What you see is not what you get! Detecting Simpson's Paradoxes during Data Exploration

Yue Guo[†], Carsten Binnig[†*], Tim Kraska [†]
[†] Brown University, Providence, USA
[*]TU Darmstadt, Darmstadt, Germany

ABSTRACT

Visual data exploration tools, such as Vizdom or Tableau, significantly simplify data exploration for domain experts and, more importantly, novice users. These tools allow to discover complex correlations and to test hypotheses and differences between various populations in an entirely visual manner with just a few clicks, unfortunately, often ignoring even the most basic statistical rules. For example, there are many statistical pitfalls that a user can "tap" into when exploring data sets.

As a result of this experience, we started to build *QUDE* [1] , the first system to Quantifying the Uncertainty in Data Exploration, which is part of Brown's Interactive Data Exploration Stack (called IDES). The goal of *QUDE* is to automatically warn and, if possible, protect users from common mistakes during the data exploration process. In this paper, we focus on a different type of error, the Simpson's Paradox, which is a special type of error in which a high-level aggregate/visualization leads to the wrong conclusion since a trend reverts when splitting the visualized data set into multiple subgroups (i.e., when executing a drill-down)..

1. INTRODUCTION

Motivation: While visual data exploration tools are key to democratizing data science, they also carry new risks. For example, it is easy to mistake a visualization (e.g., a histogram showing that more females are impacted by a certain disease) for a statistically significant fact, even though it might just be a random occurrence [1]. Similar, visual data exploration tools often increase the risk that the users are unaware of the impact data errors or incompleteness of the data [2]. We recently observed that when analyzing the age distribution of patients data in the MIMIC-II data set [10]. The distribution was visualized using histograms with a bucket-size of 10 years. Nothing was suspicious about the visualized distribution, which showed that very young and older people are slightly more often in the emergency room. Only after zooming in by we found that the data set did not contain any patients data with an age between 1-9 years but a lot of patients with an age of 0. After further investigation, we found out that the data came from an emergency room for adults and 0 was used if the age was not known. Even more severe, filtering out the 0-age patients changed significantly our models, which we built using the same visual interface, for example to analyze which diseases are most common between younger and older patients.

HILDA'17 May 14, 2017,Chicago, IL, USA

ISBN 978-1-4503-5029-7/17/05…15.00

DOI: http://dx.doi.org/10.1145/3077257.3077266

	Applicants	Admitted
Men	8442	44%
Women	4321	35%

Department	Men		Women	
	Applicants	Admitted	Applicants	Admitted
A	825	62%	108	**82%**
B	560	63%	25	**68%**
C	325	**37%**	593	34%
D	417	33%	375	**35%**
E	191	**28%**	393	24%
F	373	6%	341	**7%**

Figure 1: UC Berkeley Gender Bias

As a result of this experience, we started to build *QUDE* [1] , the first system to Quantifying the Uncertainty in Data Exploration, which is part of Brown's Interactive Data Exploration Stack (called IDES). The goal of *QUDE* is to automatically warn and, if possible, protect users from common mistakes during the data exploration process. For example, in our recent SIGMOD paper [14] we describe techniques to automatically control the multi-hypothesis problem, which can arise from interactive exploring data to find interesting insights. In this paper, we focus on a different type of error, the Simpson's Paradox, which is a special type of error in which a high-level aggregate/visualization leads to the wrong conclusion as described earlier.

Contributions: The Simpson's Paradox [6], also Yule–Simpson effect, is a paradox in which a trend reverts when splitting a data set into multiple subgroups (i.e., when executing a drill-down). The most famous example for this paradox is the gender bias among graduate school admissions to University of California, Berkeley. As shown in Figure 1[1], the overall admissions of 1973 showed that men were more likely to be admitted than women. However, when looking at the largest departments the trend actually reverted.

