Democracy Developers
A Guide to Risk Limiting Audits with RAIRE
Part 1: Auditing IRV Elections with RAIRE

Michelle Blom, Peter Stuckey and Vanessa Teague

Date: August 21, 2023
Version 1.0 DRAFT

Contents

Executive Summary

This document is intended to help election administrators, candidates and interested members of the public understand how to conduct Risk Limiting Audits (RLAs) for Instant Runoff Voting (IRV) using RAIRE.

Risk limiting audits (RLAs) are a post-election activity, performed to provide a certain level of confidence in the correctness of the reported outcome, or to correct a wrong outcome by manual recount. These audits involve randomly sampling paper ballots that have been cast by voters. After collecting and analysing such a sample, statistical computations are performed to ascertain a current level of *risk*. A Risk Limiting Audit guarantees a *Risk Limit:* the maximum probability that it will mistakenly confirm a reported outcome that was in fact wrong. The audit proceeds by sampling ballots until this risk falls below an acceptable level, or election administrators determine that a full manual count is required.

This guide describes how RAIRE operates, the software currently available to support RAIRE audits, and the tools required to perform an audit using RAIRE.

Chapter 1 gives a general overview of the audit process incorporating RAIRE. Chapter 2 reviews instant runoff voting and explains how we visualize and analyze IRV elections. Chapter 3 explains the main concept in RAIRE audits: *assertions*. Chapter 4 reviews Risk Limiting Audits. Chapter 5 brings all of this together and explains how to conduct a complete RLA for IRV elections using assertions generated by RAIRE. Part 2 of this guide provides additional more technical details on assertions, and how they are generated by RAIRE.

Chapter 1

Introduction: IRV RLAs with RAIRE

Risk limiting **A**udits for **IR**V **E**lections (RAIRE) allows election administrators to conduct Risk Limiting Audits (RLAs) for Instant Runoff Voting (IRV) elections. IRV elections are widely used across Australia, and are increasingly being introduced in the United States. They are also used in non-governmental elections including the Best Picture Academy Award. IRV RLAs with RAIRE were successfully piloted for vote-by-mail ballots in the San Fransicso District Attorney's contest of 2019 (Blom *et al.* [2020]), using the SHANGRLA RLA toolkit (Stark [2020]).

RAIRE takes an assertion-based view of auditing. Each assertion is a condition – a comparison between two categories of ballots – that we want to check. With the assertion-based view, each of these comparisons is an *assertion* that needs to be checked or verified during the audit. For any assertion, ballots are divided into three categories: those that support the assertion, those that counteract the assertion, and those that do not affect it.

The easiest way to understand assertions is to think about familiar plurality elections, where voters simply choose their favourite candidate. The assertions are so simple they do not need to be written explicitly: the candidate with the majority of votes wins. When auditing this election, we want to check that the reported winner had more votes than each of the reported losers. The assertion "Alice has more votes than Bob" compares "Votes for Alice" with "Votes for Bob". Votes for Alice support the assertion, votes for Bob counteract it, and blank ballots do not affect it.

IRV elections are more complicated because many different assertions can be required to test one election outcome. IRV involves ranked choice voting. Voters indicate their preferences over a set of candidates by ranking them in order from most to least preferred. To determine the winner of an IRV election, each candidate is initially awarded all ballots on which they are ranked first. The candidate with the least number of votes is eliminated. All ballots in their tally pile are then given to the next most preferred candidate on the ballot who has not yet been eliminated. This process of eliminating the candidate with the smallest tally continues until a single candidate has the majority of votes. This candidate is declared the winner.

What does RAIRE do?

RAIRE generates a set of *Assertions* that imply that the announced winner won. These assertions are tested with an RLA, hence making an RLA of the IRV election result.

RAIRE is a tool for finding a set of assertions to check in an RLA of an IRV election. These assertions, if verified in an audit, will confirm that the reported election outcome is correct, with a certain degree of confidence based on the achieved risk. RAIRE aims to find a least-cost set of assertions, using a rule of thumb to estimate the sample sizes required to verify them in an audit. To find these assertions, RAIRE requires the cast vote records (CVRs) corresponding to the paper ballots cast by voters. RAIRE can be used for ballot polling audits, provided CVRs are available, but is much more efficient with ballot-level comparison audits, where paper ballots are compared to their associated CVR for discrepancies.

What does RAIRE *not* do?

RAIRE can be used to verify that the announced winner won. It does *not* check whether they won by the announced elimination order. This is a deliberate design feature: RAIRE does not waste auditing effort on details that do not affect who won.

Figure 1.1 shows where RAIRE sits within an audit infrastructure. The same mathematical and statistical procedures used to compute risks for the “Alice versus Bob” comparisons in an audit for a plurality election can be applied to compute risks for each RAIRE assertion. Auditing tools designed for plurality elections can be extended to conduct audits for IRV with the integration of RAIRE. Figure 1.1 shows existing (plurality) RLA computations in purple, while new (IRV-specific) modules are green. New software is required to:

1. generate the assertions;
2. visualize the assertions and validate that they imply that the announced winner won;
3. enter the preference rankings observed on the sampled ballots; and
4. compute the discrepancies between the CVRs and the ballots, for each assertion.

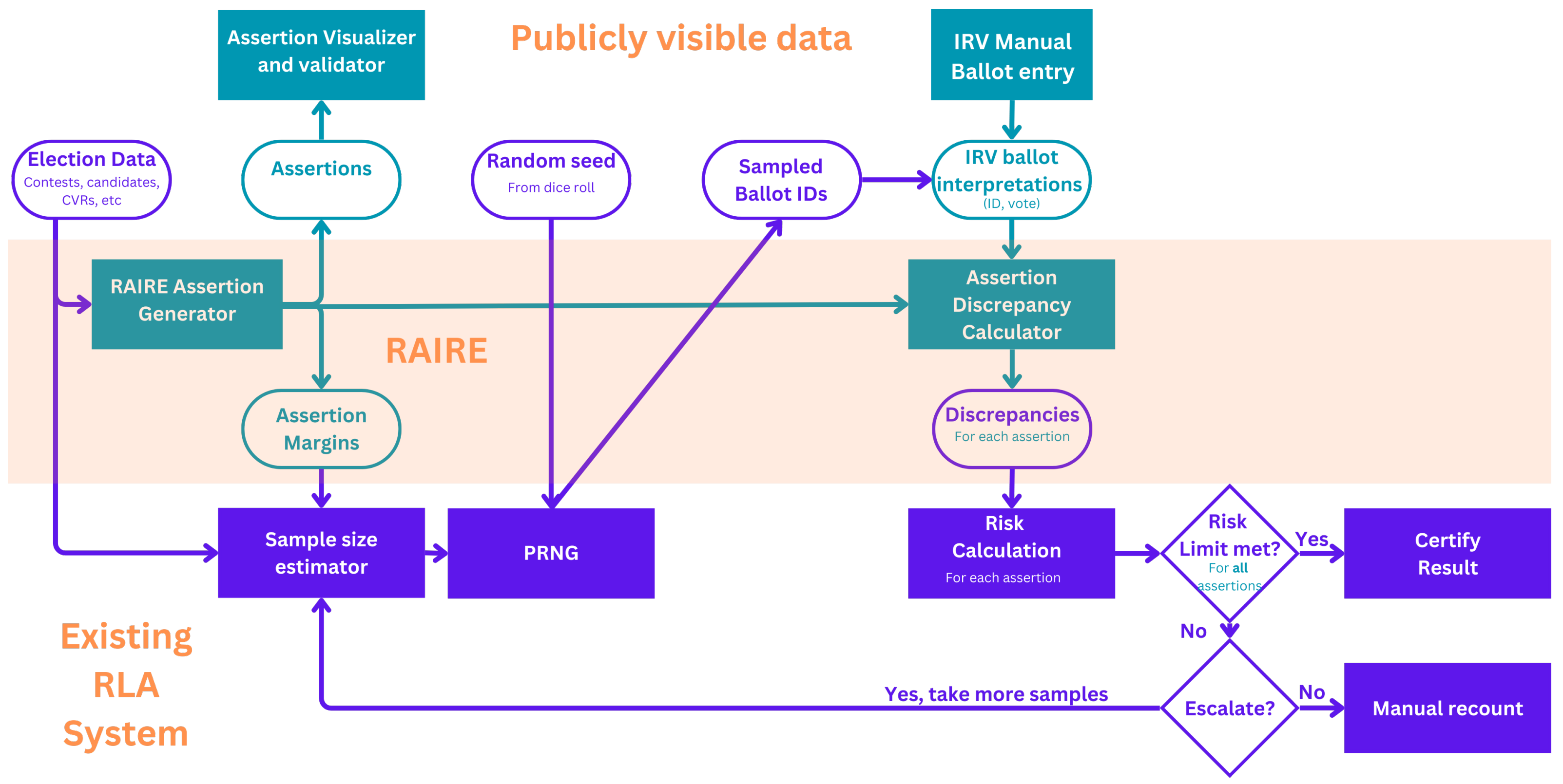


Figure 1.1: A high-level overview of an IRV RLA using RAIRE. Purple elements are existing RLA components. Green indicates new components and data structures for IRV.

