BatchVote: Voting Rules Designed for Auditability

BatchVote: Voting rules designed for auditability

Ronald L. Rivest
MIT CSAIL
rivest@mit.edu

Philip B. Stark
U.C. Berkeley Dept. Statistics
stark@stat.berkeley.edu

Zara Perumal
MIT
zperumal@mit.edu

March 31, 2017 (For Voting'17 participants only)

Abstract

We propose a family of novel social choice functions. Our goal is to explore social choice functions for which **ease of auditing** is a primary design goal, instead of being ignored or left as a puzzle to solve later.

Our proposal, "**BatchVote**," creates a social choice function f from an arbitrary "inner" social choice function g, such as instant-runoff voting (IRV), and an integer B, the number of batches.

We aim to preserve flexibility by allowing g to be arbitrary, while providing the ease of auditing of a plurality election.

To compute the winner of an election of n votes, the social choice function f partitions the votes into B batches of roughly the same size, pseudorandomly. The social choice function g is applied to each batch. The election winner, according to f, is the weighted plurality winner for the B outcomes, where the weight of each batch is the number of votes it contains. The social choice function f may be viewed as an "interpolation" between plurality (which is easily auditable) and g (which need not be).

Auditing is simple by design: we can view f as being a (weighted) plurality election by B "*supervoters*," where bth supervoter's vote is determined by applying g to the votes in batch b, and the weight of her vote is the number of votes in her batch. Since plurality elections are easy to audit, the election output can be audited by checking a random sample of "supervotes" against the corresponding paper records.

1 Introduction and Motivation

Designing or selecting a social choice function for elections requires making tradeoffs among desirable properties—it is well known that many desirable properties are incompatible. Ease of auditability does not seem to be among the properties that have been considered when selecting a social choice function. In

this paper we elevate ease of auditing to be a first-level design criterion, and propose a specific framework, called BatchVote, for ensuring ease of auditing while blending in desirable properties from another social choice function.

The paper is organized as follows: Section 2 provides terminology, notation, and orientation. Then Section 3 overviews of the BatchVote method, giving its design philosophy and major characteristics. Section 4 takes a deeper look at the properties of BatchVote, and then Section 5 shows how a BatchVote election may be audited with the a risk-limiting audit, based on known risk-limiting audit methods for plurality elections. Simulated auditing results are given in Section 6. Section 7 presents some variants of BatchVote.

2 Preliminaries

Outcomes, Ballots, Profiles. We assume that the election is designed to select an outcome from a set $\mathcal{C}$ of C alternatives (or candidates). Each of n voters casts a single ballot. A ballot may have an arbitrary format and semantics. The **profile** of cast ballots is

$$P = \{v_1, v_2, \ldots, v_n\} \ ,$$

listing the vote v_j cast by voter j, for $j = 1, \ldots, n$. The profile is best viewed as a sequence or a multiset, since it may contain repeated items (identical ballots).

Social choice functions. A **voting rule** or **social choice function** g maps profiles to a single outcome (one of the alternatives). For any profile P, $g(P)$ is the *winner* or *outcome* for the profile P.

We require a social choice function g to be deterministic, so g must break any ties that occur. We therefore allow g to take a second random input K. Here K is the *seed* for a (pseudo-)random number generator used to break ties. Thus $f(P, K)$ is deterministic. We omit f's second argument K when it is understood from context. We assume that g does not depend on the order of the votes, that is, g applied to every permutation of P must give the same result $c \in \mathcal{C}$.

One may choose a social choice function because of its mathematical properties. For example, Tideman's "ranked-pairs method"[26] has many desirable properties [29], as does the Schulze method [14]. But not all otherwise-desirable social choice functions are readily auditable. Indeed, for both Tideman's ranked-pairs method and Schulze's method there is no known efficient method for performing a risk-limiting audit.

Post-election audits Confidence in an election outcome can be derived from a **post-election audit**. We assume that voters cast votes on **paper ballots**, and that voters had the opportunity to check that their ballots reflected their choices before casting the ballots. These paper ballots were scanned and those data were electronically aggregated to provide the initial or **reported outcome** w for the election.

The paper ballots represent the "ground truth" for the election; a full and correct count of the paper ballots should give (essentially by definition) the **actual (or true) outcome** t for the election. A "compliance audit" can provide assurance that the paper trail has integrity (see Benaloh et al. [1], Lindeman and Stark [9], and Stark and Wagner [16]).

To check the election outcome, rather than recount all the ballots by hand, it is usually more efficient to audit using a statistical method based on hand examination of a sample of the paper ballots, a method first proposed by Johnson [7]. Such a **statistical (post-election) audit** can give statistical assurance that the reported outcome is indeed equal to the actual outcome, often after examining only a relatively small sample of the paper ballots. If the reported outcome is incorrect, the audit may need to examine many ballots, or even all of them, before concluding that the reported outcome was incorrect.

