CORE
Metadata, citation and similar papers at core.ac.uk
Provided by ScholarBank@NUS

Towards Exploratory Hypothesis Testing and Analysis

Guimei Liu [#1], Mengling Feng [*2], Yue Wang [##3], Limsoon Wong [#4], See-Kiong Ng [*5],
Tzia Liang Mah [*6], Edmund Jon Deoon Lee [**7]

[#] *Department of Computer Science, National University of Singapore, Singapore*
[1] `liugm@comp.nus.edu.sg`; [4] `wongls@comp.nus.edu.sg`

[*] *Data Mining Department, Institute for Infocomm Research, Singapore*
[2] `mfeng@i2r.a-star.edu.sg`; [5] `skng@i2r.a-star.edu.sg`; [6] `tlmah@i2r.a-star.edu.sg`

[##] *Graduate School for Integrative Science and Engineering, National University of Singapore, Singapore*
[3] `wangyue@nus.edu.sg`

[**] *Pharmacology Department, National University of Singapore, Singapore*
[7] `edlee@nus.edu.sg`

***Abstract*—Hypothesis testing is a well-established tool for scientific discovery. Conventional hypothesis testing is carried out in a hypothesis-driven manner. A scientist must first formulate a hypothesis based on his/her knowledge and experience, and then devise a variety of experiments to test it. Given the rapid growth of data, it has become virtually impossible for a person to manually inspect all the data to find all the interesting hypotheses for testing. In this paper, we propose and develop a data-driven system for automatic hypothesis testing and analysis. We define a hypothesis as a comparison between two or more sub-populations. We use frequent pattern mining techniques to find sub-populations for comparison, and then pair them up for statistical testing. We also generate additional information for further analysis of the hypotheses that are deemed significant. We conducted a set of experiments to show the efficiency of the proposed algorithms, and the usefulness of the generated hypotheses. The results show that our system can help users (1) identify significant hypotheses; (2) isolate the reasons behind significant hypotheses; and (3) find confounding factors that form Simpson's Paradoxes with discovered significant hypotheses.**

I. Introduction

Hypothesis testing is a well-established tool for scientific discovery. It enables scientists to distinguish results that represent systematic effects in the data from those that are due to random chance. Hypothesis testing involves a comparison of two or more sub-populations. One example hypothesis is "Smokers are more vulnerable to the H1N1 flu than non-smokers". To test this hypothesis, we need to compare the occurrence of H1N1 flu infection between two sub-populations; in this case, smokers and non-smokers. The outcome of hypothesis testing can help people make decisions. For example, knowing which group of people are more vulnerable to a certain type of flu, doctors can vaccinate this group of people first to prevent the spread of the flu. Knowing under what situation a product is more likely to fail, engineers can devote their efforts on investigating why the product fails under that situation, and then improve the design of the product to prevent that from happening again.

Conventional hypothesis testing is usually carried out in a *hypothesis-driven* manner. A scientist must first formulate a hypothesis based on his/her knowledge and experience, and then devise a variety of experiments to test it. This presents a possible *catch-22* situation, for it is often the case that people want to find something from their data, but do not know what to find. Even if a person has much domain knowledge and ample experience, the data may still contain something useful that he/she is not aware of. For example, even an experienced doctor may not know all the risk factors of a complex disease.

With the rapid development and wide usage of computer technologies, more and more data have been accumulated and stored in digital format. These data provide rich sources for making new discoveries. However, the sheer volume of the data available nowadays makes it impossible for people to inspect all the data manually. As a result, lots of useful knowledge may go undiscovered. This calls for the need of developing a system for automatic hypothesis testing in a *data-driven* manner.

Data mining is an important tool to transform data into knowledge in a data-driven manner. Data mining does not start from a pre-conception or a specific question like hypothesis testing. Instead, it aims to automatically extract useful information from large volumes of data via exploratory search, making it highly applicable for automatic knowledge discovery. In this paper, we develop a system for exploratory hypothesis testing and analysis by building on and extending existing data mining techniques. Given a dataset, we formulate and test tentative hypotheses based on the attributes in the dataset, the nature of the attributes and the statistics of the data. The space of all the possible hypotheses can be very large. We employ techniques developed for frequent pattern mining to efficiently explore the space of tentative hypotheses.

In many cases, it is not sufficient to just know whether a hypothesis is statistically significant. It is more interesting and important to know the reasons behind significant hypotheses. Therefore, we further analyze the hypotheses that are statistically significant and identify the factors that contribute to the difference. Another reason for the need of further analysis of significant hypotheses is that some of the significant hypotheses generated by exploratory search may be spurious since the exploration is not guided by domain knowledge. We examine whether there are any confounding factors that may lead to spurious hypotheses.

The rest of the paper is organized as follows. Section II formally defines the exploratory hypothesis testing and analysis problem. In Section III, we develop efficient algorithms for automatic hypothesis testing and analysis by using and extending well-established frequent pattern mining techniques. Experiment results are reported in Section IV. Section V describes related work. Finally, in Section VI, we conclude the paper with some closing discussions.

II. Problem definition

Hypothesis testing is a test of significance on a difference. A difference is *statistically significant* if it is unlikely to have occurred by chance. Hypothesis testing can be conducted on n sub-populations, where $n \geq 1$. Hypothesis testing involving only one sub-population compares the statistics of one sub-population with the parameters of the general population. However, the parameters of the general population are often unknown. The results of hypothesis testing involving more than two sub-populations are usually hard to interpret. It is not easy for users to tell which sub-population contributes the most to the difference. In the end, users still need to resort to pairwise comparisons to get a clear picture. Hence, we focus on the case when n=2 in this paper.

The hypothesis testing process consists of four main steps:

1) State the relevant null and alternative hypotheses. The null hypothesis is always "There is no difference." and the alternative hypothesis is "There is difference".
2) Choose a proper statistical test based on the type and distribution of the attribute that the sub-populations are compared on. An overview of the statistical tests and when should they be used can be found at [12].
3) Calculate the p-value using the chosen test. The p-value is the probability of obtaining a statistic at least as extreme as the one that was actually observed, given the null hypothesis is true. The lower the p-value, the less likely that the observed difference is due to random chance, thus the more statistically significant the difference is.
4) Decide whether to reject or accept the null hypothesis based on the p-value. Conventionally, a p-value of 0.05 is generally recognized as low enough to reject the null hypothesis and accept the alternative hypothesis.

In the rest of this section, we describe how to automate the testing process to test all the possible hypotheses in a given dataset. We define two tasks, exploratory hypothesis testing and hypothesis analysis, in the situation when the target attribute, on which the significance of the difference is tested, is categorical. It is straightforward to generalize the definitions to the situation when the target attribute is continuous.

PID	Race	Gender	Age	Smoke	Stage	Treatment	Response
1	Caucasian	M	45	Yes	1	A	positive
2	Asian	M	40	No	1	A	positive
3	African	F	50	Yes	2	B	negative
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
N	Caucasian	M	60	No	2	B	negative

TABLE I
An example dataset

A. Hypothesis formulation

We use the example dataset shown in Table I to illustrate how to formulate hypotheses. Each row in the dataset is the medical record of a patient. The last column is the response of the patients to a certain treatment. One example hypothesis is "Treatment A is more effective than treatment B". The two sub-populations under comparison are "patients undergone treatment A" and "patients undergone treatment B", and they are compared on attribute "Response". In this example, there are two types of attributes. One type of attributes, such as "Treatment", are used to define sub-populations, and we call them *grouping attributes*. A grouping attribute cannot be continuous. The other type of attributes, such as "Response", are the attribute on which the significance of the difference is tested, and we call them *target attributes*.

Given a dataset, we ask domain users to specify a target attribute as the objective of the exploratory hypothesis testing. If the target attribute is a categorical attribute, we further ask domain users to choose a value that is the most interesting to them, and we call this value *target attribute value*. We also ask domain users to specify the grouping attributes if possible (if not, we simply use all the categorical attributes in the given dataset as grouping attributes). This should be easy for domain users since they usually have some rough ideas on which attributes are of interest for comparison and which attributes can be used for grouping.

A sub-population is defined by a set of attribute-value pairs. We call an attribute-value pair *an item*, and a set of items *a pattern*.

