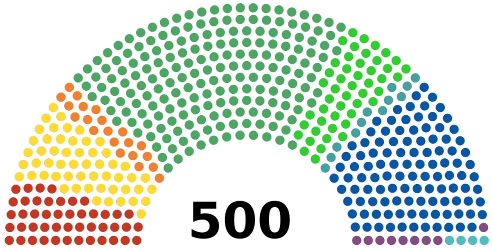
Proportional Algorithms: Apportionment

Piotr Skowron

University of Warsaw



Model: Apportionment

1. We have m political parties: $P_1, P_2, \ldots, P_m$.

Model: Apportionment

1. We have m political parties: $P_1, P_2, \ldots, P_m$.

2. We have n voters. Each voter votes for exactly one party.

Model: Apportionment

1. We have m political parties: $P_1, P_2, \ldots, P_m$.

2. We have n voters. Each voter votes for exactly one party.

   Let n_i denote the number of votes cast on party P_i

Model: Apportionment

1. We have m political parties: $P_1, P_2, \ldots, P_m$.

2. We have n voters. Each voter votes for exactly one party.

   Let n_i denote the number of votes cast on party P_i

   (of course, $\sum_{i=1}^{m} n_i = n$).

Model: Apportionment

1. We have m political parties: $P_1, P_2, \ldots, P_m$.

2. We have n voters. Each voter votes for exactly one party.

   Let n_i denote the number of votes cast on party P_i

   (of course, $\sum_{i=1}^{m} n_i = n$).

3. We have k parliamentary seats and we need to distribute them among the parties.

Model: Apportionment

1. We have m political parties: $P_1, P_2, \ldots, P_m$.

2. We have n voters. Each voter votes for exactly one party.

   Let n_i denote the number of votes cast on party P_i

   (of course, $\sum_{i=1}^{m} n_i = n$).

3. We have k parliamentary seats and we need to distribute them among the parties. (**In most cases we want to do it proportionally!**)

Apportionment: example applications

1. Parliamentary elections.

2. Distributing seats in European Parliament between countries based on their population.

3. Distributing the numbers of electoral votes between the states in the USA, based on their population.

Apportionment: two examples

number of seats: $k = 10$.

Example 1:

	Party 1	Party 2	Party 3	Party 4
#votes	10	20	20	50
#seats	?	?	?	?

Apportionment: two examples

number of seats: $k = 10$.

Example 1:

	Party 1	Party 2	Party 3	Party 4
#votes	10	20	20	50
#seats	1	2	2	5

✓

Apportionment: two examples

number of seats: $k = 10$.

Example 1:

	Party 1	Party 2	Party 3	Party 4
#votes	10	20	20	50
#seats	1	2	2	5

✓

Example 2:

	Party 1	Party 2	Party 3	Party 4
#votes	6	7	39	48
#seats	?	?	?	?

?

Apportionment: two examples

number of seats: $k = 10$.

Example 1:

	Party 1	Party 2	Party 3	Party 4
#votes	10	20	20	50
#seats	1	2	2	5

✓

Example 2:

	Party 1	Party 2	Party 3	Party 4
#votes	6	7	39	48
#seats	0.6	0.7	3.9	4.8

Not integral

Apportionment: two examples

number of seats: $k = 10$.

Example 1:

	Party 1	Party 2	Party 3	Party 4
#votes	10	20	20	50
#seats	1	2	2	5

✓

Example 2:

	Party 1	Party 2	Party 3	Party 4
#votes	6	7	39	48
#seats	0	1	4	5

?

Apportionment: two examples

number of seats: $k = 10$.

Example 1:

	Party 1	Party 2	Party 3	Party 4
#votes	10	20	20	50
#seats	1	2	2	5

✓

Example 2:

	Party 1	Party 2	Party 3	Party 4
#votes	6	7	39	48
#seats	1	1	4	4

?

Apportionment: two examples

number of seats: $k = 10$.

Example 1:

	Party 1	Party 2	Party 3	Party 4
#votes	10	20	20	50
#seats	1	2	2	5

✓

Example 2:

	Party 1	Party 2	Party 3	Party 4
#votes	6	7	39	48
#seats	0	0	4	6

?

Apportionment: two examples

number of seats: $k = 10$.