Stark [17] introduced a particular kind of statistical audit—a **risk-limiting (post-election) audit** (or **RLA**). What distinguishes an RLA is that if the reported outcome is incorrect, the RLA has a large, pre-specified chance of correcting it. Lindeman and Stark provide a "gentle introduction" to RLAs [9]. Lindeman et al. [8], Norden et al. [11], and the Risk-Limiting Audit Working Group [2] give general overviews of post-election audits. Stark and Wagner [16] promulgate "evidence-based elections," which include not only a risk-limiting audit but also ensure that the evidence trail has integrity.

A variety of statistical methods for providing RLAs have been developed [18, 6, 20, 21, 22, 15, 23, 3, 12, 10, 13, 25]; some of these methods form the foundation of our approach for auditing BatchVote. There are online tools to help condut risk-limiting audits [24].

3 BatchVote

This section gives an overview of the BatchVote design philosophy, gives details of the method, and provides an analysis of its efficiency.

3.1 Design philosophy

BatchVote derives a new social choice function f from a pre-existing social choice function g. Roughly speaking, f divides the n ballots into B "batches," applies g to each batch, then defines the overall election outcome as the (weighted) plurality result of the B batch-level outcomes.

When B is very large, most batches contain at most a single ballot, and f behaves like plurality voting. But when B is equal to one, f and g are identical. In between, BatchVote acts like a blend or "interpolation" between plurality and g. Thus, $f(P)$ is not generally equal to $g(P)$, but f may inherit some desirable features of of g.

For instance, g could be a preferential voting method that allows voters to express their preferences more clearly than with simple plurality voting. Preferential voting methods are notoriously difficult to audit [25, 30]; for many

preferential voting methods no efficient risk-limiting audit method is known.

But g is, at the top level, just (weighted) plurality, for which efficient risk-limiting auditing method are well-known (see, for example [9]).

3.2 The BatchVote method

BatchVote determines the election outcome as follows, given an "inner social choice function" g.

1. Determine the set $\mathcal{C}$ of candidates.
2. Determine the target average batch size λ.
3. Collect the n cast votes and assign each a unique "ballot ID."
4. Determine the number of batches, $B = n/\lambda$.
5. Determine the "random election seed" K, using a public dice-rolling ceremony or similar means.
6. Distribute votes to batches in a deterministic manner, based on the election seed and the ballot IDs.
7. Compute the winner of each batch, using the social choice function g.
8. Compute the overall winner using a weighted plurality method to combine the batch-level outcomes, where the weight of a batch is the number of votes it contains.

Details are given in the following subsections. See Figure 1 for an illustration.

3.3 Inner social choice function g

BatchVote can use *any* social choice function as its "inner social choice function" g; g affects basic properties of the election, such whether ballots allow choices to be ranked in some way.

BatchVote is most interesting when g has desirable properties from a social choice perspective, but is difficult to audit for the entire profile of cast ballots. That includes such as many preferential voting methods. Applying g to small batches of votes, then combining the results with (weighted) plurality, may give many of the benefits of g while being easy to audit.

When BatchVote is used with inner social choice function g, we call the result "**Batch*g***". For example, methods **BatchApproval**, **BatchIRV**, **BatchRanked-Pairs**, or **BatchSchulze** are special cases of BatchVote.

