Maintaining Bernoulli Samples Over Evolving Multisets

Rainer Gemulla Wolfgang Lehner
Technische Universität Dresden

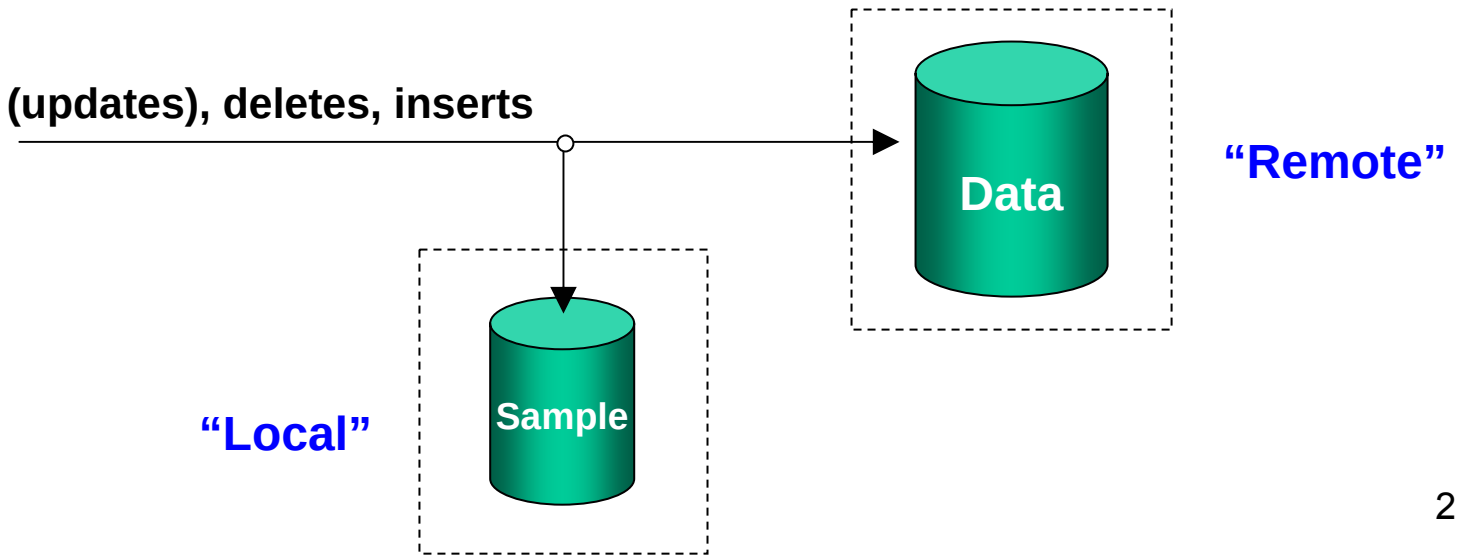
Peter J. Haas
IBM Almaden Research Center