Example 1:

	Party 1	Party 2	Party 3	Party 4
#votes	10	20	20	50
#seats	1	2	2	5

✓

Example 2:

	Party 1	Party 2	Party 3	Party 4
#votes	6	7	39	48
#seats	0	0	4	6

?

Different apportionment methods will give different results!

Apportionment: two examples

number of seats: $k = 10$.

	Party 1	Party 2	Party 3	Party 4
#votes	6	7	39	48

Apportionment: two examples

number of seats: $k = 10$.

	Party 1	Party 2	Party 3	Party 4
#votes	6	7	39	48

lower quota: party P_i should at least $\left\lfloor k \cdot \frac{n_i}{n} \right\rfloor$ seats.

Apportionment: two examples

number of seats: $k = 10$.

	Party 1	Party 2	Party 3	Party 4
#votes	6	7	39	48
lower quota	0	0	3	4

lower quota: party P_i should at least $\left\lfloor k \cdot \frac{n_i}{n} \right\rfloor$ seats.

Apportionment: two examples

number of seats: $k = 10$.

	Party 1	Party 2	Party 3	Party 4
#votes	6	7	39	48
lower quota	0	0	3	4
upper quota	1	1	4	5

lower quota: party P_i should at least $\left\lfloor k \cdot \frac{n_i}{n} \right\rfloor$ seats.

upper quota: party P_i should at most $\left\lceil k \cdot \frac{n_i}{n} \right\rceil$ seats.

The largest remainder method

(aka the Hamilton method or the Hare-Niemeyer method)

1. First, assign to each party its lower quota.
2. Next, sort the parties by the remainders $k \cdot \frac{n_i}{n} - \left\lfloor k \cdot \frac{n_i}{n} \right\rfloor$ and assign the remaining seats to the parties with the heist remainders.

The largest remainder method

(aka the Hamilton method or the Hare-Niemeyer method)

1. First, assign to each party its lower quota.
2. Next, sort the parties by the remainders $k \cdot \frac{n_i}{n} - \left\lfloor k \cdot \frac{n_i}{n} \right\rfloor$ and assign the remaining seats to the parties with the heist remainders.

number of seats: $k = 10.$

	Party 1	Party 2	Party 3	Party 4
#votes	6	7	39	48
lower quota	0	0	3	4
remainder	0.6	0.7	0.9	0.8

The largest remainder method

(aka the Hamilton method or the Hare-Niemeyer method)

1. First, assign to each party its lower quota.
2. Next, sort the parties by the remainders $k \cdot \frac{n_i}{n} - \left\lfloor k \cdot \frac{n_i}{n} \right\rfloor$ and assign the remaining seats to the parties with the heist remainders.

number of seats: $k = 10.$

	Party 1	Party 2	Party 3	Party 4
#votes	6	7	39	48
lower quota	0	0	3	4
remainder	0.6	**0.7**	**0.9**	**0.8**
#seats	0	1	4	5

The largest remainder method

(aka the Hamilton method or the Hare-Niemeyer method)

1. First, assign to each party its lower quota.
2. Next, sort the parties by the remainders $k \cdot \frac{n_i}{n} - \left\lfloor k \cdot \frac{n_i}{n} \right\rfloor$ and assign the remaining seats to the parties with the heist remainders.

number of seats: $k = 10.$

	Party 1	Party 2	Party 3	Party 4
#votes	6	7	39	48
lower quota	0	0	3	4
remainder	0.6	**0.7**	**0.9**	**0.8**
#seats	0	1	4	5

The largest remainder method satisfies lower and upper quota.

House monotonicity and Alabama paradox

House monotonicity: if we increase the number of seats k then each party should get at least the same number of seats as before the increase.

House monotonicity and Alabama paradox

House monotonicity: if we increase the number of seats k then each party should get at least the same number of seats as before the increase.

Alabama paradox: the largest remainder method fails house monotonicity

	Party 1	Party 2	Party 3
#votes	6	6	2
value $k \cdot \frac{n_i}{n}$ for $k = 10$	4.286	4.286	1.429
#seats $k = 10$	4	4	2

House monotonicity and Alabama paradox

House monotonicity: if we increase the number of seats k then each party should get at least the same number of seats as before the increase.