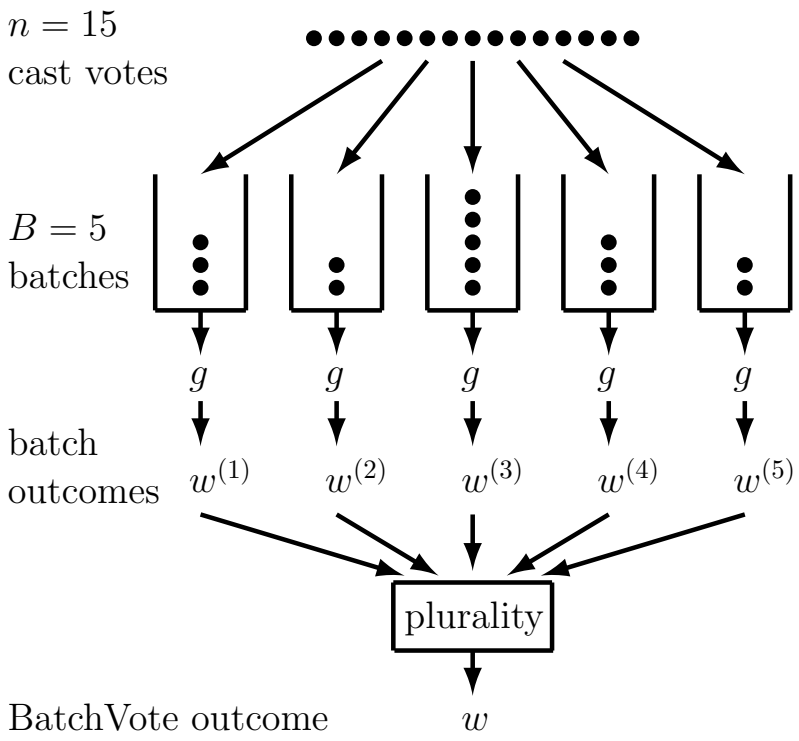

Figure 1: The BatchVote method. The n cast votes are divided pseudorandomly into B batches. The social choice function g is applied separately to each batch. The overall BatchVote outcome is the weighted plurality result of the batch outcomes.

3.4 Choosing λ

The average batch size λ is a free parameter of the BatchVote. There is no "right" choice for λ, but different choices result in different social choice functions f, and different auditing workloads. We recommend as a default $\lambda = 10C$, but other values might be preferable, depending on how much one wants f to act like g (choose λ large) or like plurality (choose λ small).

3.5 Ballot IDs

In BatchVote each ballot has a unique "ballot ID" that determines which batch the ballot is placed in. The batch a ballot is placed in does not depend on the choices made on that ballot, on other ballots, or on the number of ballots cast.

Ballot IDs are assigned by a process that does not know how ballot IDs will determine which batch a ballot will be in. In BatchVote, this isolation is accomplished by drawing a random seed K *after* all of the ballot IDs are assigned. The seed, together with the ballot IDs, determines the batch assignments.

The ballot IDs may be arbitrary strings of characters; they may be numeric, alphanumeric, or contain special characters. They may or may not contain information about where the paper ballot is located. Each ballot should have a unique ballot ID.

The ballot ID may be printed on the ballot itself when the ballot is scanned. Some optical scanners can perform this sort of operation. Some states, such as Texas, require that each ballot be "numbered."

The database of scanned ballots then contains triples of the form (assuming a single race):

$$(\textsc{ballot-ID}, \textsc{paper-ballot-location}, \textsc{voter-choices}) \ .$$

3.6 Determining the number B of batches

The number B of batches is computed as $B = N/\lambda$. (We assume for convenience that no rounding is needed here, and that B is an integer.)

BatchVote requires that B be determined *before* the random election seed K is determined, for the same reason that ballot IDs should be determined before the random election seed is determined: to prevent "gerrymandering." Once B is determined, it remains fixed, even if the number of ballots in the election changes somewhat (e.g. if a box of previously unconsidered ballots is discovered and approved for inclusion in the tally).

If B is very large (much larger than N), then many batches will be empty, and most nonempty batches will have size 1. In this case BatchVote reduces to plurality voting.

We remark that the use of a cryptographic hash function makes it harder for an adversary who can somehow contol the seed (but not the ballots themselves) to manipulate the election, in much the same spirit as the suggestions of Faliszewski et al. [5] on the use of computational complexity to protect elections from manipulation.

3.7 Random seed K

BatchVote uses a random "seed" K to help determine which batch a ballot is placed in. The seed K is also used to help break ties.

The seed K may be determined by a dice-rolling ceremony after all votes are cast. This gives a result unpredictable to an adversary. The ceremony is performed *after* all cast ballots have been collected, to prevent an adversary from manipulating the ballot IDs.

If K were predictable, an adversary who can assign ballot IDs might be able to effect the equivalent of "gerrymandering"—giving his own candidate an advantage in many batches, while the opponent gets an advantage in a few.

Our proposed process of generating a random seed *after* the votes are assigned ballot IDs prevents this sort of "gerrymandering."

3.8 Mapping ballot IDs to batches

We propose a method based on the use of a "cryptographic hash function" (specifically, SHA256) to compute the assignment of ballots to batches.

The batch number is determined by the hash of the random seed K concatenated with the ballot ID, modulo B. (The fact that 2^{256} is not an exact multiple of B is ignorable here.)

This pseudorandom method is effectively indistinguishable from a truly random mapping, for someone with a feasible amount of computational power who doesn't know the key K. We thus treat this mapping as a random mapping of ballot IDs to batches.

Appendix A gives details of our proposal.

3.9 Variability of batch sizes

BatchVote has a *variable* batch size. The number of ballots that end up in a given batch is a random variable. Different batches may have different sizes.