[1]Source: https://en.wikipedia.org/wiki/Simpson's_paradox

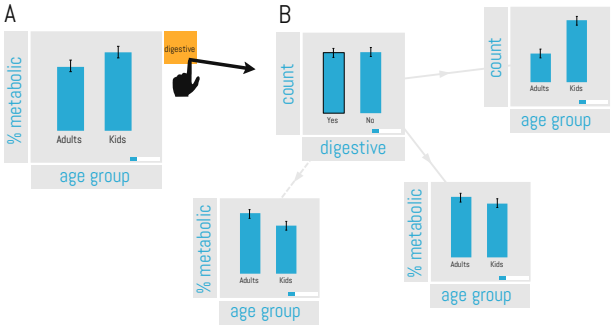


Figure 2: Simpson's Paradox Warning

In this paper, we present algorithms that detect a Simpson's Paradox online as the user explores a data set. The main contribution of this paper is the discussion of these algorithms and the evaluation on a real data set of flight records. The data set contains the details of all flights and their delays within the USA from 1987–2008, with nearly 120 million records, taking up 12 GB uncompressed.

The two main challenges to enable an efficient online detection are the sheer data size as well as the number of different attribute combinations that need to be tested. For our algorithms, we therefore applied two main techniques: (1) For dealing with large data sets, we have developed a set of approximate algorithms that can stream over the data and decide in a probabilistic manner if the data is likely contain a Simpson Paradox. This allows our algorithms to make a prediction at interactive speeds of how likely it is to find a Paradox after having seen only a small amount of data. (2) Since many different attribute combinations need to be tested to detect a Simpson Paradox, we devised a technique that leverages ideas from multi-armed bandits to find a good trade-off between exploration and exploitation. These techniques allow us to scale out to large data sets or data sets with many different attributes.

Our experiments with the flight data set show that when applying these to techniques we can efficiently warn the user at interactive speeds during an exploration session.

Outline: In Section 2 we given an overview of different statistical pitfalls and initial ideas of how to make users aware of them. Afterwards, Section 3 then discusses the problem statement this paper addresses and presents our different variants of detection algorithms to warn users if the data contains a Simpson Paradox. In Section 4, we then evaluate these algorithms using the flight data set mentioned before. Finally, we conclude in Section 6.

2. USER INTERFACE

Warning the user about a potential *Simpson's Paradox* [6] during a visual data exploration session has to be non-disruptive and needs to work even in the presence of approximate results. We integrated the algorithms presented in this paper into our visual exploration tool called Vizdom [3].

Figure 2 shows a storyboard of how a exploration session looks like to detect a Simpson's Paradox during an exploration session: Eve already has filtered down her dataset to look at patients from a particular demographic and with certain types of blood testing result values. Eve is now interested to see percentages of such patients that have blood diseases grouped by age groups (kids and adults). From looking at the histogram in **(A)** it seems like kids are more prone for these types of diseases. Afterwards, Eve however notices that the system displays a warning (yellow box). **(B)** By dragging out the warning, the system presents a set of visualizations showing that when accounting for the lurking variable "*digestive disease*" the trend reverses (i.e., kids are less prone for blood diseases for both, w digestive diseases and without).

3. DETECTION ALGORITHMS

Automatically detecting if a visualization might be impacted by the Simpson's Paradox is challenging; it requires to check if the shown relationship for any possible group-by combination will reverse.

In the following, we discuss first the naïve algorithm to find a potential Simpson's Paradox that enumerates all possible combinations and afterwards present more efficient algorithms.

3.1 Naïve Algorithm

The naïve procedure for detecting a Simpson's Paradox is shown in Algorithm 1. The main idea of the detection algorithms is that a user is currently looking at a visualization over a table T such as a histogram or a pie-chart that result from computing a relative aggregate over an attribute a such as average or a percentage grouped by a given attribute g. In our example in Figure 1 this could be the percentage of admitted students grouped by gender. The group-by attribute g is typically a binary attribute that only takes two values. For this group-by attribute our algorithm next computes the trend; i.e., for a given ordering of the group-by values, it analyses if the aggregate values are *ascending* or *descending*. In our example, we assume the ordering is *men*, *women* and the trend is descending.