Alabama paradox: the largest remainder method fails house monotonicity

	Party 1	Party 2	Party 3
#votes	6	6	2
value $k \cdot \frac{n_i}{n}$ for $k = 10$	4.286	4.286	1.429
#seats $k = 10$	4	4	2
value $k \cdot \frac{n_i}{n}$ for $k = 11$	4.714	4.714	1.571
#seats $k = 11$	5	5	1

House monotonicity and Alabama paradox

House monotonicity: if we increase the number of seats k then each party should get at least the same number of seats as before the increase.

Alabama paradox: the largest remainder method fails house monotonicity

	Party 1	Party 2	Party 3
#votes	6	6	2
value $k \cdot \frac{n_i}{n}$ for $k = 10$	4.286	4.286	1.429
#seats $k = 10$	4	4	**2**
value $k \cdot \frac{n_i}{n}$ for $k = 11$	4.714	4.714	1.571
#seats $k = 11$	5	5	**1**

Population monotonicity and yet another paradox

Population monotonicity: if there exists two parties, P_i and P_j, such that the number of votes for P_i increases with a higher rate than the number of votes for P_j (i.e., $\frac{n'_i}{n_i} > \frac{n'_j}{n_j}$, where n'_i and n'_j are the new numbers of votes for P_i and P_j, respectively), and if the number of seats assigned to P_j increases, then the number of seats assigned to P_i cannot decrease.

Population monotonicity and yet another paradox

Population monotonicity: if there exists two parties, P_i and P_j, such that the number of votes for P_i increases with a higher rate than the number of votes for P_j (i.e., $\frac{n_i'}{n_i} > \frac{n_j'}{n_j}$, where n_i' and n_j' are the new numbers of votes for P_i and P_j, respectively), and if the number of seats assigned to P_j increases, then the number of seats assigned to P_i cannot decrease.

Population paradox: the largest remainder method fails population monotonicity

Population monotonicity and yet another paradox

Population monotonicity: if there exists two parties, P_i and P_j, such that the number of votes for P_i increases with a higher rate than the number of votes for P_j (i.e., $\frac{n_i'}{n_i} > \frac{n_j'}{n_j}$, where n_i' and n_j' are the new numbers of votes for P_i and P_j, respectively), and if the number of seats assigned to P_j increases, then the number of seats assigned to P_i cannot decrease.

Population paradox: the largest remainder method fails population monotonicity

number of seats: $k = 22$

	Party 1	Party 2	Party 3	Party 4	Party 5
value $k \cdot \frac{n_i}{n}$	2.35	4.89	6.12	7.30	9.34
#seats	3	5	6	7	9

Population monotonicity and yet another paradox

Population monotonicity: if there exists two parties, P_i and P_j, such that the number of votes for P_i increases with a higher rate than the number of votes for P_j (i.e., $\frac{n_i'}{n_i} > \frac{n_j'}{n_j}$, where n_i' and n_j' are the new numbers of votes for P_i and P_j, respectively), and if the number of seats assigned to P_j increases, then the number of seats assigned to P_i cannot decrease.

Population paradox: the largest remainder method fails population monotonicity

number of seats: $k = 22$

	Party 1	Party 2	Party 3	Party 4	Party 5
value $k \cdot \frac{n_i}{n}$	2.35	4.89	6.12	7.30	9.34
#seats	3	5	6	7	9
value $k \cdot \frac{n_i}{n}$	2.4	4.77	6.12	7.30	9.41
#seats	2	5	6	7	10

Population monotonicity and yet another paradox

Population monotonicity: if there exists two parties, P_i and P_j, such that the number of votes for P_i increases with a higher rate than the number of votes for P_j (i.e., $\frac{n_i'}{n_i} > \frac{n_j'}{n_j}$, where n_i' and n_j' are the new numbers of votes for P_i and P_j, respectively), and if the number of seats assigned to P_j increases, then the number of seats assigned to P_i cannot decrease.

Population paradox: the largest remainder method fails population monotonicity

number of seats: $k = 22$

	Party 1	Party 2	Party 3	Party 4	Party 5
value $k \cdot \frac{n_i}{n}$	2.35	4.89	6.12	7.30	9.34
#seats	3	5	6	7	9
value $k \cdot \frac{n_i}{n}$	2.4	4.77	6.12	7.30	9.41
#seats	**2**	5	6	7	**10**

D'Hondt method

(aka the Jefferson method or the Hagenbach-Bischoff method)

In each iteration we assign one seat to one party. Let $s_i(r)$ denote the number of seats assigned to party P_i until iteration r. In iteration r we assign one seat to the party P_i which maximises $\frac{n_i}{s_i(r) + 1}$.

