ISSN 1745-6231

Voting matters

for the technical issues of STV

published by

The McDougall Trust

Issue 21 March 2006

About the McDougall Trust (reg. charity no. 212151)

The McDougall Trust is a charitable trust formed in 1948. The charity's purposes as stated in its governing scheme of 1959 are to advance knowledge of and encourage the study of and research into:

- political or economic science and functions of government and the services provided to the community by public and voluntary organisations; and
- methods of election of and the selection and government of representative organisations whether national, civic, commercial, industrial or social.

The Trust's work includes the maintenance and development of the Lakeman Library for Electoral Studies, a unique research resource, the production and publication of Representation: The Journal of Representative Democracy, and, of course, this publication **Voting matters**, that examines the technical issues of the single transferable vote and related electoral systems.
For further information on the Trust, please contact:

The Secretary,
McDougall Trust,
6 Chancel Street,
London SE1 0UX, UK.
Telephone: +44 (0)20 7620 1080
Facsimile: +44 (0)20 7928 1528
Email: admin@mcdougall.org.uk
Web: www.mcdougall.org.uk

For further information on this publication, please contact B A Wichmann, the Editor at the above address or by email at: `Brian.Wichmann@bcs.org.uk`

Editorial

The delay in producing this issue is due to the lack of material. An issue is produced when about 20 pages of articles are available.

There are 3 papers in this issue:

- Jeff O'Neill: *Fast Algorithms for Counting Ranked Ballots*.

  Many years ago, the speed of undertaking a computer count was an issue. Computers are now fast enough for this not to be a serious concern. This paper shows that comparatively modest changes in the way a program operates can make significant changes to the speed of counting.

- Brian Wichmann: *Changing the Irish STV Rules*.

  The Republic of Ireland has used STV since its independence, but used a counting rule in which the order of the ballot papers could potentially change the result, albeit rather infrequently. This paper considers a change to the Meek rules which is assessed by means of computer simulation.

- Franz Ombler: *Booklet position effects, and two new statistics to gauge voter understanding of the need to rank candidates in preferential elections*.

  The use of STV in New Zealand is a very welcome development. The New Zealand elections randomised the order in which candidates were listed in ballot papers for some elections, but not in an accompanying booklet given to all voters. This paper demonstrates effects of the booklet and proposes measures of voter understanding of the importance of ranking their chosen candidates.

We have an innovation with this issue which is actually some additional material under the heading **Internet Resources** on the McDougall web site. The additional material is in the form of links to papers or references that are being used in *Voting matters* contributions. Hypertext links are typically too long to handle easily by means of printing, and therefore present a problem in producing *Voting matters*. There is also an additional hazard with such links as they can be removed or their position changed. The web site should be able to record changes and record material that has been lost.

Lastly, a report on electronic voting produced by the Irish Commission should be available shortly on their web site at: `http://www.cev.ie/`.

TV voting

There is an increasing use of popular voting associated with TV programmes, which, unfortunately, does not include preferential voting. With a programme like BBC's *Big Read*, one wonders what the result would have been. For instance, if one could (somehow) arrange preferential voting in which is the voters had read the books in their list, how would *War and Peace* have compared with *Harry Potter*?

Fast Algorithms for Counting Ranked Ballots

Jeffrey C. O'Neill
jco8@cornell.edu

1 Introduction

This paper shows how some vote-counting methods can be implemented significantly faster by organizing ranked-ballot data into a tree rather than a list. I will begin by explaining how the tree data structure works and then apply it to Meek's method and Condorcet voting.

2 Tree-Packed Ballots

The most basic way of storing ballots is in a list. For example, suppose Alice, Bob, and Cindy are candidates and we have ten voters. The votes could be stored in a list, where each line corresponds to a ballot, and within each line, the candidates are listed in order of preference. I call this raw or unpacked ballot data, and an example is shown in Figure 1.1.

In this example, as is inevitable in any real election with ranked ballots, some voters will cast the exact same ballot. Instead, one could store only one copy of duplicate ballots along with the number of times the ballot occurred. I call this list-packed ballots. Figure 1.2 shows the same ballots from Figure 1.1 packed into a list.

Many vote-counting methods can use list-packed ballots instead of raw ballots and save computations. For example IRV, ERS97 STV, and Meek's method can all use list-packed ballots but Cambridge and Irish STV cannot. The reason Cambridge and Irish STV cannot is that the outcome is dependent on the order of the ballots, and order information is lost with list-packed ballots.

The ballots, however, can be packed even more densely into a tree, what I call tree-packed ballots. Figure 1.3 shows the same ballots packed into a tree. The root of the tree lists the total number of ballots, which is ten. From the root, branches go downward corresponding to the first-ranked candidates. The subsequent nodes list the number of times that candidate was ranked first on a ballot. Note that these three numbers add up to ten. The second level corresponds to the second-ranked candidates listed after the corresponding first-ranked candidates. Note that no candidate is ever ranked second after Bob. Further, note that four ballots have Alice first, but only two ballots list a candidate second after Alice. This is because two of the four voters who listed Alice first did not rank a candidate second.

For the three data structures, the size of the data struc-

Alice, Cindy
Cindy
Cindy, Alice
Bob
Bob
Alice
Cindy, Alice
Alice
Alice, Bob, Cindy
Bob

Figure 1.1: Raw ballots.

3	Bob
2	Cindy, Alice
2	Alice
1	Cindy
1	Alice, Cindy
1	Alice, Bob, Cindy

Figure 1.2: List-packed ballots.

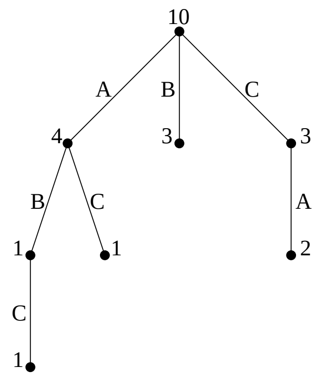

Figure 1.3: Tree-packed ballots.

Data Structure	Time
Load and No Packing	17.7
Load and List Pack	26.7
Load and Tree Pack	31.1

Table 1.1: Comparison of loading and packing times (in seconds).

Method	List	Tree
SNTV	0.6	
IRV	1.2	
ERS97 STV	5.5	
BC STV	4.7	
Meek STV	32.8	5.9 (18%)
Warren STV	30.8	3.0 (10%)
Condorcet	13.3	7.7 (59%)

Table 1.2: Timing of vote-counting methods with list-packed and tree-packed ballots (in seconds). The percentages in parenthesis indicate the computation time of the tree-packed implementation relative to the list-packed implementation.

ture corresponds to the number of entries, which is the number of times that candidate names are stored. For example, the size of the data structure in Figure 1.1 is 15, the size of the data structure in Figure 1.2 is 10, and the size of the data structure in Figure 1.3 is 7 (the root node isn't counted). Table 1.3 shows the sizes of the three data structures for the ballots from eight elections. B is the number of ballots, C is the number of candidates, and S is the number of seats to be filled.

List-packed ballots are 65% of the size of raw ballots. Tree-packed ballots are 45% of the size of list-packed ballots and 29% of the size of raw ballots. I expect the computation time of a particular implementation to be roughly proportional to the size of the data structure used. Thus, I expect the computation time with tree-packed ballots to be about 45% of the computation time with list-packed ballots. The more complicated data structures will also add some overhead that will increase the computation time to some extent.

Before presenting the details of implementing vote-counting methods with the different data structures, I will present the timing results with the different data structures. The timing results should only be considered in a rough sense since the efficiency of the particular implementations may vary. All timing results are cumulative for the above eight elections and are in seconds. First, the times in seconds for loading, loading and list packing, and loading and tree-packing are shown in Table 1.1.

Next I compare the computation times for a number of vote-counting methods using list-packed and tree-packed ballots. Because the relationship between raw and list-packed ballots is obvious, those times are not compared in this paper. [1] Further, only the slower methods are implemented with tree-packed ballots because these are the only ones that are in need of improvement. The methods are single non-transferable vote (SNTV), instant runoff voting (IRV), Electoral Reform Society STV (ERS97 STV), STV rules proposed for British Columbia in 2005 (BC STV), Meek STV, Warren STV, and Condorcet. [2] The computation times are shown in seconds in Table 1.2. The percentages in parentheses indicate the computation time of the tree-packed implementation relative to the list-packed implementation.

