# Confidence of Conditional Functional Dependencies Using Reservoir Sampling

**Tiwari Pushpa** [1], **K. Dhana sree** [2], **Dr.R.V.Krishnaiah**[3]
[1]M.Tech Student, [2]Associate Professor, [3]Principal
*[1]M.Tech Student, Department of CSE, DRK Institute of Science & Technology, Hyderabad, Ap, India,*
*[2]Associate Professor, Department of CSIT, DRK Institute of Science & Technology, Hyderabad, Ap, India,*
*[3]Principal, Department of CSE, DRK Group of Institutions, Hyderabad, Andhra Pradesh, India,*

## Abstract-

*Conditional functional dependencies (CFD), are extensions of classical functional dependencies that apply to certain subset of the relations, as specified in pattern tableau. The support and confidence of a given tableau are key properties in understanding and exploring data quality. Consider a data warehouse containing large quantities of historical data. Careful analysis of a particular relation over time may have revealed various tableaux that have high support and confidence, checking the support and confidence on new data as they arrive hour by hour is an important quality check, and can reveal new trends if the support or confidence change suddenly. Given sufficient computational resources, it is straightforward to compute the confidence of a CFD by first sorting the input on attributes of the relation. For each group in the support set, the number of matching rows and the frequency of the most common consequent that satisfies all applicable assertions, can be found, allowing direct computation of the confidence. So if proper samples of data are provided, calculating the confidence is easier. Our focus lies in coming up with providing some technique of providing sampled data. Existing sampling schemes are not sufficient as they do not provide us with reliable sampled data for which the confidence is to be calculated. We have proposed an algorithm in its naïve version, which follows the concept of reservoir sampling. The algorithm proposes a skip random variable S(n, t), which is generated in constant time by generating its continuous counterpart and then correcting it so that it has exactly the desired distribution function F(s), by a modification of von Neumann's rejection-acceptance method. Proposed Algorithm is significantly faster than other reservoir sampling algorithms.*

***Keywords****-Conditional Functional Dependencies (CFD), Functional Dependencies, pattern tableau, Support, Confidence, Reservoir Sampling, Random variable, rejection-acceptance method.*

## I. INTRODUCTION:

Conditional functional dependencies (CFDs) have recently been proposed to characterize the semantics of complex data and facilitate data cleaning [1, 2, 3, 4, 5, 6, 7, 8]. A CFD is a functional dependency (FD) that holds on a subset of the relation specified in an accompanying pattern tableau. The support of a CFD is the fraction of rows that match the tableau. Its confidence, defined formally below, is the fraction of rows satisfying the functional dependency amongst those that match the tableau. The support and confidence of a given tableau are key properties in understanding and exploring data quality. Consider a data warehouse containing large quantities of historical data. Careful analysis of a particular relation over time may have revealed various tableaux that have high support and confidence, checking the support and confidence on new data as they arrive hour by hour is an important quality check, and can reveal new trends if the support or confidence change suddenly.

**Table 1**. Customers Detail

| SN | CC | AC | PN | NM | STR | CT | ZIP |
|---|---|---|---|---|---|---|---|
| 1 | 01 | 908 | 1111 | MIK | TREE | MH | 07974 |
| 2 | 01 | 908 | 1111 | MIK | TREE | MH | 07974 |
| 3 | 01 | 212 | 2222 | JOE | 5TH STR | NY | 01202 |
| 4 | 01 | 908 | 2222 | JIM | ELM STR. | MH | 07974 |
| 5 | 44 | 131 | 3333 | BEN | HIGH STR. | EDI | EH4 |
| 6 | 44 | 131 | 4444 | IAN | HIGH STR. | EDI | EH4 |
| 7 | 44 | 908 | 4444 | IAN | PORT PI | MH | WIB |
| 8 | 01 | 131 | 2222 | SEA | 3RD STR. | UN | 01202 |

Each row specifies phone details of a customer. CC stands for country code, AC for area code, PN for (phone number), NM for (name), STR for street, CT for city, and ZIP for zip code.
The following two FDs hold in Table 1.
f1 : [CC, AC] →CT
f2 : [CC, AC, PN] →STR
Conditional functional dependencies are
ø0 : ([CC, ZIP] →STR, (44, - ‖ - ))
ø1 : ([CC, AC] →CT, (01, 908‖MH))
ø2 : ([CC, AC] →CT, (44, 131 ‖ EDI))
ø3 : ([CC, AC] →CT, (0, 212 ‖ NY))
CFDs ø0, ø2, ø2, ø3 specify special cases of FD f1. FD [CC, ZIP] →STR does not hold in data set 1, but holds for the part of the data set where CC = 44. This has been summarized as a CFD ø0. An FD can be considered as a special case of CFD when the tuple pattern contains only the unnamed variable.
FD f1 : [CC, AC] →CT
CFD f1 : ([CC;AC] →CT,( - -, ‖ -)).