The size $N^{(b)}$ of batch b is a random variable with a binomial distribution $\textsc{Bin}(k; N, p)$, where the probability $p = 1/B$ that a ballot ends up in bin b.

The standard Poisson approximation to the binomial, where $\lambda = nT/B$ is the mean batch size, gives us

$$P[N^{(b)} = k] \approx e^{-\lambda}\lambda^k/k! \ .$$

BatchVote accommodates this variability by *giving each batch a weight equal to its size* in the final plurality election. Empty batches are ignored.

Thus, each voter has the same effective "weight." A ballot in a batch of size k has an effective weight of $(1/k) * k = 1$. BatchVote feels similar to the U.S. Electoral College, where the number of electoral votes a state gets depends upon that state's population.

See Section 7 for some discussion of other ways that we considered but rejected for dividing ballots into batches.

3.10 Applying g to each batch

The application of social choice function g to each batch is straightforward, assuming that g is applicable to a batch of any size.

The social choice function g may need to resolve ties. As some batches may be small, ties may be fairly common. Lack of space precludes our giving full details of our proposal, but, roughly speaking, when g is applied to batch b it is supplied a "tie-breaking seed" K_b derived pseudorandomly from K and batch index b (say by hashing together K and b).

Appendix B provides a concrete suggestion as to how this might be done.

3.11 Efficiency

BatchVote computes the election outcome in time equal to the sum of the time taken to assign ballots into batches, plus the time taken to compute each batch outcome using g, plus the time taken to compute the weight plurality overall result. In practice, all of these operations are quickly performed, assuming that g is efficient.

4 Properties

4.1 BatchVote-specific properties

Fairness to voters Because each batch has weight equal to the number of votes it contains, BatchVote is fair to voters—each voter is treated equally.

4.2 General properties

What properties does f inherit from g and from Plurality, since it is a blend of the two systems?

Clearly, in order for BatchVote to have some property, then *both* Plurality and g must have that property, since when B is large BatchVote becomes Plurality, and when $B = 1$, BatchVote is g. Wikipedia provides a nice list of voting system properties[1].

Unfortunately, plurality itself has few voting system properties. We mention two properties here.

Ballot format Obviously, the ballot format for f is identical to the ballot format for g.

Monotonicity It is easy to see that if g is monotone, then so is f. (Monotonicity means that moving a candidate towards the front of a ballot can only help that candidate.)

[1] `https://en.wikipedia.org/wiki/Voting_system`

BatchPlurality versus Plurality BatchPlurality is similar to plurality: there is precinct level tabulation and reporting of precinct-level results to a central tabulation. However, with BatchPlurality, precincts do not report the candidate counts, just the winner for the precinct and the overall number of voters. Thus BatchPlurality is closer to how the Electoral College works.

5 Auditing

Efficient post-election audits are derived from Wald's sequential tests of statistical hypotheses [27, 28].

5.1 Ballot-polling audits

The ballot-polling post-election audit studied here is a simple modification of the ballot-polling audit method introduced by Lindeman, Stark, and Yates [10].

Consider a pair of candidates (w, ℓ) where w is a reported winner and ℓ is a reported loser. Candidate w really beat candidate ℓ in the batch plurality contest if $\sum_{b:t^{(b)}=w} N^{(b)} > \sum_{b:t^{(b)}=\ell} N^{(b)}$, i.e., if

$$p_{w\ell} \equiv \frac{\sum_{b:t^{(b)}=w} N^{(b)}}{\sum_{b:t^{(b)}=w} N^{(b)} + \sum_{b:t^{(b)}=\ell} N^{(b)}} > 1/2. \tag{1}$$

Suppose we draw a random batch $\mathbb{B}$ such that $\Pr\{\mathbb{B} = b\} = N^{(b)}/N$. Condition on the event that the true winner of batch $\mathbb{B}$ is either w or ℓ. Then the (conditional) probability that the true winner of the batch is w is $p_{w\ell}$. Wald's sequential probability ratio test [27] can test the hypothesis that $p_{w\ell} \leq 1/2$ against the alternative that $p_{w\ell} = \frac{\sum_{b:w^{(b)}=w} N^{(b)}}{\sum_{b:w^{(b)}=w} N^{(b)} + \sum_{b:w^{(b)}=\ell} N^{(b)}}$, the reported fraction of the weighted votes for either w or ℓ that are reported votes for w.

5.2 Comparison audits

This section describes a comparison audit for BatchVote.

We have a weighted plurality contest with B batches (voters). Batch b contains $N^{(b)}$ ballots. Let $N \equiv \sum_b N^{(b)}$ be the total number of ballots cast.