TECHNISCHE UNIVERSITÄT DRESDEN

IBM

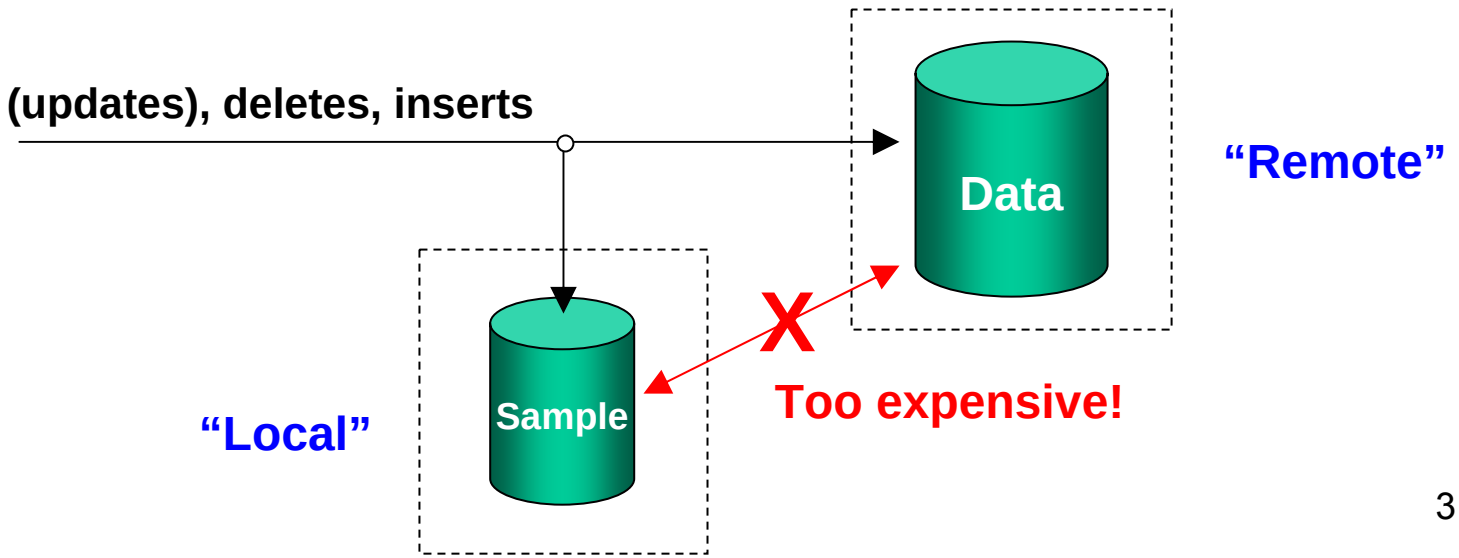
Motivation

- Sampling: crucial for information systems
  - Externally: quick approximate answers to user queries
  - Internally: Speed up design and optimization tasks
- Incremental sample maintenance
  - Key to instant availability
  - Should avoid base-data accesses



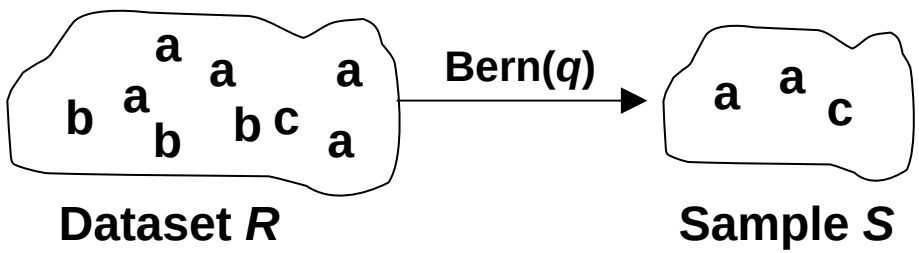
Motivation

- Sampling: crucial for information systems
  - Externally: quick approximate answers to user queries
  - Internally: Speed up design and optimization tasks
- Incremental sample maintenance
  - Key to instant availability
  - Should avoid base-data accesses



What Kind of Samples?

- Uniform sampling
  - Samples of equal size have equal probability
  - Popular and flexible, used in more complex schemes
- Bernoulli
  - Each item independently included (prob. = *q*)
  - Easy to subsample, parallelize (i.e., merge)
- Multiset sampling (most previous work is on sets)
  - Compact representation
  - Used in network monitoring, schema discovery, etc.



Outline

- Background
  - Classical Bernoulli sampling on sets
  - A naïve multiset Bernoulli sampling algorithm
- New sampling algorithm + proof sketch
  - Idea: augment sample with “tracking counters”
- Exploiting tracking counters for unbiased estimation
  - For dataset frequencies (reduced variance)
  - For # of distinct items in the dataset
- Subsampling algorithm
- Negative result on merging
- Related work

Classical Bernoulli sampling

- Bern(q) sampling of sets
  - Uniform scheme
  - Binomial sample size

$$P\{\,|S| = n\,\} = B(n;|R|,q) = \binom{|R|}{n} q^n (1-q)^{|R|-n}$$

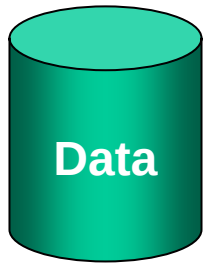
- Originally designed for insertion-only
- But handling deletions from R is easy
  - Remove deleted item from S if present

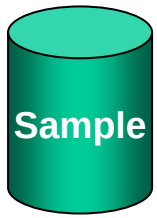
Multiset Bernoulli Sampling

- In a Bern(q) multiset sample:
  - Frequency $X(t)$ of item $t \in T$ is Binomial($N(t)$,q)
  - Item frequencies are mutually independent
- Handling insertions is easy
  - Insert item t into R and (with probability q) into S
  - I.e., increment counters (or create new counters)
- Deletions from a multiset: not obvious
  - Multiple copies of item t in both S and R
  - Only local information is available

