Generalized partition testing via Bayes linear methods

F.P.A. Coolen, M. Goldstein
Department of Mathematical Sciences
University of Durham
Science Laboratories, South Road
Durham, DH1 3LE, England

M. Munro
Department of Computer Science
University of Durham
Science Laboratories, South Road
Durham, DH1 3LE, England

Abstract

This paper explores the use of Bayes linear methods related to partion testing for software. If a partition of the input domain has been defined, the method works without the assumption of homogeneous (revealing) subdomains, and also includes the possibility to learn, from testing inputs in one subdomain, about inputs in other subdomains, through explicit definition of the correlations involved. To enable practical application an exchangeability structure needs to be defined carefully, for which means the judgements of experts with relation to the software is needed. Next to presenting the basic idea of Bayes linear methods and how it can be used to generalize partition testing, some important aspects related to applications as well as for future research are discussed.

Keywords: Bayes linear methods; expert knowledge; partition testing; software testing theory.

1 Introduction

When developing and maintaining software one of the key areas is testing. Complete exhaustive testing of any non-trivial program is easily shown to be infeasible and even impossible. Given time and cost constraints placed on testers, the main issue is what subset of all possible test cases should be chosen. An accepted approach is to partition the input domain into a number of subdomains such that each member of a subdomain is deemed to have the same effect on the system. The testing task then becomes selecting one (or possibly more) representative(s) from each subdomain, and the art of testing that of defining subdomains.

Partitioning can be based either on a black box view of the system through the use of the requirements and specification, or on a white box view through the structural analysis of the code, or on a combination of these two views. Whilst there are a number of techniques that can be used the process is still fraught with difficulties and problems [17, 22].

Testers build up a great deal of expert knowledge about the effects of particular test cases. They also have an intuitive feel for how the input domain should be partitioned. This intuition can lead to difficulties of ill defined and overlapping subdomains [17] which can possibly invalidate any analysis of test case effectiveness. The use of the testers' expert knowledge and judgement has not been fully utilised.

The usual assumptions made when discussing partition testing are:

1. test results for inputs in one subdomain do not provide any information about inputs in other subdomains; and

2. all inputs within the same subdomain lead to correct outputs or all lead to incorrect outputs.

In this paper both of these assumptions are addressed and a generalized method for partition testing that does not make these assumptions is described. The method uses the quantification of expert judgement to resolve the problems.

In section 2 partition testing is considered, in the way it is usually presented in the literature on software testing theory. In section 3 some of the ideas behind Bayes linear methods are briefly introduced, related to statistical inference from a subjective point of view. In section 4 partition testing is generalized by considering an exchangeability structure related to the partition of the input domain, together with Bayes linear updating given such a structure. As an illustration a well-known problem of triangle classification is discussed in section 5. Section 6 is a brief discussion of possible contributions from a Bayes linear approach to a fairly general software testing cycle.

2 Partition testing

Let the term 'partition' be restricted to the formal mathematical meaning of a finite exhaustive division of the input domain I into *disjoint* subdomains $I_1, I_2, \ldots, I_k$. In the literature, an ideal situation that is often assumed is that all subdomains are what has been called 'homogeneous' [17] or 'revealing' [23], meaning that within each subdomain either the program produces the correct output for every input element or the program produces an incorrect output for every input element. To enable clear presentation of the results in the following sections, assume that there is an oracle available so there is no doubt about whether output is correct or not. For future research and applications the theory will allow generalizations, including the absence of oracles.

The ideal situation described above would lead to 'perfect testing' by testing a single input per subdomain. Inconsistent with the idea of partition testing with such revealing or homogeneous subdomains is the suggestion that per subdomain several elements should be tested, which is often advocated in literature [16, 17, 22]. Obviously, such 'imperfect' testing is based on an intuition that the elements per subdomain are different in some aspects, and one can never be sure whether they all lead to similar output unless one would know all details of the program

[22, 23], or unless one would have trivial subdomains. There are several practical problems, and while attention has been paid to the choice of the subdomains, the modelling of opinions with regard to 'level of homogeneity' of different input elements has been neglected. Generally, it seems that to choose combinations of input variables in testing software and to assess the results of the tests, it is necessary to use judgements of experts, for example the writers of the software, the testers of the software, and people with specific knowledge about the environment in which the software will be used.

3 Bayes linear methods

The main method for combining expert judgements with experimental data is the Bayesian approach [1], which has been successfully applied in a wide variety of areas. However, as ever more complex problems in uncertainty are considered, the traditional Bayes paradigm, in which all uncertainties are carefully specified and analysed, becomes increasingly difficult to apply. An alternative approach, where limited aspects of uncertainties are specified and analysed, is the use of Bayes linear methods; for an overview of the methodology, see [6], a summary of the approach is given in [13]. These are methods working on expectations rather than probabilities (see also [5]), so in practice one can restrict specifications and analyses to variance structures, reflecting the judgements of experts, for the various quantities of interest. Bayes linear methods are useful if the problem at hand is too difficult to allow a full prior specification, as needed for standard Bayesian methods, or if the analysis is too difficult given such a full prior specification. Bayes linear methods have sound logical foundations [11], and a system has been developed for practical application [14]. First experiences are positive; for example, Bayes linear methods have been successfully applied to very large scale computer experiments related to reservoir simulators in the oil industry [3].