While we expected the computation times with tree-packed ballots to be 45% of the times for list-packed ballots, they are much faster for Meek and Warren STV. Why this is so will be explained below.

[1] Implementing a particular method with raw or list-packed ballots uses nearly the same code. The code iterates over the raw ballots or iterates over the list-packed ballots. The computation time is simply proportional to the number of loop iterations. In contrast, with tree-packed ballots, the code needs to be rewritten from scratch as is discussed below.

[2] The timing for Condorcet is only for computing the pairwise comparison matrix. Computing the Condorcet winner from the pairwise comparison matrix is generally much faster than computing the pairwise computation matrix.

3 Meek STV with Tree-Packed Ballots

I will now give the details of how to implement Meek STV using tree-packed ballots. The process is very similar for Warren STV. A full description of Meek STV is beyond the scope of this paper [1, 2, 3]. Instead, I will present the details most relevant to the fast implementation.

In each stage of counting votes with Meek STV, all the votes must be counted from scratch. This is distinct from other STV methods where some votes are simply transferred from one candidate to another and a full recount is not necessary at each round. With Meek STV, each candidate is assigned a fraction, f[c], where c denotes the candidate. At the beginning of the count, all the fractions are 1.0, and the fractions remain 1.0 as long as a candidate is under the quota. When a candidate has more than a quota, the fraction essentially discounts the value of that candidate's votes to bring the candidate back down to a quota. With a discount less than 1.0, the subsequently ranked candidates on a ballot will receive a portion of the vote.

In each round of a Meek STV count, the fractions f[c] will be updated and the ballots recounted. The following is a segment of Python pseudo-code for counting ballots for one round of a Meek count. Note that it uses list-packed ballots. The ith packed ballot is b.packed[i] and the corresponding weight of that packed ballot is b.weight[i].

```
# Iterate over all of the ballots.
for i in range(nBallots):
  # Each ballot is worth one vote.
  remainder = 1.0
  # Iterate over the candidates on this ballot.
  for c in b.packed[i]:
    # If the candidate is already eliminated
    # then skip to the next candidate on the
    # ballot.
    if c in losers:
      continue
    # This candidate gets a portion
    # of this ballot.  For the first non-losing
    # candidate on the ballot, the remainder will
    # be 1.0. If the candidate is under quota,
    # then f[c] is also 1.0 and this candidate
    # gets all of the ballot.  Otherwise the
    # candidate gets less than the full value,
    # and will share the ballot with
    # subsequently ranked candidates.
    count[c] += remainder * f[c] * b.weight[i]
    # Calculate how much of this ballot remains,
    # if any, to be counted for subsequently
    # ranked candidates.
    remainder *= 1 - f[c]
    # Stop if this ballot is used up.
    if remainder == 0:
      break
```

This code can be rewritten to use tree-packed ballots. The computations are exactly the same as before, they are just done in a different order so that similar computations can be done together. Consider the ten ballots presented above. Alice is ranked first on four ballots. With list-packed ballots, it would take three loop iterations to count these three ballots, but with tree-packed ballots all the first-place votes for Alice are counted at the same time, thus saving computations.

The code is more complicated, because it involves a depth-first traversal of the tree. The following shows how the nodes of the tree are accessed and also the order of a depth-first traversal.

```
tree[n] = 10
tree[Alice][n] = 4
tree[Alice][Bob][n] = 1
tree[Alice][Bob][Cindy][n] = 1
tree[Alice][Cindy][n] = 1
tree[Bob][n] = 3
tree[Cindy][n] = 3
tree[Cindy][Alice][n] = 2
```

A convenient way to implement the depth-first traversal is to use a recursive subroutine. Note that the subroutine calls itself by passing one branch of the tree, which is just a smaller tree, and possibly a diminished value for the remainder.

```
def updateCountMeek(tree, remainder):
  # Iterate over the next possible candidates.
  for c in tree.nextCands():
    # Copy the remainder for each iteration.
    rrr = remainder
    # Skip over losing candidates.
    if c not in losers:
      # Count the votes as before but weight with
      # the tree-packed data instead of the
      # list-packed data.
      count[c] += rrr * f[c] * tree[c][n]
      # Calculate how much of this ballot remains,
      # if any, to be counted for subsequently
      # ranked candidates.
      rrr *= 1 - f[c]
    # If there are any candidates ranked after
    # the current one and this ballot is not used
    # up, then recursively repeat this procedure.
    if tree[c].nextCands() != [] and rrr > 0:
      updateCountMeek(tree[c], rrr)
```

The initial call to the subroutine uses the base of the tree, and as before, the initial value of the remainder is 1.0

```
updateCountMeek(self.b.tree, 1.0)
```

Now that I have explained the fast algorithm, I can explain why it works much faster than expected. The unexpected speed increase arises from the fact that in any STV election, it is overwhelmingly the top choices on the ballots that are counted. In the first round of a Meek election, only the first-ranked candidates are counted. Consider the ballots for the Dublin North 2002 election. With list-packed ballots, one needs to count the 138,647 weighted ballots, but with tree-packed ballots, one needs to count only the twelve nodes of the tree corresponding to the first rankings of the twelve candidates. As the rounds progress, more and more nodes in the tree will be needed for the count, but generally this will be far less than the total number of nodes in the tree and even further less than the number of list-packed ballots.

Readers who understand the differences between Meek STV and Warren STV will immediately realize why Warren STV is much faster than Meek STV with the tree-packed ballots: Warren STV is less likely than Meek STV to use lower-ranked choices on a ballot.

4 Condorcet with Tree-Packed Ballots

Tree-packed ballots can also be used to compute the pairwise comparison matrix in a Condorcet election. The pairwise comparison matrix, pMat[c][d], counts the number of times that candidate c is ranked higher than candidate d on the ballots. Computing the pairwise comparison matrix is straightforward with list-packed ballots:

```
# Iterate over all the ballots.
for i in range(nBallots):
  # Copy the list of candidates.
  remainingC = candidates[:]
  # Iterate over the candidates the ballot.
  for c in b.packed[i]:
    # Get list of lower-ranked candidates.
    remainingC.remove(c)
    # Iterate over all lower-ranked candidates.
    for d in remainingC:
      # c is ranked higher than d.
      pMat[c][d] += b.weight[i]
```

This code can also be rewritten to use tree-packed ballots. As before it involves the depth-first traversal of the tree.

```
def ComputePMat(tree, remainingC):
  # remainingC is a list of candidates not higher in
  # the ballot than the current candidate.  Initially
  # it is a list of all the candidates.
  # Iterate over the next possible candidates.
  for c in tree.nextCands():
    # Copy the list of remaining candidates.
    rc = remainingC
    # Remove candidate from remaining list.
    rc.remove(c)
    for d in rc:
      # Current candidate is ranked higher than
      # candidates in remaining list.
      pMat[c][d] += tree[c][n]
    # Continue if more candidates.
    if tree[c].nextCands() != []:
      ComputePMat(tree[c], rc)

# First call is with entire tree and list of all
# candidates.
ComputePMat(tree, allCands)
```

Computing the pairwise comparison matrix is faster with tree-packed ballots, but the improvement is not nearly as great as for Meek STV. The reason for this is that computing the pairwise comparison matrix requires traversing the entire tree, thus the computation times are roughly proportional to the relative sizes of the data structures. The overhead involved with using tree-packed ballots makes the implementation with tree-packed ballots a little slower than expected.

5 Conclusions

Using tree-packed ballots instead of other data structures can greatly increase the speed of some vote-counting methods. Such speed improvements need to be weighed against the time needed to create the tree-packed ballots and the cost of maintaining more complex code. Meek and Warren STV are approximately five and ten times faster, respectively, with tree-packed ballots than with list-packed ballots. Such enormous speed improvements clearly outweigh the costs. In contrast, with Condorcet voting, the time saved is about equal to the time required for tree-packing the ballots so any benefits are minimal. Other methods, such as ERS97 STV and BC STV, are so fast with list-packed ballots that tree-packed ballots are clearly not beneficial.

My implementation of all of the vote counting methods mentioned in this paper (and others) is available for download at `http://stv.sourceforge.net`.

6 References

[1] Nicolaus Tideman, *The Single Transferable Vote*, Journal of Economic Perspectives, Vol. 9, No. 1, pp. 27-38 (1995).

[2] B. L. Meek, *A New Approach to the Single Transferable Vote*, Voting matters, Issue 1, p. 1 (Mar. 1994).