**Definition (Support of CFD)**

The support of a CFD[10], is the fraction of records in a data set that satisfies the CFD.For example, the support of Á1 is 3/8 since three tuples in the example data set satisfy Á1. Given a minimum support requirement, a CFD is frequent if its support is at least as large as the minimum support. The support is the fraction of tuples that match at least one pattern in the tableau.

**Definition (confidence of CFD)**
Following prior work[10]. We define the confidence, Cø R, on a relation R is maximum fraction of its support set that can be retained, so that if all other tuples were deleted, the remaining ones would satisfy the embedded FD and all relevant assertions.
Thus, the confidence can be computed exactly, given a table and a CFD, at the cost of sorting the whole table to form the groups. Without loss of generality, we can write the table R as a (multi)set of rows ri = (xi, yi), where xi € X is the antecedent, yi € Y is the consequent, and all other attributes are dropped.

Denote the total number of rows in R as N.
Let Nx as |{ri : xi = x}|, the number of rows sharing the antecedent x (the size of the group of x),
Nx,y as |{ri : xi = x Λ yi = y Λ V tp €Tp, tp[X] ≈ x → tp[Y ] ≈ y}|,
The number of rows with antecedent x and consequent y that satisfy all applicable assertions.
Let support set of CFD ø on R as sø (R), i.e.,Sø (R) = {ri : ∃tp €Tp. ri[X] ≈ tp[X]}, and define Sø (R) = |sø (R)|/N as the support of ø on R.

Then Cø(R)=X $\sum_{X\in s\emptyset(R)[X]} \max Nx,y/s\emptyset(R) = \sum_{x\in s\emptyset(R)[X]} Nx/|s\,\emptyset(R) \max y\ Nx,y|$
The fraction Nx/N can be interpreted as the “support” of x, while the fraction max y Nx,y/Nx can be thought of as a “confidence” of the group x.

## II. PROBLEM:

Given sufficient computational resources, it is straightforward to compute the confidence of a CFD by first sorting the input on attributes X Y . For each group in the support set, the number of matching rows (Nx), and the frequency of the most common consequent that satisfies all applicable assertions (max y Nx,y) can be found, allowing direct computation of the confidence.

We are interested in the case, where input is very huge. Records of data stored in the data warehouse. So reading each and every record is time consuming. There may be situations when CFD may not hold exactly i.e. some rows in the support set may agree with antecedent (X) but may disagree on consequent(Y). Note that simple uniform sample of rows is sufficient to estimate the support of any given CFD. Moreover any rows in the input which are not in the support set of specified CFD can be ignored. Instead we should first come up with sample records, then calculate the support and confidence for the given CFD.

## III. DRAWBACK OF EXISTING SYSTEM:

So if proper samples of data are provided calculating the confidence is easier. Our focus lies in coming up with providing some technique of providing sampled data.
Existing simple sampling schemes are not sufficient as they do not provide us with reliable sample data for which the confidence is to be calculated.

### Uniform Row Sampling:

Consider a scheme which samples a set of rows uniformly from the relation, and then tries to use this information to estimate the confidence. Let the large group in R have a unique consequent, so the confidence is either 1 or 0.75, depending on how the small groups are arranged. The sampling scheme is unlikely to pick two rows from the same small group, and so has no information to distinguish the two cases, unless the sample size is Ω(√N).

## Uniform Group Sampling:

Consider a scheme that samples uniformly from the groups (so each group is equally likely to be picked), and collects some information about the sampled groups. Let the small groups have a unique consequent, so the confidence
is either 1 or 0.5+2/N, depending on the consequents in the large group. However, unless some information is collected about the large group, then there is no way to distinguish the two cases. This only occurs if the sample size is $\Omega(\sqrt{N})$.when groups are sampled uniformly.

These examples show that simple uniform sampling approaches alone will not suffice, and instead we will need to consider more sampling schemes.