A Naïve Algorithm

***N*(*t*) copies of item *t* in dataset**

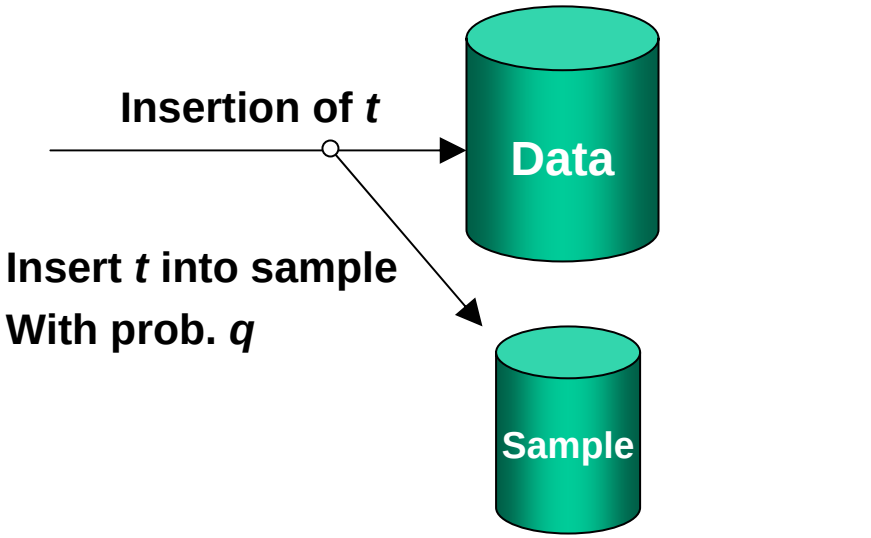




***X*(*t*) copies of item *t* in sample**

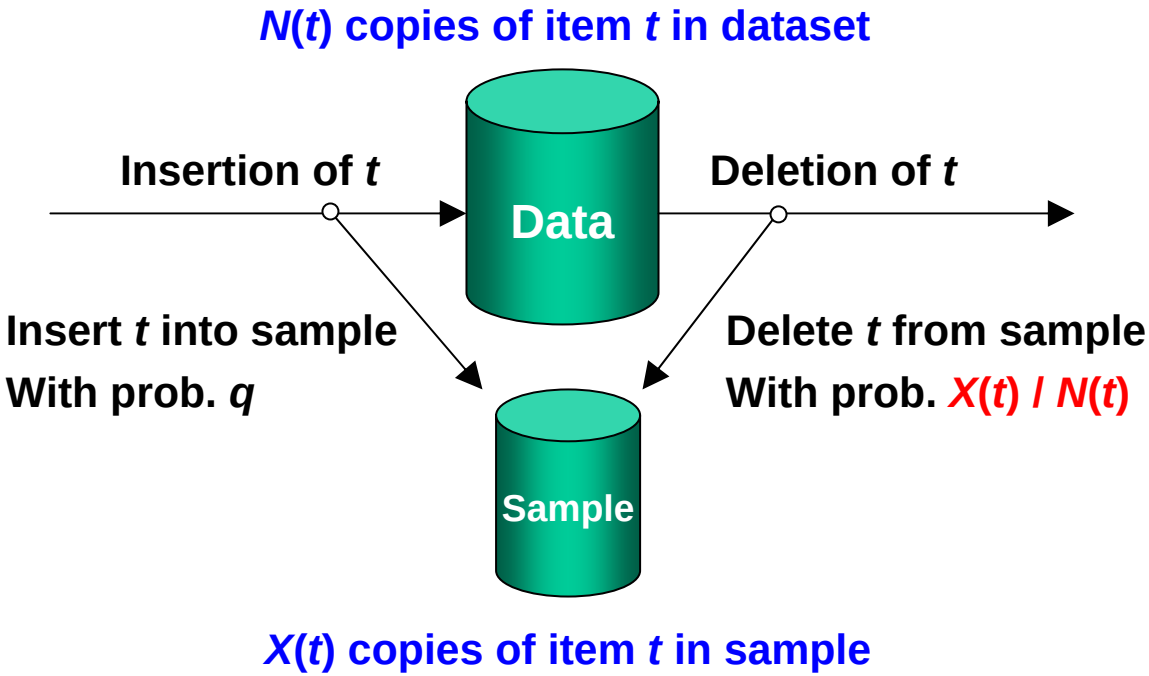
A Naïve Algorithm

***N*(*t*) copies of item *t* in dataset**

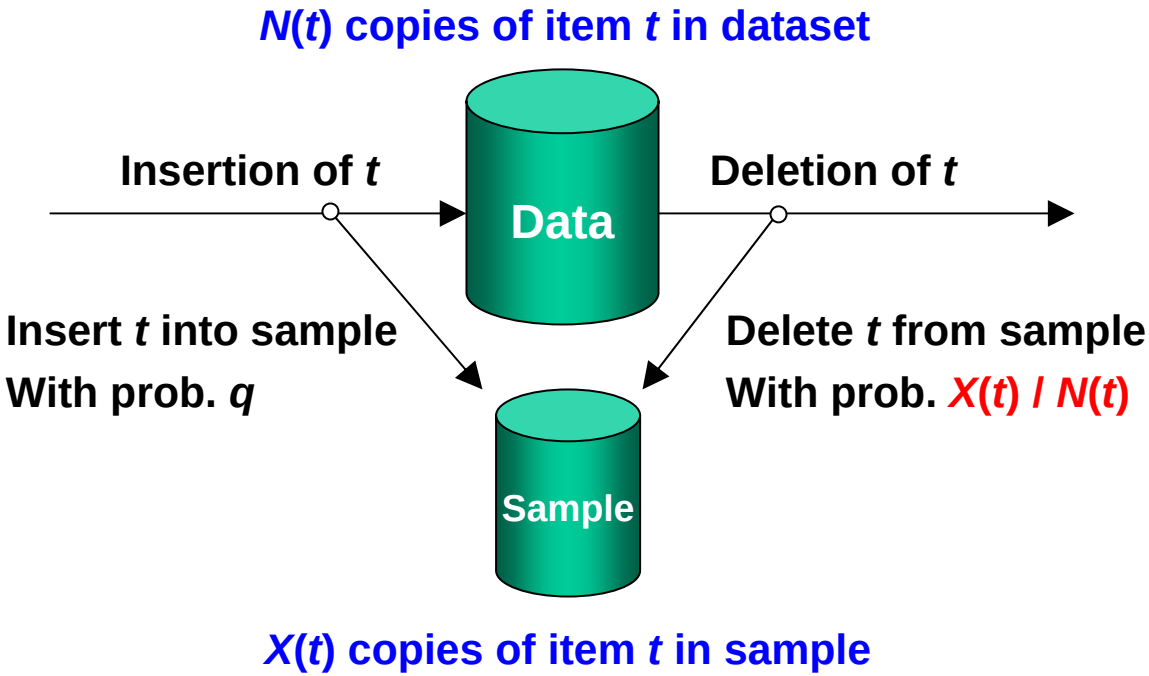


***X*(*t*) copies of item *t* in sample**

A Naïve Algorithm



A Naïve Algorithm



- **Problem:** must know $N(t)$
  - Impractical to track $N(t)$ for every distinct t in dataset
  - Track $N(t)$ only for distinct t in *sample*?
    - **No:** when t first enters, must access dataset to compute $N(t)$

New Algorithm