[3] I. D. Hill, B. A. Wichmann, and D. R. Woodall, *Algorithm 123 – Single Transferable Vote by Meek's Method*, Computer Journal, Vol. 30, No. 3, pp. 277-281 (1987).

Election	B/C/S	Raw	List	Tree
Dublin North 2002	43,942/12/4	218,933	138,647	57,568
Dublin West 2002	29,988/9/3	132,726	69,860	23,730
Meath 2002	64,081/14/5	298,106	174,737	74,105
Cambridge 1999 City Council	18,777/29/9	106,585	90,816	47,813
Cambridge 2001 City Council	17,126/28/9	95,440	79,385	40,566
Cambridge 2001 School Committee	16,489/16/6	66,254	33,860	12,907
Cambridge 2003 City Council	20,080/29/9	115,232	98,055	54,182
Cambridge 2003 School Committee	18,698/14/6	66,389	29,637	9,764
Total		1,099,665	714,997	320,635

B/C/S = Ballots/Candidates/Seats

Table 1.3: Sizes of the three data structures for the eight elections. The size of a data structure is the number of entries. See the text for more details.

Changing the Irish STV Rules

Brian Wichmann
brian.wichmann@bcs.org.uk

1 Introduction

For elections to the Dáil, the Irish Government has been using a form of STV which has remained essentially unchanged since the state was formed, in spite of small adjustments [1]. The counting rules have a significant flaw: they use a method of transferring surpluses that makes a random choice of the votes to be transferred [2]. Specifically, the rules require that the papers are placed in a random order. When a transfer is undertaken, all the relevant papers are examined in order to determine how many of them should be transferred to each continuing candidate, but the actual papers chosen for transfer depend on the random order. This method can affect the result if transferred papers are transferred again later in the count.

With the advent of computer-based counting (which is likely to be introduced shortly), the dependence upon the (random) order of the papers will become apparent. In the case of the three constituencies for which computer-based counting was used in 2002, the full ballot data was placed on the Internet (with the papers ordered as for the official count). In those three cases, the results were not order dependent, but order-dependence is bound to arise at some stage in the future. If a candidate could have been elected but was not, it is clear that a legal challenge to the result would be possible (especially if, considering all possible random orders of the papers, the aggrieved candidate was more likely to be elected than one of the candidates who actually was!).

This paper presents a study of the likely effect of changing the STV Rules for the Dáil to use the Meek method [3]. As with all modern counting rules, the Meek method has no order-dependence.

2 A method for simulating Irish voting patterns

For three Dáil elections held in 2002 we have the complete ballot data as noted above. This implies that many forms of analysis can be undertaken, for instance, the use of preferences as below:

Constituency	*Average used (Meek)*	*Average used (Irish)*	*Average given*	*Seats/ Candidates*
Dublin North	2.12	1.34	4.98	4/12
Dublin West	2.11	1.49	4.43	3/9
Meath	1.98	1.43	4.65	5/14

Here we use the data in another way. A previous paper [4] describes a way of generating simulated ballot data from a conventional STV result sheet using a simple statistical technique [5]. We wish to tailor this method to Irish voting patterns, which we can do by making the simulated ballot data more closely resemble the actual ballot data in the three Dáil elections for which the latter are known. To that end, the following changes have been made to the method described in [4]:

1. a proportion of the papers with only one or two preferences are ignored, since otherwise there would be too many such papers;
2. an appropriate proportion is added of strict party votes — all the preferences being for one party;
3. additional votes are added in which the final preferences are in ballot paper (or reverse) order because such are observed in the actual data. This is done by taking some of the generated papers which listed between a half and three quarters of the available candidates and inserting the remaining candidates;
4. for those candidates having a very small number of first preference votes, there is an adjustment to

ensure that the number of second preferences for them is also low.

The best possible outcome would be if the generated papers looked as if they came from the same population as the actual papers for the three constituencies. If fact, the results were as follows:

First preference test. This compares the distribution of first preferences for the actual and generated papers. The program construction should ensure that this test passes.

First two preferences test. Each pair of candidates is considered and also each candidate singly where no second preference is expressed. For the pairs the order of the two candidates is disregarded, counts for AB and BA being put together. The distributions formed from the actual and generated papers are then compared. It is not very surprising that this test fails because much of the necessary information about the relationships between candidates is missing in result sheets, and hence the generator's random selection will not produce a good fit. For Dublin North, for instance, the Labour and Green Party candidates appear to have a common following giving a high count to papers containing these as the first two choices. The result sheet for this election shows the high transfers at count 7 from the (elected) Green candidate to Labour, but does not show the reverse. In general, so many of the second preferences are unknown that the test cannot be expected to perform well.

Length test. This test considers the distribution of the number of preferences specified. Those that specify every candidate, and those that specify every candidate except one, are merged as their meanings are regarded as identical. This test is not passed, but does not fail so badly as to indicate a need to modify the program.

Rank test. This considers the ranking of the candidates against the ballot paper order. It passes with one of the three constituencies, and does not appear to warrant further program modification.

It is clear that the three available constituencies have different statistical properties, not all of which can be related to the differing numbers of seats (3, 4 and 5). Hence, the generator cannot be expected to obtain a good match for all of them. It is thought that any further change to the generator would be unlikely to make much improvement.

3 Generating data to match two Dáil elections

For each of the constituencies for the 1992 and 1997 Dáil elections, the result sheet is used, together with the generator described in the previous section, to produce three (related) sets, making 246 in total. The total number of candidates to be elected was 993. This ballot data could then be processed using the Irish rules and Meek. The observed differences were in 17 constituencies, 16 giving a difference of one candidate and one a difference of two. Hence the differences were in 1.8% of the candidates elected. (The difference in candidates was 18/993, while that in constituencies was 17/246, but the former is taken since that is the number which influences the Dáil.)

In all of the 17 constituencies, on completing the count with both rules, there was only one continuing candidate. In 13 of these, the set of those elected plus the continuing candidate was the same — the difference between the two rules was in the choice of the last candidate to elect.

We now need to consider ways of determining what should be the 'correct' result for these 17 cases. Two general methods are considered:

Order-dependence. We need to consider whether the Irish count was influenced in the final outcome by the order of the ballot papers. The papers were initially in random order and hence would not be expected to favour a specific candidate.

In theory, it should be possible to compute the probability of each possible outcome from the ballot papers. However, this seems rather difficult and hence the approach taken is to determine the two candidates whose position is different with the two rules. A program is then used to re-order the papers to favour the Meek outcome. Then the Irish rules are applied to the re-ordered papers to see if a different result is obtained. If a different result is produced, then it is clear that the papers *are* order-dependent, even if the probabilities of the different outcomes are not known. However, if the same result is produced, it is not possible to be sure that there is no order-dependence in the result, unless transferred surplus votes are not subsequently transferred again.

If the papers are order-dependent, then the Irish result is certainly questionable. In all such cases,

Test	*Seats*	*Withdrawn test*		*Order*
		Cands.	*Result*	*Depend.*
92/P19A	4	5	Meek	Yes
92/P22A	4	6	Irish	No?
92/P22B	4	6	Irish	No?
92/P23A	4	5	Meek	Yes
92/P24B	5	6	Meek	Yes
92/P24C	5	6	Irish	Yes
92/P26C	4	6	Meek	Yes
92/P27C	5	6	Meek	Yes
92/P35A	5	6	Meek	No?
92/P35B	5	6	Meek	Yes
92/P35C	5	7	Irish	Yes
92/P43A	4	5	Meek	No?
92/P43B	4	5	Meek	No?
97/P18C	3	4	Meek	No
97/P35B	3	4	Meek	No
97/P46B	4	5	Meek	No
97/P46C	4	5	Meek	No

Table 2.1: The differences analysed

reordering the papers can produce the Meek result.

Withdraw no-hopers. All the candidates who were neither elected nor a continuing candidate with either rule can be considered as having no hope of election. Under such circumstances, with STV, it is reasonable to assume that withdrawing these no-hopers from the count would not change the result. With the Meek rules, we know that this test *will* produce the same result, but the Irish result is uncertain. In the 17 cases under consideration, when running the Irish rules (with the papers in the same order), the result is either as with the original election, or else changes to the Meek result, as indicated in the Table 2.1.

In Table 2.1, the 6 cases in which the *withdrawn* test gives the Meek result and where there is also order-dependence, we regard as showing that the Meek result is superior. This leaves another 11 cases to consider in more depth.