In a simple form, a Bayes linear analysis concerns two vectors of quantities, namely a vector $B = (B_1, \ldots, B_r)^T$, which one wishes to learn about, and a vector $D = (D_1, \ldots, D_s)^T$ of quantities which will be observed in order to adjust beliefs about B. For these quantities one needs to assess prior specifications (that is, before the values of the vector D are actually known) $E(B), E(D), Var(B), Var(D), Cov(B, D)$, where the expectations $E(B), E(D)$ are vectors of dimension r and s, respectively, the variances $Var(B), Var(D)$ are square symmetric matrices of dimensions $r \times r$ and $s \times s$, respectively, and the covariance $Cov(B, D)$ is a matrix of dimension $r \times s$, writing $Cov(D, B) = Cov(B, D)^T$. If D is observed, the adjusted expectation $E_D(B)$ and variance $Var_D(B)$ for B given D may be assessed as

$$\begin{aligned} E_D(B) &= E(B) + Cov(B, D)[Var(D)]^{-1}[D - E(D)] \\ Var_D(B) &= Var(B) - Cov(B, D)[Var(D)]^{-1}Cov(D, B). \end{aligned}$$

If the inverse of the matrix $Var(D)$ does not exist this is replaced in these formulas by the Moore-Penrose generalized inverse of $Var(D)$ [15]. $E_D(B)$ may be interpreted as a simple approximation to a full Bayes posterior expectation based on a limited specification of prior beliefs. This approximation is exact if D is a vector of indicator functions for a partition, or if the joint probability distribution is multivariate normal. Alternatively, $E_D(B)$ may be viewed as the appropriate generalization to posterior expectation, given only the corresponding partial prior specifications [12].

4 Exchangeability structure related to the partition of the input domain

For software testing, let D be the vector of test inputs and B the vector of all other inputs. This paper considers the simple situation of binary test results, with a value 0 assigned to an input if the program gives the correct corresponding output, and a value 1 if the output is incorrect, assuming that an oracle is available so no mistakes are made in assigning these 0 - 1 values. Using these binary test results relates to counting the number of incorrect outputs. In general, for Bayes linear methods it is not necessary to restrict the test results to 0 - 1 values, neither is it necessary to assume that an oracle is available. These assumptions are made here to simplify the presentation, and these assumptions also seem to be common in literature on partition testing. For ease of notation, if an element of the input domain is denoted by X, the test result for this element is denoted by either $X = 0$ or $X = 1$. This implies that B and D are 0 - 1 valued vectors.

The adjusted expectation $E_D(B)$ and variance $Var_D(B)$ for B given D are easy to calculate, if the necessary expectations, variances and covariances are given. This is where practical problems arise. Suppose that the input domain has a known finite number of elements, say N. A full specification, enabling calculation of the adjusted expectation $E_D(B)$ and variance $Var_D(B)$ for whatever subset D of the entire input domain I is chosen as test inputs, would ask for N values for the expectation vector and $N(N+1)/2$ values for all elements in the variance and covariance matrices. This means that, e.g. for a program with only 100 different inputs, one would need to assess the expected test result value for every single one of the inputs, together with the covariance for every pair of inputs. This requires a total of 5150 numerical values to be assessed, which is obviously an impossible task in practice unless we impose some structure on the prior assessments.

To simplify the assessment of expectations, variances and covariances, exchangeability assumptions can be made, partitioning the input domain into subdomains corresponding to partition testing, but offering more flexibility than the partition testing methods presented in literature so far. For a detailed presentation of the central role of exchangeability in statistics see De Finetti [5] (or [1, 9]). In direct relation to Bayes linear methods Goldstein [11], discussing the strong conditions that exchangeability imposes on prior beliefs, presents a weaker form of exchangeability, called second order exchangeability, that is a strong enough assumption for the application of Bayes linear methods. Suppose that the following exchangeability assumptions are made, corresponding to the idea of partition testing with the input domain divided into subdomains $I_1, I_2, \dots, I_k$, with X_{ij} the j-th input tested in subdomain I_i,

$$\begin{aligned}
E(X_{ij}) &= p_i, \text{ for all } X_{ij} \in I_i, \\
Var(X_{ij}) &= v_i, \text{ for all } X_{ij} \in I_i, \\
Cov(X_{ij}, X_{il}) &= c_i, \text{ for all } X_{ij}, X_{il} \in I_i \ (j \neq l), \\
Cov(X_{ij}, X_{hl}) &= c_{ih}, \text{ for all } X_{ij} \in I_i, X_{hl} \in I_h \ (i \neq h).
\end{aligned}$$

For these 0-1 valued X_{ij} the restriction $v_i = p_i(1-p_i)$ holds. These exchangeability assumptions correspond to the judgement that it does not matter in which order we test the items within each partition element. Supposing a large number of possible inputs within each subdomain, this exchangeability structure implies the following representation for the unknown 0 - 1 valued quantities related to the inputs X_{ij} [10]

$$\begin{aligned}
X_{ij} &= M_i + R_{ij}, \text{ with} \\
E(M_i) &= p_i, \\
Var(M_i) &= c_i, \\
E(R_{ij}) &= 0, \\
Var(R_{ij}) &= v_i - c_i, \\
Cov(M_i, M_h) &= c_{ih} \qquad \text{for } i \neq h,
\end{aligned}$$

where the R_{ij} are mutually uncorrelated, and also uncorrelated with M_i. It can be remarked that $c_i \leq v_i = p_i(1-p_i)$ since X_{ij} and M_i have equal expectations p_i, but the variance of X_{ij} is the maximum possible variance related to random variables on $[0,1]$ with expectation p_i since X_{ij} is 0 - 1 valued, whereas M_i is real-valued on the interval $[0,1]$.

Although M_i is not observable, it can be regarded as the unknown average proportion of failures for inputs in subdomain I_i, and its expectation and variance can be assessed through its relations with the observable X_{ij}, as indicated above. Therefore we may choose to learn about $B = (M_1, \ldots, M_k)$. The effect of observations from tests is reduction of the variance of each M_i, with the limiting situation (for the number of observations going to infinity) that $Var(M_i)$ goes to zero, in which case the uncertainty in the values of individual X_{ij} results from (the variance of) R_{ij}. There is an analogy with tossing a coin, where sufficient tosses approximately identify the probability of heads, say p_h, so remaining uncertainty is the outcome of each toss with p_h known. A useful property of the adjusted variance $Var_D(M_i)$ is that it only depends on the number of data observed per subdomain, and not on the actual output values of the observed data [15]. We use this property for analysing choices of test suites in the examples in section 5. It should be emphasized that, in this setting, learning from test results is reflected in updated expected values for M_i and reduced variances for M_i. To adjust beliefs on M_i when observing test results the following covariances are needed (these follow from the assessments and assumptions above):

$$\begin{aligned} Cov(M_i, X_{ij}) &= Var(M_i) = c_i, \\ Cov(M_i, X_{hl}) &= Cov(M_i, M_h) = c_{ih}. \end{aligned}$$

Assuming this exchangeability structure simplifies the assessment task considerably, since now only k values p_i and c_i are needed, one for each subdomain I_i, while specification of the covariances between all pairs of subdomains requires $k(k-1)/2$ numerical values. For example, for $k = 9$, nine expectation values p_i need to be assessed together with nine variances c_i and 36 covariances c_{ih}, so a total of 54 values need to be specified. This is the minimal specification if we wish to carry out an uncertainty analysis. The need to specify a detailed full probability model would require much more consideration.