The audit process from beginning to end

The basic process of an RLA remains the same as for plurality elections. The main difference is that an IRV outcome depends on multiple assertions, and all the assertions have to be accepted before the audit can confirm the result. The main parts of the workflow are summarised here. New steps for IRV are in bold.

1. Commit to the ballot manifest and CVRs.
2. Choose contest(s) for audit.
3. **Run RAIRE to generate assertions for audit.**
4. **Use the RAIRE assertion validation and visualization module to check that the assertions imply the announced winner won.**[1]
5. Generate a trustworthy random seed, e.g. by public dice rolling.
6. Estimate the required sample size, based on the margin, **for each assertion.**
7. Use the seed to generate the list of sampled ballots.
8. Retrieve the required ballots, compare them to their CVRs, and calculate the discrepancies **for each assertion.**
9. Update the risk **for each assertion** based on the observed discrepancies.
10. For each contest under audit, if the measured risk is below the risk limit **for each assertion**, stop the audit and accept the result.
11. If some results have not yet been confirmed, decide whether to escalate (sample more ballots) or conduct a full manual count.

[1]This can be done at any time, including after the audit is complete.

Chapter 2

IRV elections and Visualizing Outcomes

How IRV counts work

In an IRV election there are a set of candidates, C, who are ranked by voters. Each voter specifies their first preference (just like in plurality elections) and may then, optionally, specify second, third and later preferences. An IRV count proceeds in a series of *elimination* steps.

Initially, every candidate's first preferences are counted. This provides the first set of tallies for *IRV elimination*. We call the tallies of candidates at this stage their *first preference tallies*.

IRV elimination proceeds as described below:

While there is more than one continuing candidate:

- From the continuing candidates, select the candidate l with the lowest tally.
- *Eliminate* l: give each vote in l's tally to the next-preferred continuing candidate on that ballot.

An *elimination order* is a list of candidates in the order they were eliminated. By convention, we write the winner (who is never eliminated) last. There can be many different elimination orders that lead to the same winner. In practice, IRV counts often stop when one candidate w has more than 50% of the votes—at that stage, there is no chance that anyone other than w will remain at the end, so we can cut the algorithm short and declare that w is guaranteed to win.

Example 1. *Suppose there are 4 candidates: Alice, Bob, Chuan and Diego. The votes are as follows:*

Preferences	***Count***
(A, B, C, D)	*5*
(B, A, C)	*1*
(B, D, A)	*1*
(C)	*2*
(C, D)	*2*
(D, C)	*4*

We first count the first preference tallies: Alice has 5 votes, Bob has 2, Chuan has 4 and Diego has 4. This means that Bob is eliminated first and his votes distributed to the next-preferred candidate on each ballot—Diego gets the (B, D, A) vote and Alice gets the (B, A, C) vote. The new tallies are Alice: 6, Bob: eliminated, Chuan: 4, Diego: 5. Now Chuan is eliminated. Diego gains two more votes–the (C, D) ballots–and wins with 7 votes compared to Alice's 6.

The full sequence of tallies is shown below.

Count	**Remaining**	**Alice's tally**	**Bob's tally**	**Chuan's tally**	**Diego's tally**	**Action**
1	A, B, C, D	5	2	4	4	Eliminate B
2	A, C, D	6	-	4	5	Eliminate C
3	A, D	6	-	-	7	Eliminate A; D wins

Visualizing all possible IRV outcomes

Let us represent the number of candidates in the set C as $|C|$. An IRV election with $|C|$ candidates has $|C|! = |C| \times (|C| - 1) \times (|C| - 2) \times \cdots \times 3 \times 2$ possible (complete) elimination sequences. We can visualize these as a collection of *trees*. A *tree*, in computer science parlance, is a way of organizing data hierarchically. Figure 2.1 depicts an example of a tree data structure. A tree contains a number of *nodes*, where each node can be thought of as a container of data. These nodes are organized hierarchically, and connected by *edges*. In the example shown in Figure 2.1, there are five nodes, each labeled from one to five. Node 1 is connected to node 2 by an edge, and to node 3 by an edge.

When describing how we visualize the space of possible outcomes of an election, and how RAIRE works, we will use some additional terminology relating to trees.

Child	A node, c, is a *child* of another node, p, if c sits underneath p and c is connected to p by a single edge. In our example, nodes 2 and 3 are children of node 1. Nodes 4 and 5 are children of node 2.

An example of a *tree*

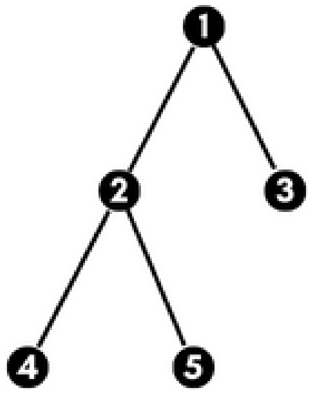


Figure 2.1: A tree data structure.

Parent	If node c is a child of node p, then node p is the *parent* of c. In our example, node 1 is the parent of nodes 2 and 3, while node 2 is the parent of nodes 4 and 5.
Ancestor	The ancestors of a node n are the set of all nodes that we can reach from n by moving up the tree from child node to parent. In our example, nodes 2 and 1 are the ancestors of nodes 4 and 5. Node 1 is the only ancestor of nodes 2 and 3.
Descendant	The descendants of a node n are the set of all nodes that we can reach from n by moving down the tree from parent node to child. In our example, nodes 4 and 5 are the descendants of node 2. Nodes 2, 3, 4 and 5 are the descendants of node 1.
Leaf	A *leaf* is a node without children.
Branch	A *branch* is a path of connected nodes starting at the very top of the tree, and ending at a leaf. In our example, the sequences of nodes 1–2–4, 1–2–5 and 1–3 represent the branches of our tree.

We visualize the space of possible elimination orders for an election as a *collection* of trees, one for each possible winner (including the reported winner). We call each of these trees an *elimination tree*. For an election between candidates Alice, Bob, Chuan, and Diego, the top level of each of these elimination trees is shown in Figure 2.2.

Consider the first node depicted in Figure 2.2. This node represents all outcomes that *end* with Alice as the winner. Similarly, the second depicted node represents all outcomes that *end* with Bob as the winner. Each node in an elimination tree represents either a *complete* or a *partial* outcome. Nodes 1 to 4 in Figure 2.2 represent partial outcomes as they do not express a complete elimination order.

RAIRE visualizes all elimination orders that end with Alice as the winner as shown in Figure 2.3. At the second level of the tree, we add a candidate as the runner-up. (The runner-up is the last candidate eliminated, though in IRV this is not necessarily the losing candidate who came closest to winning.)

Elimination Trees – First Level

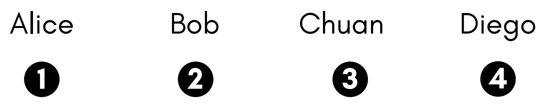


Figure 2.2: Top level of each elimination tree for a contest between Alice, Bob, Chuan, and Diego.