Based on the aggregated result, the detection algorithm tries to find an additional group-by attribute g' (e.g., department, citizenship, etc.) in the table T that reverts the trend if the user would drill-down into that attribute as well (i.e. if she groups by g and g'). In the naïve algorithm, we therefore iterate over all possible candidate group-by attributes in T and stop once we find a group-by attribute g' that reverts the trend. While the first group-by attribute g has to be a binary group-by attribute with only two distinct values (e.g., male vs. female) , the second group-by attribute g' can have an arbitrary number of distinct values. In that case, we allow that the trend does not need to revert for all group-by values in order to detect a paradox. Instead, we allow the user to define a relative threshold t for how many values the trend needs to revert (e.g., we set t to 60% or 0.6 in our algorithm). In our example in Figure 1, the trend only reverts for 4 out of 6 departments which is above our threshold and would thus be reported as a Simpson's Paradox.

3.2 Optimization 1: One-pass Algorithm / Row and Column

When implementing Algorithm 1 in a naïve way, we can use a row-based representation of T and scan T every time we call *computeTrend*; i.e., for each candidate group-by attribute g' we enumerate. A first optimization of the naïve algorithm is to use a one-pass algorithm that only needs to scan one time over T and computes the aggregates for all

Algorithm 1: Naïve Detection Algorithm

```
Input  : Table T, Group-by Attribute g, Aggregate Attribute a
Output: Group-by Attribute s for Simpson's Paradox
Algorithm firstSimpson(T, g, a)
s ← null;
// trend can be asc or desc
trendG ← computeTrend(g, a, T);
foreach Attribute g' ∈ T − {a, g} do
    trendG' ← computeTrend({ g , g' }, a, T, t=0.6);
    if reverts(trendG,trendG') then
        s ← g';
        break;
    end
end
return s;
```

possible g'. For enabling this in an efficient way, we store T in a row-wise manner.

However, for our use case of data exploration, it is often desirable that the time a detection algorithm to find the first paradox is minimized to warn the user as early as possible. This is particular true for IDES where all results are approximated to guarantee interactive latencies. Therefore, another variant is that we store T in a columnar fashion and scan only over one pair of columns g', a when we call the function *computeTrend*. This algorithm scans the aggregate attribute column a multiple times and thus has a higher scan overhead than the row-based one-pass algorithm mentioned before. However, if the algorithm detects a Simpson's Paradox when enumerating one of the first group-by columns g', it can save I/O (which we exploit in Optimization 3 further by steering the algorithm).

3.3 Optimization 2: Approximate Algorithm

As a second optimization of the before-mentioned one-pass algorithm, we have implemented two approximate algorithms (row- and column-based) that avoid reading the complete table T. The column-based version is shown in Algorithm 2.

The main idea is that we use ideas from approximate query processing to estimate the aggregate computed on the attribute a such as average or a percentage grouped by a given group-by attribute by calling *estimateTrend*. Furthermore, for each distinct group-by value we compute a confidence intervals. In our implementation, the aggregate is computed based on the tuples seen so far and the confidence interval ϵ for each group-by value is derived using the following variation of Hoeffding's inequality (where max_a and min_a are the maximum and minimum of attribute a, n is the number of rows seen so far, and p is the confidence):

$$\epsilon = (max_a - min_a)\sqrt[2]{\frac{1}{2n} ln(\frac{2}{1-p})}$$

Based on the confidence intervals for each group-by value, we can define the trend if the intervals of the individual group-by values do not overlap anymore, which is the most important fact for detecting a Simpson's Paradox. To that end, the function *estimateTrend* can return not only the result that the aggregates of the group-by values are *ascending* or *descending* but also return *unkown* (if the confidence intervals overlap). For example, let us look at the upper table of Figure 1. Assume that the confidence interval for both aggregates is 0.5 (i.e., 50%). In that case the confidence intervals would overlap and *estimateTrend* would return *unkown*. Once the trend is not *unkown*, the approximate computation terminates. Moreover, as mentioned before when computing the trend for the candidate group-by attributes g', we allow the user to define a relative threshold t for how many group-by values the trend needs to revert (e.g., we set t to 60% or 0.6 in our algorithm).