D'Hondt method

(aka the Jefferson method or the Hagenbach-Bischoff method)

In each iteration we assign one seat to one party. Let $s_i(r)$ denote the number of seats assigned to party P_i until iteration r. In iteration r we assign one seat to the party P_i which maximises $\frac{n_i}{s_i(r)+1}$.

number of seats: $k = 10$

	Party 1	Party 2	Party 3	Party 4
#votes	6	7	39	48

D'Hondt method
(aka the Jefferson method or the Hagenbach-Bischoff method)

In each iteration we assign one seat to one party. Let $s_i(r)$ denote the number of seats assigned to party P_i until iteration r. In iteration r we assign one seat to the party P_i which maximises $\frac{n_i}{s_i(r)+1}$.

number of seats: $k = 10$

	Party 1	Party 2	Party 3	Party 4
#votes	6	7	39	48
#votes/2	3	3.5	19.5	24
#votes/3	2	2.33	13	16
#votes/4	1.5	1.75	9.75	12
#votes/5	1.2	1.4	7.8	9.6
#votes/6	1	1.17	6.5	8.0
#votes/7	0.86	1	5.57	6.86

D'Hondt method

(aka the Jefferson method or the Hagenbach-Bischoff method)

In each iteration we assign one seat to one party. Let $s_i(r)$ denote the number of seats assigned to party P_i until iteration r. In iteration r we assign one seat to the party P_i which maximises $\frac{n_i}{s_i(r)+1}$.

number of seats: $k = 10$

	Party 1	Party 2	Party 3	Party 4
#votes	6	7	39	**48**
#votes/2	3	3.5	19.5	24
#votes/3	2	2.33	13	16
#votes/4	1.5	1.75	9.75	12
#votes/5	1.2	1.4	7.8	9.6
#votes/6	1	1.17	6.5	8.0
#votes/7	0.86	1	5.57	6.86

D'Hondt method
(aka the Jefferson method or the Hagenbach-Bischoff method)

In each iteration we assign one seat to one party. Let $s_i(r)$ denote the number of seats assigned to party P_i until iteration r. In iteration r we assign one seat to the party P_i which maximises $\frac{n_i}{s_i(r)+1}$.

number of seats: $k = 10$

	Party 1	Party 2	Party 3	Party 4
#votes	6	7	**39**	**48**
#votes/2	3	3.5	19.5	24
#votes/3	2	2.33	13	16
#votes/4	1.5	1.75	9.75	12
#votes/5	1.2	1.4	7.8	9.6
#votes/6	1	1.17	6.5	8.0
#votes/7	0.86	1	5.57	6.86

D'Hondt method

(aka the Jefferson method or the Hagenbach-Bischoff method)

In each iteration we assign one seat to one party. Let $s_i(r)$ denote the number of seats assigned to party P_i until iteration r. In iteration r we assign one seat to the party P_i which maximises $\frac{n_i}{s_i(r)+1}$.

number of seats: $k = 10$

	Party 1	Party 2	Party 3	Party 4
#votes	6	7	**39**	**48**
#votes/2	3	3.5	19.5	**24**
#votes/3	2	2.33	13	16
#votes/4	1.5	1.75	9.75	12
#votes/5	1.2	1.4	7.8	9.6
#votes/6	1	1.17	6.5	8.0
#votes/7	0.86	1	5.57	6.86

D'Hondt method

(aka the Jefferson method or the Hagenbach-Bischoff method)

In each iteration we assign one seat to one party. Let $s_i(r)$ denote the number of seats assigned to party P_i until iteration r. In iteration r we assign one seat to the party P_i which maximises $\frac{n_i}{s_i(r)+1}$.

number of seats: $k = 10$

	Party 1	Party 2	Party 3	Party 4
#votes	6	7	**39**	**48**
#votes/2	3	3.5	**19.5**	**24**
#votes/3	2	2.33	13	16
#votes/4	1.5	1.75	9.75	12
#votes/5	1.2	1.4	7.8	9.6
#votes/6	1	1.17	6.5	8.0
#votes/7	0.86	1	5.57	6.86