After subdomains have been identified under the assumed structure described above, the first task is assessment of the values p_i, c_i, c_{ih} $(i = 1, \ldots, k,\ i < h)$. For 0-1 valued random quantities, which, in effect, simply count failures in output, the expected value of a random quantity is equal to the probability that it has value 1, so $p_i = E(X_{ij}) = P(X_{ij} = 1)$. This is the probability that input $X_{ij} \in I_i$ leads to wrong output, where probability has a subjective interpretation, i.e. represents the judgement of the expert. An easy way to measure your probability is as follows: consider a bet that will pay £1 if the output corresponding to input X_{ij} is wrong, and nothing if the output is correct. Your probability for the event $X_{ij} = 1$ may be measured as the price £p_i that you regard fair for this bet, meaning that you are indifferent between either selling or buying the bet for that price. Less straightforward is assessment of values for c_i and c_{ih}, quantifying how the quality of the outputs for two different input values depend on each other, either within the same subdomain ($c_i = Cov(X_{ij}, X_{il}),\ j \neq l$) or in different subdomains ($c_{ih} = Cov(X_{ij}, X_{hl}),\ i \neq h$). This specification is simplified by the fact that we are dealing with two 0-1 valued random quantities, as in general for two such quantities X, Y the following holds: $Cov(X,Y) = P(Y = 1)[P(X = 1|Y = 1) - P(X = 1)]$, where $P(X = 1|Y = 1)$ is your probability for the event $X = 1$ given that $Y = 1$, namely the event that input X will lead to wrong output, given that testing input Y leads to wrong corresponding output. Hence, to assess c_i we only need to assess $p_i = P(X_{ij} = 1)$ and $p_{i|i} = P(X_{ij} = 1|X_{il} = 1)$, $j \neq l$. Having assessed c_i, we assess c_{ih} by specifying $p_{i|h} = P(X_{ij} = 1|X_{hl} = 1)$.

The assumed exchangeability structure can also be used to arrive at partition testing as it is known in the literature on software testing, showing how serious the reduction of freedom is, and therefore how strong the assumptions related to standard partition testing are.

One assumption that seems standard in the literature on partition testing is that test results for inputs in one subdomain do not provide any information about inputs in other subdomains, which in the structure above corresponds to the judgement that $c_{ih} = 0$ for all $i \neq h$. In many practical situations this assumption is highly disputable, and it is here that the method of this paper provides a useful generalization to partition testing as presented in the literature so far. As an illustration of the effect of this assumption, in section 5 we consider a well-known ‘artificial’ test problem in the literature.

A second dubious assumption is that all inputs within the same subdomain lead to correct outputs or all lead to incorrect outputs. This rather extreme assumption, that implies that testing a single input per subdomain would reveal everything about that whole subdomain, corresponds to the judgement that $v_i = c_i$, so $Var(R_{ij}) = 0$ which implies that $R_{ij} = 0$ (with probability 1) and reduces the model to $X_{ij} = M_i$, where either $M_i = 0$ (all elements in I_i leading to correct outputs) or $M_i = 1$ (all elements in I_i leading to failures), leading to the restriction $c_i = p_i(1 - p_i)$. Many researchers have, correctly, had doubts about this second assumption, which is not needed in our formulation.

Another extreme situation in testing, complete random testing without any assumed structure, is simply achieved by taking $k = 1$ in the structure above, so not dividing the input domain into subdomains.

5 Example: Triangle classification problem

To illustrate our method, we consider the triangle classification problem, which is discussed in many papers and books on software testing, e.g. see Myers [19] and Weyuker and Ostrand [23]. Briefly, the problem specifications are that three positive integers $A \geq B \geq C$ should be given as input, and as output the program should indicate if these integers represent sides of a triangle, and if so, what kind of triangle (equilateral, isosceles, scalene right, obtuse or acute).

Weyuker and Ostrand [23] form a partition of the input domain consisting of nine non-empty subdomains $I_1, \dots, I_9$, which we will use to illustrate the Bayes linear approach as described in section 4. These subdomains are (with the correct output stated between brackets):

I_1 : $(A < B) \vee (B < C)$ (illegal order)

I_2 : $(A \geq B + C) \wedge (A > B > C)$ (not a triangle)

I_3 : $(A \geq B + C) \wedge (B = C)$ (not a triangle)

I_4 : $(A = B = C)$ (equilateral)

I_5 : $(A = B > C)$ (isosceles)

I_6 : $(A > B = C) \wedge (A < B + C)$ (isosceles)

I_7 : $(A > B > C) \wedge (A^2 = B^2 + C^2)$ (right scalene)

I_8 : $(A > B > C) \wedge (A^2 < B^2 + C^2)$ (acute scalene)

I_9 : $(A > B > C) \wedge (A^2 > B^2 + C^2) \wedge (A < B + C)$ (obtuse scalene)

Weyuker and Ostrand [23] assume, as is common in the literature on partition testing, that these nine subdomains are revealing, meaning that all elements within the same subdomain lead to correct output or failure.

As an illustration of our method, assume that the tester does not have actual knowledge about the code, but uses the above partition as a representation of his judgement of exchangeability, based on his insights as to how the software should perform, as the basis for carrying out black-box testing of the code. From this perspective, the tester might express cautious beliefs as to the quality of the software, and may not want to assume revealing subdomains. This could arise from the possibility of general programming errors (e.g. unnecessary rounding of the three numbers), or particular logical oversights (e.g. wrongly classifying $(A > B = C) \wedge (A^2 = B^2 + C^2)$ as right scalene), or implementation issues (e.g. intermittent problems in correctly accessing the program inputs).