Elimination Tree with Winner Alice

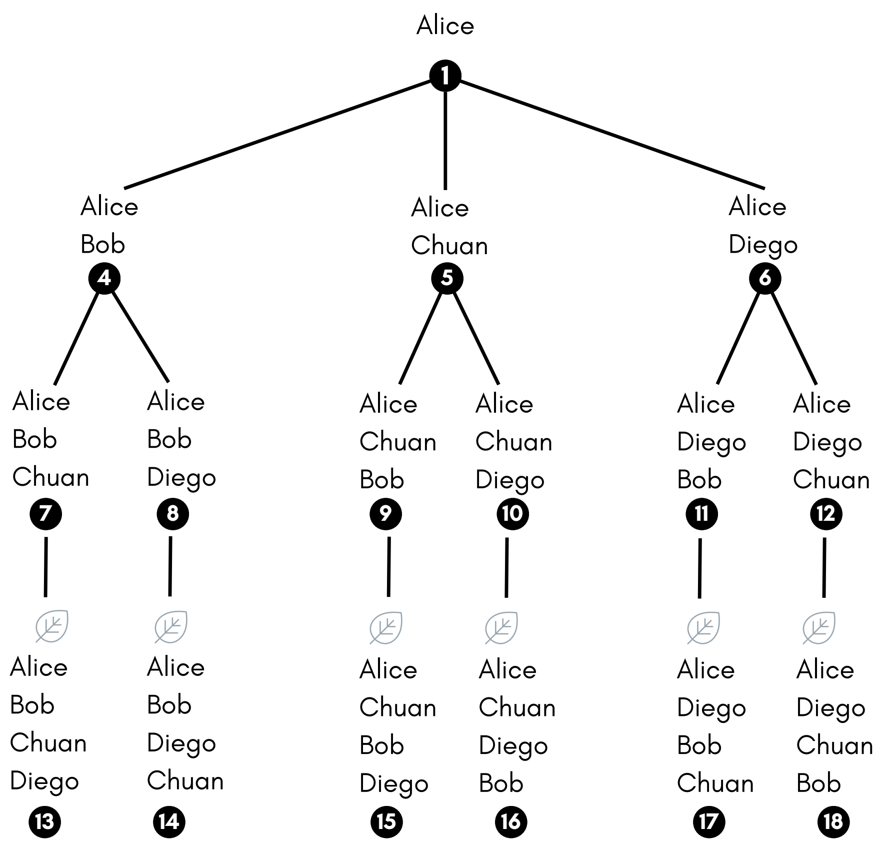


Figure 2.3: All possible elimination orders where Alice is the ultimate winner. The leaves of this elimination tree represent *complete* elimination orders.

Node 4 represents all outcomes that end with Bob as the runner-up and Alice as the winner. In nodes 5 and 6, Chuan and Diego, respectively, are the runner-up candidates.

The third level of the tree in Figure 2.3 identifies a candidate to be eliminated just prior to our runner-up. The leaves, on the fourth level, represent complete elimination orders. Node 13, for example, represents an elimination order in which Diego is eliminated first and Chuan second, leaving Bob as the runner-up, and Alice as the winner. Nodes 13–18 in Figure 2.3 represent *complete* outcomes.

The tree in Figure 2.3 captures all elimination orders that end with Alice as the ultimate winner. The complete set of **alternate outcome trees** that RAIRE considers is formed by a collection of such trees, one *for each reported loser*. To save space we often label each node with only the candidate eliminated at that step, rather than the whole elimination order. The whole order can be read by tracing up to the top. So each path up a tree represents an elimination order, with the first-eliminated candidate at the leaf and the winner at the top. We demonstrate this idea in the next example.

Example 2. *San Francisco District Attorney's contest, 2019.*

Figure 2.4 shows the complete set of elimination trees for the San Francisco 2019 DA's contest, including the (vote-by-mail) winner. Each tree shows all the different ways that a particular candidate can win. The bright red line shows the official elimination order: Dautch, Tung, Boudin, Loftus, leading to a win by Loftus. The rest of that tree shows other elimination orders that lead to a Loftus win. The other trees show wins by other candidates, and the elimination orders that lead to those wins.

2019 San Franciso DA's Race

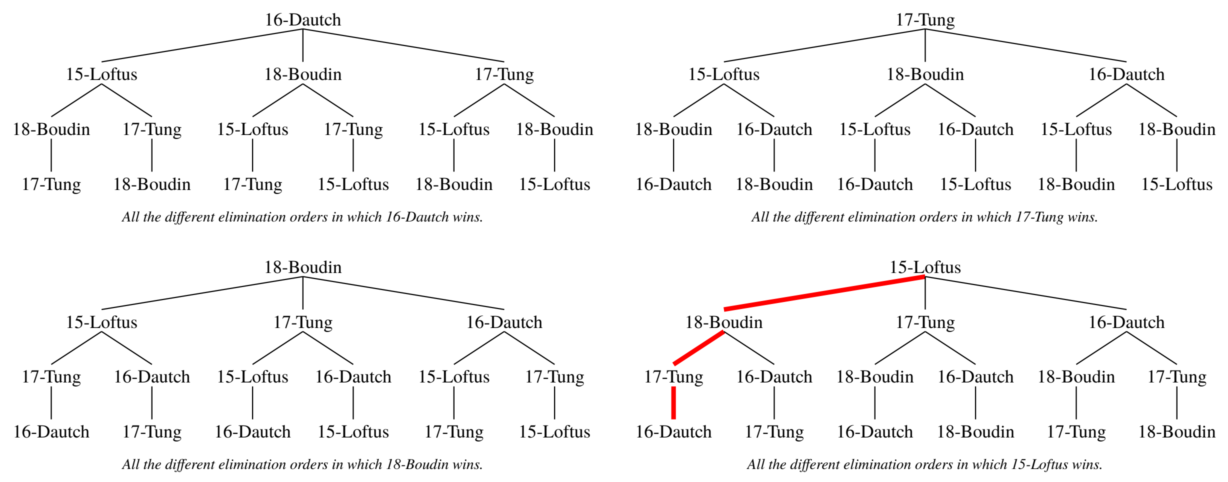


Figure 2.4: Complete elimination trees for the 2019 San Francisco DA race.

Exercises

Exercise 1. *Tally the following IRV example election.*

Preferences	***Count***
(A, B, C, D)	*50*
(A, C)	*40*
(B, C, A)	*25*
(B, D, A)	*25*
(C, A, B)	*30*
(C, D, B)	*45*
(D)	*100*

Exercise 2. *Draw the complete set of elimination trees for Exercise 1. You can ignore the tree for the apparent winner.*

Exercise 3. *Suppose there are 6 candidates.*

a) How many different complete elimination sequences are there?

b) How many leaves are there in each elimination tree?

c) How many leaves are there altogether in all the apparent losers' elimination trees?

Chapter 3

Assertions for IRV winners

We can verify the outcome of an IRV election with only two types of assertions. These are called Not Eliminated Before (NEB) assertions and Not Eliminated Next (NEN) assertions. To explain these assertion types, we will consider an election with four candidates: Alice, Bob, Chuan and Diego.

Not Eliminated Before (NEB) Assertions

Alice NEB Bob is an assertion saying that Alice cannot be eliminated before Bob, irrespective of which other candidates are continuing. In other words, no outcome is possible in which Alice is eliminated before Bob. When expressed as a comparison of tallies, this assertion says that the *smallest* number of votes Alice can have, at any point in counting, is greater than the *largest* number of votes Bob can ever have while Alice is continuing. Alice's smallest tally is equal to her first preference count – the number of ballots on which she is ranked first. The largest number of votes Bob can have while Alice is continuing is the number of ballots on which he is ranked higher than Alice, or he is ranked and Alice is not.

Example 3. *In the following example, Alice NEB Bob is true because Bob is ranked a total of 80 times without being preceded by Alice, which is less than Alice's first-preference tally of 100. However, Alice NEB Diego is not true, because Diego is ranked 125 times without being preceded by Alice, which is more than Alice's first preference tally.*

Preferences	***Count***
(A, B, C, D)	*100*
(B, D, C)	*40*
(C, B, D)	*40*
(C, D)	*45*

Not Eliminated Next (NEN) Assertions

NEN assertions compare the tallies of two candidates under the assumption that a specific set of candidates have been eliminated. An instance of this kind of assertion could look like this: *NEN: Alice > Bob if only {Alice, Bob, Diego} remain*. This means that in the context where Chuan has been eliminated, Alice cannot be eliminated next, because Bob has a lower tally. When expressed as a comparison of tallies, this assertion says that the number of ballots in Alice's tally pile, in the context where only Alice, Bob and Diego are continuing, is greater than the number of ballots in Bob's tally pile in this context. This example assumes one eliminated candidate – Chuan – however, NEN assertions can be constructed with contexts involving no eliminated candidates, or more than one eliminated candidate. The assertion *NEN: Alice > Chuan if only {Alice,Bob,Chuan,Diego} remain* says that Alice cannot be the first eliminated candidate, as she has more votes than Chuan when no candidates have yet been eliminated. The assertion *NEN: Diego > Bob if only {Bob,Diego} remain* says that Diego has more votes than Bob in the context where those two are the only continuing candidates.