The last four results in Table 2.1 are *not* order-dependent because the votes transferred after a surplus are not subsequently transferred. It is instructive to consider the first one of these further. The first stages of both Meek and the Irish rules are to exclude the five no-hopers. Hence, after these exclusions, the votes for the

Candidate	*Meek, Stage 6* / *Irish, Stage 5*	*Meek, Stage 7* / *Irish, Stage 6*	*Result*
C1	7241	7621	Elected
	7241	7317	
C3	7875	7614	Elected
	7875	7939	Elected
C5	7411	7592	
	7411	7472	Elected
C8	8316	7614	Elected
	8316	8111	Elected

Table 2.2: Test 97/P18C Analysis

(Meek results rounded to integers.)

remaining five candidates are the same for both rules. (The stages are out of step as the Irish rules exclude two in one stage, while Meek rules do not.) The *withdrawn* test shows that if the Irish rules were applied starting from this point, then the Meek result would have been produced. However, the two actual outcomes can be summarised in Table 2.2.

With the Irish rules, since the quota is calculated once at the start, C8 is elected with 639 (8111-7472) more votes than C5. The reduced quota with Meek means that many more of those people who voted first for C8 had a fraction of their vote transferred to their next preference. Moreover the 205 votes that were transferred from C8 all came from the excluded candidate C6. With Meek, all the votes for C8 are considered and an appropriate fraction retained while the rest of the votes are passed to the next preference. In our opinion, Meek can be seen to be fairer, although it requires more work to examine each vote at each stage.

All the other three cases for 1997 are similar.

We now consider the case 92/P24C in which the *withdrawn* test still produces the Irish result but we know that reordering the papers can produce the Meek result. Also, the *withdrawn* test is very simple in that only one candidate needs to be excluded. We give the result sheet for each rule in Tables 2.3 and 2.4. The elected candidates are in italics and underlined.

Comparing these two result sheets reveals the key differences as follows:

1. at the second stage, the Irish rules transfer the surplus of C2, while Meek transfers the surpluses of C1, C2 and C6. With the Irish rules, the surplus of C6 is never transferred;

C1	11156	11156	11156	−1463 9693
C2	16715	−7022 9693	9693	9693
C3	9076	+2668 11744	−2051 9693	9693
C4	6945	+1838 8783	+402 9185	+225 9410
C5	4532	+2516 7048	+1076 8124	+1238 9362
C6	9732	9732	9732	9732
Non-T	—	—	+573 573	573
Totals	58156	58156	58156	58156

Quota is 9693.

Table 2.3: Test 92/P24C, Irish rules

C1	11156	10692	9017
C2	16715	9732	9005
C3	9076	10832	9020
C4	6945	7983	8906
C5	4532	6142	9002
C6	9732	11121	9011
Non-T	—	1654	4195
Totals	58156	58156	58156
Quota	9693	9417	8993

Table 2.4: Test 92/P24C, Meek rules

2. the quota reduction of 700 votes with Meek is much larger than the difference of only 48 votes between the last two candidates (C4 and C5) under the Irish rules;

3. the number of non-transferable votes is very much larger with Meek. The reason for this is that all votes are treated the same way, while the Irish rules only transfer votes which have subsequent preferences specified (given that there are sufficient votes to do this). Some people might see this as a weakness of the Meek method, but for an opposing view, that it is a good feature of the method, see [6]— this point is considered further later.

C1	4126	+256 4382	+827 5209	+1047 6256	+243 6499
C2	4695	+191 4886	+167 5053	−5053 —	—
C3	6081	+1019 7100	+208 7308	+1120 8428	−693 7735
C4	9075	9075	−1340 7735	7735	7735
C5	5320	+172 5492	+138 5630	+820 6450	+170 6620
C6	9373	−1638 7735	7735	7735	7735
Non-T	—	—	—	+2066 2066	+280 2346
Totals	38670	38670	38670	38670	38670

Quota is 7735.

Table 2.5: Test 92/P22A, Irish rules

C1	4126	5084	5821	6997
C2	4695	5008	—	—
C3	6081	7129	7985	7070
C4	9075	7649	8178	7040
C5	5320	5587	6291	6790
C6	9373	7650	8207	7059
Non-T	—	563	2188	3714
Totals	38670	38670	38670	38670
Quota	7734	7621	7296	6991

Table 2.6: Test 92/P22A, Meek rules

With the possible exception of the issue of handling of non-transferable papers, the Meek result cannot be criticized, while the obvious imperfections in the Irish rules gives cause to doubt the result.

We now consider case 92/P22A (92/P22B is essentially the same). Again, for simplicity, we consider the *withdrawn* test rather than the full election. The two result sheets are presented in Tables 2.5 and 2.6.

It would be reasonable to ask why a further simplification could not be made by removing candidate C2, excluded by both rules. C2 is there as the continuing candidate with the Irish rules for the full election. Hence the candidate cannot be regarded as a no-hoper.

One can analyse the Irish results for evidence of order-dependence. The 191 and then 167 votes transferred to C2 are then transferred again and thus depend upon the choice of votes made. This total of 358 is greater than the 121 vote-difference between the last two candidates (C1 and C5). Hence the question mark remains: it might be possible to obtain the Meek result by a suitable re-ordering.

The number of non-transferable votes is high in both cases. Meek can compensate for this by reducing the quota, while with the Irish rules, an excessive number of papers remain with the three leading candidates. This excess amounts to about 2,000 votes, while the key difference is that C1 leads C5 by 207 votes with Meek, but by C5 leads C1 by 121 votes with the Irish rules.

Hence the primary source of the difference is the high number of non-transferable votes arising when C2 is excluded. The Meek logic is clearly superior in this case.

The three cases 92/P35A, 92/P43A and 92/P43B are all similar in having a weak order-dependence which cannot change the result by re-ordering the papers. However, in all these cases, the *withdrawn* test gives the Meek result. It is regrettable when the presence of a no-hope candidate changes an election result.

The last case, 92/P35C, is the most extreme since the closeness of the voting and the difference in the rules gives a difference of two seats. This is also exhibited by the election with the no-hopers removed, which is shown in Tables 2.7 and 2.8.

The order-dependence in this case arises from the 162 and 35 votes transferred to C3 which are subsequently transferred again and hence are subject to random sampling. However, an attempt to obtain a different result by changing the order failed (with the no-hopers removed), in spite of the original election being order-dependent (see Table 2.1).

The striking difference is that the Irish rules exclude C3 whom Meek rules eventually elect. However, the choice between C3 and C4 is close with both rules — 7 votes in favour of C3 for the Irish rules against 1 in favour of C4 with Meek. The quota reduction undertaken by Meek is enough to make the change, although this is again a consequence of the short lists logic.

C1	5407	+1264 6671	+269 6940	+1075 8015	+140 8155
C2	12008	−3158 8850	8850	8850	8850
C3	6304	+162 6466	+35 6501	−6501 —	—
C4	6290	+178 6468	+40 6508	+2558 9066	−216 8850
C5	7312	+159 7471	+33 7504	+613 8117	+76 8193
C6	9489	9489	−639 8850	8850	8850
C7	6288	+1395 7683	+262 7945	+934 8879	8879
Non-T	—	—	—	+1321 1321	1321
Totals	53098	53098	53098	53098	53098

Quota is 8850.

Table 2.7: Test 92/P35C, Irish rules

C1	5407	6846	7595	8041	8532
C2	12008	8796	9227	8756	8560
C3	6304	6497	8950	8678	8543
C4	6290	6496	—	—	—
C5	7312	7495	8131	8223	8324
C6	9489	8796	9307	8793	8569
C7	6288	7850	8458	8907	8577
Non-T	—	322	1430	1700	1993
Totals	53098	53098	53098	53098	53098
Quota	8850	8796	8611	8566	8517

Table 2.8: Test 92/P35C, Meek rules

4 Conclusions

It is possible to generate ballot data based upon Irish result sheets which is sufficiently similar to actual data to give a basis for comparing two counting rules. The analysis of the Irish rules shows that order-dependence is a significant problem, confirming the result in [2].

The Meek counting rule overcomes the order-dependence, as do all the modern counting rules (such as the Gregory rules used in Northern Ireland).

The analysis here shows that the property of Meek that the exclusion of no-hope candidates is the same as if those candidates had never entered the election is also important. Surely the intervention of such candidates should not influence the result? Other commonly used counting rules do not have this property.