Similarly, the tester may want to take into account, based on the logic behind this partition, that tests in one subdomain are also, to some extent, informative for other subdomains. For example, the subdomains I_2, I_3 have $(A \geq B + C)$ in common, whereas I_2, I_9 share $(A > B > C)$ and I_3, I_5, I_6 share the fact that exactly two sides of the triangle are equal. This raises the possibility that inputs from these subdomains may share some parts of the software, and similarly that there may be several parts of the program shared by most subdomains. Such sharing may, at one extreme, correspond to re-use of common pieces of code, while, at the other extreme, each subdomain may be separately coded, on the basis of an analysis which has common logical faults. Such errors may relate to shared logical issues in classifying the various triangles, or indirect shared errors, such as problems in handling the inputs to the program. Therefore, it seems likely that there will be correlations between different subdomains. In black-box testing, such correlations arise not from the certain knowledge of such relationships, but rather from the subjective judgement of the tester as to the likelihood of such relationships.

Thus, for example, if one learns, from testing, that the program provides correct output for an element of I_3, then this will probably increase confidence that the program will deal correctly with the elements in I_2 and I_5. Here the word 'confidence' should not be interpreted as a statistical concept, but in the every-day meaning related to beliefs. Further, an essential step in programs for the triangle problem is calculation and comparison of A^2 with $B^2 + C^2$, to distinguish between right, obtuse and acute triangles. These triangles are in different input subdomains, but learning that the program deals correctly with one of them will increase confidence about these calculations and comparison, and therefore may also increase confidence about how the program deals with the other triangles. Thus, even for programs to deal with such a simple problem, there are several aspects shared by some or all subdomains, so the assumption that the subdomains are uncorrelated seems inappropriate.

Usually, one will model correlations between different subdomains by positive covariances c_{ih}, implying that learning that the program deals well with one input element increases confidence for other input elements, and seeing an incorrect output decreases confidence for other inputs. However, the model does allow negative correlations (implying that seeing a correct output decreases confidence for other inputs), but these will only reflect true beliefs about the way one learns from software testing in the unusual case where we already know that the software contains a certain given number of errors.

We will now analyse two different collections of assessed values for this example, to illustrate the possible use of Bayes linear methods for this partition testing problem. It is not our aim to present 'good assessed values' for this triangle problem, which would of course depend on knowledge about actual software. Instead, the examples below are intended to serve as a simple illustration of our approach, and also to illustrate how differences in our judgements about the software are translated into differences in the analysis of test outcomes.

Example 1

For the first example, assume that p_i and $p_{i|i}$ values have been assessed as given below (with the corresponding values v_i and c_i, calculated as indicated above, and $Var(R_{ij}) = v_i - c_i$):

i	p_i	$p_{i\|i}$	v_i	c_i	$v_i - c_i$
1	0.2	0.24	0.16	0.008	0.152
2	0.2	0.24	0.16	0.008	0.152
3	0.2	0.24	0.16	0.008	0.152
4	0.3	0.35	0.21	0.015	0.195
5	0.3	0.35	0.21	0.015	0.195
6	0.3	0.35	0.21	0.015	0.195
7	0.4	0.45	0.24	0.02	0.22
8	0.5	0.55	0.25	0.025	0.225
9	0.5	0.55	0.25	0.025	0.225

Assume the following matrix $P_{i|h}$, where the entry in row i and column h is $p_{i|h}$ (on the diagonal we have included, between brackets, $p_{i|i}$):

$$P_{i|h} = \begin{pmatrix} (0.24) & 0.21 & 0.21 & 0.21 & 0.21 & 0.21 & 0.21 & 0.21 & 0.21 \\ & (0.24) & 0.24 & 0.21 & 0.21 & 0.21 & 0.21 & 0.21 & 0.21 \\ & & (0.24) & 0.21 & 0.21 & 0.21 & 0.21 & 0.21 & 0.21 \\ & & & (0.35) & 0.35 & 0.35 & 0.31 & 0.31 & 0.31 \\ & & & & (0.35) & 0.35 & 0.31 & 0.31 & 0.31 \\ & & & & & (0.35) & 0.31 & 0.31 & 0.31 \\ & & & & & & (0.45) & 0.42 & 0.42 \\ & & & & & & & (0.55) & 0.55 \\ & & & & & & & & (0.55) \end{pmatrix}.$$

From the assessed values above the covariance matrix C can be easily derived, where the entry in row i and column h is c_{ih}, and the diagonal elements are c_i. This matrix is symmetric, so it is sufficiently specified as follows:

$$C = \begin{pmatrix} 0.008 & 0.002 & 0.002 & 0.003 & 0.003 & 0.003 & 0.004 & 0.005 & 0.005 \\ & 0.008 & 0.008 & 0.003 & 0.003 & 0.003 & 0.004 & 0.005 & 0.005 \\ & & 0.008 & 0.003 & 0.003 & 0.003 & 0.004 & 0.005 & 0.005 \\ & & & 0.015 & 0.015 & 0.015 & 0.004 & 0.005 & 0.005 \\ & & & & 0.015 & 0.015 & 0.004 & 0.005 & 0.005 \\ & & & & & 0.015 & 0.004 & 0.005 & 0.005 \\ & & & & & & 0.02 & 0.01 & 0.01 \\ & & & & & & & 0.025 & 0.025 \\ & & & & & & & & 0.025 \end{pmatrix}.$$

A close look at the above assessments shows that the tester is apparently quite pessimistic about the quality of the code, and careful with regard to the information provided by single tests. Such judgements might occur for a variety of reasons, for example for actual testing the code might be embedded in a larger system of which the tester doubts its quality. Further, some of the nine partition elements have effectively been put together, in that they jointly act as a single subdomain. This is the case for I_2, I_3, because $p_{2|3} = p_{2|2} = p_{3|3}$. Note that I_1 does not belong to this subdomain, even though $p_1 = p_2 = p_3$ and $p_{1|1} = p_{2|2} = p_{3|3}$, as $p_{1|2} \neq p_{1|1}$. Similarly, we may view (I_4, I_5, I_6) and (I_8, I_9) as single subdomains. So in fact we can regard these assessments as leading to a partition of five different subdomains, and we will see the obvious effects of this in the results below.