Let $\mathcal{C}$ denote the set of possible election outcomes. (Here $\mathcal{C}$ standsfor "candidates," although an outcome might involve more than one candidate winning). Let $w^{(b)}$ be the reported outcome (reported winner) for batch b and let $t^{(b)}$ denote the actual (true) outcome for batch b (the outcome that a manual audit of the batch would show).

The total reported weighted vote for outcome c is

$$R_c \equiv \sum_{b:w^{(b)}=c} N^{(b)}. \tag{2}$$

The reported winner of the contest is $w \equiv \arg\max_c R_c$ (assuming no ties). Similarly, the actual (true) winner of the contest is $t \equiv \arg\max_c A_c$.

The reported losers are the candidates $\mathcal{L} \equiv \{\ell \in \mathcal{C} : \ell \neq w\}$.

To simplify the notation, define

$$R_c^{(b)} \equiv \begin{cases} N^{(b)}, & \text{if } w^{(b)} = c \\ 0, & \text{else.} \end{cases} \tag{3}$$

That is, $R_c^{(b)}$ is the number of ballots in batch b if batch b was reported to have voted for outcome c, and is zero if batch b was reported to have voted for any other outcome. Thus

$$R_c = \sum_b R_c^{(b)} \ . \tag{4}$$

For $\ell \in \mathcal{L}$, define the *reported pairwise margins*:

$$R_{w\ell} \equiv R_w - R_\ell \ . \tag{5}$$

This will be positive for all $\ell \in \mathcal{L}$ if and only if w is the reported winner w.

The total *actual* weighted vote for candidate c is

$$A_c \equiv \sum_{b:t^{(b)}=c} N^{(b)} \ . \tag{6}$$

Define

$$A_c^{(b)} \equiv \begin{cases} N^{(b)}, & \text{if } t^{(b)} = c \\ 0, & \text{else ,} \end{cases} \tag{7}$$

so

$$A_c = \sum_b A_c^{(b)} \ . \tag{8}$$

The true winner is w if $A_w > A_\ell$ for all $\ell \in \mathcal{L}$ that is, if the *actual pairwise margins*

$$A_{w\ell} \equiv A_w - A_\ell > 0, \ \ \forall \ell \in \mathcal{L}. \tag{9}$$

We now give a simple auditing procedure based a sufficient condition for the true winner to be w, couched in terms of a single scalar.

$$\begin{aligned} A_{w\ell} &= A_w - A_\ell \\ &= \sum_b A_w^{(b)} - \sum_b A_\ell^{(b)} \\ &= R_{w\ell} - \sum_b \left((A_\ell^{(b)} - A_w^{(b)}) - (R_\ell^{(b)} - R_w^{(b)}) \right). \end{aligned} \tag{10}$$

The correct outcome is w if and only if for all $\ell \neq w$,

$$R_{w\ell} - A_{w\ell} = \sum_b (A_\ell^{(b)} - A_w^{(b)}) - (R_\ell^{(b)} - R_w^{(b)}) < R_{w\ell} \ . \tag{11}$$

Define

$$\gamma^{(b)} \equiv \max_{\ell \neq w} \frac{(A_\ell^{(b)} - A_w^{(b)}) - (R_\ell^{(b)} - R_w^{(b)})}{N^{(b)} R_{w\ell}} \ . \tag{12}$$

If

$$\sum_b \gamma^{(b)} N^{(b)} < 1 \ , \tag{13}$$

then for all $\ell \neq w$,

$$\sum_b (A_\ell^{(b)} - A_w^{(b)}) - (R_\ell^{(b)} - R_w^{(b)}) < R_{w\ell} \ , \tag{14}$$

i.e., w is the true winner.

Select a batch $\mathbb{B}$ at random, with probability $N^{(b)}/N$ of selecting batch b. If batch b is selected, it can be tallied by hand, revealing $A_c^{(b)}$ for all c; then $\gamma^{(b)}$ can be calculated. Let $X = \gamma^{\mathbb{B}}$, the value of γ for the randomly selected batch. Then

$$\mathbb{E}X = \sum_b \gamma^{(b)} \Pr\{\mathbb{B} = b\} = \frac{1}{N} \sum_b \gamma^{(b)} N^{(b)} \ . \tag{15}$$

Hence, w is the true winner if $\mathbb{E}X < 1/N$. A sequential test of the hypothesis $\mathbb{E}X \geq 1/N$ can be used to construct a risk-limiting audit with risk limit α: continue to audit until either that hypothesis is rejected at significance level α or there has been a full hand count.

5.3 Masking of errors

In the presence of no errors, the BatchX method can be viewed as paying a penalty of a factor of λ compared to doing an audit of X on the entire set of ballots. Of course, this statement only makes sense when X has an efficient audit method defined for it, which it may not.