### Two pass solution

1] Sampling approach
2] Heavy hitters

In this paper we discuss reservoir sampling method We describe a solution to the CFD confidence which takes two passes through the data . In the first pass it samples tuples uniformly from the relation and in second pass for each sampled tuple the confidence of the respective group is calculated. [10]

## IV PRELIMINARIES:

### Reservoir algorithms:

The first step of any reservoir algorithm is to put the first n records of the file into a "reservoir"[9] The rest of the records are processed sequentially, records can be selected for the reservoir only as they are processed. An algorithm is a reservoir algorithm if it maintains the invariant that after each record is processed a true random sample of size n can be extracted from the current state of the reservoir.

We want n records in the sample to be in the same order as they appear in the file, so that they can be accessed sequentially .it is ideally suited to online use. Measure the performance based on CPU time.

### Previous work done:

Algorithm S [11,18] has been developed. This method sequentially selects n records at random from a file containing N records where 0<= n<= N. It generates a random variable and skips the next record if NU > n, else repeats the process. It selects the next record in the file for the sample and sets n = n+1, N= N-1.

## V. PROPOSED WORK:

The limiting restrictions on algorithms for this sampling problem are that the records must be read sequentially and at most once. This means that any algorithm for this problem must maintain a reservoir that contains a random sample of size n of the records processed so far.
Proposed algorithm revolves around random variable. We define S(n, N) to be random variable that counts the number of records to be skipped before selecting the next record.
Parameter n is the number of records remaining to be selected and N is the total number of records left in the file.
The range of S(n, N) is set of integers in the interval $0 <= s <= N$
The distribution Function [9]
$F(s) = Prob(S<=s)$, for $0 <= s <= N$ can be expressed as
$F(s) = l - [(N- n)^{\underline{s+1}}] / N^{\underline{s+1}}$
$F(s) = 0$ for $s < 0$
$F(s) = 1$ for $s >= N - n$
(The notation $a^{\underline{b}}$ denotes the "falling power"
$a(a - 1) \ldots (a - b + 1) = a!/(a - b)!$, and the corr. notation $a^{\overline{b}}$ denotes the "rising power"
$a(b + 1) * . . (a + b - 1) = (a + b - l)! / (a - l)!$.)
The Probability function f(s) [9]
$f(s) = Prob(S = s)$, for $0 <= s <= N-n$
$= F(s) - F(s-1)$
$= (n/N) *(N-n)s / (N-1)s$
$f(s) = 0$ for $s< 0$ and $s > N-n$.

The skip random variable S(n, N) is generated in constant time by generating its continuous counterpart and then correcting it so that it has exactly the desired distribution function F(s), by modification of Von Neumann's rejection-acceptance method.
Assume X is either a continuous or an integer random variable. Let g(x) be the density function of X if X is continuous or f(x) be probability function if X is integer value.
We choose a constant 'c' >= l
so that $f(\lfloor x \rfloor) <= cg(x)$
In order to generate S, we generate X and a random variate U which is uniformly distributed on unit interval.
The expected value of S is $\varepsilon sf(s) = \frac{N-n}{n+1}$

### Proposed Algorithm 'P'

(1) If n>= N, we use previously discovered algorithms, as the proposed algorithm involves the overhead of calculating X, $h(\lfloor x \rfloor)$, g(x). Proposed

algorithm only when $n <= \propto N$, where $\propto$ is a constant value ranges between (0.05, 0.15).

(2) Generate U and X . if U <= h(( └ X┘ )/ cg(X), probability is high,
then set S = └ X┘ , go to step 4

(3) If U <= f( └ X┘ ) / cg(X),
then set S = └ X┘ , else step 2.

(4) Select the (S+1)nth record. Skip over the next S(n,N) records in the file and select the following one for the sample.
Set N = N – S(n, N) - 1 and n = n-1. Go to step 2, if n >0.

## VI. RESULTS:

In this section we give three optimizations that dramatically improve the running time of the naive version of Algorithm D.

1] Threshold Optimization. Proposed algorithm works best when $n <= \propto N$.

2] Subroutine optimization. We can speed up Algorithm 'P' by a factor of almost 2 by cutting in half the number of operations of the form $x^y$, where x and y are either real (floating-point) numbers or large integers.

3]Random Optimization. Algorithm 'P' reduces the number of calls to RANDOM by a factor of 1/3

| Algorithms | Average Execution Time |
|---|---|
| S | 17N |
| A | 4N |
| C | $8n^2$ |
| P | 55n |

**Table 2.** Comparison of various algorithms

## VII. CONCLUSION:

The proposed algorithm is a naïve version for random sample of size n from file of N records.