Simple assertions sometimes work

RAIRE works by generating a set of assertions which, together, imply a particular winner. In this section, we introduce some common patterns that those sets of assertions might use. We aim to make it obvious why certain sets of assertions are enough to imply a particular winner, and to match a person's intuition about why a certain candidate won an IRV election.

One candidate dominates

Sometimes one candidate happens to be so strongly ahead of all the others that NEB assertions hold with all other candidates.

Example 4. *Suppose that for the four candidates Alice, Bob, Chuan and Diego, we have:*

Alice NEB Bob,
Alice NEB Chuan and
Alice NEB Diego.

Then Alice must be the winner—it is not possible for her to be eliminated at any stage of the IRV count.

Although this might sometimes be enough, real elections are usually closer. We need to look for other patterns of assertions that are enough to imply that one candidate won.

Two leading candidates

Now suppose there are two candidates who accumulate most of the votes: Alice and Bob.

Example 5. *Suppose Alice NEB Bob is not true, but the following weaker fact is true:*

NEN: Alice > Bob if only {Alice, Bob} remain.

This says that, after Chuan and Diego are eliminated, Alice's tally is higher than Bob's.

Assume we still have two NEB assertions:

Alice NEB Chuan and
Alice NEB Diego.

This, again, is enough to prove that Alice won. To see why, consider the last elimination step. Alice must reach this step, because she cannot have been eliminated before Chuan or Diego. If Chuan or Diego is the other remaining candidate, Alice beats them (by the NEB assertion). The only other possibility is Bob—for this case, the NEN assertion shows that, in the last round, Alice beats Bob.

Visualizing assertions

This reasoning can be visualized using elimination trees. For an audit, we need to disprove all elimination orders that result in a winner other than the announced winner.

The assertion *Alice NEB Chuan* is enough to disprove every elimination order in which Alice is eliminated before Chuan, and hence to disprove the entire tree in which Chuan wins. It also allows us to disprove the orders in Bob's tree and Diego's tree in which Alice is eliminated before Chuan.

The consequences of *Alice NEB Chuan* in Bob's tree and Chuan's tree are shown in Figure 3.1. It still allows the possibility that Bob might win via elimination orders Diego, Chuan, Alice, Bob, or Chuan, Diego, Alice, Bob, or Chuan, Alice, Diego, Bob.

Implications of NEB Assertions

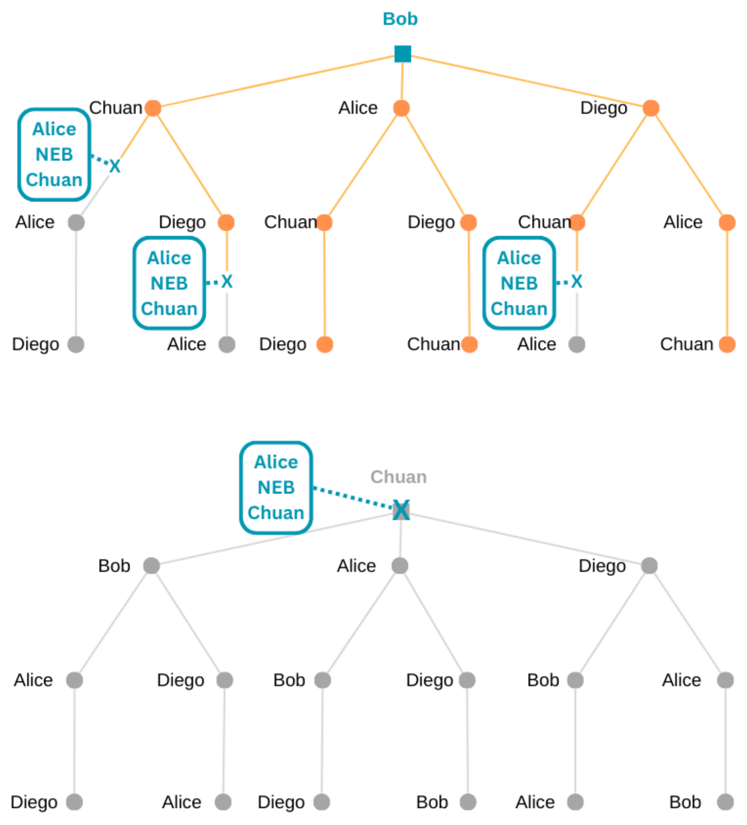


Figure 3.1: Implications of *Alice NEB Chuan* in the trees where Bob and Chuan win. Chuan cannot win, but Bob might.

Suppose we add another other NEB assertion: *Alice NEB Diego*. Now the only possible alternate winner is Bob. Figure 3.2 adds this assertion and the final assertion, *NEN: Alice > Bob if only {Alice, Bob} remain*, leaving Alice as the only possible winner.

This pattern, with two leading candidates and several others who are easily excluded, is common in IRV elections. In the Australian state of New South Wales, this set of assertions is true in most cases.

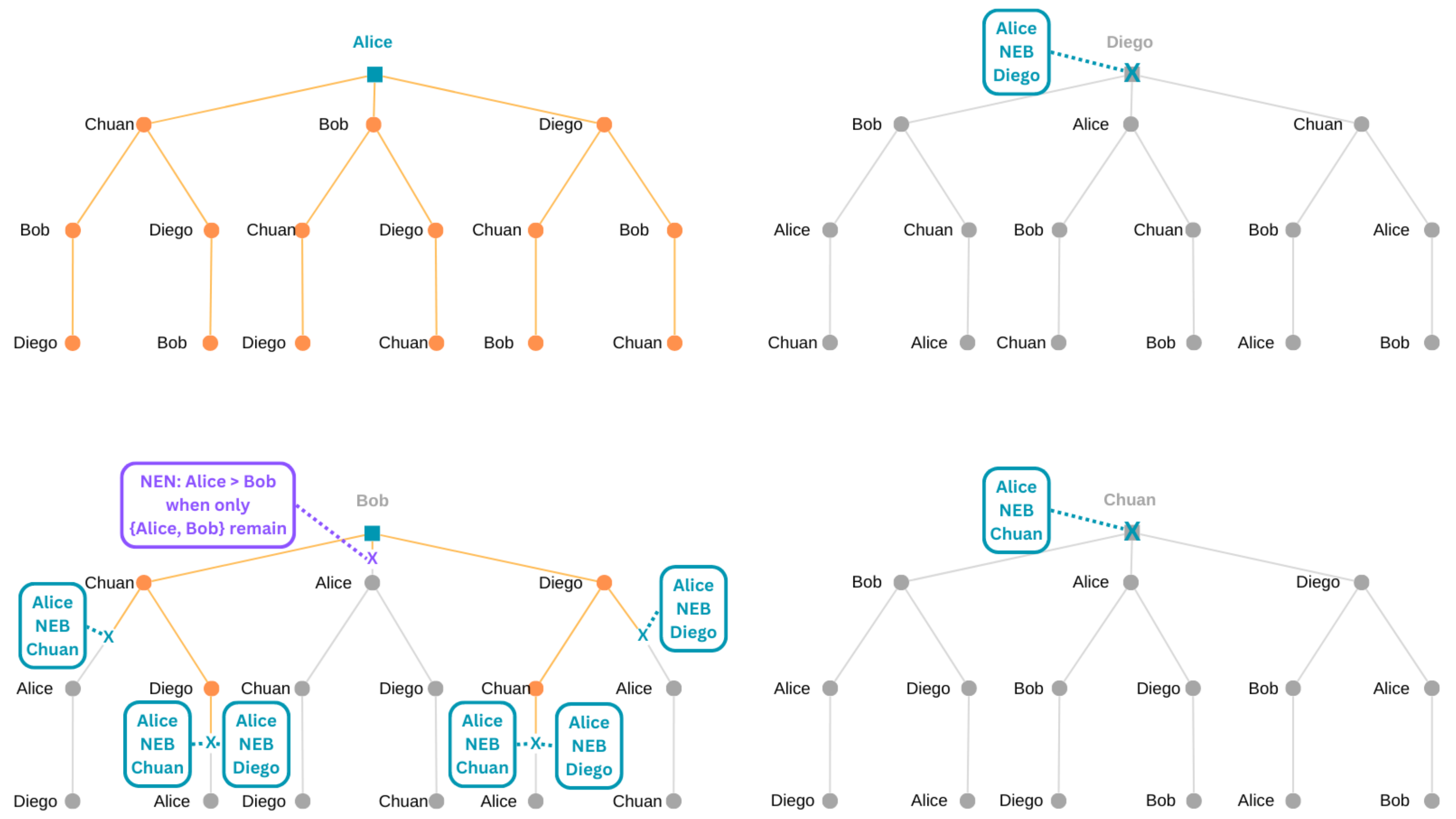


Figure 3.2: Implications of *Alice NEB Chuan* combined with *Alice NEB Diego* and *NEN: Alice > Bob if only {Alice,Bob} remain*. Alice is the only possible winner.