Definition 1 (Pattern): A pattern is a set of attribute-value pairs (items), denoted as $P = \{A_1 = v_1, A_2 = v_2, \cdots, A_k = v_k\}$, where A_i is an attribute ($1 \leq i \leq k$), v_i is a value taken by attribute A_i, and $A_i \neq A_j$ if $i \neq j$.
If an attribute-value pair $A = v$ appears in a pattern P, we say P *contains* attribute A. Each pattern defines a sub-population. For example, pattern {EthnicGroup=Caucasian, Gender= Male} defines the sub-population consisting of Caucasian male patients. In the rest of this paper, when we say pattern P, we refer to the pattern itself; when we say sub-population P, we refer to the sub-population defined by P. The support of a pattern P, is defined as the number of records containing P, denoted as $sup(P)$. The support of pattern P is equivalent to the size of sub-population P.

Given two patterns P and P', if every item in P is also in P', then P is called a *sub-pattern* of P', denoted as $P \subseteq P'$, and P' is called a *super-pattern* of P, denoted as $P' \supseteq P$. If P is a sub-pattern of P' and P has one less item than P', then we call P an *immediate sub-pattern* of P', and P' an *immediate super-pattern* of P.

In hypothesis testing, users usually study one factor at a time. Hence, in this work, we require that the defining patterns of two sub-populations under comparison differ by one and only one item. For example, comparing subgroup {EthnicGroup= Caucasian, Gender=Male} with subgroup { EthnicGroup= Caucasian, Gender=Female} is acceptable, while comparing subgroup {EthnicGroup=Caucasian, Gender= Male} with subgroup {EthnicGroup=Asian, Gender =Female} is less intuitive because even if the difference between the two sub-populations is statistically significant, it is hard for users to conjecture which attribute contributes to the difference. Now we formally define tentative hypotheses.

Definition 2 (Tentative hypothesis): Let A_{target} be a categorical target attribute, v_{target} be the target attribute value, P_1 and P_2 be two patterns that differ by one and only one item, denoted as $P_1 = P \cup \{A_{diff} = v_1\}$, $P_2 = P \cup \{A_{diff} = v_2\}$. The tentative hypothesis on the two sub-populations defined by P_1 and P_2 is represented as $H = \langle P, A_{diff} = v_1|v_2, A_{target}, v_{target}\rangle$. Pattern P is called the *context* of H, attribute A_{diff} is called the *comparing attribute* of H and P_1 and P_2 are called the two sub-populations of H. The null hypothesis is $p_1 = p_2$, and the alternative hypothesis is $p_1 \neq p_2$, where $p_i = \frac{sup(P_i \cup \{A_{target}=v_{target}\})}{sup(P_i)}$, i=1, 2.
Based on the definition, hypothesis "Treatment A is more effective than treatment B" can be represented as $\langle \{\}, Treatment = A|B,\ Response, positive\rangle$.

B. Choosing a proper test and calculating the p-value

The selection of a proper statistical test depends on the type and distribution of the target attribute. An overview of the statistical tests and when should they be used can be found at [12]. Most of these tests can be integrated into our system seamlessly. Given a statistical test, we usually need to scan the dataset and collect some statistics for the calculation of the p-value.

Let us look at the example hypothesis H: "Treatment A is more effective than treatment B". The two sub-populations are compared on attribute "Response" and it is nominal, hence we choose χ^2-test with Yates' correction[15]. The χ^2-test is a test for examining the association between two categorical attributes, and it requires four statistics to be collected: the size of the two sub-populations under comparison, denoted as n_1 and n_2, and the proportion of the target attribute value in the two sub-populations, denoted as p_1 and p_2. The four values of H are shown in Table II (a). Given the four values, the χ^2-score is calculated as follows:

$$\chi^2_{Yates} = \frac{(n_1+n_2)(n_1 n_2 |p_1 - p_2| - (n_1+n_2)/2)^2}{n_1 n_2 (n_1 p_1 + n_2 p_2)(n_1 + n_2 - n_1 p_1 - n_2 p_2)} \quad (1)$$

The degree of freedom of this score is 1. From χ^2-score and the degree of freedom, we can get the corresponding p-value by looking up a χ^2 distribution table. Table II (b) shows the χ^2-score and p-value of H.

Patterns	sup	p_i

Treatment=A
Treatment=A, Response=positive
	1000 (n_1)
	890
		p_1=89%

Treatment=B
Treatment=B, Response=positive
	1000 (n_2)
	830
		p_2=83%

(a) Statistics needed for calculating p-value using χ^2-test

Hypothesis H	χ^2-score	p-value
$\langle \{\}, Treatment = A\|B,\ Response, positive\rangle$	14.95	0.0001

(b) p-value

TABLE II
CALCULATING THE P-VALUE OF HYPOTHESIS "TREATMENT A IS MORE EFFECTIVE THAN TREATMENT B"

C. Deciding the significance of the observed difference

Conventionally, a p-value of 0.05 is generally recognized as low enough to reject the null hypothesis if one single hypothesis is tested. A p-value of 0.05 means that there is a 0.05 probability that the null hypothesis is true but we are wrongly rejecting it. Here we are testing large numbers of hypotheses simultaneously, which may generate many false positives [3]. We use two methods to control the number of false positives, Bonferroni correction [1] and Benjamini and Hochberg's method [3].

Bonferroni correction [1] is one of the most commonly used approaches for multiple comparisons. It aims at controlling family-wise error rate (FWER)—the probability of making one or more false discoveries among all the hypotheses. The basic idea is that if we test n dependent or independent hypotheses, then one way of maintaining the family-wise error rate is to test each individual hypothesis at a statistical significance level of 1/n times what it would be if only one hypothesis were tested. Bonferroni correction is computationally simple, but it can be very conservative and may inflate the rate of false negatives unnecessarily.

Benjamini and Hochberg's method [3] controls false discovery rate (FDR)—the expected proportion of falsely rejected null hypotheses, which is less stringent than family-wise error rate. Let H_1, H_2, $\cdots$, H_n be the n hypotheses tested and they are sorted in ascending order of p-value. Their corresponding p-values are p_1, p_2, $\cdots$, p_n. To control FDR at a level of q, we get the largest i, denoted as k, for which $p_i \leq \frac{i}{n}q$, and then regard all H_i, i=1, 2, $\cdots$, k, as statistically significant.

We use the above two methods in our system as follows. We ask users to specify a statistical significance threshold max_pvalue for testing one single hypothesis, and output the hypotheses with p-value$\leq max_pvalue$. During the hypothesis generation process, we count the total number of tests being performed, denoted as n. Bonferroni correction can be easily applied by replacing max_pvalue with max_pvalue/n. We then rank the hypotheses in ascending order of p-values, and apply Benjamini and Hochberg's method.

Statistical significance vs. domain significance. Sometimes a statistically significant result can have little or no domain significance. For example, given large sample sizes, a difference in 5 beats per minutes in pulse rate in a clinical trial involving two drugs can give a statistically significant difference whereas the average difference may hardly bring about a drastic metabolic change between the two groups. The reason being that domain significance mainly depends on the difference between the two means or proportions, while statistical significance also depends on standard error. If standard error is small enough, a difference can be statistically significant even if it is not domain significant. Only domain users can decide the level at which a result is regarded as domain significant.

D. Hypothesis analysis

In many cases, we are not only interested in knowing whether the difference between two sub-populations is significant, we are more interested in knowing the reasons behind the difference. For example, given the failure rate of one model of a product is significantly higher than that of another model, it is important for engineers to know under what situations the first model is more likely to fail so that they can improve its design accordingly. Table III compares the failure rate of two models of the same product. Model A has a higher failure rate than model B in general. However, after the two sub-populations are further divided using attribute "time-of-failure", we find that model A has comparable failure rate with model B at the time of ''in-operation" and "outputting", but has exceptionally high failure rate in the "loading" phase. This information is very useful since it helps engineers narrow down the problem.

pairs of sub-populations	failure rates

model=A
model=B
	4%
	2%

model=A, time-of-failure=loading
model=B, time-of-failure=loading
	6.0%
	1.9%

model=A, time-of-failure=in-operation
model=B, time-of-failure=in-operation
	2.1%
	2.1%

model=A, time-of-failure=outputting
model=B, time-of-failure=outputting
	2.0%
	1.9%

TABLE III
AN EXAMPLE INTERESTING ATTRIBUTE "TIME-OF-FAILURE". "LOADING" IS OF PARTICULAR INTEREST BECAUSE THE TWO MODELS SHOW A VERY BIG DIFFERENCE.

pairs of sub-populations	response positive	response negative	proportion of positive response	p-value
Treatment=A	890	110	89.0%	0.0001
Treatment=B	830	170	83.0%	
Treatment=A, Stage=1	800	80	90.9%	0.0807
Treatment=B, Stage=1	190	10	95%	
Treatment=A, Stage=2	90	30	75%	0.2542
Treatment=B, Stage=2	640	160	80%	

TABLE IV
AN EXAMPLE SPURIOUS HYPOTHESIS. ATTRIBUTE "STAGE" IS A CONFOUNDING FACTOR.

