On the role of statistical significance in exploratory data analysis

Inderpal Bhandari and Shriram Biyani
IBM T. J. Watson Research Center
PO Box 704
Yorktown Heights, NY 10598, USA
e-mail: isb@watson.ibm.com

Abstract

Recently, an approach to knowledge discovery, called Attribute Focusing, has been used by software development teams to discover such knowledge from categorical defect data as allows them to improve their process of software development in real time. This feedback is provided by computing the difference of the observed proportion within a selected category from an expected proportion for that category, and then, by studying those differences to identify their causes in the context of the process and product. In this paper, we consider the possibility that some differences may simply have occurred by chance, i.e., as a consequence of some random effect in the process generating the data. We develop an approach based on statistical significance to identify such differences. Preliminary, empirical results are presented which indicate that knowledge of statistical significance should be used carefully when selecting differences to be studied to identify causes. Conventional wisdom would suggest that all differences that lie beyond some small level of statistical significance be eliminated from consideration. Our results show that such elimination is not a good idea. They also show that information on statistical significance can be useful in the process of identifying a cause.

Keywords: data exploration, knowledge discovery, statistical significance, software engineering.

1 Introduction

Knowledge discovery in databases [1] often involves the identification of statistical relationships in data as a first step towards the discovery of knowledge. The main problem with such identification is that in a large data set there usually are very many statistical relationships that exist but which are quite useless in the context of knowledge discovery. For instance, some of those relationships will be well known, and hence, not be meaningful in the context of knowledge discovery. Those relationships can be screened for and removed by incorporating domain knowledge in the knowledge discovery system [2]. Other relationships will reflect purely statistical relationships, i.e., relationships that occur by chance because of random effects in the underlying processes used to generate and collect the data. Such chance relationships are undesirable as they may lead to the discovery of spurious knowledge. Those relationships can be screened for and removed by using methods of statistical validation The identification of relationships that occur by chance has always been a concern in statistical analysis, specifically, in the arena of hypothesis testing ([3], Chapter 6). The approach used to eliminate such relationships is based on determining the extent to which an observed relationship was produced by chance. One uses that extent to determine whether to accept or to reject that relationship. Clearly, a similar strategy can be used in knowledge discovery to remove chance relationships, and, it often is recommended in the literature [2, 4, 5].

This paper suggests that such a strategy be used carefully since the circumstances under which hypothesis testing is done in the context of knowledge discovery may be different from conventional applications of such testing. Hence, it may not be advisable to follow the dictates of conventional wisdom. In particular, we study the use of statistical significance in the context of the application of an approach to knowledge discovery called Attribute Focusing to software engineering. By utilizing data from a fielded application, it is shown that use of statistical significance can eliminate useful relationships, i.e., those that are in fact representative of new knowledge. Since extensive field work on applying attribute focusing to the software engineering domain has shown that the number of patterns that lead to discovery per set of data is small - an average of two to three patterns - such elimination is not desirable. Thus, our results show that one cannot simply make use of statistical significance as is conventionally accepted but instead one must understand how it should and should not be incorporated in attribute focusing.

2 Background on software process improvement

We begin by providing the necessary background on software engineering. The software production process [6, 7] provides the framework for development of software products. Deficiencies in the activities that make up the process lead to poor quality products and large production cycles. Hence, it is important to identify problems with the definition

or execution of those activities.

Software production results in defects. Examples of defects are program bugs, errors in design documents, etc.. One may view defects as being manifest by deficiencies in the process activities, hence, it is useful to learn from defect data. Such learning is referred to as *defect-based process improvement.* If it occurs during the course of development, it is referred to as *in-process* improvement. Else, if it occurs after development is completed, it is referred to as *post-process* improvement. In many production laboratories, software defects are classified by the project team in-process to produce attribute-valued data based on defects [8, 9, 10]. Those data can be used for defect-based process improvement.

Recently, Bhandari, in [11], introduced a method for exploratory data analysis called Attribute Focusing. That method provides a systematic, low-cost way for a person such as a software developer or tester, who is not an expert at data analysis, to explore attribute-valued data. A software development project team can use Attribute Focusing to explore their classified defect data and identify and correct process problems in real time. There have been other efforts in the past that focused on data exploration and learning from data but they do not focus on providing real-time feedback to a project team. We mention them for the sake of completeness. Noteworthy examples are the work by Selby and Porter [12] and the work on the Optimized Set Reduction Technique by Briand and others [13].

The application of attribute focusing to process improvement was fielded in mid 1991. Major software projects within IBM started using the method in acutal production of software. By year-end 1992, there were over 15 projects that were using the method. The extensive field experience has allowed us to document evidence [14, 15, 16, 17] that Attribute Focusing (AF) has benefitted many different development projects within IBM over and beyond other current practices for both in-process and post-process improvement. In [14], Bhandari and Roth reported on the experience of using AF to explore data collected from a survey based on system test and early field defects. In [15], Chaar, Halliday, et al, utilized AF to explore defect data based on the four attributes in the Orthogonal Defect Classification scheme [10] and other commonly used attributes to assess the effectiveness of inspection and testing. In [16], the experience of using AF to explore such data to make in-process improvements to the process of a single project was described, while in [17], such improvement was shown to have occurred for a variety of projects that covered the range of software production from mainframe computing to personal computing.

We will make use of the above-mentioned field experience later on. For the moment, we seek a high-level understanding of the main topic of this paper. Attribute focusing makes use of functions to highlight parts of the categorical data. Those functions have the form:

$$I(x) = O(x) - E(x) \quad (1)$$

where O(x) represents what was observed in the data for category x while E(x) represents what was expected for that category. Equation 1 computes a difference between two distributions. In the case of Attribute Focusing, such differences are used to call the attention of the project team to the relationship $O(x)$. The team determines whether those relationships are indicative of hidden process problems, and if so, determines how those problems must be resolved.