D'Hondt method

(aka the Jefferson method or the Hagenbach-Bischoff method)

In each iteration we assign one seat to one party. Let $s_i(r)$ denote the number of seats assigned to party P_i until iteration r. In iteration r we assign one seat to the party P_i which maximises $\frac{n_i}{s_i(r)+1}$.

number of seats: $k = 10$

	Party 1	Party 2	Party 3	Party 4
#votes	6	7	**39**	**48**
#votes/2	3	3.5	**19.5**	**24**
#votes/3	2	2.33	13	**16**
#votes/4	1.5	1.75	9.75	12
#votes/5	1.2	1.4	7.8	9.6
#votes/6	1	1.17	6.5	8.0
#votes/7	0.86	1	5.57	6.86

D'Hondt method

(aka the Jefferson method or the Hagenbach-Bischoff method)

In each iteration we assign one seat to one party. Let $s_i(r)$ denote the number of seats assigned to party P_i until iteration r. In iteration r we assign one seat to the party P_i which maximises $\frac{n_i}{s_i(r)+1}$.

number of seats: $k = 10$

	Party 1	Party 2	Party 3	Party 4
#votes	6	7	**39**	**48**
#votes/2	3	3.5	**19.5**	**24**
#votes/3	2	2.33	**13**	**16**
#votes/4	1.5	1.75	9.75	12
#votes/5	1.2	1.4	7.8	9.6
#votes/6	1	1.17	6.5	8.0
#votes/7	0.86	1	5.57	6.86

D'Hondt method

(aka the Jefferson method or the Hagenbach-Bischoff method)

In each iteration we assign one seat to one party. Let $s_i(r)$ denote the number of seats assigned to party P_i until iteration r. In iteration r we assign one seat to the party P_i which maximises $\frac{n_i}{s_i(r)+1}$.

number of seats: $k = 10$

	Party 1	Party 2	Party 3	Party 4
#votes	6	7	**39**	**48**
#votes/2	3	3.5	**19.5**	**24**
#votes/3	2	2.33	**13**	**16**
#votes/4	1.5	1.75	9.75	**12**
#votes/5	1.2	1.4	7.8	9.6
#votes/6	1	1.17	6.5	8.0
#votes/7	0.86	1	5.57	6.86

D'Hondt method

(aka the Jefferson method or the Hagenbach-Bischoff method)

In each iteration we assign one seat to one party. Let $s_i(r)$ denote the number of seats assigned to party P_i until iteration r. In iteration r we assign one seat to the party P_i which maximises $\frac{n_i}{s_i(r)+1}$.

number of seats: $k = 10$

	Party 1	Party 2	Party 3	Party 4
#votes	6	7	**39**	**48**
#votes/2	3	3.5	**19.5**	**24**
#votes/3	2	2.33	**13**	**16**
#votes/4	1.5	1.75	**9.75**	**12**
#votes/5	1.2	1.4	7.8	9.6
#votes/6	1	1.17	6.5	8.0
#votes/7	0.86	1	5.57	6.86

D'Hondt method

(aka the Jefferson method or the Hagenbach-Bischoff method)

In each iteration we assign one seat to one party. Let $s_i(r)$ denote the number of seats assigned to party P_i until iteration r. In iteration r we assign one seat to the party P_i which maximises $\frac{n_i}{s_i(r)+1}$.

number of seats: $k = 10$

	Party 1	Party 2	Party 3	Party 4
#votes	6	7	**39**	**48**
#votes/2	3	3.5	**19.5**	**24**
#votes/3	2	2.33	**13**	**16**
#votes/4	1.5	1.75	**9.75**	**12**
#votes/5	1.2	1.4	7.8	**9.6**
#votes/6	1	1.17	6.5	8.0
#votes/7	0.86	1	5.57	6.86

D'Hondt method

(aka the Jefferson method or the Hagenbach-Bischoff method)

In each iteration we assign one seat to one party. Let $s_i(r)$ denote the number of seats assigned to party P_i until iteration r. In iteration r we assign one seat to the party P_i which maximises $\frac{n_i}{s_i(r)+1}$.