Finally, in order to further optimize the execution, we do not iterate over the rows of T individually but use fixed-size blocks.

Algorithm 2: Approximate Detection Algorithm

```
Input  : Table T, Group-by Attribute g, Aggregate Attribute a
Output: Group-by Attribute s for Simpson's Paradox
Algorithm firstSimpson(T, g, a)
s ← null;
// trend can be asc, desc, or unkown
foreach Row r ∈ T do
    trendG ← estimateTrend(g, a, r);
    if trendG! = unkown then
        break;
    end
end
foreach Attribute g' ∈ T − {a, g} do
    foreach Row r ∈ T do
        trendG' ← estimateTrend({ g , g' }, a, r, t=0.6);
        if trendG! = unkown ∧ reverts(trendG,trendG') then
            s ← g';
            break;
        end
    end
end
return s;
```

3.4 Optimization 3: Steered Approximate

For the last optimization, the observation is that after scanning the first rows of T it is often clear that for some candidate group-by attributes g' a Simpson's Paradox is more likely than for other candidate group-by attributes.

The intuition is shown in Figure 3. Assume the user looks at the admissions of students per gender and sees that there are more males than females as shown in Figure 3 on the very left. In order to warn the user about a potential paradox, the system enumerates different different additional group-by attributes (e.g., departments, gender in that case) to find out for which attribute the trend might revert. As shown on the right hand, after scanning the first rows of the table T, simply by looking at the approximate result one can see that the likelihood to detect a Simpson's Paradox when grouping by departments is much higher than grouping by citizenship.

The main notion of the steered algorithm is to leverage the fact that some group-by attributes are more likely to reveal a Simpson's Paradox than others. Thus, instead of enumerating over all candidate attributes g' in a sequential manner, we try to steer the algorithm to the most promising attribute. To that end, the detection algorithm can be cast as a k-armed bandit problem where as each candidate attribute g' is modeled as one arm. The reward of each candidate attribute g' is defined as probability that the trend reverses. This can be modeled by using one random variable for each distinct group-by value $X_1, X_2, ..., X_n$ and and then computing the probability of $\mathrm{P}(X_1 < X_2, ... < X_n)$ if $X_1 > X2 > ... > X_n$ holds for g.

For computing this probability, we use the following formulation:

$$\mathrm{P}(X_1 \geq X_2) = \mathrm{P}(X_1 - X_2 \geq 0)$$

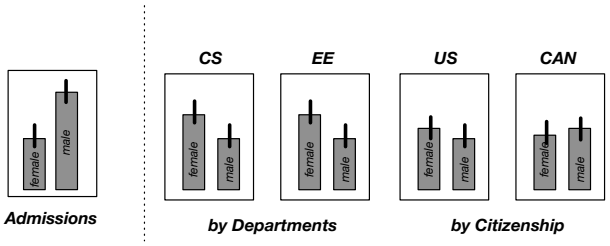


Figure 3: Likelihood of a Simpson's Paradox

Assuming a normal distribution for X_1 and X_2, we can compute the mean and variance for $Z = X_1 - X_2$ as $\mu_z = \mu_{x1} - \mu_{x2}$ and $\sigma_z^2 = \sigma_{x1}^2 + \sigma_{x2}^2$. Note that the mean of X_1 and X_2 can be approximated while streaming over T. Based on μ_z and σ_z, we can than then compute $\mathrm{P}(Z \geq 0)$. This can easily be generalized to $\mathrm{P}(X_1 < X_2, ... < X_n)$ with n random variables and thus directly be used as a reward function for the k-armed bandit problem.