Another reason for the need of further analysis of significant hypotheses is that some of the significant hypotheses generated via exploratory search may be spurious since the search is not guided by domain knowledge. For example, hypothesis H in Table II(b) is actually a spurious hypothesis as shown in Table IV. The original hypothesis H indicates that treatment A is more effective than B. However, after we further divide the two sub-populations of H into smaller subgroups using attribute "stage" , we find that treatment B is more effective than treatment A for both patients at stage 1 and patients at stage 2. This phenomenon is called Simpson's Paradox [13], and it is caused by the fact that "stage" has associations with both treatment and response: doctors tend to give treatment A to patients at stage 1, and treatment B to patients at stage 2; patients at stage 1 are easier to cure than patients at stage 2. Attributes that have associations with both the comparing attribute and the target attribute are called *confounding factors*.

The above examples show that further investigation of hypotheses is often very useful, and it can be conducted by dividing the two sub-populations under comparison into finer subgroups, and then inspecting whether unexpected result is observed in pairs of the finer subgroups.

Difference lift of items and attributes

To divide the two sub-populations of a hypothesis H into smaller subgroups, we can add more items into the context P of H. A simple way to measure the impact of an item $A = v$ to H is to see how much the difference is lifted.

Definition 3 (DiffLift(A=v|H)): Let $H = \langle P, A_{diff} = v_1|v_2, A_{target}, v_{target}\rangle$ be a hypothesis, A_{target} be categorical, $P_1 = P \cup \{A_{diff} = v_1\}$ and $P_2 = P \cup \{A_{diff} = v_2\}$ be the two sub-populations of H, $A = v$ be an item not in H, that is, $A \neq A_{diff}$, $A \neq A_{target}$ and $A = v \notin P$. After adding item $A = v$ to P, we get two new sub-populations: $P_1' = P_1 \cup \{A = v\}$ and $P_2' = P_2 \cup \{A = v\}$. The lift of difference after adding $A = v$ to H is defined as *DiffLift(A=v|H)*=$\frac{p_1'-p_2'}{p_1-p_2}$, where p_i is the proportion of v_{target} in sub-population P_i, and p_i' is the proportion of v_{target} in sub-population P_i', i=1, 2.

We can divide the values of *DiffLift(A=v|H)* into three ranges, and each range represents a different situation.

Situation 1: *DiffLift(A=v|H)*>1. The new difference is wider than the old difference and it is also of the same direction as the old difference. The larger the *DiffLift*, the more interesting the item. For example, in Table III, the *DiffLift*s of "time-of-failure=loading", "time-of-failure=in-operation", "time-of-call=outputting" are 2.05, 0, 0.05 respectively. Hence "time-of-failure=loading" is an very interesting item.

Situation 2: $0 \leq$ ***DiffLift(A=v|H)***≤ 1**.** The new difference is of the same direction as the old difference, but it is narrower than the old difference. The items satisfying this condition usually are not very interesting, e.g., "time-of-failure=in-operation" and "time-of-failure=outputting" in Table III.

Situation 3: *DiffLift(A=v|H)*< 0. The new difference is of the opposite direction of the old one. If the values of an attribute all satisfy this condition, then we have a Simpson's Paradox as in Table IV where *DiffLift(stage=1*$|H_1$*)*$= -0.683$ and *DiffLift(stage=2*$|H_1$*)* $= -0.833$.

Definition 4 (Simpson's Paradox): Given a hypothesis H and an attribute A not in H, if for every value v of A, we have *DiffLift(A=v|H)*< 0, then we say H and A form a Simpson's Paradox.

We define the *DiffLift* of an attribute A as the average of the absolute *DiffLift* of its attribute values.

Definition 5 (DiffLift(A|H)): Let $v_1, v_2, \cdots, v_k$ be the set of values taken by attribute A. Then the difference lift of A with respect to H is defined as $DiffLift(A|H) = \frac{\sum_{i=1}^{k}|DiffLift(A=v_i|H)|}{k}$.

Contribution of items and attributes

In some cases, *DiffLift(A=v|H)* is not sufficient to capture all the information. Let n_i be the size of sub-population P_i, n'_i be the size of sub-population P'_i, i=1, 2. If n'_i is extremely small compared with n_i, i=1, 2, then even if *DiffLift(A=v|H)* is positive and very large, item $A = v$ can hardly have any material impact on H. We are more interested in finding those attribute values that have a big positive *DiffLift(A=v|H)*, and are also associated with a large number of records, so that acting upon such attribute values, we are able to make much bigger impact than acting upon attribute values that are associated with few records.

If we divide sub-population P_1 into several disjoint subsets P_{11}, P_{12}, $\cdots$, P_{1k}, then the proportion of v_{target} in P_1 can be expressed as $p_1 = \sum_{i=1}^{k}\frac{n_{1i}}{n_1}p_{1i}$, where n_1 is the size of sub-population P_1, n_{1i} is the size of subset P_{1i} and p_{1i} is the proportion of v_{target} in subset P_{1i}. Hence $\frac{n_{1i}}{n_1}(p_{1i} - p_1)$ can be regarded as the overall contribution of P_{1i} to P_1, denoted as $Contribution(P_{1i}|P_1)$. We have $\sum_{i=1}^{k}\frac{n_{1i}}{n_1}(p_{1i} - p_1) = \sum_{i=1}^{k}\frac{n_{1i}}{n_1}p_{1i} - p_1\frac{\sum_{i=1}^{k}n_{1i}}{n_1} = 0$. The contribution of $A = v$ to H is determined by which one is bigger, the contribution of P'_1 to P_1, or the contribution of P'_2 to P_2.

Definition 6 ($Contribution(A = v|H)$): Let H, $A = v$, P_i, p_i, P'_i and p'_i, i=1, 2, be defined as in Definition 3. The contribution of $A = v$ to H is defined as $Contribution(A = v|H) = (\frac{n'_1}{n_1}(p'_1 - p_1) - \frac{n'_2}{n_2}(p'_2 - p_2))/(p_1 - p_2)$.
If $Contribution(A = v|H) > 0$, we say $A = v$ contributes positively to H; if $Contribution(A = v|H) < 0$, we say $A = v$ contributes negatively to H; otherwise, we say $A = v$ makes no contributes to H. We define the $Contribution$ of an attribute A as the average of the absolute $Contribution$ of its attribute values.

Definition 7 ($Contribution(A|H)$): Let $v_1, v_2, \cdots, v_k$ be the set of values taken by attribute A. Then the contribution of A to H is defined as $Contribution(A|H) = \frac{\sum_{i=1}^{k}|Contribution(A=v_i|H)|}{k}$.

To help users have a better understanding of the discovered hypotheses, for each significant hypothesis H, we identify those attributes and items that have either high *DiffLift* or high $Contribution$, and generate a ranked list of these attributes and items. We also identify the attributes that can form Simpson's Paradoxes with H. Users can then browse the generated information to find things that are interesting to them.

E. Generalizations

Generalization to continuous target attribute. When A_{target} is continuous, we can simply use m_i to replace p_i, and m'_i to replace p'_i in Definitions 2, 3 and 6, where m_i and m'_i is the mean of A_{target} in sub-population P_i and P'_i respectively, i=1, 2. If A_{target} is normally distributed, we can use t-test to calculate p-values; otherwise, we can use Mann-Whitney test.

Generalization to hypotheses involving only one sample or more than two samples. We can simply represent a hypothesis involving one sample as $H = \langle P, A_{target}, v_{target}\rangle$. In this case, the statistics of sub-population P is compared with the whole population. *DiffLift* and $Contribution$ can be defined accordingly. It is also straightforward to formulate and test hypotheses involving more than two samples. A hypothesis involving k ($k > 2$) samples can be represented as $H = \langle P, A_{diff} = v_1|v_2|\cdots|v_k, A_{target}, v_{target}\rangle$. Statistical tests for more than two samples are also available. However, the analysis of hypothesis involving more than two samples is much more complicated. We leave it to our future work.