number of seats: $k = 10$

	Party 1	Party 2	Party 3	Party 4
#votes	6	7	**39**	**48**
#votes/2	3	3.5	**19.5**	**24**
#votes/3	2	2.33	**13**	**16**
#votes/4	1.5	1.75	**9.75**	**12**
#votes/5	1.2	1.4	7.8	**9.6**
#votes/6	1	1.17	6.5	**8.0**
#votes/7	0.86	1	5.57	6.86

D'Hondt method

(aka the Jefferson method or the Hagenbach-Bischoff method)

In each iteration we assign one seat to one party. Let $s_i(r)$ denote the number of seats assigned to party P_i until iteration r. In iteration r we assign one seat to the party P_i which maximises $\frac{n_i}{s_i(r) + 1}$.

number of seats: $k = 10$

	Party 1	Party 2	Party 3	Party 4
#votes	6	7	**39**	**48**
#votes/2	3	3.5	**19.5**	**24**
#votes/3	2	2.33	**13**	**16**
#votes/4	1.5	1.75	**9.75**	**12**
#votes/5	1.2	1.4	7.8	**9.6**
#votes/6	1	1.17	6.5	**8.0**
#votes/7	0.86	1	5.57	6.86
#seats	**0**	**0**	**4**	**6**

D’Hondt method

(aka the Jefferson method or the Hagenbach-Bischoff method)

In each iteration we assign one seat to one party. Let $s_i(r)$ denote the number of seats assigned to party P_i until iteration r. In iteration r we assign one seat to the party P_i which maximises $\frac{n_i}{s_i(r)+1}$.

Fact: D’Hondt method satisfies lower quota.

D'Hondt method

(aka the Jefferson method or the Hagenbach-Bischoff method)

In each iteration we assign one seat to one party. Let $s_i(r)$ denote the number of seats assigned to party P_i until iteration r. In iteration r we assign one seat to the party P_i which maximises $\frac{n_i}{s_i(r)+1}$.

number of seats: $k = 10$

	Party 1	Party 2	Party 3	Party 4
#votes	6	7	**39**	**48**
#votes/2	3	3.5	**19.5**	**24**
#votes/3	2	2.33	**13**	**16**
#votes/4	1.5	1.75	**9.75**	**12**
#votes/5	1.2	1.4	7.8	**9.6**
#votes/6	1	1.17	6.5	**8.0**
#votes/7	0.86	1	5.57	6.86
#seats	**0**	**0**	**4**	**6**

D'Hondt method

(aka the Jefferson method or the Hagenbach-Bischoff method)

In each iteration we assign one seat to one party. Let $s_i(r)$ denote the number of seats assigned to party P_i until iteration r. In iteration r we assign one seat to the party P_i which maximises $\frac{n_i}{s_i(r)+1}$.

number of seats: $k = 10$

	Party 1	Party 2	Party 3	Party 4
#votes	6	7	**39**	**48**
#votes/2	3	3.5	**19.5**	**24**
#votes/3	2	2.33	**13**	**16**
#votes/4	1.5	1.75	**9.75**	**12**
#votes/5	1.2	1.4	7.8	**9.6**
#votes/6	1	1.17	6.5	**8.0**
#votes/7	0.86	1	5.57	6.86
#seats	**0**	**0**	**4**	**6**

D'Hondt method fails upper quota

D'Hondt method

(aka the Jefferson method or the Hagenbach-Bischoff method)

In each iteration we assign one seat to one party. Let $s_i(r)$ denote the number of seats assigned to party P_i until iteration r. In iteration r we assign one seat to the party P_i which maximises $\frac{n_i}{s_i(r)+1}$.

number of seats: $k = 10$

	Party 1	Party 2	Party 3	Party 4
#votes	6	7	**39**	**48**
#votes/2	3	3.5	**19.5**	**24**
#votes/3	2	2.33	**13**	**16**
#votes/4	1.5	1.75	**9.75**	**12**
#votes/5	1.2	1.4	7.8	**9.6**
#votes/6	1	1.17	6.5	**8.0**
#votes/7	0.86	1	5.57	6.86
#seats	0	0	4	**6**

D'Hondt method fails upper quota
It favours large parties.

Baliński and Young impossibility theorem (1983)

There exists no method of apportionment that satisfies population monotonicity, lower and upper quota.