But errors (discrepancies discovered between paper ballots and their electronic counterparts) may be masked here, as changing a ballot in a batch to its correct value may have no effect on the batch outcome.

6 Experimental results

We have code available on github[2]. We have experimented with both synthetic data sets and data sets from real elections[3].

We also estimated audit workloads versus λ; see Figure 2. The number of batches that need to be examined for a risk limit of $\alpha = 0.05$ is about $6/m^2$ for a ballot-polling audit and about $6/m$ for a comparison audit, where m is the margin between the top two candidates. The estimated audit workload is then λ times larger, since auditing a batch requires looking at about λ ballots. Note that the workload peaks when the winner changes. Our recommendation for choosing $\lambda = 10C$ is an attempt to choose a point a bit to the right of the winner-crossover peak, if it exists.

[2]The github repo is `https://github.com/ron-rivest/2016-batchvote-code`. This is currently private, but will be made public.

[3]Real data sets available at: `http://rangevoting.org/TidemanData.html`

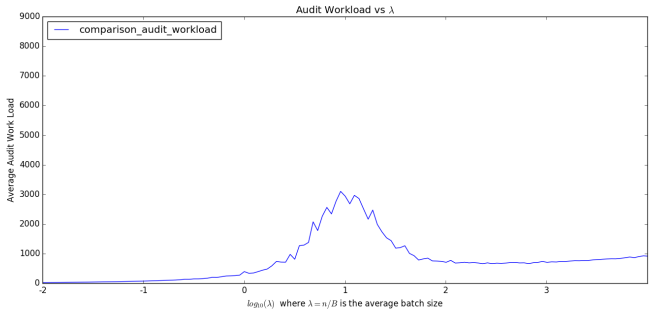

Figure 2: Comparison audit workload estimatefor BatchRankedPairs for the Burlington VT 2009 Mayoral Race data.

7 Variants

This section describes possible variations of BatchVote.

Replication One might replicate each ballot T times, for some $T > 1$, giving extra flexibility to the process allocating ballots to batches.

Fixed-size batches We explored a number of ways of forcing each batch to have very nearly the same size. For example, one could require all batches to have a fixed size λ (an integer). (Using a replication factor $T = \lambda$ may help.)

However, no alternative method seemed sufficiently simple and "random;" we prefer the proposed BatchVote method for its simplicity.

Using replications and all subsets of size λ BatchVote is sensitive to the random seed K. For a close election, a different value of K might yield a different election outcome. This is not surprising, as K controls tie-breaking.

However, K also controls the placement of ballots in batches, and one might prefer to have a social choice function that is somehow insensitive to the allocation of ballots to batches.

One could consider *all* subsets of size λ, and apply plurality to their batch-level results. However, such an approach is too expensive.

MajorityThenBatch or CondorcetThenBatch First check to see if there is a majority or Condorcet winner. If so, then proclaim that candidate to be the winner. Else, proceed with the BatchVote method. (This is a common approach for forcing a voting system to be Majoritarian or Condorcet.)

Question: How to audit such combined systems?

Write-in votes We can treat write-in candidates as regular candidates; deriving the list of candidates from ballots cast.

Multiple races How should one use the BatchVote method when there are multiple races in an election? In our description so far, we have implicitly assumed that there is only one race being audited. See Stark [19, 21] and Benaloh et al. [1] for approaches for addressing this issue.

Acknowledgments

Ronald L. Rivest gratefully acknowledges support for his work on this project received from the Center for Science of Information (CSoI), an NSF Science and Technology Center, under grant agreement CCF-0939370, and from the Department of Statistics, University of California, Berkeley, which hosted his sabbatical visit when this work began.

References

[1] J. Benaloh, D. Jones, E. Lazarus, M. Lindeman, and P.B. Stark. SOBA: Secrecy-preserving observable ballot-level audit. In *Proceedings 2011 Electronic Voting Technology Workshop/Workshop on Trustworthy Elections (EVT/WOTE '11)*, 2011. `http://static.usenix.org/events/evtwote11/tech/final_files/Benaloh.pdf`.

[2] J. Bretschneider, S. Flaherty, S. Goodman, M. Halvorson, R. Johnston, M. Lindeman, R.L. Rivest, P. Smith, and P.B. Stark. Risk-limiting post-election audits: Why and how?, Oct. 2012. (ver. 1.1) `http://people.csail.mit.edu/rivest/pubs.html#RLAWG12`.

[3] S. Checkoway, A. Sarwate, and H. Shacham. Single-ballot risk-limiting audits using convex optimization. In D. Jones, J.-J. Quisquater, and E. Rescorla, editors, *Proceedings 2010 EVT/WOTE Conference*. USENIX/ACCURATE/IAVoSS, August 2010.