F. Problem statement

If the size of the two sub-populations under comparison is too small, statistical tests usually do not have enough power to detect the difference even if the difference is real. Hence testing hypotheses involving very small sub-populations is often futile. To save computation cost, we put a minimum support constraint min_sup on the size of the two sub-populations. If the size of a sub-population is no less than

Our system requires users to supply the following parameters: (1) a minimum support threshold min_sup; (2) a maximum p-value threshold max_pvalue which indicates the level of statistical significance if one single hypothesis is tested; (3) a minimum difference threshold min_diff which reflects the level of domain significance; (4) a target attribute A_{target} and a target attribute value v_{target} if A_{target} is categorical; (5) a set of grouping attributes $\mathcal{A}_{grouping}$, and the grouping attributes must be categorical. Users can set the parameters based on their requirements and domain knowledge. The last parameter is optional. If users do not specify the grouping attributes, then we use all the categorical attributes in the given dataset as grouping attributes.

Exploratory hypothesis testing. Given a dataset D and the above parameters, the hypothesis testing task aims to find all the hypotheses $H = \langle P, A_{diff} = v_1|v_2, A_{target}, v_{target}\rangle$ satisfying the following conditions: **1)** $\forall$ item $A = v$ in P, $A \in \mathcal{A}_{grouping}$, and $A_{diff} \in \mathcal{A}_{grouping}$. **2)** $sup(P_1) \geq min\_sup$, $sup(P_2) \geq min\_sup$, where $P_1 = P \cup \{A_{diff} = v_1\}$, $P_2 = P \cup \{A_{diff} = v_2\}$. **3)** p-value$(H) \leq max_pvalue$. **4)** If A_{target} is categorical, $|p_1 - p_2| \geq min\_diff$, where p_i is the proportion of v_{target} in sub-population P_i, i=1, 2. If A_{target} is continuous, then $|m_1 - m_2| \geq min\_diff$, where m_i is the mean of A_{target} in sub-population P_i, i=1, 2.

Hypothesis analysis. For each significant hypothesis H, the hypothesis analysis task generates the following information for further analysis: **1)** The set of Simpson's Paradoxes formed by H with attributes not in H. **2)** The list of items not in H ranked in descending order of *DiffLift*$(A = v|H)$ and $Contribution(A = v|H)$ respectively. **3)** The list of attributes not in H ranked in descending order of *DiffLift*$(A|H)$ and $Contribution(A|H)$ respectively.

III. Automatic hypothesis testing and analysis

We generate hypotheses in two steps. First, we use existing frequent pattern mining techniques to generate large sub-populations. Extensive efforts have been devoted to frequent pattern mining, and many efficient algorithms have been developed [6]. Most of these algorithms can be used to generate large sub-populations. In the second step, we pair the large sub-populations up to form tentative hypotheses. The number of large sub-populations generated may be very large, so we need to store the patterns in a structure that supports efficient pattern retrieval. A suitable candidate for this task is the CFP-tree structure [11].

In the rest of this section, we first present the hypotheses generation and analysis algorithms without restricting ourselves to any frequent pattern mining algorithm or any structure for storing patterns, and then describe how the CFP-tree structure can be utilized to improve the efficiency of the algorithms.

A. Finding large sub-populations

Existing frequent pattern mining algorithms generate only frequent patterns and their supports. Additional information are needed for calculating p-values and for analyzing significant hypotheses. We modified the frequent pattern mining algorithm used in our system as follows.

We collect more statistics of the large sub-populations besides the support of the patterns defining them for the calculation of p-values. Here we take the two commonly used tests, χ^2-test and t-test, as examples. Let P be a frequent pattern. When χ^2-test is used, we need to collect $sup(P)$ and $sup(P \cup \{A_{target} = v_{target}\})$ to get the proportion of $A_{target} = v_{target}$ in sub-population P. When t-test is used, we need to get the mean m_P and the standard deviation s_P of A_{target} in sub-population P. These two statistics are defined as follows: $m_P = \frac{\sum_{i=1}^{sup(P)} v_i}{sup(P)}$ and $s_P = \sqrt{\frac{\sum_{i=1}^{sup(P)}(v_i - m_P)^2}{sup(P)-1}}$ = $\sqrt{\frac{\sum_{i=1}^{sup(P)} v_i^2 - sup(P)\cdot m_P^2}{sup(P)-1}}$, where v_i is the A_{target} value of the i-th record in sub-population P. Hence we need to collect $sup(P)$, $\sum_{i=1}^{sup(P)} v_i$ and $\sum_{i=1}^{sup(P)} v_i^2$ to calculate m_P and s_P. All these information can be collected as we count the support of pattern P. No additional scan of data is needed.

When analyzing a significant hypothesis H, we divide the two sub-populations of H into finer sub-populations by adding an item to the defining patterns of the two sub-populations. The resultant patterns may not be frequent. Hence we need to generate not only frequent patterns, but also some infrequent patterns to make sure that all the information needed for analyzing a hypothesis is available. The generated infrequent patterns are the immediate super-patterns of some frequent patterns. In the original algorithm, a pattern is not extended if it is not frequent. In the modified version, an infrequent pattern is still extended if it has at least one frequent immediate sub-pattern. That is, a pattern is not extended only if all of its immediate sub-patterns are infrequent. We explore the search space in a way such that the sub-patterns of a pattern are generated before the pattern itself. This exploration strategy has been used in mining frequent generators [10].

B. Generating tentative hypotheses

Given a hypothesis $H = \langle P, A_{diff} = v_1|v_2, A_{target}, v_{target}\rangle$, the defining patterns of the two sub-populations of H contain one more item than the context pattern P, so they are immediate super-patterns of P. We generate tentative hypotheses as follows. For each frequent pattern P, we use it as a context and retrieve all of its frequent immediate super-patterns. We then group these super-patterns of P based on the attributes not in P. Patterns that have the same attribute not in P are placed in the same group. Patterns in the same group are then paired up to form hypotheses. The pseudo-codes for generating hypotheses are shown in Algorithm 1.

Algorithm 1 GenHypotheses

```
Input:
    $\mathcal{P}$ is the set of frequent patterns;
    $min\_sup$ is the minimum support threshold;
    $max\_pvalue$ is the maximum p-value threshold;
    $min\_diff$ is the minimum difference threshold;
    $A_{target}$ is the target attribute;
    $v_{target}$ is the target attribute value;
Description:
 1: for all frequent pattern $P$ in $\mathcal{P}$ do
 2:     Retrieve all frequent immediate super-patterns of $P$, denoted
        as $\mathcal{S}$;
 3:     Let $\mathcal{A} = \{A_1, A_2, \cdots, A_k\}$ be the set of attributes not in $P$.
        Every pattern in $\mathcal{S}$ contains exactly one attribute in $\mathcal{A}$;
 4:     Group patterns in $\mathcal{S}$ into $k$ groups based on the attributes in
        $\mathcal{A}$: $G_i = \{P'|P' \in \mathcal{S}, P'\ contains\ A_i\}$, $i = 1, 2, \cdots, k$;
 5:     for all $i$=1 to $k$ do
 6:         for all pair of patterns $P_1$, $P_2$ in $G_i$ do
 7:             if the difference between sub-populations $P_1$ and $P_2 \geq$
                $min\_diff$ then
 8:                 Calculate p-value of $H = \{P, A_i = v_1|v_2, A_{target},$
                    $v_{target}\}$, where $A_i = v_j$ is in $P_j$, $j$=1, 2;
 9:                 if p-value of $H \leq max\_pvalue$ then
10:                     Output $H$;
11:                     AnalyzeHypothesis($H$, $\mathcal{A} - \{A_i\}$);
12: Rank all the generated significant hypotheses in ascending order
    of their p-values;
```

C. Generating information for hypothesis analysis

If a hypothesis is statistically significant, we generate information for its further analysis . The pseudo-codes are shown in Algorithm 2. Given a significant hypothesis H with two sub-populations P_1 and P_2, we retrieve the immediate super-patterns of P_1 and P_2 (line 1-2). Let P_i' be an immediate super-pattern of P_i, i=1, 2. If P_1' and P_2' contain the same item that does not appear in neither P_1 nor P_2, P_1' and P_2' form a pair (P_1', P_2'). We group these pairs based on the attributes not in H (line 3). For each item $A = v$ that is not in H, *DiffLift*$(A = v|H)$ and $Contribution(A = v|H)$