In this paper, we consider the possibility that some differences may simply have occurred by chance, i.e., as a consequence of some random effect in the software development process. There are many possible sources in a complicated process such as software development which could contribute to such an effect. A simple example is the process of classifying data itself. A person could misclassify defects by simple oversight, which could be modeled mathematically by a random process.

After presenting background information (Section 3), we discuss the effect of such chance-based differences on Attribute Focusing (Section 4). We develop an approach based on statistical significance to determine the degree to which a difference could have occurred by chance (Section 5). Preliminary, empirical results are presented and used to illustrate how that approach should and should not be incorporated in Attribute Focusing (Section 6). We discuss the general implications of our results, and, in conclusion, summarize the limitations and lessons of this paper (Section 7).

3 Attribute Focusing

To make this paper self contained, we present background information on Attribute Focusing and how it is used to explore classified defect data. The method is used on a table of attribute-valued data. Let us describe how such a table is created from defect data.

Recall (Section 2) that defects are classified by the project team. The classification of defects results in a table of attribute-valued data. Let us illustrate what such a table looks like. We begin by reviewing some attributes which are being used at IBM to classify defects. Those attributes are the same as or modifications of attributes that have been reported in the past in the considerable literature which exists on defect classification schemes.

The attributes *missing/incorrect* and *type* capture information on what had to be done to fix the defect after it was found. *Missing/incorrect* has two possible values which may be chosen, namely, *missing* and *incorrect.* For instance, a defect is classified *missing* if it had to be fixed by adding something new to a design document, and classified *incorrect* if it could be fixed by making an in-place correction in the document. *Type* has eight possible values. One of those values is *function.* The *type* of a defect is chosen to be *function* if it had to be fixed by correcting major product functionality. The attribute

trigger captures information about the specific inspection focus or test strategy which allowed the defect to surface. One of its possible values is *backward compatibility*. The *trigger* of a defect found by thinking about the compatibility of the current release to the previous releases is chosen to be *backward compatibility*. The attribute *component* is used to indicate the software component of the product in which the defect was found. Its set of possible values is simply the set of names of the components which make up the product.

The software development process consists of a sequence of major activities such as design, code, test, etc.. After every major activity the set of defects that are generated as a result of that activity are first, classified, and then, explored by the project team using Attribute Focusing. A set of classified defects may be represented by an attribute-valued table. Every row represents a defect and every column represents an attribute of the classification scheme. The value at a particular location in the table is simply the value chosen for the corresponding attribute to classify the corresponding defect for that location. Table 1 illustrates a single row of such a table. The names of the attributes are in upper case and the values of those attributes chosen for the defect are in lower case. The single row in the table represents a defect which had to fixed by introducing new product functionality, was found in a component called *India* and detected by considering the compatibility of the product with previous releases.

Once an attribute-valued table has been created from the defect data, a project team can use Attribute Focusing to explore the data to identify problems with their production process. That exploration has two steps. The first step utilizes automatiq procedures to select patterns in the data which are called to the attention of the team. Next, the patterns are studied by the project team to identify process problems. Let us review those steps in turn.

3.1 Computing Differences

Patterns in the data are called to the attention of the project team based on the differences between observed statistics and theoretical distributions. Those differences are defined by two functions, I_1 and I_2, which are discussed in turn below.

$$I_1(X = a) = p(X = a) - 1/Choice(X) \quad (2)$$

where X is an attribute, a is a possible value for that attribute, $p(X = a)$ is the proportion of rows in the data for which $X = a$, and $Choice(X)$ is the total number of possible values that one may choose for X. I_1 is used to produce an ordered list of all attribute-values. Table 2 shows a part of such an ordering.

The table shows the observed and expected proportions and their difference for two attribute-values as computed by I_1. The column *Attribute-value* corresponds to $X = a$ in Equation 2, while *Observed* corresponds to $p(X = a)$, *Expected* corresponds to $1/Choice(X = a)$ and *Difference* corresponds to $I_1(X = a)$. Thus, we see that 27% of the defects were classified *function* and 44% of the defects were classified *missing*. The expected proportions are computed based on what the proportions of the attribute-values would be if their respective attributes were uniformly distributed. For instance, *function* is a value for the atttribute *type*, which has eight possible values. Hence, its expected proportion is 12.5%. *Missing* is a value of the attribute *Missing/Incorrect* which has two possible values. Therefore, the expected proportion is 50%. The column *difference* in the table simply computes the difference between the *observed* and *expected* column for every row in the table. The table is ordered by the absolute value of that difference. A similar table is computed based on I_2 (*Equation 3*) as discussed below.

$I_2(X = a, Y = b) = p(X = a, Y = b) - p(X = a) * p(Y = b)$, where $p(X = a)$ is the proportion of defects in the data for which $X = a$, while $p(X = a, Y = b)$ is the proportion of records in the data for which $X = a$ and $Y = b$. I_2 is used to produce an ordered list of all possible pairs of attribute values that can be chosen for a defect. Table 3 illustrates a part of such an ordering.

The columns *value1, value2* denote attribute-values. For brevity, we have not omitted the attributes themselves and only specified the values of interest. *Value1, value2* together define the category of defects represented by a row. For instance, the first row has information about those defects for which the attribute *type* was classified *function* and the attribute *missing/incorrect* was classified *missing*. The second row has information about those defects for which the attribute *component* was classified *India* and the attribute *trigger* was classified *backward compatibility*.