4. EXPERIMENTAL EVALUATION

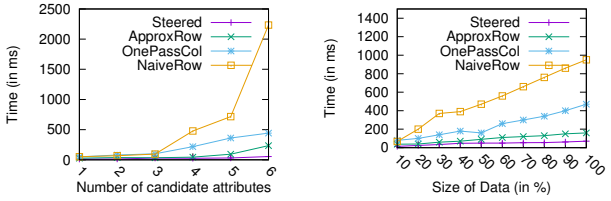
For our initial evaluation, we used a real-world data set that describes flight delays and has a size of 12GB uncompressed [2] and tested the different algorithms as described in the section before. For running the experiments, we used a machine that has two Intel Xeon E5-2660 v2 processors (each with 10 cores) and 256GB RAM. As operating system, we used Ubuntu 14.01 Server Edition (kernel 3.13.0-35-generic). The prototype algorithms were all implemented in C++ implementation and used 10 threads for the execution.

An important factor is to evaluate the efficiency of our algorithms for data exploration. A recent study [9] has shown that visual delays of more than 500ms tend to decrease both end-user activity and data set coverage due to the reduction in rates of user interaction, which is crucial for overall observation, generalization and hypothesis. For our algorithms, we also want to warn the user at interactive speeds about a Simpson's Paradox while she is looking at a given visualization. Two important factor that influence the runtime of our detection algorithms are the number of candidate group-by attributes that need to be enumerated as well as the number of rows in a data set. In the following, we therefore analyze the runtime of all detection algorithms when varying these two parameters. We report the time it takes until the algorithms find the first paradox.

Exp 1a - Varying #of Attributes: In the first experiment, we executed the different detection algorithms by varying the number of candidate group-by attributes g' available in the data set that need to be enumerated by our detection algorithms from $1-6$. Moreover, we fixed the initial group-by attribute g and the aggregate attribute a. This simulates the scenario where the user is looking a particular visualization over a and g (e.g., the histogram of admissions by gender) and the system enumerates all possible $1-6$ drill-downs to the different candidate using $\{g, g'\}$ as group-by attributes. Figure 4(a) shows the results.

The *naïve* algorithm performs the worst and requires more than $2s$ for 6 attributes. The reasons are that this algorithm requires multiple passes over the data and needs to compute multiple drill-downs for a growing number of candidate group-by attributes g'. The second algorithm that we analyzed is the *one-pass* algorithm that uses a row-oriented format. This already performs much better than the naïve since it requires only one pass independent of the number of attributes. This is directly reflected in the runtime which is approximately $6\times$ faster for 6 candidate group-by attributes. Moreover, we can see that this algorithm already achieves interactive latencies of less than 500ms. However, many real-data sets are often much larger than $12GB$ and typically have more attributes than only 6. To that end, the runtime on larger data sets with more attributes will be much higher and exceed the interactiveness threshold. The third algorithm, is the *approximate* algorithm that uses a row-oriented format. Compared to the *one-pass* algorithm, the runtime grows much slower with a increasing number of candidate group-by attributes that need to be enumerated. Finally, our *steered* algorithm is the most optimal and the runtime is the least sensitive towards the number of attributes that need to be enumerated.

[2]http://stat-computing.org/dataexpo/2009/the-data.html



(a) Varying #of Attributes (b) Varying Data Size

Figure 4: Efficiency of Detection Algorithms

Exp 1b - Varying Data Size: In this experiment, we used the same data set as before but fixed the number of candidate attributes to 5 and varied the number of rows in T by creating samples of different sizes ($10\% - 100\%$). While the naïve and the one-pass algorithm grow with the data size and exceed 500 ms, the approximate and steered algorithm are almost independent. Moreover, again the steered algorithm is the most optimal one since it exploits the most likely candidate first.

5. RELATED WORK

While several authors have focused on the important issue of discovering unexpected/surprising patterns in data mining [11, 12, 13, 8], only a few algorithms discuss on how to detect a Simpson's Paradox [5, 7]. However, all these algorithms have not been designed to efficiently warn the user in an exploration session at interactive speeds. For example, [5, 7] discuss algorithms that are similar to the idea of our naïve algorithm, which is neither able to warn users at interactive speeds nor scales to larger data sizes or data sets with many different attributes.