These assessed values represent rather strong doubts about the quality of the software at this moment (considering the p_i values), and also reflect beliefs that the inputs, even those in the same partition element, are quite heterogeneous (considering the $p_{i|i}$ and $p_{i|h}$ values). In this assessment, the c_i are much smaller than the corresponding v_i values, implying that only a small part of the entire variation is due to current lack of knowledge about the M_i values, as represented in $Var(M_i)$, about which we learn when taking observations (performing tests). One possible way to measure the effect of tests is by considering the reduction of each $Var(M_i)$ after adjusting on the basis of the test results, which is a convenient measure since it only depends on the number of observations [15]. We shall briefly discuss this effect for some possible test suits (numbers of inputs per partition element), and focus on which partition elements to take inputs from when the number of observations is fixed.

Before any test results are used to update the beliefs given above, the sum of the variances c_i for the nine M_i's equals 0.139. As a measure of how much we learn from our tests we use the ratio of the sum of the adjusted variances for the M_i's and the corresponding prior value 0.139. Let us denote this by RSV (ratio of the sum of the variances). If we can take only one single observation, then which of the nine partition elements should we choose? Note that we do not distinguish between inputs within the same partition element, and so we only have to choose the partition element, and can take any input within that subdomain.

The single observation from which we learn most, using the RSV as a measure, is achieved by testing one input from either I_8 or I_9, giving RSV=0.957, so a reduction of the sum of variances for the M_i's by 4.3 percent. The direct effect on the variances for the individual M_i's is as follows, where each value gives the ratio of adjusted variance and prior variance when observing an input from I_8 or I_9. (We call this the vector of the ratios of variances, VRV):

$$(0.988, 0.988, 0.988, 0.993, 0.993, 0.993, 0.980, 0.900, 0.900).$$

As expected, the learning effect (relative reduction in variance for M_i) is largest for I_8 and I_9 when applying a test of an input from these subdomains. But it also reduces the variances for the other M_i's, a feature that was not included in previous discussions of partition testing and that is the consequence of the assessed values $p_{i|h}$. The exact change on the expected values of the M_i's will of course depend on the outcome of the test, failure or non-failure. Since all covariances are positive, the changes in the expected values will be in the 'logical direction', in that an observed failure increases all failure probabilities, and an observed non-failure decreases all failure probabilities. At the end of this example, we briefly show the effect of tests without any failures on the expected values of the M_i's.

If we would take our only observation from I_1 instead, this would give RSV=0.993 and the following vector VRV of variance reduction ratios for the individual M_i's:

$$(0.950, 0.997, 0.997, 0.996, 0.996, 0.996, 0.995, 0.994, 0.994).$$

If we were allowed to take two observations, the best we could do would be to take both out of I_8 or I_9, leading to RSV=0.921 (so after testing two such inputs 92.1 percent of, say, 'explainable' variation would still be left) and VRV:

$$(0.977, 0.977, 0.977, 0.988, 0.988, 0.988, 0.964, 0.818, 0.818).$$

However, it would be almost equally effective to take one input out of I_4, I_5, I_6 and one out of I_8, I_9, leading to RSV=0.928 and VRV:

$$(0.982, 0.982, 0.982, 0.923, 0.923, 0.923, 0.977, 0.896, 0.896).$$

In many practical problems, it is more important to identify errors in some of the subdomains than in others. In such cases, we would choose tests to minimise a weighted sum of the adjusted variances, where the weights are chosen to reflect the importance of the different types of error. For example, in the above comparison, we would prefer one choice from (I_4, I_5, I_6) and one from (I_8, I_9) to both from (I_8, I_9) if we had put a slightly higher relative weight to the variance terms of (I_4, I_5, I_6) than those of (I_8, I_9).

If we can select three subdomains, and take one observation from each, the choice that minimises RSV would be one out of I_4, I_5, I_6, one out of I_8 and one out of I_9 (taking these last two from the same subdomain would not make any difference in this example, as mentioned before), leading to RSV=0.899 and VRV:

$$(0.973, 0.973, 0, 973, 0.918, 0.918, 0.918, 0.960, 0.815, 0.815).$$

In contrast, if we were to take, for example, one observation from each of I_2, I_4 and I_8, this would give RSV=0.921 and VRV:

$$(0.980, 0.934, 0.934, 0.920, 0.920, 0.920, 0.972, 0.891, 0.891).$$

Similarly, if we can select six subdomains and take one observation from each, the choice minimising RSV would be to take one input out of each of $I_4, \ldots, I_9$, with RSV=0.842 and VRV:

$$(0.958, 0.958, 0.958, 0.802, 0.802, 0.802, 0.879, 0.799, 0.799),$$

whereas, for example, one observation out of every subdomain except I_3, I_6, I_9 would give RSV=0.878 and VRV:

$$(0.922, 0.922, 0.922, 0.854, 0.854, 0.854, 0.887, 0.871, 0.871),$$

which resolves less total variance but reduces all these individual variances more equally.

The effect of more observations per subdomain can be seen when considering taking, for example, 10 or 100 observations per selected subdomain; remember that the subdomains are far from homogeneous or 'revealing' in this example. This could be of practical interest if there are actual experimental costs related to testing per subdomain, but we do not consider such matters any further and use this for illustration only. If we are allowed to select six subdomains, and take 10 inputs out of each of these six, the best we could do to minimise RSV is to drop the subdomains I_1, I_2, I_4 (where I_2 can be replaced by I_3, and I_4 by I_5 or I_6 with the same results), leading to RSV=0.417 and VRV:

$$(0.863, 0.596, 0.596, 0.382, 0.382, 0.382, 0.475, 0.295, 0.295).$$

If we could select six subdomains while being allowed 100 observations per subdomain, we could drop I_3, I_6, I_8 as one of several options leading to minimal value RSV=0.086 and VRV:

$$(0.154, 0.154, 0.154, 0.060, 0.060, 0.060, 0.096, 0.080, 0.080).$$

In this example so far, using RSV as design criterion, two aspects related to the variances of the M_i's play an important role. First, reduction of variance is preferred for those variances that are large in the prior stage, which leads to subdomains I_8 or I_9 included in all the optimal choices. Secondly, although some of the subdomains (like I_8, I_9) are closely related, so that in effect they could be reduced to one subdomain in this example, this is not taken into account in the RSV, which just takes the sum of adjusted variances for the M_i's even though some of the M_i's are strongly correlated.