(1) The algorithm uses rejection acception technique to do the sampling in optimum time. Proposed Algorithm is significantly faster than other reservoir sampling algorithm as shown the table.
(2) Main idea is generating S quickly by generating fast approximation and correcting it so that its result distribution is the desired F(s)
(3) Elimination of costly calls to mathematical subroutines
(4) Speed is achieved by accepting the value of 'S' only when U <= <= h(( └ X┘ )/ cg(X), else we reject and repeat the process with a new set of X, U.
(5) The calculation of probability function f(s) is expensive and requires O(min{n,S}) time. Instead we quickly compute approximation h(s), such that h(s) <= f(s). Only with low probability we calculate f(s)
(6) All reservoir algorithms require Q(n(1 + log(N/t))) time, whereas proposed runs in O(n) time and generates approximately n uniform random variables and performs n exponentiation operations.

.

## REFERENCES

[1] L. Bravo, W. Fan, F. Geerts, and S. Ma. Increasing the expressivity of conditional functional dependencies without extra complexity. In IEEE International Conference on Data Engineering, 2008.
[2] L. Bravo, W. Fan, and S. Ma. Extending dependencies with Bases, 2007.
[3] F. Chiang and R. Miller. Discovering data quality rules. In International Conference on Very Large Data Bases, 2008.
[4] G. Cong, W. Fan, F. Geerts, X. Jia, and S. Ma. Improving data quality: Consistency and accuracy. In International Conference on Very Large Data Bases, 2007.
[5] W. Fan, F. Geerts, X. Jia, and A. Kementsietsidis. Conditional functional dependencies for capturing data inconsistencies. ACM Trans. Database Syst., 33(2), 2008.
[6] W. Fan, F. Geerts, L. Lakshmanan, and M. Xiong. Discovering conditional functional dependencies. In IEEE International Conference on Data Engineering, 2009.
[7] W. Fan, S. Ma, Y. Hu, J. Liu, and Y. Wu. Propagating functional dependencies with conditions. In International Conference on Very Large Data Bases, 2008.
[8] Lukasz Golab, Howard Karloff, Flip Korn, Divesh Srivastava, and Bei Yu. On generating near-optimal tableaux for conditional functional dependencies. In International Conference on Very Large Data Bases, 2008.
[9] J. S. Vitter. Random sampling with a reservoir. ACM Transactions on Mathematical Software, 11(1):37–57, March 1985.
[10] Graham Cormode, Lukasz Golab, Flip Korn, Andrew Mcgregor, Divesh Srivastava and Xi Zang, Estimating the confidence of conditional functional dependencies
[11] VITTER, J. S. Faster methods for random sampling. Commun. ACM 27, 7 (July 1984). 703-718.

[12] VITTER, J. S. An Efficient Algorithm for Sequential. Random Sampling. ACM Transactions on Mathematical Software,Volume 13 Issue 1, March 1987, Pages 58-67
[13] FELLER, W. An Introduction to Probability Theory and Its Applications, vol. I, 3rd ed. Wiley, 1968.
[14] FELLER, W. An Introduction to Probability Theory and Its Applications, vol. II, 2nd ed. Wiley, 1971
[15] FAN, C. T., MULLER, M. E., AND REZUCHA, I. Development of sampling plans by using sequential (item by item) selection techniques and digital computers. Am. Stat. Assoc. J. 57 (June 1962), 387-402.

## Author:

**TIWARI PUSHPA** has completed MCA from Maharaja Sayajirao University, Baroda, and pursuing M.Tech (C.S.E) in DRK Institute of Science and Technology, JNTUH, Hyderabad, Andhra Pradesh, India. Her main research interest includes Data Mining & Databases.

**K. Dhana sree** M.Tech, (Ph.D) associate professor dept. Of csit, drk IST, jntu, Hyderabad. Research area

Datamining, computer networks and securtiy computing.previously published 3 international journal papers and 1 international conference paper.

**Dr.R.V.Krishnaiah**, did M.Tech (EIE) from NIT Waranagal, MTech (CSE) form JNTU, Ph.D, from JNTU Ananthapur, He has memberships in

professional bodies MIE, MIETE, MISTE. His main research interests include Image Processing, Security systems, Sensors, Intelligent Systems, Computer networks, Data mining, Software Engineering, network protection and security control. He has published many papers and Editorial Member and Reviewer for some national and international journals.