Let us understand the information in the first row. *obs1* specifies that 27% of the defects were classified *function*. *obs2* specifies that 44% of the defects were classified *missing*. *obs12* indicates that 15% of the defects were classified both *function* and *missing*. *Expec12* is simply the product of *obs1* and *obs2*, as per the I_2 function. Thus the expected value is computed based on what the proportion of defects that were classified using two attribute-values would be if the selection of those attribute-values was statistically independent. *Diff* is the difference of *obs12* and *expec12*. The table is ordered by the absolute value of that difference.

Equations 2 and 3 can both be cast in the form of Equation 1. Note that in both cases the expected distributions are based on well-known theoretical concepts. E_1 is based on the concept of uniform distribution while I_2 is based on statistically independent distributions. To appreciate the exploratory nature of I_1 and I_2, note that the choice of those expected distributions is not based on experience with software production, but is based instead on simple information-theoretic arguments. An attribute will convey maximum information if we expect it

TABLE 1

MISSING/INCORRECT	TYPE	TRIGGER	COMPONENT
missing	function	back. compatibility	India

TABLE 2

Attribute-value	Observed	Expected	Difference
Type= function	27%	12.5%	14.5%
Missing/Incorrect= missing	44%	50%	6%

to have a uniform distribution, while two attributes will convey maximum information if their expected distributions are statistically independent. To get a quick appreciation of that argument, consider the converse case. If a single value is always chosen for an attribute, we may as well discard that attribute. It is useless from the point of view of presenting new information. Similarly, if two attributes are perfectly related so that we can predict the value of one knowing the value of the other, we may as well discard one of those attributes.

Continuing with the description of Attribute Focusing, I_1 and I_2 produce orderings of attribute-values and pairs of attribute-values respectively. The top few rows of those orderings are presented to the project team for interpretation. The selection of those items is based on the amount of time the project team is willing to spend interpreting the data. That time is usually around 2 hours for a set of data that has been generated. As we shall see, the team uses a specified model to interpret the data that is presented to them. Hence, we can calibrate the time taken to interpret items in the data and determine how many rows should be presented to the team for each ordering.

For the purpose of this paper, we omit the details of calibration and simplify the details of presentation and interpretation. We will assume that there are two tables presented to the team, one table based on I_1 and one table based on I_2; Both tables have been appropriately cut-off as explained above, and that the team interprets a table, one row at a time. In actuality, the team interprets a collection of smaller tables which when combined would result in the larger tables based on I_1 or I_2. The team will also often consider the relationships between items in different rows that are presented to them. All those details are adequately covered in [11, 14, 16, 17]. For the purpose of this paper it is the total number of items which are interpreted that is important and not the form those items are presented in, or the fashion in which they are interpreted. Hence, we use the simplified description above.

3.2 Interpretation of data

The team interprets the information about the attribute-values in each table, one row at a time. A row in the tables based on I_1 or I_2 presents information about the magnitude or association of attribute-values respectively. We will refer to that information as an *item*. The team relates every item to their product and process by determining the *cause* and *implication* of that magnitude or association, and by *corroborating* their findings. The implication is determined by considering whether that magnitude or association signifies an undesirable trend, what damage such a trend may have done up to the current time, and what effect it may have in the future if it is left uncorrected. The cause is determined by considering what physical events in the domain could have caused that magnitude or association to occur. The cause and implication are addressed simultaneously, and are corroborated by finding information about the project that is not present in the classified data but which confirms that the cause exists and the implication is accurate. Details of the interpretation process used by the team may be found in [17, 16]. We will not replicate those details here but will illustrate the process of interpretation by using the items in Table 3. Those items are taken from a project experience that was the subject of [16].

Example 1: The first item in Table 3 shows an association which indicates that defects which were classified *function* also tended to be classified *missing*. Recall, that *function* defects have to be fixed by making a major change to functionality, and *missing* defects have to be fixed by adding new material as opposed to an in-place correction. The implication is that the fixes will be complicated, quite possibly introducing new defects as new material is added. Hence, this is an undesirable trend which should be corrected. The team also determined that the cause of the trend was that the design of the recovery function, the ability of the system to recover from an erroneous state, was incomplete. The tex-

TABLE 3

value1	value2	obs1	obs2	obs12	expec12	diff
function	missing	27%	44%	15%	12%	3%
india	back. compat.	39%	10%	2%	4%	-2%

tual descriptions of the defects that were classified *function* and *missing* were studied in detail. It was found that all such defects pertained to the recovery function, hence, corroborating the cause.

Example 2: The second item in Table 3 shows a disassociation which indicates that defects which were classified *India* did not tend to be classified *backward compatibility*. The team determined that the cause of the trend was that issues pertaining to compatibility with previous releases had not been addressed adequately in the design of component *India*. In other words, since they had missed out on an important aspect of the design of the component, there were few defects to be found that pertained to that aspect The implication was that, if the design was not corrected, existing customer applications would fail after the product was shipped. The existence of the problem was corroborated by noting the lack of compatibility features in the design of component *India*, the existence of the compatibility requirement for the product, and the consensus amongst the designers in the team that component *India* should address that requirement.

The project team goes through the tables item by item in the above fashion. Items for which the implication is undesirable and the cause is determined lead to corrective actions, which are usually implemented before proceeding to the next phase of development. Items for which the implication is undesirable but the cause cannot be determined in the interpretation session are investigated further at a later time by examining the detailed descriptions of the relevant defects (that were classified using the attribute-values in the item under investigation). If the cause is still not found after such examination, the items are not considered any further. We will say that such items have been *dismissed*.

4 The effects of chance

Let us consider what would happen if an item in the tables presented to the team had occurred purely by chance. It is certainly possible that item may have an undesirable implication, since the implication is often determined by considering the meanings of the attribute values in the item. In Example 1 (Section 3.2), the meanings of the words *function* and *missing* were used to conclude that the trend was undesirable. Let us examine what would happen if the relevant association for that example, the first item in Table 3, was the result of a chance effect and did not have a physical cause ?