[4] Berj Chilingirian, Zara Perumal, Ronald L. Rivest, Grahame Bowland, Andrew Conway, Philip B. Stark, Michelle Blom, Chris Culnane, and Vanessa Teague. Auditing Australian Senate ballots. *CoRR*, abs/1610.00127, 2016. `arxiv.org/abs/1610.00127`.

[5] Piotr Faliszewski, Edith Hemaspaandra, and Lane A. Hemaspaandra. Using complexity to protect elections. *CACM*, 53(11):74–82, Nov 2010.

[6] J. L. Hall, L. W. Miratrix, P. B. Stark, M. Briones, E. Ginnold, F. Oakley, M. Peaden, G. Pellerin, T. Stanionis, and T. Webber. Implementing risk-limiting post-election audits in California. In *Proc. 2009 Electronic Voting Technology Workshop/Workshop on Trustworthy Elections (EVT/WOTE '09, Montreal, Canada)*. USENIX, Aug 2009. `http://www.usenix.org/event/evtwote09/tech/full_papers/hall.pdf`.

[7] K. Johnson. Election verification by statistical audit of voter-verified paper ballots. `http://ssrn.com/abstract=640943`, Oct. 31 2004.

[8] M. Lindeman, M. Halvorseon, P. Smith, L. Garland, V. Addona, and D. McCrea. Principle and best practices for post-election audits. `www.electionaudits.org/files/best%20practices%20final_0.pdf`, 2008.

[9] Mark Lindeman and Philip B. Stark. A gentle introduction to risk-limiting audits. *IEEE Security and Privacy*, 10:42–49, 2012.

[10] Mark Lindeman, Philip B. Stark, and Vincent S. Yates. BRAVO: Ballot-polling risk-limiting audits to verify outcomes. In Alex Halderman and Olivier Pereira, editors, *Proceedings 2012 EVT/WOTE Conference*, 2012.

[11] Lawrence Norden, Aaron Burstein, Joseph Lorenzo Hall, and Margaret Chen. Post-election audits: Restoring trust in elections. Technical report, Brennan Center for Justice and Samuelson Law, Technology & Public Policy Clinic, 2007.

[12] California Secretary of State. Post-election risk-limiting audit pilot program, 2011-2013. `http://www.sos.ca.gov/elections/voting-systems/oversight/post-election-auditing-regulations-and-reports/post-election-risk-limiting-audit-pilot-program/`.

[13] A. D. Sarwate, S. Checkoway, and H. Shacham. Risk-limiting audits and the margin of victory in nonplurality elections. *Politics and Policy*, 3(3):29–64, 2013.

[14] Markus Schulze. A new monotonic, clone-independent, reversal symmetric, and condorcet-consistent single-winner election method. *Social Choice and Welfare*, 36(2):267–303, 2011.

[15] P. B. Stark. Risk-limiting vote-tabulation audits: The importance of cluster size. *Chance*, 23(3):9–12, 2010.

[16] P. B. Stark and D. A. Wagner. Evidence-based elections. *IEEE Security and Privacy*, 10(05):33–41, Sep-Oct 2012.

[17] Philip B. Stark. Conservative statistical post-election audits. *Ann. Appl. Stat.*, 2:550–581, 2008.

[18] Philip B. Stark. A sharper discrepancy measure for post-election audits. *Ann. Appl. Stat.*, 2:982–985, 2008.

[19] Philip B. Stark. Auditing a collection of races simultaneously. `https://arxiv.org/abs/0905.1422v1`, 2009.

[20] Philip B. Stark. CAST: Canvass audits by sampling and testing. *IEEE Trans. Inform. Forensics and Security*, 4(4):708–717, Dec. 2009.

[21] Philip B. Stark. Efficient post-election audits of multiple contests: 2009 California tests. `ssrn.com/abstract=1443314`, 2009. 2009 Conf. Empirical Legal Studies.

[22] Philip B. Stark. Risk-limiting post-election audits: P-values from common probability inequalities. *IEEE Trans. on Information Forensics and Security*, 4:1005–1014, 2009.

[23] Philip B. Stark. Super-simple simultaneous single-ballot risk-limiting audits. In *Proc. 2010 EVT/WOTE Workshop*, 2010. `http://www.usenix.org/events/evtwote10/tech/full_papers/Stark.pdf`.

[24] Philip B. Stark. Tools for comparison risk-limiting election audits. `http://www.stat.berkeley.edu/~stark/Vote/auditTools.htm`, 2015.

[25] Philip B. Stark and Vanessa Teague. Veriable european elections: Risk-limiting audits for d'hondt and its relatives. *USENIX Journal of Election Technology and Systems (JETS)*, 1(3):18–39, 2014.

