2016 AUS Senate Audit Workflow

1. Overview

This document summarizes the election workflow used to audit the 2016 Australian Senate Election.

There are many ways that error can alter the outcome of elections. The Australian Senate Election has complex voting rules and delicate dependence on software, including OCR, which make the outcome particularly vulnerable to small errors. Auditing electronic results against the underlying voter-marked paper records can help determine the level of confidence warranted.

2. Workflow

The audits we propose are incremental, in a series of rounds, starting with an initial sample of ballots. Tests determine whether the sample provides strong evidence that the electoral outcome is correct. If not, the sample is augmented and the process repeats, round after round, until the outcome is confirmed, until all ballots have been examined, or until a preset limit on time or audit resources has been met.

Each time a sample is enlarged, randomly selected paper ballots are retrieved and interpreted manually (hand-to-eye), and those manual interpretations are recorded electronically. Those new electronic records are the basis of the audit computations.

2.1 Stage 1: Planning

The in-charge of the audit will identify the individuals to participate in performing the audit. They will meet to review and refine the steps to be performed, the observation of each step, the allocation of resources, the time-table involved, any physical security protocols to be followed, the funding necessary for any expenses, the reporting protocols to be followed, the maintenance of chain of custody over the paper ballots, the determination as to which computers (laptops) will be employed and how they will initialized with the appropriate software, what procedures will be followed for handling exceptions or errors, etc.

2.2 Stage 2: Choosing Ballots to be Audited

Paper ballots are selected at random for transcription and auditing, using a pseudo-random number generator seeded with a publicly-chosen random value. The output of this stage is a set of listing specified which paper ballots will be audited in each round of the audit.

2.2.1 Generate Random Seed Publicly

The seed in the pseudo-random number generator is set using a physical source of randomness, by rolling 24 10-sided dice. This ensures that the ballots to be selected are truly unpredictable, but that the process is repeatable, starting from the archived seed. The dice-rolling should be a public ceremony, observed by a number of scrutineers from the various parties.

2.2.2 Running *Sampler -- Choosing Random Numbers*

To choose the paper ballots to review, we run *sampler.py.*
We run sampler with the following parameters:

1. Election Id: “2016 AUS Senate Election [STATE] Audit Round [#]”
2. Random seed: 24 decimal digits generated in step 2.1.1
3. Output range: 2 to number of rows in *aec-senate-formalpreferences-STATE.csv*.
4. Duplicates ok: no
5.
    a. Expanded version of sample: No if first round, yes for subsequent round. If non-first round enter the total cumulative number of ballots previously sampled.
    b. How many ballots: **1500**
6. Output file: “2016_AUS_Senate_Election_[STATE]_Audit Round_[#].txt”

2.2.3 Listing Ballots

At each round after sampler.py has been run, a csv will be generated named selected_ballots_round_# to facilitate retrieval of ballots by election officials. This CSV will be generated by running sampleballotgenerator.py using the sampler.py output and *aec-senate-formalpreferences-STATE.csv* as input. The generated csv named *selected_ballots_round_#.csv* can be printed with 47 rows per page. The original preference information is *not* listed here, to prevent bias on the part of the auditors.

The CSV header *selected_ballots_round_#.csv* is as follows:

ElectorateNm, VoteCollectionPointNm, VoteCollectionPointId, BatchNo, PaperNo, Preferences

2.3 Stage 3: Obtaining and transcribing the selected paper ballots

The selected_ballots_round_#.csv will be sent to the officials for the counting step. At least two election officials are required at this stage to verify correctness at each step.

2.3.1 Retrieving ballots

At this step a pair of election officials will take the a page of *SelectedBallots.csv* and retrieve the ballots listed on the sheet. They will take the sheet and retrieve ballots in the following manner. For each ballot in the list, both officials will confirm its placement in the pile. When a ballot is extracted, officials will place a yellow sticky note in the batch with the appropriate ballot information: (District, batch, paper no) and place a pink sticky note with identical information on the extracted ballot.

These ballots will be placed in order in a ballot box and brought over for manual entry, with the corresponding sheet. Batch's open

2.3.2 Ballot Entry

The preferences observed on the obtained paper ballots will be entered into *selected_ballots_round_#.csv.* From the box of collected ballots, each ballot will be read and the preferences reinterpreted by both election officials directly from the paper ballots. Once the election officials agree on the preferences, they will enter the preferences in the *selected_ballots_round_#.csv* preference field as a list of candidate IDs.

2.3.3 Entry review

Once the all the ballots have been entered into the *selected_ballots_round_#.csv.* The form will be sent to the audit group to run the audits on the new records.

2.4 Running Audits

2.4.1 Generating Audit Data

From the filled out *selected_ballots_round_#.csv and aec-senate-formalpreferences-STATE.csv* generate_recount_data.py will be run to create a new CSV with the recount preferences and whether the initial and recounted preferences match. The header for the *aec_senate_formal_preferences_audit_round_#_.csv* will be as follows*:* *ElectorateNm, VoteCollectionPointNm, VoteCollectionPointId, BatchNo, PaperNo, Preferences, RecountPreferences, Match*

2.4.1 Concatenate Data

From the new *aec_senate_formal_preferences_audit_round_#_.csv* the data will be appended to *aec_senate_formal_preferences_audit_aggregate.csv* to form a combined data set for the audit round.

2.4.2 Run Audit

From the *aec_senate_formal_preferences_audit_aggregate.csv,* various audits will be run to determine whether to proceed with another round.

3 Design Considerations / Open Questions

3.1 Number of Ballots

1500 ballots were chosen as the size of ballots per round.

3.2 Current Tabulation System

Although the current tabulation system operates of of scans, we require hand to eye interpretations of ballots as the most accurate means of verifying the outcome.

4 Requirements

4.1 Ballot Selection

For ballot selection, two distinct people and some ten-sided dice are needed. Also, the following software andtools are required: Python, sampler.py, *aec-senate-formalpreferences.csv*. , and ballot_selector.py

4.2 Ballot Counting

Each ballot counting stage requires 2 people per entry or ballot retrieval step, aec_senate_formal_preferences_audit_round_#_.csv and selected_ballots.csv.

4.3 Running Audit

In order to run the audit the following software and tools are required:

The audit code here: https://github.com/ron-rivest/2016-aus-senate-audit needs to be installed with the dividebatur code in the root directory of the 2016-aus-senate-audit directory: https://github.com/grahame/dividebatur/tree/master/dividebatur . These tools require Python 3, git lfs, and numpy.

These tools are currently being refactored, and access to an AWS image with the libraries and bash run scripts can be requested by emailing zperumal@mit.edu.

5 Future Work