- Key idea: use **tracking counters** (GM98)
  - After j-th transaction, *augmented* sample S_j is

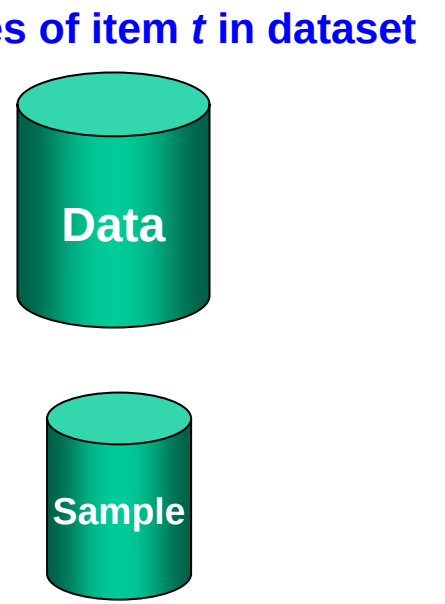
    $S_j = \{ (X_j(t), Y_j(t)) : t \in T \text{ and } X_j(t) > 0 \}$
    - $X_j(t)$ = frequency of item t in the sample
    - $Y_j(t)$ = net # of insertions of t into R since t joined sample

New Algorithm

- Key idea: use **tracking counters** (GM98)
  - After j-th transaction, *augmented* sample S_j is

    $S_j = \{ (X_j(t), Y_j(t)) : t \in T \text{ and } X_j(t) > 0 \}$

    - $X_j(t)$ = frequency of item t in the sample
    - $Y_j(t)$ = net # of insertions of t into R since t joined sample