Another interesting approach for detecting a Simpson's Paradox is discussed in [4]. Different from [5, 7], in [4] the authors discuss the idea of a magnitude of a Simpson's Paradox. Different from our approach, the paper however does not use the magnitude to steer the computation and achieve interactive response times. Instead the authors use the magnitude of a Simpson's Paradox to rank the different paradoxes presented to the user.

6. CONCLUSION AND FUTURE WORK

In this paper, we made a first attempt to automatically detect the Simpson's Paradox For the detection, we presented different algorithms that leverage ideas from approximate query processing and steering using a k-armed bandit strategy, which significantly improved the run-time to detect a potential Simpson's paradox. For the future, we see two main directions.

Finding only statistical significant Simpson's Paradoxes: By automatically trying to hundreds or more attribute combinations to find a potentially Simpson's Paradox the chance increases to mark a random occurrence as a Simpson's Paradox. The underlying assumption here is, that the aggregates/visualization are actually used to derive insights (e.g., that an admission bias exist preferring men over women) rather than just a descriptive statistic (e.g., more men were admitted than women), and as such, that the insight and the reverse of it (i.e., caused by the Simpson's Paradox) have to be statistical significant. This problem is strongly related to the visual recommender or hypothesis finder problem as outlined in [1] and likely solvable by similar solutions.

Detecting other Statistical Pitfalls: We believe that the here devised techniques based on bandits could potentially be modified to find other visual problems. For example, we believe that we could use a similar bandit-based approach to discover *Pseudoreplication*, a very common problem with data collected in life-sciences. Furthermore, another challenge is to test for the many different pitfalls at interactive speeds. One idea is to memoize intermediate results of our detection algorithms and reuse them to detect other pitfalls.

7. ACKNOWLEDGMENTS

This research is funded in part by the Intel Science and Technology Center for Big Data, DARPA Award 16-43-D3M-FP-040, NSF CAREER Award IIS-1453171, NSF Award IIS-1514491, NSF Award IIS-1562657, Air Force YIP AWARD FA9550-15-1-0144, and gifts from Google, VMware, Mellanox, and Oracle.

8. REFERENCES

[1] C. Binnig et al. Toward sustainable insights, or why polygamy is bad for you. In *CIDR*, 2017.

[2] Y. Chung et al. Estimating the impact of unknown unknowns on aggregate query results. In *SIGMOD*, pages 861–876, 2016.

[3] A. Crotty et al. Vizdom: Interactive analytics through pen and touch. *PVLDB*, 8(12), 2015.

[4] C. C. Fabris et al. Discovering surprising instances of simpson's paradox in hierarchical multidimensional data. *IJDWM*, 2(1):27–49, 2006.

[5] C. Glymour et al. Statistical themes and lessons for data mining. *Data Min. Knowl. Discov.*, 1(1):11–28, 1997.

[6] R. Kievit et al. Simpson's paradox in psychological science: a practical guide. *Frontiers in psychology*, 4, 2013.

[7] R. Kievit et al. Simpson's paradox in psychological science: a practical guide. *Frontiers in Psychology*, 4:513, 2013.

[8] Y. Kuo et al. Mining surprising patterns and their explanations in clinical data. *Applied Artificial Intelligence*, 28(2):111–138, 2014.

[9] Z. Liu et al. The effects of interactive latency on exploratory visual analysis. *IEEE Trans. Vis. Comput. Graph.*, 20(12), 2014.

[10] Mimic ii data set. https://mimic.physionet.org/. Accessed: 2014-11-03.

[11] B. Padmanabhan et al. A belief-driven method for discovering unexpected patterns. In *KDD*, pages 94–100, 1998.

[12] A. Silberschatz et al. What makes patterns interesting in knowledge discovery systems. *IEEE Trans. Knowl. Data Eng.*, 8(6):970–974, 1996.

[13] E. Suzuki et al. Discovery of surprising exception rules based on intensity of implication. In *PKDD*, pages 10–18, 1998.

[14] Z. Zhao et al. Controlling false discoveries during interactive data exploration. In *SIGMOD*, 2017.