Farrow and Goldstein [7] describe a criterion that can be used to avoid this second problem. In our approach we are basically learning about the expected values for $M_i, i = 1, \ldots, 9$, and since expectation is a linear function this implies learning about all linear combinations (weighted sums) of these M_i's. If the M_i are correlated, as in our example, linear combinations of the M_i's can be chosen that are uncorrelated, with prior variance one. Since these variables are uncorrelated, the sums of their variances are more informative measures to judge the effect of updating the collection by new information. We will study the effect of using this alternative criterion in relation to software testing in future work, related to practical applications.

For the given assessments in this example the suggested tests could be improved by allowing different numbers of observations per subdomain; such choices are straightforward to identify [15].

We finish this first example by considering the effect of tests without any failures on the expected values of the M_i's, when a single input is tested per subdomain. To see the effect, Table 1 gives the adjusted expected values $E_D(M_i)$ for the M_i for nine situations; each $E_D(M_i)$ is equal to the adjusted probability that a randomly selected element of the subdomain will fail. First we consider only testing a single input in I_9, then one in both I_8 and I_9, and adding one subdomain at a time, finally we consider the adjusted expected values when testing one input from every subdomain (and observing no failures). For comparison with the expected values before testing, these are given again in the first line.

tested	$E_D(M_1)$	$E_D(M_2)$	$E_D(M_3)$	$E_D(M_4)$	$E_D(M_5)$	$E_D(M_6)$	$E_D(M_7)$	$E_D(M_8)$	$E_D(M_9)$
prior	0.200	0.200	0.200	0.300	0.300	0.300	0.400	0.500	0.500
I_9	0.190	0.190	0.190	0.290	0.290	0.290	0.380	0.450	0.450
$I_8 - I_9$	0.182	0.182	0.182	0.282	0.282	0.282	0.364	0.409	0.409
$I_7 - I_9$	0.176	0.176	0.176	0.276	0.276	0.276	0.334	0.397	0.397
$I_6 - I_9$	0.173	0.173	0.173	0.257	0.257	0.257	0.330	0.391	0.391
$I_5 - I_9$	0.169	0.169	0.169	0.240	0.240	0.240	0.326	0.387	0.387
$I_4 - I_9$	0.167	0.167	0.167	0.225	0.225	0.225	0.323	0.383	0.383
$I_3 - I_9$	0.165	0.159	0.159	0.223	0.223	0.223	0.319	0.379	0.379
$I_2 - I_9$	0.163	0.151	0.151	0.221	0.221	0.221	0.316	0.375	0.375
$I_1 - I_9$	0.156	0.150	0.150	0.218	0.218	0.218	0.313	0.371	0.371

Table 1: *Adjusted failure probabilities for testing with no observed failures (example 1)*

The effects of the positive correlations between different subdomains is clearly seen in Table 1, as testing one input in any subdomain affects our beliefs in all subdomains. In particular, as all correlations are positive in this example, a test revealing no failures makes us more optimistic for all subdomains in the sense that our probabilities for failures decrease.

Example 2

For the second example, we use the same values as in example 1, except for the $p_{i|h}$ and the covariances, which we assume to be:

$$P_{i|h} = \begin{pmatrix} (0.24) & 0.23 & 0.23 & 0.20 & 0.20 & 0.20 & 0.20 & 0.20 & 0.20 \\ & (0.24) & 0.23 & 0.20 & 0.20 & 0.20 & 0.20 & 0.20 & 0.20 \\ & & (0.24) & 0.20 & 0.20 & 0.20 & 0.20 & 0.20 & 0.20 \\ & & & (0.35) & 0.30 & 0.30 & 0.30 & 0.30 & 0.30 \\ & & & & (0.35) & 0.33 & 0.30 & 0.30 & 0.30 \\ & & & & & (0.35) & 0.30 & 0.30 & 0.30 \\ & & & & & & (0.45) & 0.41 & 0.41 \\ & & & & & & & (0.55) & 0.53 \\ & & & & & & & & (0.55) \end{pmatrix}.$$

The corresponding covariance matrix C is:

$$C = \begin{pmatrix} 0.008 & 0.006 & 0.006 & 0 & 0 & 0 & 0 & 0 & 0 \\ & 0.008 & 0.006 & 0 & 0 & 0 & 0 & 0 & 0 \\ & & 0.008 & 0 & 0 & 0 & 0 & 0 & 0 \\ & & & 0.015 & 0 & 0 & 0 & 0 & 0 \\ & & & & 0.015 & 0.009 & 0 & 0 & 0 \\ & & & & & 0.015 & 0 & 0 & 0 \\ & & & & & & 0.02 & 0.005 & 0.005 \\ & & & & & & & 0.025 & 0.015 \\ & & & & & & & & 0.025 \end{pmatrix}.$$

This example differs from example 1 in the assessed values for $p_{i|h}$, so in the correlations between different subdomains. Although there are again groups of subdomains that are strongly correlated (I_1, I_2, I_3 and I_5, I_6 and I_8, I_9, with some correlation between these last two and I_7), they could not be replaced by a single subdomain as the correlations between inputs within the same subdomain are greater than between inputs from different subdomains. Further, many subdomains are uncorrelated, whereas in example 1 there was always some positive correlation between all subdomains. The obvious effects of choosing some subdomains to be uncorrelated is easily seen in the following results.

Suppose that we can only take a single observation from one subdomain. As before, I_8 minimises RSV, leading to RSV=0.978 and VRV=$(1, 1, 1, 1, 1, 1, 0.995, 0.900, 0.964)$. Observe that testing one input in I_8 now does not tell us anything about $I_1, \ldots, I_6$.