Algorithm 2 AnalyzeHypothesis

```
Input:
    Hypothesis $H = \{P, A_{diff} = v_1|v_2, A_{target}, v_{target}\}$;
    $\mathcal{A} = \{A_1, A_2, \cdots, A_k\}$ is the set of attributes not in $H$;
Description:
 1: Retrieve the immediate super-patterns of $P_1 = P \cup \{A_{diff} = v_1\}$, denoted as $\mathcal{S}_1$;
 2: Retrieve the immediate super-patterns of $P_2 = P \cup \{A_{diff} = v_2\}$, denoted as $\mathcal{S}_2$;
 3: Pair super-patterns of $P_1$ and $P_2$ and then group them into $k$ groups based on attributes in $\mathcal{A}$: $G_i = \{(P_1', P_2')|P_1' \in \mathcal{S}_1, P_2' \in \mathcal{S}_2$, $P_1'$ and $P_2'$ contain a same value of $A_i\}$, $i = 1, 2, \cdots, k$;
 4: for all $i$=1 to $k$ do
 5:     for all element $(P_1', P_2')$ in $G_i$ do
 6:         Let $v$ be the value of $A_i$ contained in $P_1'$ and $P_2'$;
 7:         Calculate DiffLift$(A = v|H)$ and $Contribution(A = v|H)$ using the statistics of $P_1$, $P_2$, $P_1'$ and $P_2'$;
 8:     DiffLift$(A_i|H) = \frac{\sum_{v \in Dom(A_i)} |DiffLift(A_i=v|H)|}{|Dom(A_i)|}$;
 9:     $Contribution(A_i|H) = \frac{\sum_{v \in Dom(A_i)} |Contribution(A_i=v|H)|}{|Dom(A_i)|}$;
10:     if $\forall v \in Dom(A_i)$, DiffLift$(A_i = v|H) < 0$ then
11:         Output Simpson's Paradox ($H$, $A_i$);
12: Rank the attributes in descending order of DiffLift and $Contribution$ respectively;
13: Rank the items in descending order of DiffLift and $Contribution$ respectively;
```

are calculated using the statistics of P_1, P_2, $P_1 \cup \{A = v\}$ and $P_2 \cup \{A = v\}$ (line 4-7). The *DiffLift* and $Contribution$ of each attribute are calculated using that of its items (line 8-9). Simpson's Paradoxes are generated based on *DiffLift* (line 10-11). After all the information for analysis are generated, we then output lists of items and attributes ranked in descending order of *DiffLift* or $Contribution$ (line 12-13). At line 8-10, $Dom(A_i)$ is the set of values taken by attribute A_i.

D. Implementation

The cost of the operations in Algorithm 1 and 2 is not high except the cost for retrieving immediate super-patterns. When the number of patterns generated is very large, the cost for retrieving immediate super-patterns can be very high. We use the CFP-tree structure [11] to speed-up this operation. Here we introduce the basic features of the CFP-tree structure that are relevant to this work. For more details on the CFP-tree structure, please refer to the original paper.

CFP-tree is a very compact structure for storing patterns. It allows different patterns to share the storage of their prefixes as well as suffixes. It has been shown that a CFP-tree storing all frequent patterns can be orders of magnitude smaller than its corresponding flat representation [11]. CFP-tree supports efficient exact match and subset/superset search, which are repeatedly used in our algorithms.

Given a database D, the set of possible patterns in D can be represented as a set-enumeration tree. Figure 1 shows an example set-enumeration tree on a dataset with three attributes a, b and c. The three attributes are sorted lexicographically. Each attribute x has two values: x_1 and x_2, x=a, b, c. For brevity, we use x_i to represent item $x = x_i$, where x=a, b, c and i=1, 2. A node in the set-numeration tree represents a pattern. Each pattern can be extended only by the items belonging to the attributes before it. For example, items b_1 and b_2 can be used to extend pattern $\{c_1\}$, but they cannot be used to extend pattern $\{a_1\}$. The set of items that can be used to extend an pattern P are called candidate extensions of P, denoted as $cand\_exts(P)$.

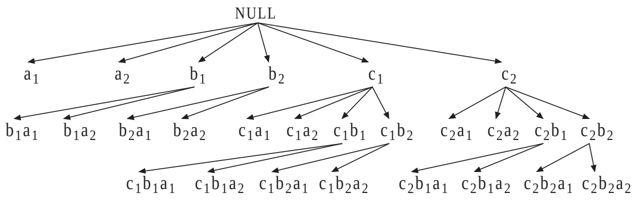


Fig. 1. An example search space tree with 3 attributes a, b, and c. Each attribute x has two values: x_1 and x_2, x=a, b, c

The CFP-tree structure [11] is a prefix-trie structure. It is similar to the set-enumeration tree but it allows suffix sharing. Figure 2(c) shows an example CFP-tree, and it stores the set of frequent patterns in the dataset shown in Figure 2(a). The dataset has four attributes: a, b, c and d. The minimum support threshold is set to 3. The set of frequent patterns are shown in Figure 2(b). For brevity, a frequent pattern $\{i_1, i_2, \cdots, i_m\}$ with support n is represented as $i_1 i_2 \cdots i_m : n$.

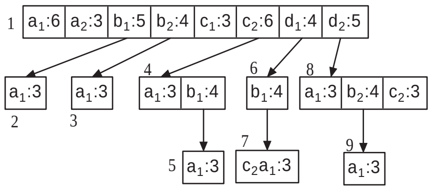
Each node in a CFP-tree consists of a number of entries. Each entry contains exactly one item if the node has multiple entires. An entry can contain multiple items if it is the only entry of its parent. For example, node 7 in Figure 2(c) has only one entry and this entry contains two items c_2 and a_1. A path starting from an entry in the root node represents one or more frequent patterns and these patterns have the same support. Let E be an entry and P be a pattern represented by the path from root to E. Entry E stores several pieces of information: (1) n items ($n \geq 1$), (2) the support of P and other information for calculating p-values, and (3) a pointer pointing to the child node of E.

1	a_1, b_1, c_2, d_1	**6**	a_1, b_2, c_2, d_2
2	a_2, b_1, c_1, d_1	**7**	a_1, b_2, c_1, d_2
3	a_1, b_2, c_1, d_2	**8**	a_1, b_1, c_2, d_1
4	a_1, b_1, c_2, d_1	**9**	a_2, b_1, c_2, d_2
5	a_2, b_2, c_2, d_2		

(a) Dataset

Freqeunt Itemsets
a_1:6, a_2:3, b_1:5, b_2:4, c_1:3, c_2:6, d_1:4, d_2:5
b_1a_1:3, b_2a_1:3, c_2a_1:3, c_2b_1:4, d_1b_1:4, d_1a_1:3, d_1c_2:3, d_2a_1:3, d_2b_2:4, d_2c_2:3,
$c_2b_1a_1$:3, $d_1b_1a_1$:3, $d_1b_1c_2$:3, $d_1c_2a_1$:3, $d_2b_2a_1$:3
$d_1b_1c_2a_1$:3

(b) Frequent patterns (min_sup=3)



(c) A CFP-tree

Fig. 2. An example CFP-tree

The CFP-tree structure allows different patterns to share the storage of their prefixes as well as suffixes, which makes it a very compact structure for storing frequent patterns. Prefix sharing is easy to understand. For example, patterns $\{c_2, a_1\}$ and $\{c_2, b_1\}$ share the same prefix c_2 in Figure 2(c). Suffix sharing is not as obvious as prefix sharing. Suffix sharing occurs when a pattern has the same support as its prefix. Let P be pattern. If there exists a candidate extension i of P such that i occurs in every records containing P, that is, $sup(P \cup \{i\}) = sup(P)$, then for every other candidate extension i' of P, we have $sup(P \cup \{i\} \cup \{i'\}) = sup(P \cup \{i'\})$. In this case, pattern $P \cup \{i\}$ and P share suffixes. In Figure 2(a), item b_1 appears in every records containing pattern $\{d_1\}$. Therefore, patterns $\{d_1\}$ and $\{d_1, b_1\}$ share suffixes, which is node 7. Node 7 represents 6 patterns: $\{d_1, c_2\}$, $\{d_1, a_1\}$, $\{d_1, c_2, a_1\}$, $\{d_1, b_1, c_2\}$, $\{d_1, b_1, a_1\}$, $\{d_1, b_1, c_2, a_1\}$.