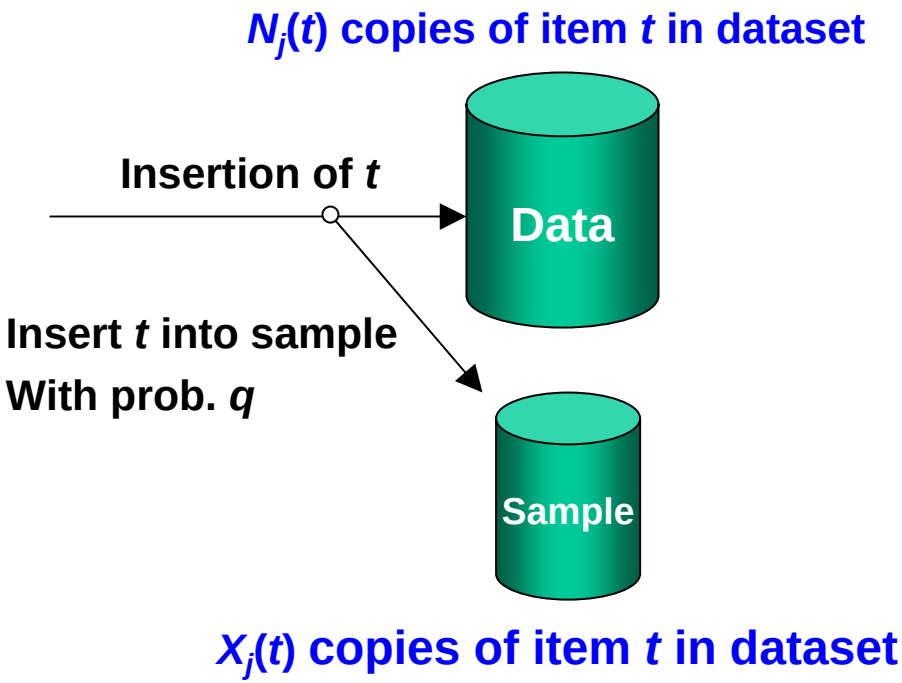


New Algorithm

- Key idea: use **tracking counters** (GM98)
  - After *j*-th transaction, *augmented* sample S_j is

    $S_j = \{ (X_j(t), Y_j(t)) : t \in T \text{ and } X_j(t) > 0 \}$

    - $X_j(t)$ = frequency of item t in the sample
    - $Y_j(t)$ = net # of insertions of t into R since t joined sample

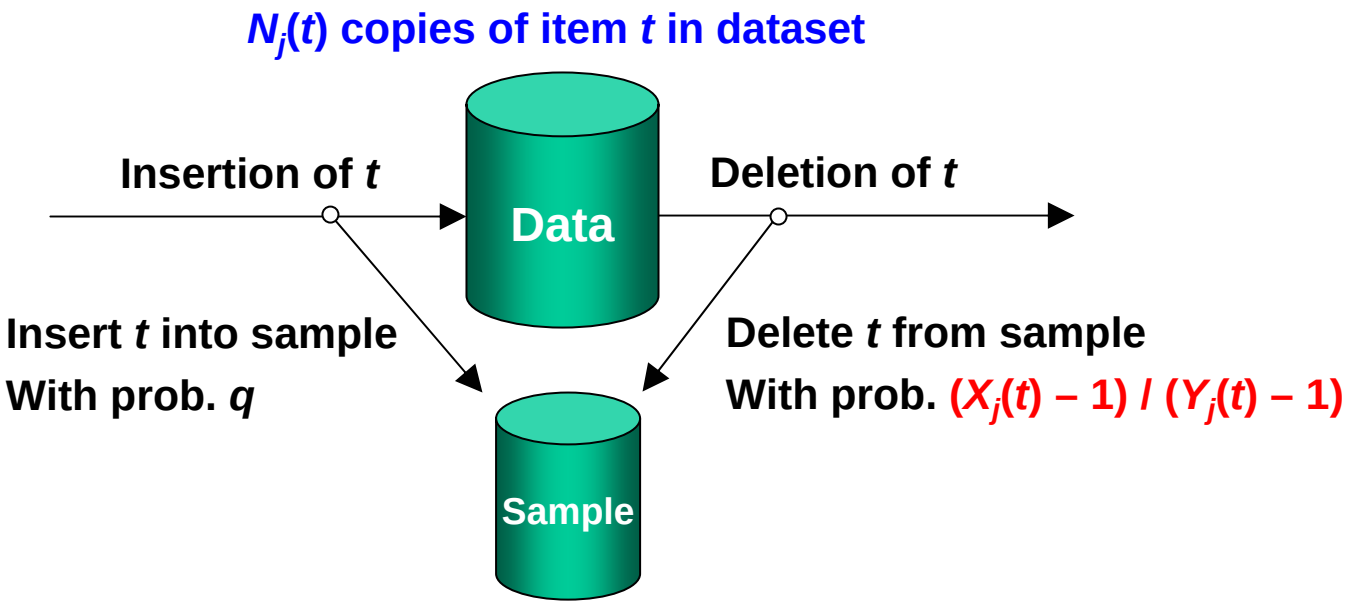


New Algorithm

- Key idea: use **tracking counters** (GM98)
  - After j-th transaction, *augmented* sample S_j is

    $S_j = \{ (X_j(t), Y_j(t)) : t \in T \text{ and } X_j(t) > 0 \}$

    - $X_j(t)$ = frequency of item t in the sample
    - $Y_j(t)$ = net # of insertions of t into R since t joined sample