Why not audit every elimination step?

A common suggestion is to conduct an IRV RLA by testing every elimination step. This is valid, but it can waste a lot of effort verifying comparisons that do not alter who wins.

Example 6. *Suppose there are 4 candidates, Alice, Bob, Chuan and Diego. The votes are as follows:*

Preferences	***Count***
(A)	*10,000*
(B, A, C)	*5,000*
(C, A, B)	*4,999*
(D, C, B, A)	*16,000*

The full sequence of tallies is shown below.

Count	***Continuing***	***Alice's tally***	***Bob's tally***	***Chuan's tally***	***Diego's tally***	***Action***
1	A, B, C, D	*10,000*	*5,000*	*4,999*	*16,000*	*Eliminate* C
2	A, C, D	*14,999*	*5,000*	-	*16,000*	*Eliminate* B
3	A, D	*19,999*	-	-	*16,000*	A *wins*

Auditing the first elimination step would need a lot of samples, because the difference between the two lowest candidates, Bob and Chuan, is only one vote. However, it is easy to see that it does not matter whether Bob or Chuan is eliminated first. If the CVRs are wrong, and the truth is instead that there are 4,999 (B, A, C) *votes and 5,000* (C, A, B) *votes, then the correct tally sequence is the following.*

Count	***Continuing***	***Alice's tally***	***Bob's tally***	***Chuan's tally***	***Diego's tally***	***Action***
1	A, B, C, D	*10,000*	*4,999*	*5,000*	*16,000*	*Eliminate* B
2	A, C, D	*14,999*	-	*5,000*	*16,000*	*Eliminate* C
3	A, D	*19,999*	-	-	*16,000*	A *wins*

This produces the same winner.

Auditing effort—the number of ballots that need to be sampled in order to be confident that the announced winner won—tends to be higher for small margins. Auditing every elimination step would waste effort, not because there are many assertions, but because the very close comparison between Bob and Chuan would require a large sample to confirm. This effort does not need to be expended in this example, because the comparison between Bob and Chuan is irrelevant to who wins. RAIRE does not generate any assertion to test that comparison.

Of course, for some IRV elections, early eliminations do matter. RAIRE examines all the CVRs and determines which comparisons can be ignored and which need to be tested with an RLA.

More complicated sets of assertions

Sometimes the simple assertion sets above are not enough, or do not lead to an efficient audit. In these cases, we need to use RAIRE to generate a set of assertions that is both sufficient to prove that the announced winner won, and that minimizes the number of ballots auditors have to sample. This is described in Part 2 of this guide.

Verifying that a set of assertions implies a winner

Each assertion (if true) rules out certain elimination orders, removing the corresponding paths in the trees that visualize all IRV outcomes. If some candidate's tree has all paths down to its leaves removed, then that candidate cannot win: all the elimination orders that let them win have been ruled out.

How can we verify that a certain set of assertions truly implies that the announced winner won? There are three ways:

- Using the visualization trees to check that, for all other candidates, all the paths down to their leaves are cut by an assertion, or
- Using a logical argument like the example in the section Two Leading Candidates on p. 16, or
- Using software to do the validation.

This is a crucial part of the RAIRE audit, especially when the assertions are generated by RAIRE and do not fit into one of the simple, familiar patterns above.

Using the online visualiser and explainer tool

We have developed an online assertion visualiser—called "Explain Assertions"—at `https://github.com/DemocracyDevelopers/raire-rs/tree/main/WebContent/example_assertions` which automates the process of drawing the elimination trees and verifying that the announced winner won.

You can start with the default assertions (already there) or add your own.

Some examples from this book are available from `https://github.com/DemocracyDevelopers/raire-rs/tree/main/WebContent/example_assertions`. Click on the .json file you want, select the whole contents, and copy-paste it into the 'input' field of the Assertion Explainer. Alternatively, you can write your own.

Click the "Explain" button to display the trees.

If a tree has a large part that is not relevant to the display, it is summarised by a small triangle with a number in it indicating how many nodes it summarises.

There are two different display options.

Step-by-step The step-by-step form shows the effects of each assertion, one by one. At each step, the green node is the winner candidate in the assertion, while the pink node is the loser. Paths turn grey, and are then removed, when the assertion rules them out.

This version is most useful if you are learning about assertions.

Compressed The compressed form shows all the assertion implications at once, labelling each branch with the assertion that rules it out.

This version is most useful if you are already familiar with assertions and are using the visualiser to validate RAIRE's output for an audit.

In either case, it is important to check that all paths on all trees for all non-winners are removed.

The tool allows for some other options:

- Expand fully all elimination orders at start
- Draw as text rather than trees
- Don't bother drawing the (technically unnecessary) trees for the winning candidate.

All of these can be turned on or off according to your preference.

Exercises

Exercise 4. *Suppose there are three candidates: Alice, Bob and Chuan.*

a) Draw out all the elimination trees.

b) Suppose that the announced elimination order is (Bob, Alice, Chuan), so Chuan is the announced winner. Identify that elimination order in your trees.

c) Add the assertion Chuan NEB Alice. Show which elimination orders are removed from your trees.

d) Think of one more assertion which, if combined with Chuan NEB Alice, implies that Chuan wins. Show its implications on your trees.

e) Is your answer to Part d) the only possible answer?

Exercise 5. *Consider the San Francisco DA's contest with four candidates: Tung, Loftus, Dautch and Boudin. Suppose you are given a set of three assertions:*

- *Loftus NEB Tung*
- *Loftus NEB Dautch*
- *NEN: Loftus $>$ Boudin if only {Loftus, Boudin, Dautch} remain*

Does this imply that Loftus won? Either argue that it does, or provide an alternative winner via an elimination order that is consistent with all these assertions.

Exercise 6. *Write all the assertions for checking every elimination step in Example 6 to show that Alice won. Hint: you will need a lot of NEN assertions, with sets that match the set of candidates still continuing in the table.*

Exercise 7. *Write a different set of assertions to check that Alice won in Example 6 without checking every elimination step. Hint: try some NEB assertions.*

Exercise 8. *Make up an example IRV election in which the comparison between the lowest two candidates in the first elimination round does matter.*

Chapter 4

Risk Limiting Audits

This section reviews Risk Limiting Audits. Although RAIRE can work with any RLA, we concentrate on *ballot-level comparison RLAs* because these are currently used by the State of Colorado. Our presentation follows Lindeman and Stark's *Super-simple simultaneous Risk Limiting Audits* (Stark [2010]). Skip this chapter if you are already familiar with RLAs.

The aim of an RLA is to test an *apparent election outcome*—usually a single winning candidate w—by examining a random sample of ballot papers. An RLA may either accept the outcome, or fall back to a full manual recount in order to find the true winner. The *Risk Limit* α is chosen in advance, often by legislation. If the apparent outcome is wrong, the audit is guaranteed not to accept it except with probability at most α.

For plurality elections, the *apparent margin* between an apparent winner w and and apparent loser l is simply the winner's tally minus the loser's, according to the CVRs. The *actual margin* is $w's$ tally minus $l's$ according to the ballots. If the actual margin is negative, w is not the true winner. The *diluted margin* is the apparent margin as a fraction of total votes cast in the contest. Stark [2010] defines the diluted margin as the apparent margin divided by the total number of *ballots cast in contests under audit*, but since CO audits each contest separately, our simpler definition is equivalent.

First, administrators commit to a *ballot manifest*—a trustworthy list of the paper ballots—and a *cast vote record* (CVR) for each ballot. Having seen the CVR commitments, the public can then participate in a dice rolling ceremony to seed a *pseudorandom number generator* (PRNG) which generates a series of ballot IDs for sampling. The number of ballots to be sampled is a function of the risk limit, the margin, and the expected number (and type) of errors.