Suffix sharing is implemented by singleton nodes in a CFP-tree. More specifically, if patterns P and $P \bigcup X$, where $X \subseteq cand\_exts(P)$, satisfy $sup(P \cup X) = sup(P)$, then the child node of P contains only one entry and the set of items of the entry is X. For example, the child node of $\{d_1\}$ is a singleton node containing item b_1. If there are more than one singleton node on a path, then the space saved by suffix sharing multiplies. The space saved by suffix sharing is dramatic. A CFP-tree storing a complete set of frequent patterns can be orders of magnitude smaller than the corresponding flat representation [11].

Constructing a CFP-tree

Constructing a CFP-tree from a dataset is almost the same as mining frequent patterns from the dataset. The main difference is the output format: a CFP-tree or a flat file. Since we store patterns in a CFP-tree, we use the CFP-tree construction algorithm described in [11] to find large sub-populations.

Besides the two modifications described in Section III-A, we made another modification to the original CFP-tree construction algorithm. In the original algorithm, items are sorted in descending order of their support. In the modified version, items that are of the same attribute are sorted to be adjacent to each other, and their attributes are sorted in ascending order of the number of distinct values. The benefit of this sorting strategy is that in Algorithm 1 and 2, the retrieved immediate super-patterns of a pattern are already grouped based on the attributes not in the pattern when they are retrieved. The cost for grouping them is thus saved.

Query processing on the CFP-tree structure

The CFP-tree structure has an important property that makes it very suitable for exact match and subset/superset search.

Property 1: Let $cnode$ be a non-singleton CFP-tree node and E be an entry of $cnode$. The item of E can appear only in the subtrees pointed by entries after E.

For example, in node 1 of Figure 2(c), item c_2 can appear in the subtrees pointed by d_1 and d_2, but it cannot appear in the subtrees pointed by a_1, a_2, b_1, b_2 and c_1. Based on this property, a sub-pattern of pattern is either a prefix of the pattern or appears on the left of the pattern; a super-pattern of a pattern appears either in the subtree pointed by the pattern or on the right of the pattern.

We need to perform two types of queries on a CFP-tree: searching for the immediate sub-patterns of a pattern and searching for the immediate super-patterns of a pattern. The former is used during the generation of large sub-populations for checking whether an infrequent pattern has a frequent immediate sub-pattern. The latter is used for hypothesis generation and analysis. We convert both queries to exact match. A pattern P has $|P|$ immediate subsets, so searching for the immediate sub-patterns of a pattern P can be converted to $|P|$ exact matches. For immediate super-pattern queries, only one additional item is allowed. Hence once there is a mismatch at an entry, i.e., the item in the entry is the additional item, then the subsequent searching in the subtree rooted at that entry must be exact match.

Algorithm 3 shows the pseudo-codes for exact match. When it is first called, P is the pattern to be searched, and $cnode$ is the root node of the CFP-tree. At each node being visited, we need to search further in at most one subtree of the node, and this subtree is pointed by the rightmost entry whose item is contained in the pattern to be searched. For example, to search pattern $P = \{c_2, a_1\}$ in the CFP-tree in Figure 2(c), we only need to visit the subtree pointed by item c_2 in node 1 because the subtrees pointed by entries before c_2 do not contain c_2, and the entries after c_2 contain items that are not in P.

Algorithm 3 ExactMatch

```
Input:
    $P$ is a frequent pattern;
    $cnode$ is the CFP-tree node currently being visited;
Description:
 1: if $cnode$ is a singleton node then
 2:     $E$=the entry of $cnode$;
 3: else
 4:     $E$=the rightmost entry of $cnode$ whose item is in $P$;
 5: Removes items of $E$ from $P$;
 6: if $P$ is emtpy then
 7:     return $E$;
 8: else if $E.child$ is empty then
 9:     return NULL;
10: else
11:     ExactMatch($P$, $E.child$);
```

IV. A Performance Study

In this section, we study the efficiency of the proposed algorithms, and demonstrate the usefulness of the generated results via two case studies. We conducted the experiments on a PC with 2.33Ghz Intel(R) Core(TM) Duo CPU and 3.25GB memory. The operating system is Windows XP professional. Our algorithms were implemented in C++ and complied using Visual Studio 2005.

Datasets	#instances	#continuous attributes	#categorical attributes	A_{target}/ v_{target}
adult	48842	6	9	class=>50K
mushroom	8124	0	23	class=poisonous
DrugTestI	141	13	74	logAUCT
DrugTestII	138	13	74	logAUCT

TABLE V
DATASETS

We used four datasets in our experiments: adult, mushroom, DrugTestI and DrugTestII. Table V shows some statistics of these four datasets. Datasets *adult*, and *mushroom* are obtained from UCI machine learning repository(`http://archive.ics.uci.edu/ml/`). Datasets *DrugTest I* and *DrugTest II* are two pharmacogenetics datasets that study the associations between mutations in several genes and drug responses. Single Nucleotide Polymorphisms (SNPs)—single DNA sequence mutations—in the target genes were genotyped in volunteers from several populations.

A. *Mining efficiency*

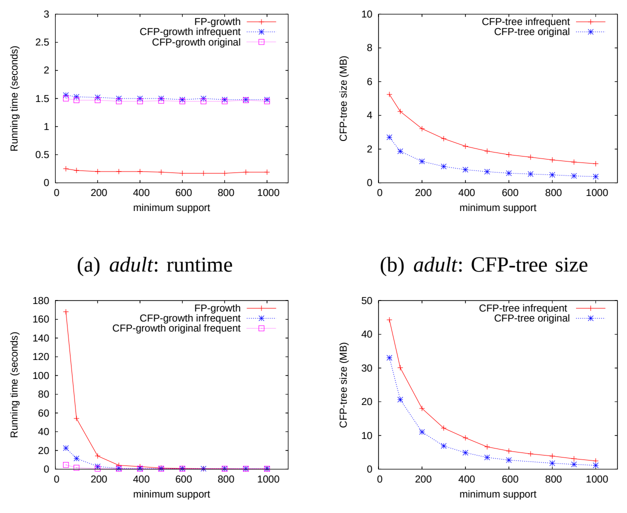
The performance of the CFP-tree construction algorithm, denoted as CFP-growth, is comparable to the start-of-the-art frequent pattern mining algorithms [11]. The first experiment studies the impact of generating infrequent patterns on the efficiency of CFP-growth and the size of the generated CFP-tree. Figure 3 shows the results on datasets *adult* and *mushroom*. The target attributes are shown in the last column of Table V. All categorical attributes except A_{target} are used as grouping attributes.

On *adult*, the running time is almost the same for both cases, while the CFP-tree size doubles when infrequent patterns are included. On *mushroom*, the running time increases greatly when infrequent patterns are included, but the size of the CFP-tree increases just slightly. The reason being that *mushroom* is much denser than *adult*, and the average pattern length in *mushroom* is much longer than that in *adult*. There is lots of sharing among patterns in *mushroom*, hence the increase in tree size is not obvious when infrequent patterns are included. However, the CFP-tree constructed on *mushroom* is much larger than that on *adult*, so the cost for subset checking is much more expensive, which increases the running time significantly. We have also studied the size of the conditional databases generated during the mining process, and we found that the size is almost the same in the two cases on both datasets.

In practice, we do not expect users generate very long patterns as the contexts of hypotheses because the contexts would be too specific. If we set a length limit to the patterns, the increase in running time and tree size should not be dramatic when infrequent patterns are included.

We also compared the running time of CFP-growth with FP-growth [7]. On *adult*, FP-growth is faster than CFP-growth. However, most of the time of CFP-growth is spent on data preprocessing such as mapping attribute values to items (integers) and converting the original dataset to a transaction dataset. On *mushroom*, CFP-growth is faster than FP-growth when min_sup is low even when CFP-growth generates infrequent patterns. This is because mushroom is very dense and many patterns are generated. FP-growth has a much longer output time than CFP-growth since it stores patterns in a flat format.

The second experiment studies the efficiency of the algorithms for hypotheses generation and analysis. The target attribute on each dataset is shown in the last column of Table V. Table VI shows that on all the four datasets, identifying significant hypotheses took less than 1 second, and the average time for generating the information for analyzing one single



(a) *adult*: runtime (b) *adult*: CFP-tree size

(c) *mushroom*: runtime (d) *mushroom*:CFP-tree size

Fig. 3. Running time of the CFP-tree construction algorithm and the size of CFP-tree with and without generating infrequent patterns that have at least one frequent immediate sub-pattern. “CFP-growth original” represents the original CFP-tree construction algorithm. “CFP-growth infrequent” refers to the modified algorithm. “CFP-tree original” refers to the CFP-tree storing only frequent patterns. “CFP-tree infrequent” refers to the CFP-tree storing both frequent and infrequent patterns.

hypothesis is below 0.1 second. The above results suggest that it is possible to identify significant hypotheses and analyze them on demand given that frequent patterns and their immediate infrequent super-patterns have been materialized. The high efficiency of the algorithms for hypotheses generation and analysis is attributed to the high efficiency of the CFP-tree structure in supporting superset queries.