Since the implication was undesirable, the team would investigate that trend. Therefore, an obvious adverse effect of the chance occurrence is that it would waste the time of the team.

What would be the result of that investigation ? There are two possibilities, a cause would be found or a cause would not be found. We discuss each possibility in turn to determine how the process of interpretation could be improved to address chance effects.

Let us examine how a cause could be found for such an item. If a cause is identified, it implies that a mistake was made by the team since we have assumed that the item occurred by chance and had no physical cause. Such a mistaken cause could not be corroborated unless a mistake was made during the corroboration as well. Finally, both those mistakes, in cause identification and cause corroboration, have to be subtle enough to fool all the members of the team who were interpreting the item. Under that circumstance a cause could be identified mistakenly with possibly adverse consequences for the project. Hence, we conclude that the identification of a cause by mistake is unlikely but not impossible. To further reduce the chance of such a mistake occurring, it would be good to identify items in the tables that were a result of chance effects and eliminate them to avoid such mistakes.

If, correctly, no cause was found for the item, the item would be dismissed from further consideration. But it would be more effective if one knew to what extent the item being dismissed was a chance occurrence. If the likelihood was high, one could indeed dismiss the item. Else, one could devote more effort to find the cause.

In summary, there are three adverse effects that can occur on account of chance effects. The team may waste time considering items that occurred by chance, the team may find a cause by mistake for such an item, and, finally, the team may dismiss an item that should be pursued further. We develop an approach based on statistical significance below to address those concerns.

5 Statistical significance

It is hard to characterize the effects that occurred by chance when the data are generated by a complex physical process such as software development. Indeed, if it was possible to identify all variations in that process, it would imply that we have a very good understanding of how software should be developed. The truth is we do not, and we are a long way from such an understanding. Hence, the approach we use to address chance occurrences is

based on the well-known concept of statistical significance (See Chapter 6, [3]). To develop such an approach we do not require a sound understanding of software development. Instead, we use the following device. We define a random process which can generate classified defect data. Then, we determine how easily that process could replicate an item that was observed in the actual defect data. If an effect is easily replicated, we have shown that it can easily occur by chance. Else, the effect is unusual, since it is not easy to replicate using the random process. The ease with which the random process can replicate an item is referred to as the statistical significance of that item.

5.1 Statistical significance based on magnitude

The calculation of the statistical significance for the items produced by the function I_1 can be performed as follows:

Recall, that the proportion of data records with value a for attribute X is $p(X = a) = N(X = a)/N(X)$, where $N(X)$ = number of cases for which the value of attribute X is defined, and $N(X = a)$ is the number of cases with value a for attribute X. Under the hypothesis that all categories are equally likely, the expected value of this proportion, $E(X, a)$, was $1/Choice(X)$ in Equation 2.

The classical notion of statistical significance is based on the tail probability beyond the actually observed value, under the theoretical distribution of a statistic. Let us explain what that means. Let us define the following random process to generate classified defect data. We assume that all defects are generated independently and that a defect is classified $X = a$ with probability $E(X, a) = 1/Choice(X)$. Let us determine how easy or difficult it would be for a random process to replicate $p(X = a)$. This is easily done as follows. If $p(X, a)$ is greater than or equal to $E(X, a)$, we can determine the probability with which $N(X = a)$ or more defects could be classified $X = a$. That probability would give us an idea of how easily the random process would produce an effect that was equal or greater in magnitude than $P(X = a)$. If $p(X, a)$ is less than $E(X, a)$, we can determine the probability with which defects less than $N(X = a)$ could be classified $X = a$. That probability would give us an idea of how easily the random process would produce an effect that was smaller in magnitude than $P(X = a)$.

This probability, commonly known as a *P-value* is an indicator of the likelihood of an observed difference occurring by pure chance. If the P-value is small, such as .001, then the observed difference is hard to replicate using the random process. This can be considered as evidence that a true difference exists in the underlying software development process. If the P-value is large, such as .4, the difference can be easily replicated by the random process. This can be considered as evidence that the difference may not exist in the software development process and may have occurred by chance. The P-value is an inverse measure of 'unusualness'- the smaller the P-value, the harder it is to believe that a deviation is a chance occurrence. The formal development is given below.

Under the random sampling assumption, the P-value for testing whether the observed proportion $p(X = a)$ is consistent with the null hypothesis $E(X, a)$ can be calculated as a tail probability under a binomial distribution. Note that this distribution is conditional on the total number of records $N(X)$ for which the value of attribute X is defined. The parameters of the binomial distribution are: the number of trials $n = N(X)$, and the probability of 'success' at each trial $p = E(X, a)$.

We use either the upper or the lower tail of the binomial distribution, depending on whether the observed proportion $p(X = a)$ is above or below the expected proportion $E(X, a)$.

Let $b(x; n, p)$ denote the binomial probability function. This is the probability of x successes in n trials, when the probability of success at each trial is p. It is given by the equation:

$$b(x; n, p) = \binom{n}{x} p^x (1-p)^{n-x} \qquad (3)$$

The relevant tail probability can be defined as:

$I_{s1}(X, a) = \sum_{i=0}^{N(X,a)} b(i; n, p)$, if $p(X = a) < p$,

$I_{s1}(X, a) = \sum_{i=N(X,a)}^{n} b(i; n, p)$, if $p(X = a) > p$,

where $n = N(q)$, and $p = E(X, a)$.

In practice, the calculation of the above probabilities is best accomplished using readily available programs to compute the binomial cumulative distribution function. The computational cost is much less than the above formula would suggest, since fast and accurate algorithms are available for most cases of practical interest.