A Correctness Proof

i	R_i	S_i
i^*-1	{t t}	{ }
i^*	{t t t}	{t}
i^*+1	{t t t t}	{t t}
…	…	…
j	{t t t t … t}	{t t … t}
	Y_j - 1 items	X_j - 1 items

A Correctness Proof

i	R_i	S_i
i^*-1	{t t}	{ }
i^*	{t t t}	{t}
i^*+1	{t t t t}	{t t}
…	…	…
j	{t t t t … t}	{t t … t}
	Y_j - 1 items	X_j - 1 items

Red sample obtained from red dataset via **naïve** algorithm, hence Bern(q)

$$P(X_j = k \mid Y_j = m) = P(X_j - 1 = k - 1 \mid Y_j - 1 = m - 1) = B(k - 1; m - 1, q)$$

Proof (Continued)

- Can show (by induction)

$$P(Y_j = m) = \begin{cases} (1-q)^{N_j} & \text{if } m = 0 \\ q(1-q)^{N_j - m} & \text{otherwise} \end{cases}$$

  - Intuitive when insertions only ($N_j = j$)

- Uncondition on Y_j to finish proof

$$P(X_j = k) = \sum_m P(X_j = k \mid Y_j = m)\, P(Y_j = m)$$

= $B(k\text{-}1; m\text{-}1, q)$ by previous slide

Frequency Estimation

- Naïve (Horvitz-Thompson) unbiased estimator

$$\hat{N}_{X_i} = \frac{X_i}{q} = \frac{X_i - 1}{q} + \frac{1}{q}$$

- Exploit tracking counter:

$$\hat{N}_{Y_i} = \begin{cases} Y_i - 1 + q^{-1} & \text{if } Y_i > 0 \\ 0 & \text{if } Y_i = 0 \end{cases}$$

- Theorem

$$E[\hat{N}_{Y_i}] = N_i \text{ and } V[\hat{N}_{Y_i}] \leq V[\hat{N}_{X_i}]$$

- Can extend to other aggregates (see paper)

Estimating Distinct-Value Counts

- If usual DV estimators unavailable (BH+07)
- Obtain S' from S: insert $t \in D(S)$ with probability

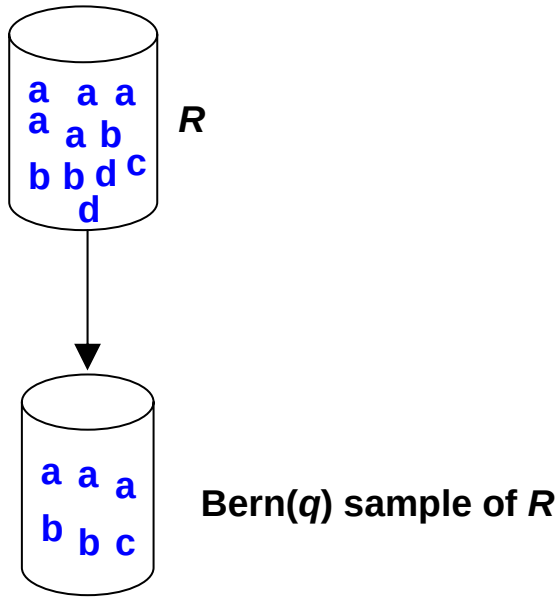
$$p(t) = \begin{cases} 1 & \text{if } Y(t) = 1 \\ q & \text{if } Y(t) > 1 \end{cases}$$

- Can show: $P(t \in S') = q$ for $t \in D(R)$
- HT unbiased estimator: $\hat{D}_{HT} = |S'| / q$
- Improve via conditioning (Var[E[U|V]] ≤ Var[U]):