The analysis also reveals that Meek usually has a much higher number of non-transferable papers than the Irish rules. It is the author's view that Meek is correct in this regard since every vote is handled in an identical fashion, while in the Irish rules (as with most of the hand-counting rules), the logic is dependent upon the other votes. This can easily have the effect of totally ignoring the wishes of those votes which gave few preferences in the sense that no transfer to non-transferables is undertaken. Whatever the reader might conclude on this point, this is a smaller effect than those arising from order-dependence and the influence of no-hope candidates noted above.

Although the difference in those elected is quite small (1.8% of the candidates elected), such a difference could be critical in the Dáil. The two major parties are frequently very nearly tied, so that the proportion of seats to them is critical in the formation of a Government. An actual counting error of 1.8% would be correctly regarded as quite unacceptable.

It might be maintained that the 'complexity' of using the Meek algorithm is not justified in view of the small differences observed in this analysis. However, in Ireland, when computers are being used, the complexity is not what it seems. An implementation of the Irish rules in Java amounts to around 2,000 lines of code [7], while the author's implementation of Meek in Ada is less than half that. There are a lot of exceptional cases in the Irish rules but virtually none in the Meek rules.

5 Acknowledgements

The paper is based upon a joint work with David Hill [8].

A significant fraction of this work would not have been possible without the ability to run a program of Joe Otten that implements the Irish rules [1].

6 References

[1] Electoral Act 1992 as amended by the Electoral (Amendment) Act 2001. Republic of Ireland.

[2] M. Gallagher and A. R. Unwin. Electoral Distortion under STV Random Sampling Procedures. *B.J.Pol.S.* Vol 16, pp243-268. 1986.

[3] I. D. Hill, B. A. Wichmann and D. R. Woodall. Algorithm 123 — Single Transferable Vote by Meek's method. *Computer Journal*. Vol 30, pp277-281, 1987.

[4] B. A. Wichmann. Producing plausible party election data, *Voting matters* Issue 5 pp. 6-10, January 1996.

[5] B. A. Wichmann. A simple model of voter behaviour, *Voting matters*. Issue 4. pp3-5. August 1995.

[6] I D Hill. Are non-transferables bad? *Voting matters*, Issue 8. p4. May 1997.

[7] Secrecy, Accuracy and Testing of the Chosen Electronic Voting System. Commission on Electronic Voting (Ireland). December 2004. (Appendix 2E gives details of the 'CD' implementation — the author has been told this is 2,000 lines.) See: http://www.cev.ie/

[8] David Hill and Brian Wichmann. STV in the Republic of Ireland. December 2003.

Booklet position effects, and two new statistics to gauge voter understanding of the need to rank candidates in preferential elections

Franz Ombler
franz@franzo.com

1 Introduction

In 2004 the Single Transferable Vote (STV) method replaced plurality for the election of members of New Zealand's District Health Boards (DHBs) [1]. While being unable to assess ballot position effects due to unrecorded random ordering of candidates' names on each ballot paper this article demonstrates effects that may be explained by the order of candidates' names in an accompanying booklet of the candidates' profiles. Such effects undermine the intended benefits from randomly ordering candidates' names on ballot papers, but prove useful in questioning voter understanding of the need to rank candidates. Two new statistics are proposed to better gauge voter understanding of a preferential voting method: the percentage of plurality style informal ballots and a rank indifferent percentage.

2 The elections

Two elections are considered: the Canterbury DHB election and the Otago DHB election. In both cases seven candidates were to be elected. Ballot papers were sent to voters by post. The ballots for the DHBs were printed with candidates' names randomly ordered such that each ballot paper might be unique. An accompanying booklet with candidates' profiles listed the candidates alphabetically [2]. It seems likely that few candidates for the elections were previously known to voters and the election would seem relatively non-partisan. Voters were allowed to rank order any number of candidates and a ballot was deemed informal if there was no 'unique first preference' indicated on the ballot [3].

2.1 Canterbury

The Canterbury DHB election was run alongside other territorial elections including those for the Christchurch City Council mayor, ward councillors and Canterbury Regional Council. These other elections continued to use plurality, so the voter had to contend with two methods in their ballot papers. There were 29 candidates. Of 117,852 non-blank ballots, 8,986 (7.6%) were deemed 'informal' and removed from the count. Of these, 7,579 (84.0% of informal votes, or 6.4% of total votes) marked all of the candidates for whom they voted as a first preference (either with a tick, or by writing '1'), presumably unaware of the need to rank candidates and thus voting as if it were a plurality election.

2.2 Otago

The Otago DHB election was run alongside territorial elections like those for Canterbury, but all elections were conducted using STV. There were 26 candidates. Of 65,389 non-blank ballots, 3,016 (4.6%) were deemed 'informal' and removed from the count. Of these, 1,315 (43.6% of informal votes, or 2.0% of total votes) marked candidates as if it were a plurality election.

As can be seen from the second-last row of Table 3.1, Canterbury DHB voters were over three times more likely to waste their vote by treating the election as a plurality election (6.4% versus 2.0%). This is probably because the Otago DHB election voters were more familiar with STV due to its use for all the elections on the Otago ballot papers. To better gauge voter understanding of preferential elections the percentage of plurality style informal ballots could be reported alongside the more usually reported total number of informal ballots.

	Canterbury	Otago
Number of seats	7	7
Candidates	29	26
Non-blank ballots	117,852	65,389
Formal ballots	108,866	62,373
Informal ballots	8,986 (7.6%)	3,016 (4.6%)
Informal ballots with multiple first preferences only (plurality-style)	7,579 (6.4%)	1,315 (2.0%)
Rank indifferent (see below)	5.1%	2.9%

Table 3.1: The Canterbury and Otago DHB elections

3 Ballot position effects

The voter burden of ordering the candidates is higher when the candidates are unfamiliar to voters, when there are so many candidates (29 for Canterbury, 26 for Otago), and where the district magnitude is high (seven) [4]. Furthermore, due to the lack of familiarity with candidates, position effects are probably greater [4], and these effects have greater consequences when voters are required to rank order candidates [5]. These effects may also be expected to be amplified by voters' lack of experience in rank ordering candidates, especially when they have to contend with multiple methods on their ballot papers as in the Canterbury election.

Candidates' names were randomly ordered during ballot paper printing, presumably to prevent ballot position effects, that is, where the positions of the candidates' names on the ballot affect voters' selection or ranking of the candidates. Randomising candidate name order should certainly have reduced the effect of 'donkey votes': ballots in which the voter ranked all the candidates in the order in which they appeared on the ballot. However, the number of donkey votes cannot be assessed due to the absence of information as to the order in which the candidates were listed on each ballot sheet. For the same reason, other ballot position effects cannot be assessed either.

4 'Booklet position effects'

Due to voters' lack of familiarity with the candidates many voters would have relied heavily on the booklet of candidates' profiles to draft their selections and rankings. The booklet listed the candidates alphabetically. We might call ensuing effects 'booklet position effects', which will dilute the intended benefits from randomly ordering the candidates' names on ballot papers; indeed it is interesting to consider (although not demonstrated here) whether booklet position effects may be greater than ballot position effects in elections in which voters are less familiar with the candidates. Certainly, the cost-effectiveness of randomising ballot paper candidate name order is questionable if the order of candidates' profiles in an accompanying booklet is not also randomised.

Assigning the candidates numbers according to their positions in the booklet (alphabetically) helps compare the rankings of candidates on each ballot with the order in which they appear in the booklet. The real ballot '2 10 14 17 19 24 26', where this voter has ranked candidate number 2 first (that is, they wrote the number one beside the candidate who appeared second in the booklet), candidate number 10 second and so on, may be described as perfectly ordered as it lists the candidates in the same order in which they appeared in the booklet. Similarly, '9 6 14 19 21 24 27' seems near perfectly ordered.

Spearman's rank correlation coefficient (r_s) may be used to assess the correlation of two rankings. We can apply this to each ballot, finding the r_s of the rankings of candidates in the ballot and the same ballot with candidates re-ordered alphabetically. For example, the r_s of the ballot '2 10 14 17 19 24 26' with its ordered self (the same ballot) is exactly 1.0, showing a perfect positive correlation; while r_s for '9 6 14 19 21 24 27' and its ordered self ('6 9 14 19 21 24 27') is 0.96.