The last four columns of Table VI show the number of tests performed and the number of significant hypotheses generated. On the two large datasets *mushroom* and *adult*, most of the significant hypotheses are still statistically significant after correction using Bonferroni correction and Benjamini and Hochberg’s method. However, the opposite is observed on the two DrugTest datasets, especially when Bonferroni correction is used. This is possibly because the number of records in the two DrugTest datasets is too small. This can be overcome by collecting more data.

The datasets used in our experiments are not very large. We expect that our algorithms can run on much larger datasets because our algorithms need to access the original datasets only in the large sub-population generation phase using frequent pattern mining techniques, and many efficient and scalable algorithms have been developed for frequent pattern mining. The CFP-tree storing the sub-populations may be large, but the cost for retrieving patterns from a CFP-tree is not very sensitive to the size of the CFP-tree. Hence the algorithms for hypotheses generation and analysis are scalable with respect to the size of the CFP-tree.

B. *Case study I: DrugTestII*

The objective of case study I is to demonstrate the ability of the proposed system in identifying significant hypotheses

Datasets	min_sup	min_diff	GenH	AnalyzeH	AvgAnalyzeT	#tests	#SignH	#SignH_BC	#SignH_FDR0.05
adult	500	0.05	0.22 sec	67.9 sec	0.016 sec	5593	4256	3758	4251
mushroom	500	0.1	0.55 sec	141.18 sec	0.015 sec	16949	9323	9162	9323
DrugTestI	20	0.5	0.08 sec	1.22 sec	0.061 sec	3627	20	0	0
DrugTestII	20	0.5	0.08 sec	3.26 sec	0.034 sec	4441	97	0	97

TABLE VI

RUNNING TIME (MEASURED IN SECONDS) AND NUMBER OF SIGNIFICANT HYPOTHESES GENERATED. *max_pvalue is set to 0.05 on all datasets. "GenH": time for testing all hypotheses; "AnalyzeH": time for analysis of all significant hypotheses; "AvgAnalyzeT": average time for analyzing a single hypothesis; "#test": total number of tests performed; "#SignH": #significant hypotheses with p-value$\leq max_pvalue$; "#SignH_BC": #significant hypotheses with p-value$\leq max_pvalue$/#test (Bonferroni correction); "#SignH_FDR0.05": #significant hypotheses when FDR is set at 0.05 (Benjamini and Hochberg's method).*

ID	Context	Comparing Groups	sup	$mean$	p-value
H_1	{}	Ethnic_Group=Japanese	25	**5.256**	≤ 0.001
		Ethnic_Group=Caucasian	22	**4.750**	

(a) An example of significant hypotheses in dataset *DrugTest II*

Context	Extra Attribute	Comparing Groups	sup	$mean$
{}	SNP_OATPB_14=0	Ethnic_Group=Japanese	7	**4.749**
		Ethnic_Group=Caucasian	21	**4.785**
	SNP_OATPB_14=1	Ethnic_Group=Japanese	17	**5.489**
		Ethnic_Group=Caucasian	1	**4.009**

(b) The attribute with the highest $contribution$ with respect to H_1.

TABLE VII

AN EXAMPLE OF A SIGNIFICANT HYPOTHESIS IDENTIFIED FROM DATASET *DrugTestII* AND POSSIBLE REASONS BEHIND IT. *SNP_OATPB_14 is a SNP at locus 14 of gene OATPB.*

and the reasons behind them. This experiment is conducted on *DrugTestII*. We choose continuous attribute "logAUCT" as the target attribute, and it measures the responses of the individuals to the drug.

Table 6(a) shows an example significant hypothesis identified from dataset *DrugTestII*. Hypothesis H_1 indicates that Japanese have higher logAUCT than Caucasian on average. Table 6(b) shows the SNP with the highest contribution with respect to H_1. When SNP_OATPB_14=0, the two ethnic groups have similar logAUCT values. However, when SNP_OATPB_14 =1, Japanese individuals have exceptionally high logAUCT, and the number of Japanese with SNP_OATPB_14=1 is quite large.

C. Case study II: adult

Case study II is conducted on dataset *adult*. It demonstrates the ability of the proposed system in finding interesting hypotheses and Simpson's Paradoxes behind them. Attribute "class" is chosen as the target attribute, and ">50K" is chosen as the target value.

Table 7(a) shows two example significant hypotheses observed in *adult*. Hypothesis H_1 implies that among white people, craft repairers earn more than administration clerks. However, if we take a further look by dividing the instances using the "Sex" attribute, we observe the opposite phenomenon in both "male" white people and "female" white people. That is, as shown in Table 7(b), for both sexes, craft repairers actually earn less than administration clerks. This Simpson's Paradox is caused by the fact that white men earn much more than white women on average, and many more white men work as craft repairers than white women. Therefore, the true hypothesis should be "white men earn more than white women on average", and this hypothesis is also detected by our system (H_2 in Table 7(a)).

Simpson's Paradoxes are caused by confounding factors that have associations with both the comparing attribute and the target attribute. In Table 7, the confounding factor is attribute "Sex". If a hypothesis $H = \langle P, A_{diff} = v_1|v_2, A_{target}, v_{target}\rangle$ and an attribute A form a Simpson's Paradox, it suggests two things: one is that H may be wrong, and the other is that $H' = \langle P, A = v_1|\cdots|v_k, A_{target}, v_{target}\rangle$ may be a more interesting hypothesis than H, where $v_1, \cdots, v_k$ are the set of values taken by attribute A.

V. RELATED WORK

Exploratory hypothesis testing needs to explore a large space of tentative hypotheses. We employ frequent pattern mining techniques for efficient exploration. Frequent pattern mining was first proposed by Agrawal *et al.*[2] in 1993, and it has become an important problem in the data mining area since then. Many efficient algorithms have been proposed. A frequent itemset mining implementations repository has been set up [6](`http://fimi.cs.helsinki.fi/src/`). Recently, Kirsch et al. [8] develop a methodology to identify a threshold s^* such that the set of patterns with support at least s^* can be flagged as statistically significant with a small false discovery rate. Their work focuses on the statistical significance of the frequency of the patterns. Our work studies the statistical significance of the difference between two subpopulations.

Pattern and association rule mining algorithms often produce a large number of patterns/rules, and not all of them

ID	Context	Comparing Groups	sup	$P_{>50K}$	p-value
H_1	Race = White	Occupation = Craft-repair Occupation = Adm-clerical	3694 3084	**22.84%** **14.23%**	$\leq$1.00E-08
H_2	Race = White	Sex=Male Sex=Female	19174 8642	**31.76%** **11.90%**	$\leq$ 1.00E-08

(a) Examples of significant hypotheses in dataset *adult*

Context	Extra Attribute	Comparing Groups	sup	$P_{>50K}$
Race = White	Sex = Male	Occupation = Craft-repair	3524	**23.5%**
		Occupation = Adm-clerical	1038	**24.2%**
	Sex = Female	Occupation = Craft-repair	107	**8.8%**
		Occupation = Adm-clerical	2046	**9.2%**

(b) A Simpson's Paradox behind H_1.

TABLE VIII

(A) EXAMPLES OF SIGNIFICANT HYPOTHESES IDENTIFIED FROM DATASET *adult*. (B) A SIMPSON'S PARADOX BEHIND HYPOTHESIS H_1. COLUMN $P_{>50K}$ IS THE PROPORTION OF INSTANCES WITH ANNUAL INCOME $>$50K.

are interesting. Various interestingness measures have been proposed to capture the interestingness of patterns/rules. Tan *et al.*[14] and Geng *et al.*[5] surveyed various measures proposed in the literature. Most existing pattern/rule interestingness measures involve no comparison, and they look at one pattern/rule at a time. Exploratory hypothesis testing compares the difference between patterns/rules to find deviations. We also present the findings in the form of comparison (hypotheses), which allows users to look at the data from another perspective. Furthermore, we do not stop at simply comparing two patterns/rules. We also investigate the reasons behind the difference and look at issues like Simpson' Paradox.