Since the apportionment methods are only about rounding, what's the whole deal about?

Komitet wyborczy[36][37]		Głosy			Mandaty		
		Liczba	%	+/–	Liczba	+/–	%
	Prawo i Sprawiedliwość	8 051 935	**43,59**	▲ 6,01	**235**	–	51,09
	KKW Koalicja Obywatelska PO .N iPL Zieloni	5 060 355	**27,40**	▼ 4,29[a]	**134**	▼ 32[b]	29,13
	Sojusz Lewicy Demokratycznej	2 319 946	**12,56**	▲ 5,01[c]	**49**	▲ 49[d]	10,65
	Polskie Stronnictwo Ludowe	1 578 523	**8,55**	▲ 3,42	**30**	▲ 14	6,52
	Konfederacja Wolność i Niepodległość	1 256 953	**6,81**	—	**11**	—	2,39
	KWW Mniejszość Niemiecka	32 094	**0,17**	▼ 0,01	**1**	–	0,22

Since the apportionment methods are only about rounding, what's the whole deal about?

Komitet wyborczy[36][37]	Głosy			Mandaty		
	Liczba	%	+/–	Liczba	+/–	%
Prawo i Sprawiedliwość	8 051 935	**43,59**	▲ 6,01	**235**	–	51,09
KKW Koalicja Obywatelska PO .N iPL Zieloni	5 060 355	**27,40**	▼ 4,29[a]	**134**	▼ 32[b]	29,13
Sojusz Lewicy Demokratycznej	2 319 946	**12,56**	▲ 5,01[c]	**49**	▲ 49[d]	10,65
Polskie Stronnictwo Ludowe	1 578 523	**8,55**	▲ 3,42	**30**	▲ 14	6,52
Konfederacja Wolność i Niepodległość	1 256 953	**6,81**	—	**11**	—	2,39
KWW Mniejszość Niemiecka	32 094	**0,17**	▼ 0,01	**1**	–	0,22

Proportionality with respect to party affiliation and geographic district.

D'Hondt method used independently in districts

	P1	P2
votes	450	550
seats	4	6

$$k = 10$$

district 1

Proportionality with respect to party affiliation and geographic district.

D'Hondt method used independently in districts

	P1	P2
votes	450	550
seats	4	6

$k = 10$

district 1

...

	P1	P2
votes	450	550
seats	4	6

$k = 10$

district 10

Proportionality with respect to party affiliation and geographic district.

D’Hondt method used independently in districts

	P1	P2
votes	450	550
seats	4	6

$k = 10$

district 1

…

	P1	P2
votes	450	550
seats	4	6

$k = 10$

district 10

	P1	P2
seats	160	240

D’Hondt method used independently in districts

	P1	P2
votes	450	550
seats	4	6

$k = 10$

district 1

…

	P1	P2
votes	450	550
seats	4	6

$k = 10$

district 10

	P1	P2
seats	160	240
lower quota	180	220

Are we fated to disproportionality?

Are we fated to disproportionality?

NO!

Are we fated to disproportionality?

NO!

Bi-apportionment

Novel committee election methods

Are we fated to disproportionality?

NO!

Bi-apportionment

Novel committee Election methods

Bi-apportionment

Input:

1. A matrix $(v_{ij}) \in \mathbb{N}^{m \times d}$ where v_{ij} is the number of votes cast on pary i in district j
2. A vector $(h_j) \in \mathbb{N}^d$ with $\sum_{i=1}^{m} h_j = k$;

   here h_j is the number of seats we should assign to district j.

Bi-apportionment

Input:

1. A matrix $(v_{ij}) \in \mathbb{N}^{m \times d}$ where v_{ij} is the number of votes cast on pary i in district j
2. A vector $(h_j) \in \mathbb{N}^d$ with $\sum_{i=1}^{m} h_j = k$;

   here h_j is the number of seats we should assign to district j.

3. From (v_{ij}) we compute the vector $(s_i) \in \mathbb{N}^m$ where s_i is the number of seats that should be given to party i (we can compute that using an apportionment method).

Bi-apportionment

Input:

1. A matrix $(v_{ij}) \in \mathbb{N}^{m \times d}$ where v_{ij} is the number of votes cast on pary i in district j
2. A vector $(h_j) \in \mathbb{N}^d$ with $\sum_{i=1}^{m} h_j = k$;

   here h_j is the number of seats we should assign to district j.