The average r_s of each formal ballot's ranking of candidates with its ordered self is only 0.06 for Canterbury and 0.03 for Otago, showing such weak positive correlations that one might be tempted to infer an absence of booklet position effects. This is likely to draw criticism that it proves nothing due to 'failure to randomly assign groups of voters to different name orders' [4]. Indeed it would be consistent with this bare analysis to claim that position effects were present to a large degree and that if the booklets had been printed randomly that we would have seen a lower average r_s. This might be true to some extent but we are unable to assess it properly due to the absence of information about the order of names on each ballot; however, even without this information, booklet position effects can be demonstrated.

If we assess the frequency of the various values of r_s for the ballots, we find inordinately high numbers of perfectly ordered and near perfectly ordered ballots. Figure 3.1 (the data for which is presented in Table 3.2) shows such an analysis of the 51,730 ballots that listed exactly seven candidates in the Canterbury election. In light grey is the exact distribution of r_s for N=7, as would be approximated by randomly ordering these same ballots. Clearly there is a heavy tail on the right for the real ballots. Focussing on the rightmost bar, these 1,286 ballots (2.49%) are listed perfectly in order, but the expected number of ballots to be found in order for these 51,730 voters is only ten (0.02%) if preferences are randomly distributed.

Analyses of ballots listing other numbers of candidates (but more than 1) also find a notably higher than expected number of perfectly ordered ballots, 2,962 more than expected in total (see Table 3.3).

The Otago DHB election shows a similar but less prominent pattern (Figure 3.2). Given the similarity of the elections in other respects, this difference might be best explained by the use of STV in all of the elections on the Otago ballot papers and therefore greater voter awareness and understanding of the method.

Booklet position effects are apparent, but there are other potential explanations. It is conceivable that some voters are strongly biased towards candidates whose names start with letters nearer the beginning of the alphabet and admittedly booklet position effects cannot be distinguished from alphabetic effects in this election [4]. It is also possible that a group of candidates may actually be preferred in alphabetical order, perhaps by a small group of voters, perhaps following how-to-vote cards with candidates ordered alphabetically. However, as discussed above, the Canterbury voters would have been less aware of STV, they were more than three times more likely than Otago voters to vote as if the election were being run as a plurality election, and the charts show a greater percentage of perfectly ordered ballots for the Canterbury election. I contend that the charts' heavy tails primarily demonstrate ignorance of, or indifference towards, the ranking of candidates.

5 A measure of voter indifference to ranking

Where booklet or ballot position order can be assessed it may be worthwhile reporting a 'rank indifferent' statistic alongside the percentage of informal votes usually reported in elections. However, it isn't easy to say how many voters are rank indifferent.

Considering the Canterbury DHB election, it certainly seems reasonable to assert that most of the 1,286 voters who listed seven candidates in perfect order were rank indifferent: all but the ten expected, perhaps (refer Table 3.3). It would also seem true of the remaining 151 who listed more than seven candidates in perfect order, as the probability of this occurring is so low. It is less compelling to argue that 38 of the 2,526 voters who listed only two candidates in perfect order should also count, as the probability of this occurring by chance is so much greater. The appropriateness of this measure would then depend on some aspects of the election: if the number of candidates is low or if there are few candidates with popular support, sincere preferences are far more likely to happen to accord with ballot or booklet position and this may result in an inordinate number of perfectly ordered or near perfectly ordered ballots.

One way to avoid this problem is to count the higher than expected number of ordered ballots only when the probability of this occurring is extremely low, below 1% perhaps, which would only assess ballots listing five or more candidates. The Canterbury DHB election would then have a statistic of 2%. However, this seems conservative given the significantly more than expected number of near perfectly ordered ballots shown in the second-to-rightmost bar in Figure 3.1. Therefore one might also consider those ballots with an r_s, such that, say, less than 1% of ballots are to be expected to be found with this r_s or higher. The appropriate choice of r_s will then depend on the number of candidates in the ballot.

Taking this approach encapsulates the above in which we ignored ballots with less than five candidates, as with fewer than five candidates, there are fewer possible values of r_s and the probability of finding ordered ballots is greater than 1%. For example, where a ballot ranks only two candidates, there are only two possible arrangements resulting in an r_s (with its ordered self) of either 1 or -1, and with a probability of 50% either way. With three candidates there are only four possible values of $r_s : -1, -0.5, 0.5$ and 1, and the expected number of ballots having an r_s of 1 is one in six (16.7%) [6]. For four candidates, the expected number of ballots with an r_s of 1 is 4%. It is not until we reach five candidates that the expected number of ballots with an r_s of 1 drops below 1%. For six candidates, the expected number of ballots with an $r_s \geq 0.94$ (an r_s of either 0.94 or 1) is less than 1%, so we now count near perfectly ordered ballots as well as perfectly ordered ballots.

The appropriate values to use for r_s are thus the critical values to be found tabulated in textbooks. The expected number of ballots can be calculated from the probability of an r_s greater than or equal to the critical value: this might be assumed to be 1%, but it varies due to the discrete nature of r_s. Thus we also need to look up the probability of this value of r_s and calculate the number of ballots that may be expected to have this r_s if the ballots were randomly ordered. Critical values for the number of candidates in the ballot from 5 through 50 and the probabilities of finding these values are listed in Table 3.4.

Thus one can step through each ballot that ranks five or more candidates, correlating the ballot with its ordered self, and counting those that are 'highly ordered', that is, those with an r_s greater than the critical value for its number of candidates. One can then subtract the expected number of highly ordered ballots, which can be simply calculated by counting the number of ballots with each number of candidates and multiplying this by the probabilities listed in Table 3.4. Dividing this difference by the total number of formal ballots provides an accessible statistic. This statistic may be interpreted as the percentage of voters that were almost certainly rank indifferent. For the Canterbury DHB election this is 3.8% and for the Otago DHB election it is 1.9%.

However, the probability of a voter being rank indifferent can be expected to be unrelated to the length of the ballot even though we cannot identify rank indifference in shorter ballots with confidence. This seems reasonable when one considers that there is no reason to believe that voters who ranked fewer candidates might have had any greater understanding of STV than those who listed five or more candidates. Therefore, we should really divide the difference by the number of formal ballots that listed five or more candidates. For the Canterbury DHB election the rank indifferent statistic is then 5.1% and for the Otago DHB election it is 2.9% (see Table 3.5 for working).

6 Conclusions

Booklet position effects should be considered when assessing the cost-effectiveness of randomising the order of candidates' names on the ballot paper, especially if voters are unfamiliar with the candidates or if the need to rank candidates might be poorly understood.

Two new statistics may be reported to better gauge voter understanding of preferential voting: first, the percentage of plurality-style informal ballots, that is, ballots in which the voter marked all of the candidates (for whom they voted) with a tick or a '1'; and second, for elections where voters might be expected to rank order five or more candidates, the percentage of voters that were almost certainly rank indifferent. However, in interpreting the rank indifferent percentage one should be wary of other potential causes of perfectly ordered or near perfectly ordered ballots such as how-to-vote cards.

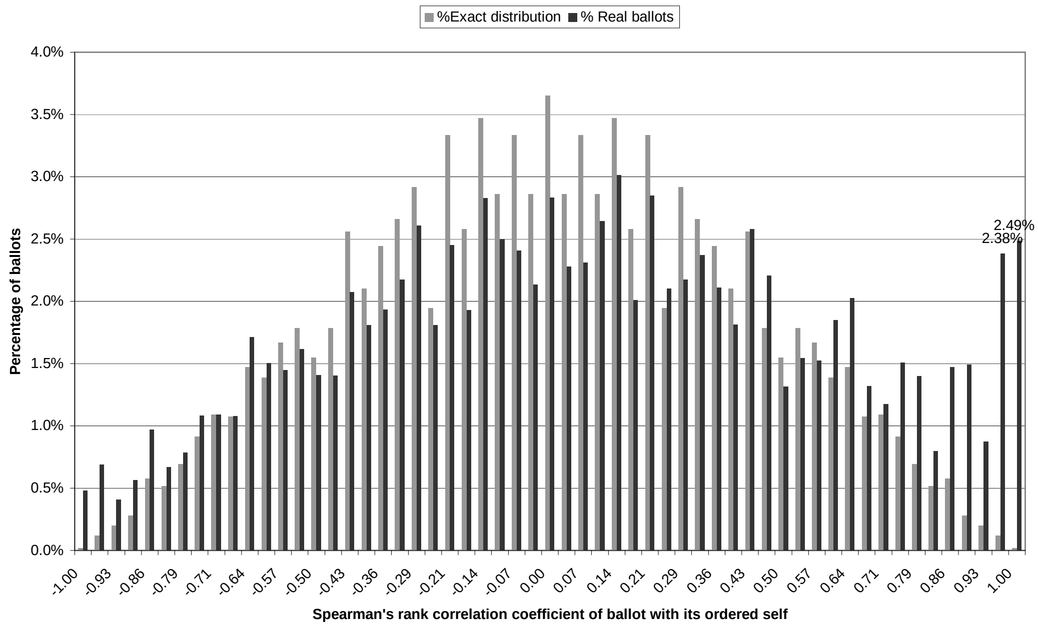

Figure 3.1: Canterbury DHB: frequency of ballots for Spearman rank-order correlation coefficients of voters' ballots with their ballots ordered alphabetically, for ballots listing seven candidates.