Ballot-level comparison RLAs proceed by taking a random sample of ballot papers and checking each one against its corresponding CVR. An error that advantages the apparent winner is an *overstatement*; an error that advantages an apparent loser is an *understatement*. The *discrepancy* is the difference between the CVR and the actual ballot—by convention, overstatements are positive discrepancies, while understatements are negative.

Example 7. *Suppose Diego is the reported winner and Chuan is a reported loser in a plurality contest.*

- *If the CVR records a vote for Diego but the ballot paper has a vote for Chuan, that is a discrepancy of 2, i.e. a* 2-vote overstatement.
- *If the CVR records an invalid vote, but the ballot paper has a vote for Chuan, that is a discrepancy of 1, i.e. a* 1-vote overstatement.
- *If the CVR and the ballot paper are identical, that is a discrepancy of 0.*
- *If the CVR records an invalid vote, but the ballot paper has a vote for Diego, that is a discrepancy of -1, a* 1-vote understatement.
- *If the CVR records a vote for Chuan but the ballot paper has a vote for Diego, that is a discrepancy of -2, i.e. a* 2-vote understatement.

Obviously overstatements are far more important than understatements—if there are enough overstatements, the apparent winner may be wrong.

The discrepancies for all the sampled ballots are input into to a *risk measuring function*. If the measured risk is less than the Risk Limit α, the audit can stop and accept the announced election result. Otherwise, officials can decide to either *escalate*—take more samples and continue the audit—or perform a full hand count to establish who really won.

For more careful explanations of how to run an RLA, see the Democracy Fund's RLA Standard Operating Procedures 2022[1] or their excellent "Knowing it's right" series[2].

[1] https://static1.squarespace.com/static/61da19f9986c0c0a9fd8435f/t/620e599548201f38d44ada3d/1645107606630/RLA+SOP.pdf

[2] https://democracyfund.org/idea/knowing-its-right-limiting-the-risk-of-certifying-elections/

Chapter 5

Using assertions to audit IRV outcomes

Now for the audit! The main insight of RAIRE is *we can verify an IRV outcome using assertions, without having to verify the correctness of every elimination step.*

We have a set of assertions which imply that the announced winner won. We could (but we won't) verify each assertion by manually examining all the ballot papers. For example, *Diego NEB Chuan* could be verified by manually counting all the first preferences for Diego and checking that that tally was greater than the (manually counted) total number of times Chuan was preferenced without being preceded by Diego. Similarly, NEN assertions could be verified by counting the tallies ignoring candidates not specified in the assertion. For example, we could verify *NEN: Bob > Chuan if only {Bob,Chuan,Diego} remain* by sorting every ballot into a tally pile according to which of Bob, Chuan, and Diego was the highest preference, ignoring any preference for Alice, and then checking that Bob's total was higher than Chuan's. If we did this for every assertion in the set, it would be a logically sound way to verify the election result, but it would be very inefficient.

Instead, we test each of the assertions using an RLA at some risk limit α. If the audit accepts them all, we conclude the audit and accept the IRV election result.

If the announced winner is wrong, then at least one of the assertions must be false. Since we test each assertion with an RLA at risk limit α, the RLA for the wrong assertion will mistakenly accept it with probability at most α. Hence the overall process is a valid Risk Limiting Audit— it will mistakenly accept the wrong outcome with probability at most α.

Both types of assertions—NEB and NEN—can be tested with standard RLA systems, but they need to be carefully transformed into the right form.

NEB Assertions

Scoring NEB assertions

An NEB assertion, for example *Alice NEB Bob*, says that Alice's first preferences exceed the total number of mentions of Bob that are not preceded by a higher preference for Alice.

We start with a set of CVRs and count them as follows:

Ballot contents	**Counted for**	**Example**
First preference for Alice	Alice 1st preference	(A, B, C, D)
Mentions Bob *without* higher preference for Alice	Bob mention	(C, B, D, A)
Mentions Bob *with* higher preference for Alice (not first)	Neither	(C, A, B)
Other	Neither	(C, A, D)

This fits naturally into any existing Risk-limiting audit process, except that our two candidates are "first preferences for Alice" and "mentions of Bob not preceded by Alice."

Auditing NEB assertions

Consider again the assertion *Alice NEB Bob*. To conduct a ballot-level comparison audit of this assertion, the process for randomly selecting ballots for audit is the same as any other RLA. When a ballot is selected, overstatements are errors that advantage the "first preferences for Alice" candidate, while understatements are errors that advantage the "mentions of Bob (not preceded by a higher preference for Alice)" candidate. An *overstatement* is an error that either mistakenly records a first preference for Alice, or mistakenly omits a mention of Bob not preceded by Alice.

For example, if a CVR says that a vote is a first-preference for Alice, but the ballot paper shows only a second preference for her (and a first preference for some other candidate, say Diego), then this is a one-vote overstatement. If the ballot paper actually shows a mention of Bob, not preceded by a higher preference for Alice, then the error is a two-vote overstatement.

NEN Assertions

Scoring NEN assertions

An NEN assertion, for example *NEN: Alice $>$ Bob if only {Alice,Bob,Chuan} remain*, says that Alice beats Bob when Alice, Bob, and Chuan are the only continuing candidates.

This is also easy to fit into any existing Risk-limiting audit process, except that our two candidates are "Alice's tally when only Alice, Bob, Chuan remain" and "Bob's tally when only Alice, Bob, Chuan remain."

Assertion A	Scoring vote $\mathbf{b} = (\mathbf{c_1}, \mathbf{c_2}, \ldots, \mathbf{c_n})$ SCORE$_A(b)$	Notes
c_1 *NEB* c_k where $k > 1$	1	Supports 1st prefs for c_1
c_j *NEB* c_k where $k > j > 1$	0	c_j precedes c_k but is not first
c_j *NEB* c_k where $k < j$	-1	a mention of c_k preceding c_j
NEN: $c_i > c_k$ if only $\{\mathcal{S}\}$ remain		
where c_i is b's first-ranked candidate in S	1	counts for c_i (expected)
where neither c_i nor c_k is b's first-ranked candidate in S	0	counts for neither c_j nor c_k
where c_k is b's first-ranked candidate in S	-1	counts for c_k (unexpected)

Table 5.1: Complete scoring for vote $(c_1, c_2, \ldots, c_n)$ for all assertions. Overstatements are counted by subtracting the ballot paper score from the CVR score. Understatements are counted by subtracting the CVR score the ballot score.

We start with a set of CVRs and count them as follows:

Ballot contents	Counted for	Example
Alice, not preceded by Bob or Chuan	Alice	(A, B, C, D)
Bob, not preceded by Alice or Chuan	Bob	(D, B, C, A)
Chuan, not preceded by Alice or Bob	Neither	(D, C)
Anything else	Neither	(D)

We simply allocate the vote as if Diego has been eliminated. The same works when sets of more than one candidate have been eliminated, or when there are more than 4 candidates: simply score the vote for the first-ranked candidate among those continuing.

Auditing NEN assertions

For an NEN assertion *NEN: Alice > Bob if only $\{\mathcal{S}\}$ remain*, an *overstatement* is an error that advantages Alice by mistakenly listing her as the highest preference in set $\mathcal{S}$, or disadvantages Bob by mistakenly not listing him first among $\mathcal{S}$. An understatement is the opposite.

For example, for the assertion *NEN: Alice > Bob if only {Alice,Bob,Chuan} remain*, if the CVR was (Diego, Alice, Bob, Chuan), but the ballot paper actually contained (Diego, Bob, Alice, Chuan), that would be a two-vote overstatement. (The first preference for Diego is ignored because we consider only {Alice, Bob, Chuan} as continuing.)

Assertion scoring summary

Table 5.1 shows how to score each ballot for each possible kind of assertion. Note that the score is a function of the assertion and the vote only - it does not depend on the apparent outcome.

The overstatement counts (discrepancies) are derived by simply subtracting the ballot paper score from the CVR score (and vice versa for understatements–understatements are derived from subtracting the CVR score from the ballot paper score). For example, if a CVR says that a vote contained a first preference for w, but the actual ballot contains a mention of l that precedes w, then it overstates the *w NEB l* assertion by $1 - -1 = 2$. By convention, overstatements are written as a positive discrepancy, while understatements are negative.