With two observations, the choice minimising RSV is again I_8 and I_9, with RSV=0.950 and VRV=$(1, 1, 1, 1, 1, 1, 0.991, 0.871, 0.871)$. If we can select three subdomains (one observation from each) then we should again include I_8 and I_9, and any of the other seven subdomains can be added as they all lead to (approximately) the same RSV. For example, if we add I_1 then RSV=0.942 and VRV=$(0.950, 0.972, 0.972, 1, 1, 1, 0.991, 0.871, 0.871)$. Note that the observation I_1 tells quite a bit about I_1, less about I_2, I_3 and nothing about the other subdomains, again due to the uncorrelated subdomains in this assessment. If we can select one observation from each of six subdomains, dropping I_1, I_2, I_5 is one of several choices leading to minimal RSV=0.914, with VRV=$(0.972, 0.972, 0.950, 0.929, 0.974, 0.929, 0.909, 0.868, 0.868)$.

Remaining with six subdomains, but allowing 10 observations per subdomain, the same possible choices all still perform well, for example dropping I_1, I_2, I_5 now leads to the minimal RSV=0.568 and VRV=$(0.806, 0.806, 0.655, 0.565, 0.843, 0.565, 0.513, 0.423, 0.423)$.

Finally, again we briefly present adjusted expected values for M_i in case of no failures in testing. As in Example 1 (Table 1), we assume a single input tested per subdomain, the results are given in Table 2.

tested	$E_D(M_1)$	$E_D(M_2)$	$E_D(M_3)$	$E_D(M_4)$	$E_D(M_5)$	$E_D(M_6)$	$E_D(M_7)$	$E_D(M_8)$	$E_D(M_9)$
prior	0.200	0.200	0.200	0.300	0.300	0.300	0.400	0.500	0.500
I_9	0.200	0.200	0.200	0.300	0.300	0.300	0.390	0.470	0.470
$I_8 - I_9$	0.200	0.200	0.200	0.300	0.300	0.300	0.381	0.425	0.425
$I_7 - I_9$	0.200	0.200	0.200	0.300	0.300	0.300	0.350	0.418	0.418
$I_6 - I_9$	0.200	0.200	0.200	0.300	0.287	0.279	0.350	0.418	0.418
$I_5 - I_9$	0.200	0.200	0.200	0.300	0.267	0.267	0.350	0.418	0.418
$I_4 - I_9$	0.200	0.200	0.200	0.279	0.267	0.267	0.350	0.418	0.418
$I_3 - I_9$	0.193	0.193	0.190	0.279	0.267	0.267	0.350	0.418	0.418
$I_2 - I_9$	0.186	0.183	0.183	0.279	0.267	0.267	0.350	0.418	0.418
$I_1 - I_9$	0.177	0.177	0.177	0.279	0.267	0.267	0.350	0.418	0.418

Table 2: *Adjusted failure probabilities for testing with no observed failures (example 2)*

The results in Table 2 are similar to those in Table 1, but now the difference is clear for subdomains that are not correlated, testing in one subdomain only influences our beliefs for that particular subdomain and the ones that are correlated with it. For example, a correct output in subdomain I_9 decreases our beliefs of failure for I_7, I_8 and I_9, but not for the other subdomains. This is also shown by the fact that the probability of a failure in I_9 is not changed anymore after the third line in Table 2, as testing in any of I_1 to I_6 does not influence our beliefs for I_9 in this example.

6 Discussion

This paper introduces Bayes linear methods as a statistical tool for software testing, building on, and extending, the theory of partition testing as presented in literature. Essential for acceptance of powerful statistical methods to support software testing is acknowledgement of the fact that quality of software is uncertain before testing, and remains uncertain after testing. The main goal of testing, therefore, is reduction of uncertainty, and statistical methods are useful for designing effective tests and analysing the effect that tests have on uncertainty. We have argued that Bayes linear methods are suitable for many natural goals in software testing, but this can only be justified by practical applications. In this section we briefly discuss some topics which we think are important for practical software testing, and we indicate how Bayes linear methods could be used to support software testing.

A fairly general cycle of software testing exists, which at each stage consists of

1. *Assess objectives:* It is likely that there are different objectives for testing at different stages in a software development process. In the early stages of development, testing may give general insights into the software, and be used to identify which parts seem to lead to many errors. In a later stage, more detailed testing may be aimed at finding inputs that the software does not deal with correctly in an attempt to relate failures to faults and try to improve the software. In a stage close to release, emphasis may particularly be on building confidence that the software will deal well with its tasks after release. Different objectives should be reflected by different utilities in the decision process that underlies the choice of test suites.

2. *Identify main sources of uncertainty and information:* Given that testing is aimed at reducing uncertainty, an obvious task is to obtain a good overview of the uncertainty involved in the software and its applications. Closely related to this is careful analysis of information that has an effect on our beliefs, which may result from e.g. earlier tests, experiences in related software development processes, use of earlier versions of the software and many other possible sources.

3. *Assess the exchangeability structure:* Deriving a meaningful partition of the input domain is of great importance. For black-box testing, the partition is necessarily based on the tester's judgement about the tasks that the software must perform. If actual information on the code is available, this can be used to assist in definition of the partition.

4. *Elicit the qualitative uncertainty structure:* A powerful method for representing the uncertainties of the tester is through a Bayesian graphical model, for which the nodes represent uncertain random quantities, joined by arrows representing probabilistic relationships between the uncertain quantities. For a practical account of the construction of such models to support Bayes linear analyses see Farrow, *et al.* [8]. For partition testing, the child nodes correspond to outcomes of tests within subdomains, and the parent nodes might correspond to possible sources of error, which the graphical model would share across the subdomains. When dealing with black-box testing, we have found it helpful to construct pseudo-code representing the tester's view of the likely processes carried out by software in order to jointly identify the partition and the sources of error which form the basis of

the graphical model. We are currently investigating ways of linking such tester's diagrams to information that might be available from UML representations of the code [21], using for example functional diagrams to relate sources of error and category diagrams to help identify the partition.