An interesting hypothesis consists of two rules that are significantly different. These two rules are usually ranked far apart using existing rule interestingness measures. Hence it is very tedious and time-consuming for users to manually identify interesting hypotheses from a large collection of rules, even when rules are ranked. The situation may get worse when some rules are discarded because they fail to pass the interestingness threshold. Therefore, rule ranking methods cannot be directly used for exploratory hypothesis testing.

An association rule $X \rightarrow Y$ can be viewed as a special case of hypothesis, i.e., it compares the proportion of Y in sub-population X with that in the general population. When pattern X contains multiple items, users can know the combined effect of these items, but it is hard for them to conjecture which item contributes to the difference. In exploratory hypothesis testing, we know explicitly which attribute is underlying the difference. The two approaches can be combined in order to have a more comprehensive view of the data.

One work that is close to ours is the work of Freitas [4]. Freitas's work compares the class label of a rule with that of its minimum generalizations (sub-patterns), and uses the proportion of the minimum generalizations that have a different class label as a surprisingness measure. The outputs of the work are still rules. It is difficult for users to conjecture with respect to which minimum generalization, the rule is surprising. Another difference is that Freitas's work compares rules whose transaction sets have subset-superset relationships, while our work compares rules whose transaction sets have no overlap. We also provide the statistical evidence of the results, which has been largely overlooked in the past. In the same paper, Freitas also proposes an algorithm to identify Simpson's Paradox based on the change of class labels. We find Simpson's Paradox in a different situation, and we use it to identify spurious hypotheses. Freitas gave the sketches of his algorithms, but he did not discuss how to implement them efficiently. We give an efficient implementation of the proposed algorithms. We believe that our techniques can also be applied to Freitas's work.

Recently, Liu *et al.*[9] developed a system called Opportunity Map, which casts rule analysis as OLAP operations and general impression mining. Users can use OLAP operations to explore rules. The same group of authors later found that although the operations on rule cubes are flexible, each operation is primitive and has to be initiated by the user. Finding a piece of actionable knowledge typically involves many operations and intense visual inspections. To solve this problem, Zhang *et al.*[16] proposed the idea of identifying actionable knowledge via automated comparison. In their approach, users need to manually select two attribute values to form two sub-populations for comparison, and the system then ranks all the other attributes based on their levels of interestingness. What they do is similar to the hypothesis analysis step in our approach, but the hypotheses in their system are provided by users manually instead of being generated automatically as in our approach. They do not identify Simpson's Paradoxes either.

VI. DISCUSSION AND CONCLUSION

In this paper, we have formulated the exploratory hypothesis testing and analysis problem and proposed a data mining approach to solve the problem. Conventional hypothesis testing allows just one or a few hypotheses to be tested at one time, while exploratory hypothesis testing enables researchers to use computational methods to examine large numbers of hypotheses and to identify those that have a reasonable chance of being true. In this fashion, human oversights and limitations can be complemented by computers.

This is our first attempt to connect hypothesis testing to database concepts and terminology by building on and extending existing data mining techniques. Our approach still has to be improved in many aspects:

- **Controlling false positive rate.** Controlling false positive rate in multiple hypothesis testing is a hard problem. This problem deserves much more discuss than what have been done in the paper. We used Bonferroni correction [1] and Benjamini and Hochberg's method [3] to control false positive rate. These two methods are very simple but they may inflate the number of false negatives unnecessarily. We will conduct an in-depth study of different correction methods in our system.
- **Measures for hypothesis generation and analysis.** We used the absolute difference between two sub-populations to measure the domain significance. Other measures can be used here as well. For example, odds ratio and relative risk have been commonly used when the target attribute is nominal. For hypothesis analysis, we have defined two simple measures, $Contribution$ and *DiffLift*, to measure the impact of an item or an attribute to a hypothesis. We do not claim that the measures used in the paper are the best, but we believe that they do capture useful information and can serve the purpose. In association rule mining, many interestingness measures have been proposed, but none of them is superior to all the others in every aspect. The situation is the same here. It will be difficult to find a measure that is better than all other possible measures in every aspect. We can use several measures together to get a more comprehensive understanding of the data.
- **Avoiding obvious hypotheses.** Some of the significant hypotheses generated may be obvious to domain users. If there are too many such hypotheses, then it will be difficult for domain users to find things that are really interesting to them. Prior domain knowledge can be used to filter these obvious hypotheses. We will study how to let users supply their domain knowledge to our system conveniently. Nevertheless, our system allows users to investigate the reasons behind hypotheses, which can help users to gain new insights into the cause of the hypotheses even if the hypotheses are obvious.
- **Concise representations of hypotheses.** It is well-known that the complete set of frequent patterns contains a lot of redundant information. Several concepts have been proposed to remove the redundancy from the complete set of frequent patterns, such as closed patterns, maximal patterns, generators, equivalent classes. The same thing can be done for hypotheses. The techniques developed for concise representations of frequent patterns can be employed and extended to represent hypotheses concisely.
- **Visualization of the generated hypotheses.** It is very important to organize and present the hypotheses properly so that users can explore them to find interesting things easily, especially when many significant hypotheses are generated. Currently, we rank the hypotheses in ascending order of their p-values and highlight the hypotheses with Simpson's Paradox. We will explore more sophisticated methods to visualize the output in the future.

It is not our intention to replace conventional hypothesis testing with exploratory hypothesis testing. Instead, we believe that the two approaches are complementary to each other. Exploratory hypothesis testing can be employed to find potentially interesting things in the data quickly via extensive computation, which is tedious and time-consuming for scientists to do manually, especially with the large-scale datasets that are available nowadays. The generated significant hypotheses can provide the starting points for scientists to explore, but they are not confirmatory conclusions. To confirm these hypotheses, scientists still need to perform a rigorous evaluation using conventional hypothesis testing.

REFERENCES

[1] H. Abdi. Bonferroni and $\check{S}$id$\acute{a}$k corrections for multiple comparisons". *N.J. Salkind (ed.). Encyclopedia of Measurement and Statistics. Thousand Oaks, CA: Sage*, 2007.

[2] R. Agrawal, T. Imielinski, and A. N. Swami. Mining association rules between sets of items in large databases. In *SIGMOD*, pages 207–216, 1993.

[3] Y. Benjamini and Y. Hochberg. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society*, 57(1):125–133, 1995.

[4] A. A. Freitas. On objective measures of rule surprisingness. In *PKDD*, pages 1–9, 1998.

[5] L. Geng and H. J. Hamilton. Interestingness measures for data mining: a survey. *ACM Computing Surveys*, 38(3), 2006.

[6] B. Goethals and M. J. Zaki, editors. *Proceedings of the ICDM 2003 Workshop on Frequent Itemset Mining Implementations*, volume 90 of *CEUR Workshop Proceedings*, 2003.

[7] J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In *SIGMOD*, pages 1–12, 2000.

[8] A. Kirsch, M. Mitzenmacher, A. Pietracaprina, G. Pucci, E. Upfal, and F. Vandin. An efficient rigorous approach for identifying statistically significant frequent itemsets. In *PODS*, pages 117–126, 2009.

[9] B. Liu, K. Zhao, J. Benkler, and W. Xiao. Rule interestingness analysis using olap operations. In *SIGKDD*, pages 297–306, 2006.

[10] G. Liu, J. Li, and L. Wong. A new concise representation of frequent itemsets using generators and a positive border. *KAIS*, 17(1):35–56, 2008.

[11] G. Liu, H. Lu, and J. X. Yu. Cfp-tree: A compact disk-based structure for storing and querying frequent itemsets. *Information Systems*, 32(2):295–319, 2007.

[12] H. J. Motulsky. http://www.graphpad.com/www/book/choose.htm.

[13] E. H. Simpson. The interpretation of interaction in contingency tables. *Journal of the Royal Statistical Society*, 13(2):238–241, 1951.

[14] P.-N. Tan, V. Kumar, and J. Srivastava. Selecting the right interestingness measure for association patterns. In *SIGKDD*, pages 32–41, 2002.

[15] F. Yates. Contingency table involving small numbers and the χ^2 test. *Supplement to the Journal of the Royal Statistical Society*, 1(2):217–235, 1934.

[16] L. Zhang, B. Liu, J. Benkler, and C. Zhou. Finding actionable knowledge via automated comparison. In *ICDE*, pages 1419–1430, 2009.