Examples

Returning to Example 5 on p. 16, some example scores for the CVR and (actual) ballots are given below, for each assertion. The overstatement is computed by subtracting the ballot (actual) score from the CVR (apparent) score.

Example 8. *Suppose the CVR is (A, B, C, D) and the ballot is (B, A, C, D). Scores for the CVR and ballot are given below, with the corresponding overstatement.*

Assertion	*CVR Score*	*Ballot Score*	*Overstatement*
Alice NEB Chuan	*1*	*0*	*1*
Alice NEB Diego	*1*	*0*	*1*
NEN: Alice > Bob if only {Alice, Bob} remain	*1*	*-1*	*2*

This error overstates Alice's win—too many errors like this may mean that the announced election outcome is wrong.

Example 9. *Suppose the CVR is (D, C, B, A) and the ballot is (C, D, B, A). Scores for the CVR and ballot are given below, with the corresponding overstatement.*

Assertion	*CVR Score*	*Ballot Score*	*Overstatement*
Alice NEB Chuan	*-1*	*-1*	*0*
Alice NEB Diego	*-1*	*-1*	*0*
NEN: Alice > Bob if only {Alice, Bob} remain	*-1*	*-1*	*0*

Although there is an error, it causes no discrepancies for any of the assertions. This means the error does not affect the logic for proving who wins.

Doing the audit

Now the audit can proceed exactly like a plurality audit, using existing plurality audit software, except that an IRV outcome should be confirmed only if *all* of its assertions are confirmed.

In order to estimate the required sample size, we need to compute an *Assertion Margin* for each assertion—this is simply the sum of the CVR scores for that assertion. This is the number of overstatements that would need have occurred for that assertion to be truly false (according to the paper ballots). Just like plurality contests, IRV assertions with smaller margins tend to require larger samples to confirm. The overall sample size is approximately the sample size required by the closest margin.

As ballots are sampled, their discrepancies are computed as described above. This can be done by hand, but it is easier for people if they can simply enter the ballot they see into some software which calculates the discrepancies for each assertion. The discrepancies are then entered into the risk calculation, which is the same as for plurality.

The audit of an IRV contest can stop and confirm the result when all of its assertions are confirmed. If there are some assertions which remain above the risk limit after the whole sample is taken, officials can decide whether to take a larger sample or do a manual recount.

Assertions, audits and outcomes: logical summary

- If all of the assertions are true, and the set of assertions implies that the announced winner won, then the announced winner won.
- The announced winner may be the true winner, but not via the announced elimination order. RAIRE does not verify the elimination order, just the winner.
- The announced winner may be the true winner, but some of the assertions may be false if they won via a different elimination order from the announced one. In this case (with probability at least $1 - \alpha$) RAIRE will resort to a full hand count, which will confirm the announced winner.
- If the announced winner is wrong, then at least one of the assertions must be false. In this case, with probability at least $1 - \alpha$, RAIRE will fall back to a full hand count (with IRV elimination steps) and thus find the true winner.
- The audit may pass even if the announced winner is wrong, but this happens with probability at most α, the risk limit.

Exercises

Exercise 9. *Suppose the CVR is (D, C, B, A) and the ballot is (C, D, B, A). Fill in the table below with the CVR score, ballot score and overstatement.*

Assertion	*CVR Score*	*Ballot Score*	*Overstatement*
Chuan NEB Alice			
Chuan NEB Diego			
NEN: Chuan > Bob if only {Alice, Bob, Chuan} remain			

Exercise 10. *Suppose the CVR is (B) and the ballot is (B, A, C, D). Fill in the table below with the CVR score, ballot score and overstatement.*

Assertion	*CVR Score*	*Ballot Score*	*Overstatement*
Bob NEB Chuan			
NEN: Alice $>$ Chuan if only {Alice, Bob, Chuan} remain			
NEN: Alice $>$ Chuan if only {Alice, Chuan, Diego} remain			
NEN: Chuan $>$ Bob if only {Alice, Bob, Chuan} remain			

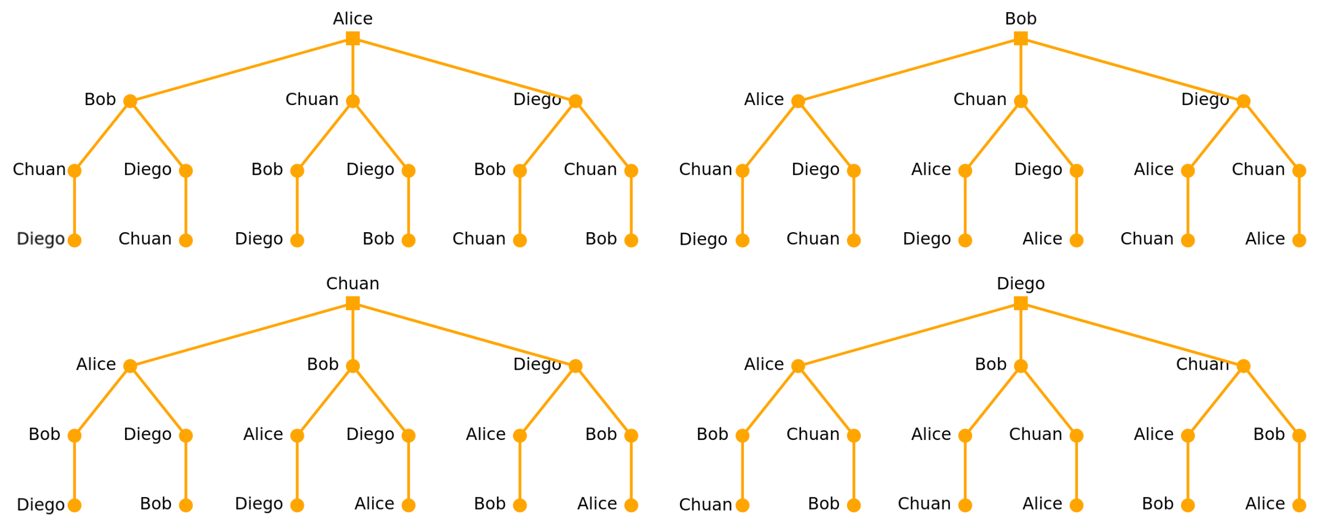
Chapter 6

Solutions to Exercises

Exercise 1

Count	Remaining	Alice's tally	Bob's tally	Chuan's tally	Diego's tally	Action
1	A, B, C, D	90	50	75	100	Eliminate B
2	A, C, D	90	-	100	125	Eliminate A
3	A, D	-	-	190	125	Eliminate D; C wins

Exercise 2 The complete set of trees is:



You can omit the tree for Chuan.

Exercise 3

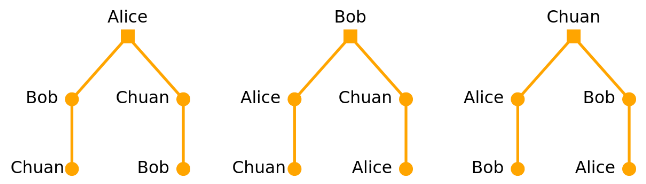
a) $6! = 6 \times 5 \times 4 \times 3 \times 2 = 720.$

b) $5! = 5 \times 4 \times 3 \times 2 = 120.$

c) $5 \times 5 \times 4 \times 3 \times 2 = 600.$

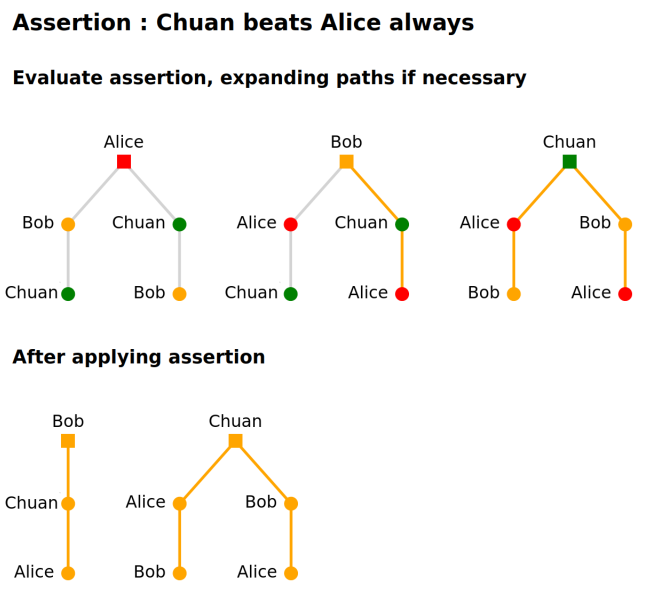
Exercise 4

a) The trees are:



b) It is the branch that has Bob as a leaf and Chuan at the top.

