From: [redacted]
To: [redacted]
Subject: For Information: Independent test of senate counts [SEC=UNCLASSIFIED]
Date: Wednesday, 20 July 2016 2:52:00 PM
Attachments: image001.gif
image002.gif

UNCLASSIFIED

As discussed a member of the WA Greens, Grahame Bowland, wrote python code to replicate the Senate count. He found no issue with the AEC results. It was a completely independent activity, using the legislation as a source, and comparing to the distribution of preferences (I think that is what he is referring to).

His blog post on the matter is not available through the SOE, so I've replicated the text below. For reference, the software is still hosted on GitHub here: https://github.com/grahame/dividebatur

Counting the West Australian Senate Election

Posted on January 25, 2014 by Grahame Bowland

tl;dr – check out angrygoats.net/wasenate/ to see the full details of the WA senate election distribution – precisely how the count was processed by the AEC for both the initial count (#14votes) and the recount, and to see what happens if the "lost" votes are added back in, assuming they were counted right the first time. grab the code and run the distribution yourself, it's over on github.

Western Australia went to the polls with the rest of the country on September 7 2013, but failed to clearly elect six senators. When the votes were tallied a crucial exclusion on a margin of 14 votes was the difference between the election of Scott Ludlam (Greens) and Wayne Dropulich (Sports Party), or Louise Pratt (ALP) and Zhenya Wang (Palmer). The ALP & Palmer won on this count, but no result was declared and a recount was conducted.

After the initial count, several people asked me how the Australian Electoral Commission (AEC) processed the distribution of preferences, and whether there was a way to verify it. To my surprise the AEC software which takes all above-the-line and below-the-line votes and runs a Single Transferable Vote (STV) distribution is closed source and not available for download. I was unable to find any publicly available software to run the count. This made it extremely difficult to verify the correctness of AEC's results.

While the recount was going on I got to work on some software to process the distribution. This was greatly helped by a detailed run log from the AEC's software I managed to obtain. This told me exactly how the AEC's software ran the distribution, including totals of votes for each candidate at each count, and precisely when candidates were elected and excluded.

STV is really a family of voting systems – some details vary between jurisdictions. The information I could find on the AEC website on how to process the distribution was adequate only for educational purposes, and did not go into the fine details. This lead to much late night study of the Commonwealth Electoral Act. A bill of Parliament is a pretty awful means to document an algorithm, but everything needed is in Part XVIII.

The AEC publish all the data necessary to run a count on their Senate downloads page. It is simply a matter of taking the number of above-the-line (ticket) votes and counting these votes as that number of papers following the party's ticket(s). They also publish the form of every below-the-line paper lodged. Add those papers into one collection and you have all the information necessary to run the distribution, and determine who the elected candidates are.

In two long nights of work, I was able to get this done – exactly reproducing the AEC's results in my software. The code was pretty rough and ready, but I had functional tests which checked my results matched the AEC's. Being afraid of rounding errors I made sure to stick with rational number representations of transfer values, and was slightly hopeful this might turn up a minor error in the AEC's software. Disappointingly, but reassuringly, I found no fault at all in the AEC's software.

The AEC published above-the-line vote counts as the recount progressed. This made me hopeful I'd be able to run a count myself once the recount was complete and know the result before the official distribution was run. Unfortunately they did not enter the changes to the below-the-line votes (ballots rescued from the informal pile) until the last day, and did not make the data available until a couple of hours after they ran the count and declared the result. Gratifyingly my software agreed with theirs, the election now tipping to Scott Ludlam and Wayne Dropulich, the same crucial exclusion now going 12 votes the other way.

Of course, the AEC lost 1370 votes which were not included in the recount. The preferences of those votes as interpreted during the first count were published by the AEC, making it simple to add them back into the recount. This swings the crucial exclusion back to a margin of 1 vote, electing Louise Pratt and Zhenya Wang. It's impossible to know if these votes would have stood as they did if scrutinised again, and votes may also have come back from

lost informal votes.
The full senate vote distribution for these three scenarios can be viewed in detail. You can flick through the counts one-by-one, or the summary page for each scenario will link you to the count in which any given candidate was elected or excluded.
My software outputs JSON data descr bing the distribution process, and the user interface is an angularjs application which presents that JSON data. If anyone is interested in analysis or visualisation of the full distribution (as opposed to a simplified distribution only including above-the-line votes) the JSON files are linked for download. I would love to see some great animations appear from this data!
If you want to check out the code behind this, it's over on github. I have named the project dividebatur (Latin for 'distribution') and it is open source under the Apache license. If you're interested in the STV algorithm itself, it's implemented in these 700 lines of Python code (hopefully quite readable).
I learnt a bunch of surprising things implementing the distribution. A bundle of "x" papers distributed at a given transfer value translates into a "x" times the transfer value votes, rounded down. This means that occasionally votes will be "lost" to rounding. What I didn't realise is that those votes can also come back – if those papers are distributed in an exclusion, they may break by next preference in a way that avoids the rounding issue. Subtle details like this (and split group voting tickets, exclusion tie resolution, …) were tricky to get right!
In conclusion, we've now got some software which lets us as voters verify that the rather complex senate distribution is correct, and study hypothetical scenarios if we want to. It is easy to determine what would happen if a party changed their ticket vote in a particular way, or consider what would happen if a state-that-shall-not-be-named stuffed up their senate election. I hope you find it useful, and please send me feedback and/or patches!
@angrygoat on twitter or grahame@angrygoats.net.
Disclosure: I'm a member of The Greens (WA). I don't believe anything in this post is partisan, and I am speaking for myself. My software is open source and can be verified at your leisure :-)

[redacted] **| Senior Research Officer**
Strategic Research & Analysis Section | Roll Management Branch
Australian Electoral Commission
[redacted]

UNCLASSIFIED