[26] T. N. Tideman. Independence of clones as a criterion for voting rules. *Social Choice and Welfare*, 4(3):185–206, 1987.

[27] A. Wald. Sequential tests of statistical hypotheses. *Ann. Math. Stat.*, 16(2):117–186, 1945.

[28] A. Wald. *Sequential analysis*. Dover (Mineola, New York), 2004.

[29] Wikipedia. Voting system. `https://en.wikipedia.org/wiki/Voting_system`.

[30] Lirong Xia. Computing the margin of victory for various voting rules. In *Proceedings of the 13th ACM Conference on Electronic Commerce (EC-12)*, 2012.

Appendix A. Possible details of batch assignment method.

We illustrate the proposed procedure with an example. Suppose the random seed K is the 24-digit decimal number

$$K = \texttt{067541877022641091953584}$$

and suppose that a ballot has the 37-character ballot ID

$$\text{ID} = \texttt{2016-11-08-maricopa-az-1562-7631-5515}\ .$$

Then the batch to which this ballot is assigned is starting with the concatenation of these two strings—that is:

$$K||ID = \texttt{067541877022641091953584}2016\texttt{-11-08-maricopa-az-1562-7631-5515}\ .$$

Applying SHA256 to this byte string yields the hexadecimal result

$$\texttt{db5d8603dcf6e4e122e7b0ff231d4069cb4626f45ab1686cb1b6dd9d424480d9}$$

which, when interpreted as a base-16 integer, yields

$$\texttt{99221755554920309225844359348330608520995333449296550547451312649783275192537}$$

(decimal). Finally, we take the result modulo B, the number of batches, and add one. Suppose $B = 10000$. Then the batch number for this ballot is

$$\texttt{2538}\ .$$

Because the result is obtained modulo B, plus one, the batches are numbered 1 to B, inclusive.

Appendix B. Guidelines for breaking ties

We provide each of the B invocations of g with its own random number seed to use in tie-breaking. Suppose the overall election-random seed is

$$\texttt{067541877022641091953584}\ .$$

Suppose we wish to provide the 15th invocation of g with its own tie-breaking seed. Then the tie-breaking seed $K^{(b)}$ provided will be

$$K^{(b)} = \texttt{067541877022641091953584:batch:15}$$

That is, the overall election seed, followed by "`:batch:`", followed by the batch number b in decimal, for $b = 1, 2, \ldots, B$. This seed can be concatenated with other values within g to break ties, and then SHA256 may be applied to the result.

Of course, the specification of g needs to clearly specify how ties are to be broken, given the tie-breaking seed $K^{(b)}$. (We have, for example, python code that illustrates this for various social choice functions g.)

Each instance of g receives a different tie-breaking seed $K^{(b)}$, to remove the possibility of obviously correlated tie-breaking between the various batches. Although the seeds for different batches are related, they are nonetheless different, and the pseudo-random character of SHA256 makes it computationally infeasible to find statistical correlations in their tie-breaking use.

Notation

This note summarizes notational conventions we use in this paper.

B Number of batches.

b A particular batch. $b = 1, 2, \ldots, B$.

$\mathcal{C}$ The set of candidates.

C Number of candidates (possble election outcomes.

c A particular candidate. (Also w, ℓ sometimes, for winner, loser.)

n Number of cast votes.

T Replication factor; how many times each vote is replicated.

N Number of ballots being tabulated; $N = nT$.

$N^{(b)}$ Number of ballots in batch b (so $\sum_b N^{(b)} = N$).

λ Average batch size ($\lambda = N/B$).

R_c Total reported tabulation in favor of candidate c. (i.e. electronic tabulation)

A_c Total actual tabulation in favor of candidate c (i.e. paper ballot tabulation)

$R_c^{(b)}$ Reported tabulation for candidate c in batch b (Either $N^{(b)}$ or 0).

$A_c^{(b)}$ Actual tabulation for candidate c in batch b. (Either $N^{(b)}$ or 0).

$R_{w\ell}$ Reported margin of candidate w over candidate ℓ ($R_{w\ell} = R_w - R_\ell$).

$A_{w\ell}$ Actual margin of candidate w over candidate ℓ ($A_{w\ell} = A_w - A_\ell$).

$R_{w\ell}^{(b)}, A_{w\ell}^{(b)}$ Margins particularized to batch b.

$t^{(b)}$ True winner of batch b.

$w^{(b)}$ Reported winner of batch b.

K Random number seed for the election.

$K^{(b)}$ Random number seed for batch b.

$\mathbb{B}$ A randomly selected batch, with $\Pr\{\mathbb{B} = b\} = N^{(b)}/N$.