Example: Consider two attributes X and Y with 2 and 20 values, respectively. Assume that for each attribute, all values are equally likely, so that the expected proportion in each category is .5 for X and .05 for Y. Suppose that the actual data contains 100 records with values of X and Y defined, and 55 of these have $X = a$ and 8 have $Y = b$, where a and b are two specific values of attributes X and Y.

I_1 gives:

$I_1(X = a) = .55 - .5 = .05$, and

$I_1(Y = b) = .08 - .05 = .03$.

Now consider the statistical significance I_{s1} of the two attribute values:

$I_{s1}(X, a) = \sum_{x=55}^{100} b(x; 100, 0.5) = 0.16$

$I_{s1}(Y, b) = \sum_{x=8}^{100} b(x; 100, 0.05) = 0.11$

Notice that in this example, I_1 gives a higher value to the first deviation than the second, but the statistical significance I_{s1} indicates that the latter is the more unusual occurrence, in the sense that it is less likely to occur by pure chance. This situation is an exception rather than the rule. In general, the larger deviations will tend to have smaller P-values.

5.2 Statistical significance based on association

A similar approach is used to derive P-values for items produced by I_2. The formal development is given below.

For a given pair of attributes (X, Y), let $N(X, Y)$ be the total number of cases for which both X and Y are defined. For a specific pair of values (a, b) of (X, Y), let $N'(X = a)$ and $N'(Y = b)$ be the number of cases with $X = a$ and $Y = b$, respectively. Let $p(X = a, Y = b)$ be the proportion of cases with both $X = a$ and $Y = b$. The corresponding expected proportion under the hypothesis of statistical independence of the attributes X and Y (conditional on $N(X = a)$ and $N(Y = b)$) is $E_{ab} = p(X = a)p(Y = b)$. The function $I_2(X = a, Y = b)$ measures the absolute difference between the actual and the expected proportions. We define the corresponding statistical significance as follows.

For short, let us write, $p_a = p(X = a)$, $p_b = p(Y = b)$, and $p_{ab} = p(X = a, Y = b)$. Let,

$I_{s2}(X, a, Y, b) = Prob(Z \leq N(X = a, Y = b))$, if $p_{ab} < p_a p_b$,

$I_{s2}(X, a, Y, b) = Prob(Z \geq N(X = a, Y = b))$, if $p_{ab} > p_a p_b$.

I_{s2} measures the tail probability of a value as extreme as or more extreme than the actual observed count $N(X = a, Y = b)$. We calculate this probability conditional on the marginal totals $N'(X = a)$, $N'(Y = b)$ and $N(X, Y)$.

The exact calculation of I_{s2} is based on the hypergeometric distribution. The probability of a specific value x within the range of Z is:

$$h(x; N, A, B) = \binom{A}{x}\binom{N-A}{B-x} / \binom{N}{B} \quad (4)$$

where, $N = N(X, Y)$, $A = N'(X = a)$ and $B = N'(Y = b)$.

The significance level is calculated by summing $h(x; N, A, B)$ over the left or right tail as indicated above. In practice, the computation can be done using the cumulative distribution function of the hypergeometric distribution, available in major statistical packages.

Example: Table 4 shows data on three pairs of attribute values, taken from a data set with a total of 71 observations.

Notice that the second pair has a higher value of the function I_2 than the first one, but the P-value (I_{s2}) suggests that the first pair is the more unusual of the two. Also observe that the first and third pairs have nearly equal values of I_2, but there is a big difference in their P-values.

6 Empirical results and lessons

On surface, there seems to be an obvious way to incorporate the statistical significance approach in Attribute Focusing. As discussed in Section 5, the smaller P-values suggest the more unusual items while larger P-values suggest that the items could have occurred by chance. Usually, some small P-value such as 0.05 is used to decide which effects are statistically significant. Effects that have a P-value beyond that small value are considered statistically insignificant while effects that have a smaller P-value are considered statistically significant.

Conventional wisdom would suggest that such a small P-value be used to remove the statistically insignificant items from the tables that are studied by the project team. Such removal would eliminate the concerns mentioned in Section 4, since items that were likely to be a result of chance occurrences would simply not be considered by the team. However, the empirical results below show that such removal is not a good idea.

Table 5 is based on the data from the project experience that was reported in detail in [16]. There were seven process problems that were identified in that experience as a result of Attribute Focusing. Those problems are labelled A through G in the table. The first column, *Table*, indicates the different tables that were interpreted by the team in five feedback sessions. The tables are identified as XS or XP, where X is a numeral that indicates that the table was interpreted in the X'th feedback session. S indicates the table involved single attribute values since it was based on I_1, while P indicates pairs of attribute values since the table was based on I_2. There is one table that is specified as 1+2P. That table was based on I_2 applied to the combined data from both the first and second feedback sessions. Such combination is used for validating that the desired effect of corrective actions that were implemented as a result of earlier feedback sessions is reflected in the later data (see Section 2.2.3 [16] for specific details, Chapter 12 [18] for general examples of validation). Sometimes, if there are items that go beyond such validation, the combined data is also be explored by the team. Those tables have been included in Table 5.