c) The grey lines in the picture below show what is removed by *Chuan NEB Alice*



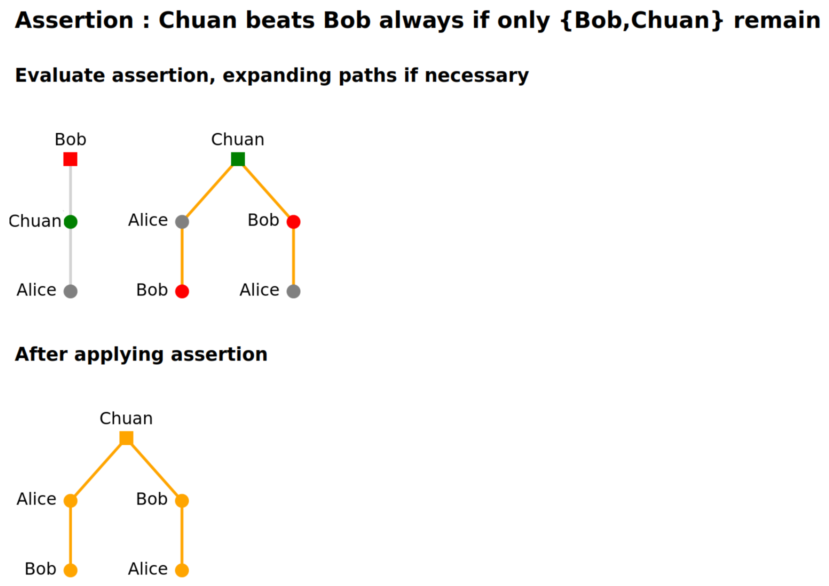
The picture was derived by entering the following data into the explainer at `https://democracydevelopers.github.io/raire-rs`. You may need to select the "show step-by-step" option and "expand fully all elimination orders."

```
{
  "metadata":{
    "candidates":["Alice","Bob","Chuan"],
    "note":"One possible solution for Exercise 4 of the Guide to RAIRE"
  },
  "solution":{
    "Ok":{
      "assertions":[
        {"assertion":{"type":"NEB","winner":2,"loser":0}},
        {"assertion":{"type":"NEN","winner":2,"loser":1,"continuing":[1,2]}}
      ],
      "winner":2,
      "num_candidates":3
    }
  }
}
```

d) Any one of:

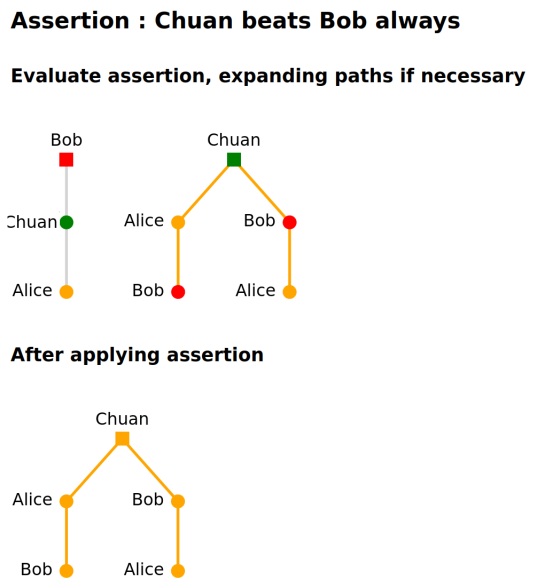
1. *NEN: Chuan > Bob if only {Chuan, Bob} remain*
2. *Chuan NEB Bob*
3. *NEN: Alice > Chuan if only {Alice, Bob, Chuan} remain*

The tree diagram with *NEN: Chuan > Bob if only {Chuan, Bob} remain* added is:


Assertion : Chuan beats Bob always if only {Bob,Chuan} remain
Evaluate assertion, expanding paths if necessary
Bob
Chuan
Alice
Chuan
Alice
Bob
Bob
Alice
After applying assertion
Chuan
Alice
Bob
Bob
Alice

It was generated using the assertions given in part (c).

If you added *Chuan NEB Bob* instead, the tree diagram is:


Assertion : Chuan beats Bob always
Evaluate assertion, expanding paths if necessary
Bob
Chuan
Alice
Chuan
Alice
Bob
Bob
Alice
After applying assertion
Chuan
Alice
Bob
Bob
Alice

It was generated by replacing the NEN assertion in part (c) with

```
"assertion":"type":"NEB","winner":2,"loser":1.
```

d) No, there are (at least) 3 possible answers (see part d)).

Exercise 5 No, these assertions do not imply that Loftus won. Boudin could win by either elimination sequence Tung, Dautch, Loftus, Boudin or Dautch, Tung, Loftus, Boudin.

Exercise 6 There are several possible correct sets of assertions. For each elimination step, for each candidate who does not get eliminated at that step, we need to write an assertion that they have a greater tally than some other candidate.

One option is to use NEN assertions only.

Round 1: all of

- *NEN: Alice > Bob if only {Alice Bob Chuan Diego} remain* (or instead *NEN: Alice > Chuan if only {Alice Bob Chuan Diego} remain*)
- *NEN: Bob > Chuan if only {Alice Bob Chuan Diego} remain*
- *NEN: Diego > Bob if only {Alice Bob Chuan Diego} remain* (or instead *NEN: Diego > Chuan if only {Alice Bob Chuan Diego} remain* or *NEN: Diego > Alice if only {Alice Bob Chuan Diego} remain*)

Round 2: all of

- *NEN: Alice > Bob if only {Alice Bob Diego} remain*
- *NEN: Diego > Bob if only {Alice Bob Diego} remain* (or instead *NEN: Diego > Alice if only {Alice Bob Diego} remain*)

Round 3: all of

- *NEN: Alice > Diego if only {Alice Diego} remain*

Exercise 7 There are many different correct solutions. In this example, *Diego NEB Bob* and *Diego NEB Chuan* are both true. These can be used instead of NEN assertions with Diego as the winner.

One option is:

- *Diego NEB Bob*
- *Diego NEB Chuan*
- *NEN: Alice > Diego if only {Alice Diego} remain*
- *NEN: Alice > Bob if only {Alice Bob Diego} remain*
- *NEN: Alice > Chuan if only {Alice Chuan Diego} remain*
- *NEN: Alice > Bob if only {Alice Bob Chuan Diego} remain*

This allows for either elimination sequence Bob, Chuan, Diego, Alice or Chuan, Bob, Diego, Alice.

Exercise 8 Here is one simple example.

Preferences	**Count**
(A)	8,000
(B, A, C)	5,000
(C, B)	4,999

If B is eliminated first, A wins with 13,000 votes. If C is eliminated first, B wins with 9,999 votes.

Exercise 9

Assertion	CVR Score	Ballot Score	Overstatement
Chuan NEB Alice	0	1	-1
Chuan NEB Diego	-1	1	-2
NEN: Chuan > Bob if only {Alice, Bob, Chuan} remain	1	1	0

Exercise 10

Assertion	CVR Score	Ballot Score	Overstatement
Bob NEB Chuan	1	1	0
NEN: Alice > Chuan if only {Alice, Bob, Chuan} remain	0	0	0
NEN: Alice > Chuan if only {Alice, Chuan, Diego} remain	0	1	-1
NEN: Chuan > Bob if only {Alice, Bob, Chuan} remain	-1	-1	0

Bibliography

Michelle Blom, Andrew Conway, Dan King, Laurent Sandrolini, Philip B. Stark, Peter J. Stuckey, and Vanessa Teague. You can do RLAs for IRV. In *E-Vote-ID*, TalTech Press Proceedings, pages 296–310, 2020.

Philip B Stark. Super-simple simultaneous single-ballot risk-limiting audits. In *EVT/WOTE*, 2010. `https://www.usenix.org/legacy/events/evtwote10/tech/full_papers/Stark.pdf`.

Philip B. Stark. Sets of half-average nulls generate risk-limiting audits: SHANGRLA. In *Financial Cryptography and Data Security. FC 2020*, volume 12063 of *Lecture Notes in Computer Science*, pages 319–336, 2020. Preprint: arXiv:1911.10035.