5. *Quantify the probabilities:* Quantifying the probabilities, given an assumed exchangeability structure, is also of great importance, and must necessarily be studied in real-world applications. The Bayesian graphical model offers a systematic approach to quantifying the probabilities required for the analysis. Each source of error is given an uncertainty value, based on expert judgements related to the complexity of the task, information on previous causes of error in similar software, details of prior test results by the software development team, and so forth. These uncertainties are propagated to the partition outcomes through the formal structure of the model.

6. *Develop test suite and perform test:* Powerful statistical methods should be used to assist in the testing process, indicating useful inputs to be tested on the basis of current uncertainty and explicitly stated objectives. Such methods should be able to deal with many constraints, e.g. related to time and financial resources, that will naturally influence practical testing opportunities.

7. *Analyse the results:* Statistical methods used to assist software testing should also enable analysis of the testing results, which may be an end-point of the process but will often be an intermediate stage, after which the cycle may be repeated with altered objectives. The approach should enable analysis within reasonably short periods of time, allowing sequential testing when needed even under severe pressure, for example by fixed target release dates.

8. *Sensitivity analysis:* While the analyses that we have described were carried out with precise probability specifications, we will usually investigate the effect on the analysis of varying these specifications, in order to assess how much more optimistic or pessimistic we would have to be about the quality of the software, a priori, to substantially alter our conclusions. Such sensitivity analysis is most usefully carried out by varying the inputs into the associated Bayesian graphical model.

The Bayes linear approach supports this software testing process by

- Giving qualitative structures to represent our uncertainties; e.g. exchangeability [8], or by graphical modelling [18, 20].

- Providing elicitation methods to quantify uncertainty [2, 4].

- The possibility to exploit decision analysis based on utilities which suggests criteria for the current stage of testing [6, 15].

- Enabling analysis of test results in large-scale applications, and offering diagnostic tools [7, 14].

We are involved in collaborative work with an industrial partner, using Bayes linear methods to support testing of software in a telecommunications environment. In this work, the above mentioned cycle is recognized and all stages are carefully considered. This work will not only be important as an example of the ability of Bayes linear methods to deal with a problem as complex as practical software testing, but it will also raise many interesting related research topics.

References

[1] Bernardo, J.M. and Smith, A.F.M. (1994). *Bayesian Theory.* Wiley, Chichester.

[2] Cooke, R.M. (1991). *Experts in Uncertainty.* Oxford University Press.

[3] Craig, P.S., Goldstein, M., Seheult, A.H. and Smith, J.A. (1997). Pressure matching for hydrocarbon reservoirs: a case study in the use of Bayes linear strategies for large computer experiments (with discussion). In: *Case Studies in Bayesian Statistics*, C. Gatsonis, J.S. Hodges, R.E. Kass, R. McCulloch, P. Rossi and N.D. Singpurwalla (eds.). Springer, New-York, 37-93.

[4] Craig, P.S., Goldstein, M., Seheult, A.H. and Smith, J.A. (1998). Constructing partial prior specifications for models of complex physical systems. *The Statistician*, **47**, 37-53.

[5] De Finetti, B. (1974). *Theory of Probability* (2 volumes). Wiley, Chichester.

[6] Farrow, M. and Goldstein, M. (1992). Reconciling costs and benefits in experimental design. In: *Bayesian Statistics 4*, J.M. Bernardo, J.O. Berger, A.P. Dawid and A.F.M. Smith (eds.). Oxford University Press, 607-615.

[7] Farrow, M. and Goldstein, M. (1993). Bayes linear methods for grouped multivariate repeated measurement studies with application to crossover trials. *Biometrika*, **80**, 39-59.

[8] Farrow, M., Goldstein, M. and Spiropoulos, T. (1997). Developing a Bayes linear decision support system for a brewery. In: S. French and J.Q. Smith (eds.), *The Practice of Bayesian Analysis.* Edward Arnold, 71-106.

[9] Gelman, A., Carlin, J.B., Stern, H.S. and Rubin, D.B. (1995). *Bayesian Data Analysis.* Chapman & Hall, London.

[10] Goldstein, M. (1986). Exchangeable belief structures. *Journal of the American Statistical Association*, **81**, 971-976.

[11] Goldstein, M. (1994). Revising exchangeable beliefs: subjectivist foundations for the inductive argument. In: *Aspects of Uncertainty*, P.R. Freeman and A.F.M. Smith (eds.). Wiley, Chichester, 201-222.

[12] Goldstein, M. (1996). Prior inferences for posterior judgements. In: *Proceedings of the 10th International Congress of Logic, Methodology and Phylosophy of Science*, M.L.D. Chiara, *et al* (eds.). Kluwer.

[13] Goldstein, M. (1999). Bayes linear analysis. In: *Encyclopaedia of Statistical Sciences*, update volume 3, S. Kotz, C.B. Read and D.L. Banks (eds.). Wiley, New York, 29-34.

[14] Goldstein, M. and Wooff, D.A. (1995). Bayes linear computation: concepts, implementation and programs. *Statistics and Computing*, **5**, 327-341.

[15] Goldstein, M. and Wooff, D.A. (1997). Choosing sample sizes in balanced experimental designs: a Bayes linear approach. *The Statistician*, **46**, 167-183.

[16] Hamlet, D. (1992). Are we testing for true reliability?. *IEEE Software*, July 1992, 21-27.

[17] Hamlet, D. and Taylor, R. (1990). Partition testing does not inspire confidence. *IEEE Transactions on Software Engineering*, **16**, 1402-1411.

[18] Lauritzen, S.L. (1996). *Graphical Models*. Oxford University Press.

[19] Myers, G. (1979). *The Art of Software Testing*. Wiley, New York.

[20] Oliver, R.M. and Smith, J.Q. (eds.) (1990). *Influence Diagrams, Belief Nets and Decision Analysis*. Wiley, London.

[21] Stevens, P. and Pooley, R. (2000). *Using UML; Software Engineering with Objects and Components* (updated version). Addison-Wesley, Harlow, England.

[22] Weyuker, E.J. and Jeng, B. (1991). Analyzing partition testing strategies. *IEEE Transactions on Software Engineering*, **17**, 703-711.

[23] Weyuker, E.J. and Ostrand, T.J. (1980). Theories of program testing and the application of revealing subdomains. *IEEE Transactions on Software Engineering*, **6**, 236-246.