The column "1.00" summarizes information about the project experience as reported in [16]. The number of items that were interpreted by the team in every table is provided along with the process problem identified as a result of that interpretation. For instance, 1S had 23 items. The interpretation of those problems led to the identification of problems A and B. 1P had 47 items but their interpretation did not lead to the identification of any problems, and so on. The other columns in Table 5 contain information about the effect on a table had items beyond the significance level specified at the head of the column been eliminated from the table. For instance, let us look at the column with the heading "0.05". We see that $1S$ would have been reduced to a table with only 6 items had we eliminated items that had a significance level greater than 0.05. The item corresponding to Process problem B would have been retained after such reduction. The column *Size* specifies the sample size or the number of defects in the data that was used to create the tables. XS and XP were created from the same data, hence, only one size is provided, in the row

TABLE 4

value1	value2	obs1	obs2	obs12	expec	diff	P-value
a	b	10/71 =14%	6/71 =8%	3/71 =4%	.14*.08 =1%	3%	3.3%
c	d	35/71 =49%	17/71 =24%	12/71 =17%	.49*.24 =12%	5%	4.1%
c	e	35/71 =49%	24/71 =34%	14/71 =20%	.49*.34 =17%	3%	20%

corresponding to XP. The information in the last three rows of the table will be explained in the next section.

The above table shows that it is possible for items that are statistically insignificant to be physically significant, i.e., they are not chance occurrences but instead have a definite physical cause. In other words, if items are eliminated from the tables based on some small P-value, it is possible that some of those items are the very items that would have led to the identification of process problems. For instance, conventional wisdom would suggest that all items that had a significance level more than 0.05 should be eliminated from the tables. From column "0.05", we see that while such reduction would indeed reduce the size of the tables to be interpreted, it would eliminate four of the seven items that led to the identification of process problems. Only the entries corresponding to B,D and G will be retained. Hence, had we used such reduction, it would have been difficult for the project team to identify the other process problems.

There are fundamental reasons why items can be statistically insignificant but physically significant. First, note that the existence of such a possibility can be inferred from our approach itself. The information on statistical significance is extraneous to software development, i.e., we did not require any knowledge of software development to compute the P-values. Instead, we used knowledge of the behavior of a process that generated defects randomly. Therefore, if an item in a table has a low statistical significance, it tells us that we could replicate that effect rather easily by using a random process. It does not tell us, necessarily, whether that effect is easy or difficult to replicate using the software development process. In other words, statistical significance does not necessarily suggest physical significance.

Let us acquire a deeper understanding by examining suitable examples from Table 5. We will make use of the process problems A, C, E, F which would have been missed had we used the significance level of 0.05 to limit the number of entries to be presented to the team.

6.1 Hidden causes

Fundamentally speaking, the items corresponding to process problems A and F were statistically insignificant but physically significant for the same reason: the presence of a hidden cause. Let us illustrate that reason by using Process problem F. The identification of Process Problem F occurred when the team considered the association between *function* and *missing* which is presented in the first row in Table 3. The statistical significance for that association (computed using Equation 4) was found to be 0.39. In other words, the random process model that was used to derive Equation 4 could have produced that association or a stronger association nearly 40% of the time. The high P-value suggests that the association between *function* and *missing* is statistically weak.

We described how Process problem F was found in Example 1 in Section 3.2. Let us go back to that description. Recall that the team found that all defects that were classified both *function* and *missing* pertained to a specific functionality, namely, recovery - the ability of the system to recover from an erroneous state. Note that the classified data did not capture any information about that functionality. In other words, it was the hidden cause for the weak association between *missing* and *function*. Another way to look at it is that had the classified data captured information about the specific functions that were implicated by a defect, we would have seen an association between *missing* and *recovery* and another association between *function* and *recovery*. Those associations would have been more significant than the the weak association between *function* and *missing*.

An informal way to understand this is as follows. An association may be viewed as an overlap between two categories. Since the classified data did not capture information about the recovery functionality, we did not see the strong overlaps between *function* and *recovery* and *missing* and *recovery*. Instead, we saw only the indirect effect of those overlaps, namely, the weaker overlap between *missing* and *function* defects which had occurred as a consequence of the hidden overlaps. The recovery function was a hidden cause underlying the observed association.

TABLE 5

Statistical Significance Levels

Table	1.00	0.05	0.10	0.20	0.30	0.40	0.50	Size
1S	23,A,B	6,B	10,B	21,A,B	21,A,B	21,A,B	21,A,B	
1P	47	3	5	8	14	18	29	30
2S	23	22	23	23	23	23	23	
2P	33,C	7	17	23,C	27,C	33,C	33,C	74
1+2P*	56,D	38,D	47,D	49,D	51,D	54,D	56,D	104
3S	28	27	28	28	28	28	28	
3P	62	39	44	52	60	62	62	121
4S	35	28	30	33	33	33	35	
4P	93,E,F	32	40	57	68	82,F	85,E,F	59
5S	39	39	39	39	39	39	39	
5P	36,G	35,G	36,G	36,G	36,G	36,G	36,G	239
Sum	475	276	319	369	400	429	447	
Hits	7	3	3	5	5	6	7	
Rate	68	92	106	73	80	72	64	

This is a good point to consider the difference between conventional hypothesis testing and hypothesis testing in the context of knowledge discovery. The identification of hidden causes is always a concern in statistical analysis. In the context of exploratory data analysis, that concern is exacerbated for the following reason. In conventional statistical analysis, a carefully designed experiment is used to test a given hypothesis. For instance, one may test the hypothesis: Is smoking associated with cancer. A carefully designed experiment to collect and analyse data that screens out many hidden causes can be conducted to test that hypothesis.

In exploratory data analysis, there is not a single hypothesis that one is interested in. In fact, we do not know what hypotheses are interesting. But then, how can one guarantee that one is collecting the right information in the first place to test a given hypothesis in a rigorous fashion ? The answer is one cannot.