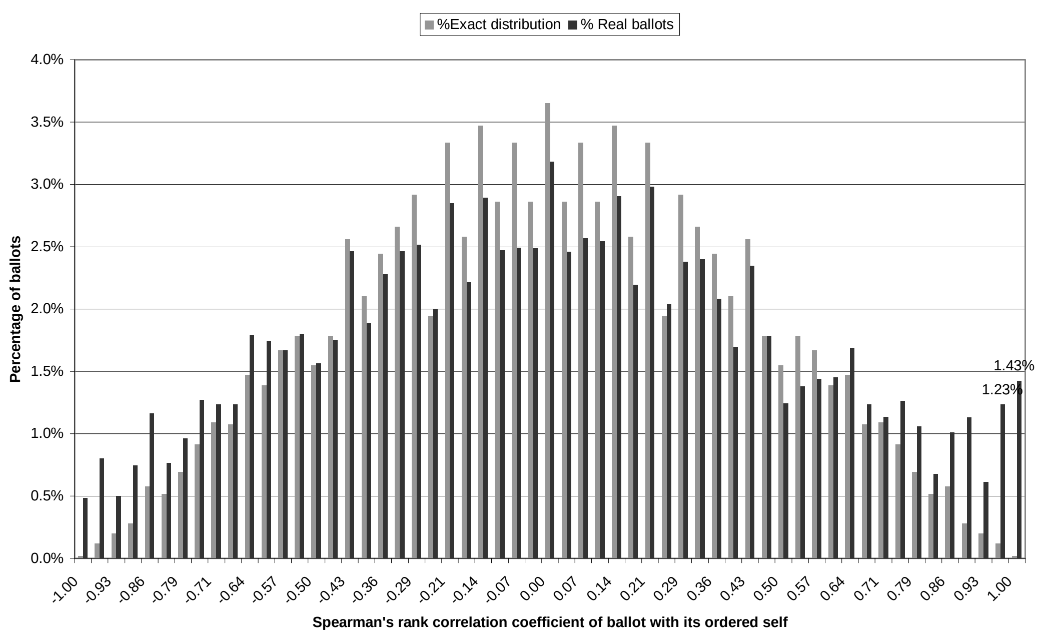

Figure 3.2: Otago DHB: frequency of ballots for Spearman rank-order correlation coefficients of voters' ballots with their ballots ordered alphabetically, for ballots listing seven candidates.

Canterbury DHB data				
Spearman's rank correlation coefficient	%Exact distribution	Real ballots	% Real ballots	Exact distribution
-1.00	0.02%	249	0.48%	10
-0.96	0.12%	357	0.69%	62
-0.93	0.20%	211	0.41%	103
-0.89	0.28%	292	0.56%	144
-0.86	0.58%	501	0.97%	298
-0.82	0.52%	346	0.67%	267
-0.79	0.69%	406	0.78%	359
-0.75	0.91%	559	1.08%	472
-0.71	1.09%	564	1.09%	565
-0.68	1.07%	557	1.08%	554
-0.64	1.47%	885	1.71%	760
-0.61	1.39%	778	1.50%	718
-0.57	1.67%	749	1.45%	862
-0.54	1.79%	836	1.62%	924
-0.50	1.55%	729	1.41%	801
-0.46	1.79%	726	1.40%	924
-0.43	2.56%	1072	2.07%	1324
-0.39	2.10%	935	1.81%	1088
-0.36	2.44%	1000	1.93%	1262
-0.32	2.66%	1123	2.17%	1375
-0.29	2.92%	1348	2.61%	1509
-0.25	1.94%	936	1.81%	1006
-0.21	3.33%	1268	2.45%	1724
-0.18	2.58%	999	1.93%	1334
-0.14	3.47%	1463	2.83%	1796
-0.11	2.86%	1293	2.50%	1478
-0.07	3.33%	1245	2.41%	1724
-0.04	2.86%	1103	2.13%	1478
0.00	3.65%	1465	2.83%	1889
0.04	2.86%	1179	2.28%	1478
0.07	3.33%	1196	2.31%	1724
0.11	2.86%	1367	2.64%	1478
0.14	3.47%	1559	3.01%	1796
0.18	2.58%	1041	2.01%	1334
0.21	3.33%	1473	2.85%	1724
0.25	1.94%	1086	2.10%	1006
0.29	2.92%	1124	2.17%	1509
0.32	2.66%	1227	2.37%	1375
0.36	2.44%	1090	2.11%	1262
0.39	2.10%	939	1.82%	1088
0.43	2.56%	1333	2.58%	1324
0.46	1.79%	1141	2.21%	924
0.50	1.55%	681	1.32%	801
0.54	1.79%	799	1.54%	924
0.57	1.67%	788	1.52%	862
0.61	1.39%	957	1.85%	718
0.64	1.47%	1048	2.03%	760
0.68	1.07%	682	1.32%	554
0.71	1.09%	606	1.17%	565
0.75	0.91%	779	1.51%	472
0.79	0.69%	724	1.40%	359
0.82	0.52%	412	0.80%	267
0.86	0.58%	761	1.47%	298
0.89	0.28%	771	1.49%	144
0.93	0.20%	453	0.88%	103
0.96	0.12%	1233	2.38%	62
1.00	0.02%	1286	2.49%	10
	100.00%	51730	100.00%	51730

Otago DHB Data				
Spearman's rank correlation coefficient	%Exact distribution	Real ballots	% Real ballots	Exact distribution
-1.00	0.02%	123	0.48%	5
-0.96	0.12%	204	0.80%	30
-0.93	0.20%	126	0.50%	50
-0.89	0.28%	189	0.74%	71
-0.86	0.58%	295	1.16%	146
-0.82	0.52%	194	0.76%	131
-0.79	0.69%	244	0.96%	176
-0.75	0.91%	322	1.27%	232
-0.71	1.09%	314	1.24%	277
-0.68	1.07%	313	1.23%	272
-0.64	1.47%	455	1.79%	373
-0.61	1.39%	443	1.74%	353
-0.57	1.67%	423	1.67%	423
-0.54	1.79%	457	1.80%	453
-0.50	1.55%	397	1.56%	393
-0.46	1.79%	445	1.75%	453
-0.43	2.56%	625	2.46%	650
-0.39	2.10%	479	1.89%	534
-0.36	2.44%	578	2.28%	620
-0.32	2.66%	625	2.46%	675
-0.29	2.92%	638	2.51%	741
-0.25	1.94%	507	2.00%	494
-0.21	3.33%	723	2.85%	846
-0.18	2.58%	562	2.21%	655
-0.14	3.47%	734	2.89%	882
-0.11	2.86%	627	2.47%	725
-0.07	3.33%	632	2.49%	846
-0.04	2.86%	631	2.49%	725
0.00	3.65%	808	3.18%	927
0.04	2.86%	624	2.46%	725
0.07	3.33%	652	2.57%	846
0.11	2.86%	645	2.54%	725
0.14	3.47%	737	2.90%	882
0.18	2.58%	557	2.19%	655
0.21	3.33%	757	2.98%	846
0.25	1.94%	518	2.04%	494
0.29	2.92%	604	2.38%	741
0.32	2.66%	609	2.40%	675
0.36	2.44%	528	2.08%	620
0.39	2.10%	431	1.70%	534
0.43	2.56%	595	2.34%	650
0.46	1.79%	453	1.78%	453
0.50	1.55%	315	1.24%	393
0.54	1.79%	350	1.38%	453
0.57	1.67%	365	1.44%	423
0.61	1.39%	369	1.45%	353
0.64	1.47%	429	1.69%	373
0.68	1.07%	313	1.23%	272
0.71	1.09%	288	1.13%	277
0.75	0.91%	321	1.26%	232
0.79	0.69%	269	1.06%	176
0.82	0.52%	172	0.68%	131
0.86	0.58%	257	1.01%	146
0.89	0.28%	287	1.13%	71
0.93	0.20%	156	0.61%	50
0.96	0.12%	313	1.23%	30
1.00	0.02%	362	1.43%	5
	100.00%	25389	100.00%	25389