$$\hat{D}_Y = E[\hat{D}_{HT} \mid S] = \sum_{t \in D(S)} p(t)/q$$

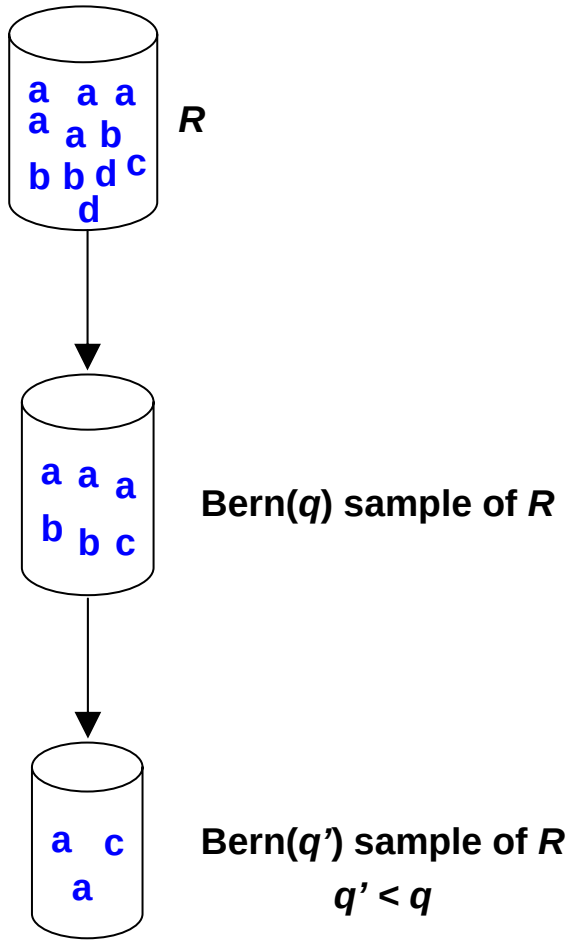
Subsampling

- Why needed?
  - Sample is too large
  - For merging
- Challenge:
  - Generate statistically correct tracking-counter value *Y*'
- New algorithm
  - See paper



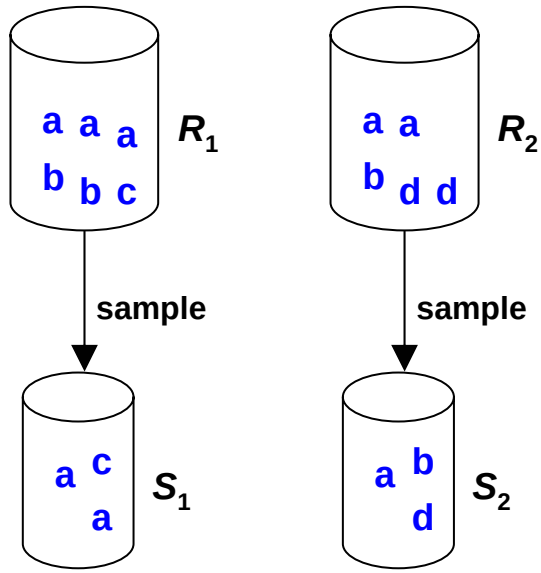
Subsampling

- Why needed?
  - Sample is too large
  - For merging
- Challenge:
  - Generate statistically correct tracking-counter value *Y*'
- New algorithm
  - See paper



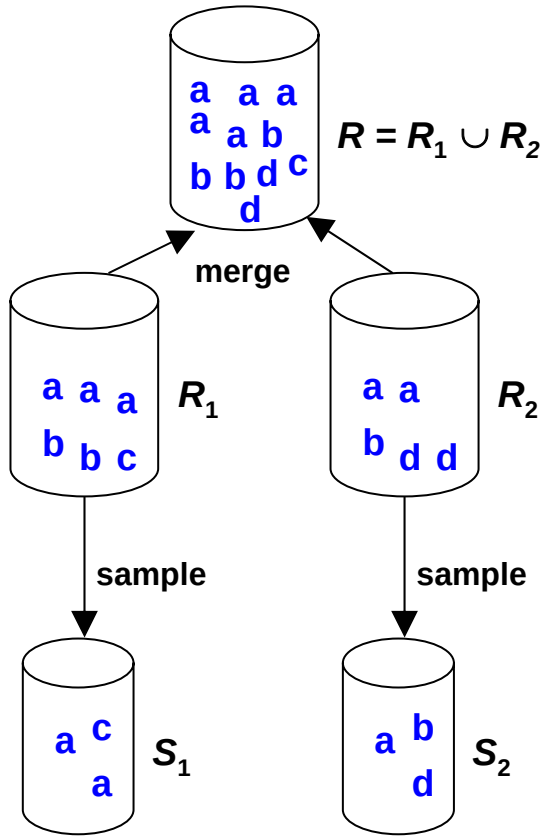
Merging

- Easy case
  - Set sampling or no further maintenance
  - $S = S_1 \cup S_2$
- Otherwise:
  - If $R_1 \cap R_2 \neq \emptyset$ and $0 < q < 1$, then there exists no statistically correct merging algorithm