It is easy to appreciate the above argument in the context of software development. Recall from Section 2 that Attribute Focusing is used to find process problems that the team does not know to look for. Well, since we don't know what we are going to find, how can we know we are capturing the right data to find it. Clearly, we cannot know. Process problem *F* is a good example. It shows that the data did not capture relevant information on specific functions such as recovery. More likely than not, if a complex process such as software development is being used to generate data, that data will not capture all the relevant information that is required to indicate a problem. Instead, it will capture some effect of the unknown problem and it will be up to the team to identify the hidden cause. If statistical significance is used to eliminate items, those indirect effects can be eliminated as well, and consequently, the team will not identify the hidden causes that underlie such effects.

6.2 The advantage of highlighting information

Process problems *C* and *E* were physically significant but not statistically significant for the same reason. Let us use the identification of *E* as an example to understand that reason. The identification of Process Problem E occurred when the team considered the disassociation between *India* and *backward compatibility* which is presented in the second row in Table 3. The P-value for that disassociation (computed using Equation 4) was found to be 0.47. In other words, the random process model that was used to derive Equation 4 could have produced that disassociation or a stronger disassociation nearly 50% of the time, indicating that the lack of association was rather weak.

We described how that problem was found in Example 2 in Section 3.2. Let us go back to that description. Recall that the problem was found because the team realized that there should be a strong association between *India* and *backward compatibility* instead of the disassociation which was called to their attention.

Consider what would have happened if we eliminated that disassociation from Table 3 based on its weak statistical significance. The team would have had difficulty realizing that there should be a strong association between *India* and *backward compatibility* since there would be no entry in the table to call their attention to it. However, had they noticed the absence of the strong association, they would have reasoned exactly as they did to find the process problem. In other words, the information on statistical significance is entirely consistent with the line of reasoning used by the team to find the process problem. Omitting the entry simply makes it harder for the team to notice the problem while retaining the entry serves to highlight the problem.

7 Lessons, discussion and conclusions

Let us return to the concerns that were summarized at the end of Section 4. The first concern had to do with saving the time spent on interpreting items that occurred by chance. If there was a way to clearly identify those items, such removal would indeed be a good idea. We believe such identification is hard when the data are generated are by a complex physical process such as software development. Hence, we believe our approach developed using statistical significance is a reasonable and pragmatic way to try and address chance effects. As is clear from the reported experience, we should not try to save the team's time by limiting the number of items that are explored on the basis of a small P-value.

Well, then how should we proceed ? The last three rows of Table 5 suggest a promising direction for further research. The row *Sum* is simply the sum of of the items totalled over all interpretation sessions that would have been considered by the team if we used a particular significance level to select items. The row *Hits* indicates the number of process problems that would have been found, while *Rate* is the division of *Sum* by *Hits*. Hence, *Rate* gives us an idea of the rate of discovery, i.e., the number of items interpreted to find a problem. We see that had we used a significance level of 0.50, the rate of discovery would have been improved slightly. However, if we used conventional wisdom and used 0.05 to select items, the rate would have been substantially worse.

The above argument suggests that while it is possible to improve the rate of discovery by using the significance level, it is also possible to reduce the rate substantially. More work is required to identify the right level that should be used. This direction is consistent with other work on data exploration. For instance, Hoschka and Klosgen [5] (see Page 333 of the reference) talk about varying the significance levels with sample size. Larger significance levels are used with smaller samples and vice versa. The column *Size* in Table 5 indicates the sample sizes for the experience reported in this paper. We see that the smaller sample sizes, 30, 59, and 74, do indeed require the larger P-values to cover all prob-

lems. However, we also see that the sample of size 59 requires the largest P-value to cover all problems and not the sample of size 30, indicating that the relationship of sample size to P-value may not be a linear one. We plan to investigate such heuristics in the future. It is also plausible that the right significance level not only depends on the sample size but also varies from domain to domain. Hence, empirical studies are essential. For instance, we may find that a P-value of 0.50 almost always improves the rate of discovery for defect data, and hence, is a useful heuristic to select items in defect data. We plan to do more empirical studies to find such heuristics.

The results in Table 5 suggest that it may be necessary to explore statistically insignificant items. If statistically insignificant items are to be explored, then how do we address the remaining two concerns raised in Section 4 ?

The other two concerns that were raised in Section 4 can be addressed by using the information on statistical significance. One concern was that if a cause could not be determined even after an item was investigated, that item would not be considered further. Using the information on statistical significance, the team can decide if that should indeed be the case. If the item is statistically significant, they may decide to continue to investigate that item.

Another concern was that a sequence of errors on the part of the team could lead them to find a cause based on an item that had occurred purely by chance. The team can use the information on statistical significance to double check their findings. If an item that was statistically insignificant was found to be physically significant, their findings should be consistent with the item being statistically insignificant. A hidden cause that was clearly corroborated as in Example 1, or an item that was merely highlighted as in Example 2 (Section 3.2), are examples of such consistency.

7.1 Limitation and strength

Let us make explicit the limitation and strength of the work that has been presented in this paper. They are transposed in point-counterpoint fashion.

- **Limitation:** As is the case with empirical studies, strictly speaking, the conclusions are specific to the conditions of the experiment. Hence, the results presented here tell us primarily about the application of attribute focusing to one software project.
- **Strength:** While the results are indeed specific to one experiment, it is useful to learn from the underlying explanations since they may have general implications. Furthermore, there is not much available by way of field experience in the literature on knowledge discovery. Since the results are based on data from a fielded application, such an exercise is especially relevant.

Accordingly, (what we believe are) the general lessons to be drawn from this paper are listed below. The supporting evidence is indicated within parentheses.