Table 3.2: Data for Figures 3.1 and 3.2: the numbers of ballots for each possible value of r_s and the exact distribution (as would be approximated by randomly ordered ballots) for ballots ranking seven candidates [7]

Candidates in ballot (*n*)	Ballots (*b*)	Perfectly ordered (*p*)	Probability of being in order (1/*n*!)	Expected ballots in order (*b*/*n*!)	% found in order (*p*/*b*)	Number of times more than expected (*p*/(*b*/*n*!))
1	5691	5691	1.000000	5691	100.00%	1.00
2	4977	2526	0.500000	2489	50.75%	1.02
3	8483	1766	0.166667	1414	20.82%	1.25
4	8030	817	0.041667	335	10.17%	2.44
5	8639	514	0.008333	72	5.95%	7.14
6	5857	229	0.001389	8	3.91%	28.15
7	51730	1286	0.000198	10	2.49%	125.29
8	3331	55	0.000025	8.3E-02	1.65%	665.75
9	2224	39	2.8E-06	6.1E-03	1.75%	6363.45
10	2721	27	2.8E-07	7.5E-04	0.99%	36007.94
11	1107	12	2.5E-08	2.8E-05	1.08%	4.33E+05
12	1170	3	2.1E-09	2.4E-06	0.26%	1.23E+06
13	503	6	1.6E-10	8.1E-08	1.19%	7.43E+07
14	507	4	1.1E-11	5.8E-09	0.79%	6.88E+08
15	361	1	7.6E-13	2.8E-10	0.28%	3.62E+09
16	294	0	4.8E-14	1.4E-11	0.00%	0.00
17	166	2	2.8E-15	4.7E-13	1.20%	4.29E+12
18	131	0	1.6E-16	2.0E-14	0.00%	0.00
19	91	0	8.2E-18	7.5E-16	0.00%	0.00
20	112	0	4.1E-19	4.6E-17	0.00%	0.00
21	68	0	2.0E-20	1.3E-18	0.00%	0.00
22	50	0	8.9E-22	4.4E-20	0.00%	0.00
23	37	0	3.9E-23	1.4E-21	0.00%	0.00
24	47	0	1.6E-24	7.6E-23	0.00%	0.00
25	33	0	6.4E-26	2.1E-24	0.00%	0.00
26	49	0	2.5E-27	1.2E-25	0.00%	0.00
27	45	0	9.2E-29	4.1E-27	0.00%	0.00
28	47	0	3.3E-30	1.5E-28	0.00%	0.00
29	2365	2	1.1E-31	2.7E-28	0.08%	7.48E+27

Table 3.3: Perfectly ordered ballots in the Canterbury DHB election

Number of candidates selected on ballot	Minimum r_s	Probability of finding such a ballot
5	1.000	0.00833
6	0.943	0.00833
7	0.893	0.00615
8	0.833	0.00769
9	0.783	0.00861
10	0.745	0.00870
11	0.709	0.00910
12	0.678	0.00926
13	0.648	0.00971
14	0.626	0.00953
15	0.604	0.00973
16	0.582	0.00999
17	0.566	0.00983
18	0.550	0.00986
19	0.535	0.01
20	0.520	0.01
21	0.508	0.01
22	0.496	0.01
23	0.486	0.01
24	0.476	0.01
25	0.466	0.01
26	0.457	0.01
27	0.448	0.01
28	0.440	0.01
29	0.433	0.01
30	0.425	0.01
31	0.418	0.01
32	0.412	0.01
33	0.405	0.01
34	0.399	0.01
35	0.394	0.01
36	0.388	0.01
37	0.383	0.01
38	0.378	0.01
39	0.373	0.01
40	0.368	0.01
41	0.364	0.01
42	0.359	0.01
43	0.355	0.01
44	0.351	0.01
45	0.347	0.01
46	0.343	0.01
47	0.340	0.01
48	0.336	0.01
49	0.333	0.01
50	0.329	0.01

Table 3.4: Critical values and probabilities for r_s

[6, 7]

Canterbury DHB				
Candidates in ballot (*n*)	**Ballots**	**Expected highly ordered**	**Found highly ordered**	**Difference**
1	5691			
2	4977			
3	8483			
4	8030			
5	8639	72.0	514	442.0
6	5857	48.8	229	180.2
7	51730	318.1	2972	2653.9
8	3331	25.6	237	211.4
9	2224	19.1	169	149.9
10	2721	23.7	222	198.3
11	1107	10.1	81	70.9
12	1170	10.8	78	67.2
13	503	4.9	45	40.1
14	507	4.8	33	28.2
15	361	3.5	30	26.5
16	294	2.9	27	24.1
17	166	1.6	12	10.4
18	131	1.3	8	6.7
19	91	0.9	6	5.1
20	112	1.1	4	2.9
21	68	0.7	3	2.3
22	50	0.5	5	4.5
23	37	0.4	2	1.6
24	47	0.5	5	4.5
25	33	0.3	4	3.7
26	49	0.5	3	2.5
27	45	0.5	3	2.6
28	47	0.5	0	-0.5
29	2365	23.7	76	52.4
		576.8	4768	4191.2
Total	108866		Rank indifferent *n* >= 1	3.8%
Total *n* >= 5	81685		Rank indifferent *n* >= 5	**5.1%**

Otago DHB				
Candidates in ballot (*n*)	**Ballots**	**Expected highly ordered**	**Found highly ordered**	**Difference**
1	4323			
2	4196			
3	6047			
4	5515			
5	4573	38.1	157	118.9
6	4100	34.2	86	51.8
7	25389	156.1	831	674.9
8	1905	14.6	115	100.4
9	1112	9.6	62	52.4
10	1470	12.8	90	77.2
11	502	4.6	25	20.4
12	577	5.3	26	20.7
13	225	2.2	9	6.8
14	282	2.7	10	7.3
15	148	1.4	7	5.6
16	117	1.2	8	6.8
17	54	0.5	1	0.5
18	56	0.6	3	2.4
19	34	0.3	1	0.7
20	56	0.6	2	1.4
21	23	0.2	1	0.8
22	26	0.3	0	-0.3
23	11	0.1	1	0.9
24	21	0.2	2	1.8
25	81	0.8	1	0.2
26	1530	15.3	79	63.7
		301.7	1517	1215.3
Total	62373		Rank indifferent *n* >= 1	1.9%
Total *n* >= 5	42292		Rank indifferent *n* >= 5	**2.9%**

Table 3.5: Manual calculation of rank indifferent statistic

Further information and computer programs to automate the production of these statistics are available from the author on request.

7 References

[1] Department of Internal Affairs, 'STV Information', 2004, URL on web site.

[2] Personal communication with Christchurch City Council, 13 April 2005.

[3] New Zealand Government, 'Part 4 - Conduct of Elections and Polls using Single Transferable Voting Electoral System', *Local Electoral Amendment Regulations 2002.*

[4] Jon A. Krosnik, Joanne M. Miller, and Michael P. Tichy, 'An unrecognized need for ballot reform', in Ann N. Crigler, Marion R. Just, and Edward J. McCaffery (eds.), *Rethinking the Vote : The Politics and Prospects of American Election Reform* (New York: Oxford University Press, 2004), pp. 52, 53, 63.

[5] Susan Banducci, Michael Thrasher, Colin Rallings and Jeffrey A. Karp, 'Candidate appearance cues in low-information elections', 2003, Paper presented at the Annual Conference of the American Political Science Association (Philadelphia), URL on web site. p. 14.

[6] Sidney Siegel and N. John Castellan, Jr., *Nonparametric statistics for the behavioral sciences*, second edition (New York: McGraw-Hill, 1988), pp. 242, 360-361.

[7] Mark van de Wiel, Computer program `spearman.c` for calculating the 'Exact distribution of Spearman's rank statistic', available from URL on web site.