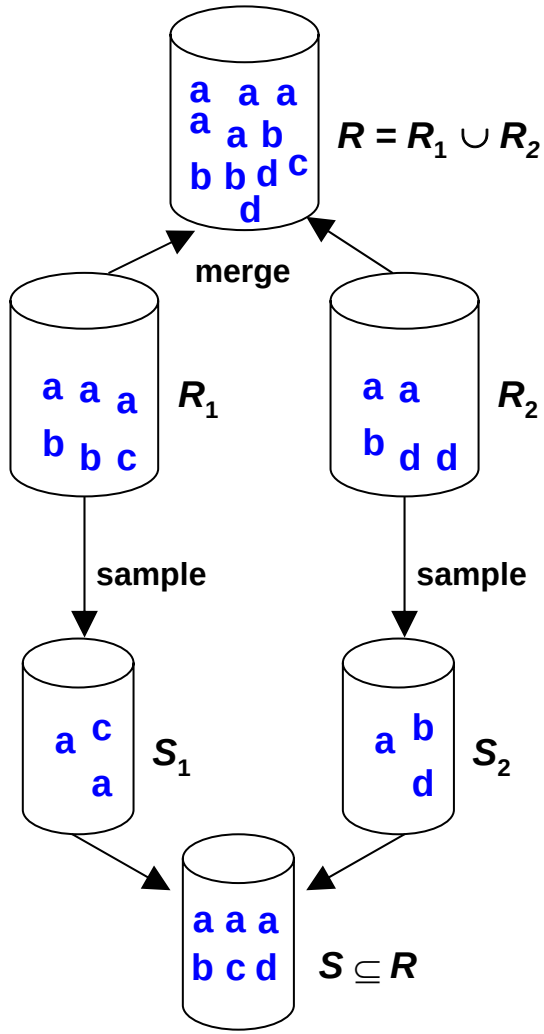
Merging

- Easy case
  - Set sampling or no further maintenance
  - $S = S_1 \cup S_2$
- Otherwise:
  - If $R_1 \cap R_2 \neq \emptyset$ and $0 < q < 1$, then there exists no statistically correct merging algorithm



Merging

- Easy case
  - Set sampling or no further maintenance
  - $S = S_1 \cup S_2$
- Otherwise:
  - If $R_1 \cap R_2 \neq \emptyset$ and $0 < q < 1$, then there exists no statistically correct merging algorithm



Related Work on Multiset Sampling

- Gibbons and Matias [1998]
  - Concise samples: maintain $X_i(t)$, handles inserts only
  - Counting Samples: maintain $Y_i(t)$, compute $X_i(t)$ on demand
  - Frequency estimator for hot items: $Y_i(t) - 1 + 0.418 / q$
    - Biased, higher mean-squared error than new estimator
- Distinct-Item sampling [CMR05,FIS05,Gi01]
  - Simple random sample of $(t, N(t))$ pairs
  - High space, time overhead

Maintaining Bernoulli Samples Over Evolving Multisets

Rainer Gemulla Wolfgang Lehner
Technische Universität Dresden

Peter J. Haas
IBM Almaden Research Center

TECHNISCHE UNIVERSITÄT DRESDEN

IBM

Backup Slides

Subsampling

- Easy case (no further maintenance)
  - Take Bern(q^*) subsample, where $q^* = q' / q$
  - Actually, just generate X' directly as Binomial(X,q^*)
- Hard case (to continue maintenance)
  - Must also generate new tracking-counter value Y'
- Approach: generate X' then $Y' \mid \{X', Y\}$

Subsampling: The Hard Case

- Generate $X' = \Phi + \Upsilon$
  - $\Phi = 1$ iff item included in S at time i^* is retained
    - $P(\Phi = 1) = 1 - P(\Phi = 0) = q^*$
  - Υ is # of other items in S that are retained in S'
    - Υ is Binomial(X-1,q^*)
- Generate Y' according to "correct" distribution
  - $P(Y' = m \mid X', Y, \Phi)$

Subsampling (Continued)

	i: 1	2	3	4	5	6	
Original sample	t	t	t	t	t	t	$Y_6 = 5, X_6 = 3$
$\Phi = 1$	t	t	t	t	t	t	$Y_6' = 5, X_6' = 2$
$\Phi = 0$	t	t	t	t	t	t	$Y_6' = 3, X_6' = 2$

$$P(Y' = m \mid X', Y, \Phi = 1) = [1 \text{ if } m = Y \text{ and } 0 \text{ otherwise }]$$

$$P(Y' = m \mid X', Y, \Phi = 0) = \frac{X'}{m} \prod_{i=m+1}^{Y-1} \left(1 - \frac{X'}{i}\right) \quad \text{when } X' > 0$$

Generate Y - Y' using acceptance/rejection (Vitter 1984)

Related Work on Set-Based Sampling

- Methods that access dataset
  - CAR, CARWOR [OR86], backing samples [GMP02]
- Methods (for bounded samples) that do not access dataset
  - Reservoir sampling [FMR62, Vi85] (inserts only)
  - Stream sampling [BDM02] (sliding window)
  - Random pairing [GLH06] (resizing also discussed)

More Motivation: A Sample Warehouse