis, $PS_p^{I_{j(k)}} = P\left(I^{-}_{j(k)} \,\middle|\, \tilde{Y}_{j(k)}=0\right)$. We derive expressions for $PS_e^{I_{j(k)}}$ and $PS_p^{I_{j(k)}}$ when $c_j > 1$, and we assume that the diagnostic test results are independent, conditional on the true status of the group (or individual) being tested (McMahan et al., 2012a). Additional discussion of the conditional independence assumption is available in Litvak et al. (1994).

D.2.1. Pooling sensitivity

For informative Dorfman testing, individual $I_{j(k)}$ is categorized as positive when both its group test and individual test are positive. McMahan et al. (2012a) showed that an individual's pooling sensitivity can be rewritten as

$$\begin{aligned} PS_e^{I_{j(k)}} &= P\left(G_j=1 \,\middle|\, \tilde{Y}_{j(k)}=1\right) P\left(Y_{j(k)}=1 \,\middle|\, G_j=1, \tilde{Y}_{j(k)}=1\right) \\ &= S_e^2. \end{aligned}$$

This derivation uses the conditional independence assumption previously mentioned in Section D.1 to result in the product of the sensitivities at each stage of the algorithm. Allowing for S_e to differ across the stages of the algorithm, the pooling sensitivity simply becomes

$$PS_e^{I_{j(k)}} = S_{e:1}S_{e:2}.$$

D.2.2. Pooling specificity

We now consider the pooling specificity. Individual $I_{j(k)}$ can be categorized as negative in two situations: 1) its group test is negative, or 2) its group test is positive but its individual test is negative. McMahan et al. (2012a) expressed the pooling specificity for an individual as

$$\begin{aligned} PS_p^{I_{j(k)}} &= P\left(G_j = 0 \,\middle|\, \tilde{Y}_{j(k)} = 0\right) + \\ &\quad P\left(G_j = 1, Y_{j(k)} = 0 \,\middle|\, \tilde{Y}_{j(k)} = 0\right). \end{aligned} \tag{D.2.1}$$

Using the Law of Total Probability, McMahan et al. (2012a) wrote the first term in equation (D.2.1) as

$$\begin{aligned} &P\left(G_j = 0 \,\middle|\, \tilde{Y}_{j(k)} = 0\right) \\ &\quad = P\left(G_j = 0 \,\middle|\, \tilde{G}_j = 0\right) P\left(\tilde{G}_j = 0 \,\middle|\, \tilde{Y}_{j(k)} = 0\right) + \\ &\qquad P\left(G_j = 0 \,\middle|\, \tilde{G}_j = 1\right) P\left(\tilde{G}_j = 1 \,\middle|\, \tilde{Y}_{j(k)} = 0\right). \end{aligned}$$

Using the definitions for $S_{e:1}$ and $S_{p:1}$, we can write

$$P\left(G_j = 0 \,\middle|\, \tilde{Y}_{j(k)} = 0\right)$$

$$
\begin{aligned}
&= S_{p:1}\left\{1 - P\left(\tilde{G}_j = 1 \,\middle|\, \tilde{Y}_{j(k)} = 0\right)\right\} + \\
&\quad (1 - S_{e:1})\, P\left(\tilde{G}_j = 1 \,\middle|\, \tilde{Y}_{j(k)} = 0\right) \\
&\quad S_{p:1} + (1 - S_{e:1} - S_{p:1})\, P\left(\tilde{G}_j = 1 \,\middle|\, \tilde{Y}_{j(k)} = 0\right). \qquad \text{(D.2.2)}
\end{aligned}
$$

Note that $P\left(\tilde{G}_j = 1 \,\middle|\, \tilde{Y}_{j(k)} = 0\right)$ represents the probability that at least one individual in the jth group, other than the kth ordered individual, is truly positive. This can be written as

$$
P\left(\tilde{G}_j = 1 \,\middle|\, \tilde{Y}_{j(k)} = 0\right) = 1 - \prod_{k' \in A_j^{(k)}} \left(1 - p_{j(k')}\right), \qquad \text{(D.2.3)}
$$

where $A_j^{(k)} = \{1, ..., k-1, k+1, ..., c_j\}$. Substituting equation (D.2.3) into equation (D.2.2) gives

$$
\begin{aligned}
&P\left(G_j = 0 \,\middle|\, \tilde{Y}_{j(k)} = 0\right) \\
&\qquad = S_{p:1} + (1 - S_{e:1} - S_{p:1}) \left\{1 - \prod_{k' \in A_j^{(k)}} \left(1 - p_{j(k')}\right)\right\} \\
&\qquad = (1 - S_{e:1}) - (1 - S_{e:1} - S_{p:1}) \prod_{k' \in A_j^{(k)}} \left(1 - p_{j(k')}\right). \qquad \text{(D.2.4)}
\end{aligned}
$$