7.2 General Implications

In the context of knowledge discovery:

1. In any given set of data, there are very few statistical hypotheses that will actually lead to the discovery of knowledge. (Note that Table 5 shows that only seven problems (based on seven relationships) were found from five sets of data. Field data from the application of attribute focusing to software development has shown that, on average, two to three such relationships are found per set of data [11, 17]).
2. The data are not collected to support testing the validity of a particular statistical hypothesis. (See the related discussion towards the second half of Section 6.1).
3. Data will often come from processes that are poorly understood. It is hard to characterize the statistical properties of a complex data generating process. Hence, statistical validation of hypotheses in data from such processes will usually be done by using general hypothesis testing methods such as statistical significance. (See the discussion at the beginning of Section 5. See also the work by Courtney and Gustafson [19] for another example of such use).
4. Items 1, 2 and 3, above suggest that while the use of statistical hypothesis testing methods may often be the only practical way to eliminate relationships that occur by chance, they will have to be used carefully to make sure that one does not also eliminate from a set of data the few relationships that are useful.

7.3 Conclusions

In the context of an exploratory data analysis technique such as Attribute Focusing, knowledge of statistical significance:

- Should be used carefully when selecting differences which should be studied to identify causes. Conventional wisdom would suggest that all differences that lie beyond some small level of statistical significance be eliminated from consideration. Our results show that such elimination is not a good idea.
- May be useful to improve the rate of discovery of causes during exploration. More work is required to understand how such improvement can occur.
- Can be used to address the concern that the items being explored have occurred by chance in two ways. First, if a cause is found while reasoning about a a statistically insignificant item, that line of reasoning should be examined to ensure it is consistent with the item being statistically insignificant. Second, if a cause is not found for a statistically significant deviation, further investigation of the item should be considered prior to dismissing that item.

8 Acknowledgements

We would like to thank Jarir Chaar, Ram Chillarege, Mike Halliday, Art Nadas and Peter Santhanam, all of IBM, for the support and insights they have given us during the course of this work.

References

[1] G. Piatetsky-Shapiro and W. Frawley, eds., *Knowledge Discovery in Databases*. Menlo Park, California: AAAI Press/The MIT Press, 1991.

[2] C. Matheus, P. Chan, and G. Piatetsky-Shapiro, "Systems for knowledge discovery in databases," *IEEE Transactions on Knowledge and Data Engineering*, vol. 5, pp. 903–913, December 1993.

[3] B. Ostle and R. Mensing, *Statistics in Research*. Ames: Iowa State University Press, 1975.

[4] V. Dhar and A. Tushilin, "Abstract-driven pattern discovery in databases," *IEEE Transactions on Knowledge and Data Engineering*, vol. 5, December 1993.

[5] P. Hoschka and W. Klosgen, "A support system for interpreting statistical data," in *Knowledge Discovery in Databases* (G. Piatetsky-Shapiro and W. Frawley, eds.), Menlo Park, California: AAAI Press/The MIT Press, 1991.

[6] W. S. Humphrey, *Managing the Software Process*. Addison-Wesley Publishing Company, 1989.

[7] R. A. Radice, N. K. Roth, A. C. O'Hara, and W. A. Ciarfella, "A Programming Process Architecture," *IBM Systems Journal*, vol. 24, no. 2, 1985.

[8] A. Endres, "An Analysis of Errors and Their Causes in System Programs," *IEEE Transactions on Software Engineering*, vol. 1, no. 2, pp. 140–149, 1975.

[9] V. R. Basili and H. D. Rombach, "Tailoring the Software Process to Project Goals and Environments," in *Proc. of the International Conference on Software Engineering*, pp. 345–357, April 1987.

[10] R. Chillarege, I. Bhandari, J. Chaar, M. Halliday, D. Moebus, B. K. Ray, and M. Y. Wong, "Orthogonal defect classification – a concept for in-process measurements," *IEEE Transactions on Software Engineering*, vol. 9, November 1992.

[11] I. Bhandari, "Attribute Focusing: Machine-Assisted Knowledge Discovery Applied to Software Production Process Control," *Knowledge Acquisition*. Accepted for publication. Abbreviated verion appeared in Proceedings of the Workshop on Knowledge Discovery in Databases, AAAI Technical Report Series, Number WF-93-02, July 1993.

[12] R. W. Selby and A. A. Porter, "Learning from examples: Generation and evaluation of decision trees for software resource analysis," *IEEE Transactions on Software Engineering*, vol. 14, pp. 1743–57, December 1988.

[13] L. C. Briand, V. R. Basili, and C. J. Hetmanski, "Developing Interpretable Models with Optimised Set Reduction for Identifying High Risk Software Components," *IEEE Transactions on Software Engineering*, November 1993. To appear in a special issue on Software Reliability.

[14] I. S. Bhandari and N. K. Roth, "Post-process Feedback with and without Attribute Focusing: A Comparative Evaluation," in *Proc. of the International Conference on Software Engineering*, pp. 89–98, May 1993.

[15] J. K. Chaar, M. J. Halliday, I. S. Bhandari, and R. Chillarege, "In-Process Metrics for Software Inspection and Test Evaluations," *IEEE Transactions on Software Engineering*, November 1993.

[16] I. Bhandari, M. Halliday, E. Tarver, D. Brown, J. Chaar, and R. Chillarege, "A case study of software process improvement during development," *IEEE Transactions on Software Engineering*, December 1993.

[17] I. Bhandari, M. Halliday, , J. Chaar, R. Chillarege, K. Jones, J. Atkinson, C. Lepori-Costello, P. Jasper, E. T. adn C. Carranza-Lewis, and M. Yonesawa, "In-process Improvement through Defect Data Interpretation," *IBM Systems Journal*, 1st Qtr 1994. To appear. A draft is available as IBM Research Report Log 81922, 1993.

[18] A. Mayerhauser, *Software Engineering - Methods and Management*. San Diego: Academic Press, Inc., 1990.

[19] R. E. Courtney and D. A. Gustafson, "Shotgun Correlations in Software Metrics," *Software Engineering Journal*, January 1993.