McMahan et al. (2012a) expressed the second term in equation (D.2.1) as

$$
\begin{aligned}
&P\left(G_j = 1, Y_{j(k)} = 0 \,\middle|\, \tilde{Y}_{j(k)} = 0\right) \\
&\qquad = P\left(G_j = 1 \,\middle|\, \tilde{Y}_{j(k)} = 0\right) P\left(Y_{j(k)} = 0 \,\middle|\, G_j = 1, \tilde{Y}_{j(k)} = 0\right).
\end{aligned}
$$

Allowing for unequal sensitivity and unequal specificity values across stages of testing, we can write

$$\begin{aligned}
&P\left(G_j = 1, Y_{j(k)} = 0 \;\middle|\; \tilde{Y}_{j(k)} = 0\right) \\
&\quad = P\left(G_j = 1 \;\middle|\; \tilde{Y}_{j(k)} = 0\right) S_{p:2} \\
&\quad = S_{p:2}\left\{P\left(G_j = 1 \;\middle|\; \tilde{G}_j = 0\right) P\left(\tilde{G}_j = 0 \;\middle|\; \tilde{Y}_{j(k)} = 0\right) + \right. \\
&\qquad \left. P\left(G_j = 1 \;\middle|\; \tilde{G}_j = 1\right) P\left(\tilde{G}_j = 1 \;\middle|\; \tilde{Y}_{j(k)} = 0\right)\right\} \\
&\quad = S_{p:2}\left[(1 - S_{p:1})\left\{1 - P\left(\tilde{G}_j = 1 \;\middle|\; \tilde{Y}_{j(k)} = 0\right)\right\} + \right. \\
&\qquad \left. S_{e:1} P\left(\tilde{G}_j = 1 \;\middle|\; \tilde{Y}_{j(k)} = 0\right)\right] \\
&\quad = S_{p:2}\left[(1 - S_{p:1}) - (1 - S_{e:1} - S_{p:1}) P\left(\tilde{G}_j = 1 \;\middle|\; \tilde{Y}_{j(k)} = 0\right)\right].
\end{aligned}$$

Substituting equation (D.2.3) into the above equation gives

$$\begin{aligned}
&P\left(G_j = 1, Y_{j(k)} = 0 \;\middle|\; \tilde{Y}_{j(k)} = 0\right) \\
&\quad = S_{p:2}\left[(1 - S_{p:1}) - (1 - S_{e:1} - S_{p:1}) \times \right. \\
&\qquad \left. \left\{1 - \prod_{k' \in A_j^{(k)}} \left(1 - p_{j(k')}\right)\right\}\right] \\
&\quad = S_{e:1} S_{p:2} + (1 - S_{e:1} - S_{p:1}) S_{p:2} \prod_{k' \in A_j^{(k)}} \left(1 - p_{j(k')}\right). \qquad \text{(D.2.5)}
\end{aligned}$$

Substituting equations (D.2.4) and (D.2.5) into equation (D.2.1), we get

$$\begin{aligned} PS_p^{I_{j(k)}} &= (1 - S_{e:1}) - (1 - S_{e:1} - S_{p:1}) \prod_{k' \in A_j^{(k)}} \left(1 - p_{j(k')}\right) + \\ &\quad S_{e:1} S_{p:2} + (1 - S_{e:1} - S_{p:1}) S_{p:2} \prod_{k' \in A_j^{(k)}} \left(1 - p_{j(k')}\right) \\ &= 1 - (1 - S_{p:2}) \left\{ S_{e:1} + \right. \\ &\quad \left. (1 - S_{e:1} - S_{p:1}) \prod_{k' \in A_j^{(k)}} \left(1 - p_{j(k')}\right) \right\}. \end{aligned}$$

The pooling positive predictive value is defined as $PPPV^{I_{j(k)}} = P\left(\tilde{Y}_{j(k)} = 1 \,\middle|\, I_{j(k)}^{+}\right)$ and the pooling negative predictive value is defined as $PNPV^{I_{j(k)}} = P\left(\tilde{Y}_{j(k)} = 0 \,\middle|\, I_{j(k)}^{-}\right)$. Expressions for these values follow from those given in McMahan et al. (2012a).