3. From (v_{ij}) we compute the vector $(s_i) \in \mathbb{N}^m$ where s_i is the number of seats that should be given to party i (we can compute that using an apportionment method).

Output:

A matrix $(s_{ij}) \in \mathbb{N}^{m \times d}$ where s_{ij} is the number of seats given to party i in district j.

For each i it must hold that $\sum_{j=1}^{d} s_{ij} = s_i$ and for each j we must have $\sum_{i=1}^{m} s_{ij} = d_j$.

Bi-apportionment: a two step procedure.

Input: $(v_{ij}) \in \mathbb{N}^{m \times d}$, $(h_j) \in \mathbb{N}^d$, $(s_i) \in \mathbb{N}^m$.

Output: $(s_{ij}) \in \mathbb{N}^{m \times d}$.

The procedure:

1. First we find a possibly non-integral matrix $(f_{ij}) \in \mathbb{Q}_+^{m \times d}$ such that $\sum_{j=1}^{d} f_{ij} = s_i$ for each i and $\sum_{i=1}^{m} s_{ij} = d_j$ for each j.

2. Next, we round (f_{ij}) to obtain (s_{ij}).

Bi-apportionment: iterative proportional fitting

Input: $(v_{ij}) \in \mathbb{N}^{m\times d}$, $(h_j) \in \mathbb{N}^{d}$, $(s_i) \in \mathbb{N}^{m}$.

Intermediate step: $(f_{ij}) \in \mathbb{Q}_{+}^{m\times d}$.

Bi-apportionment: iterative proportional fitting

Input: $(v_{ij}) \in \mathbb{N}^{m \times d}$, $(h_j) \in \mathbb{N}^d$, $(s_i) \in \mathbb{N}^m$.

Intermediate step: $(f_{ij}) \in \mathbb{Q}_+^{m \times d}$.

Idea: rescale the matrix (v_{ij}) so that it satisfies constraints for rows and columns.

Bi-apportionment: iterative proportional fitting

Input: $(v_{ij}) \in \mathbb{N}^{m \times d}$, $(h_j) \in \mathbb{N}^d$, $(s_i) \in \mathbb{N}^m$.

Intermediate step: $(f_{ij}) \in \mathbb{Q}_+^{m \times d}$.

Idea: rescale the matrix (v_{ij}) so that it satisfies constraints for rows and columns.

Problem: it might not be possible to achieve that by rescaling the matrix by a single constant.

Bi-apportionment: iterative proportional fitting

Input: $(v_{ij}) \in \mathbb{N}^{m \times d}$, $(h_j) \in \mathbb{N}^d$, $(s_i) \in \mathbb{N}^m$.

Intermediate step: $(f_{ij}) \in \mathbb{Q}_+^{m \times d}$.

Idea: rescale the matrix (v_{ij}) so that it satisfies constraints for rows and columns.

Problem: it might not be possible to achieve that by rescaling the matrix by a single constant.

Solution: there exists a unique matrix of the form $f_{ij} = \lambda_i v_{ij} \gamma_j$. This matrix is called the fair share matrix and is characterised by the axioms of exactness, homogeneity, and uniformity.

Bi-apportionment: iterative proportional fitting

Input: $(v_{ij}) \in \mathbb{N}^{m \times d}$, $(h_j) \in \mathbb{N}^d$, $(s_i) \in \mathbb{N}^m$.

Intermediate step: $(f_{ij}) \in \mathbb{Q}_+^{m \times d}$.

Idea: rescale the matrix (v_{ij}) so that it satisfies constraints for rows and columns.

Problem: it might not be possible to achieve that by rescaling the matrix by a single constant.

Solution: there exists a unique matrix of the form $f_{ij} = \lambda_i v_{ij} \gamma_j$. This matrix is called the fair share matrix and is characterised by the axioms of exactness, homogeneity, and uniformity.

Iterative proportional fitting is an algorithm for computing the fair share matrix.

First rescale rows, then columns, then rows, etc.

Iterative proportional fitting: example

Iterative proportional fitting is an algorithm for computing the fair share matrix. First rescale rows, then columns, then rows, etc.

Converting (v_{ij}) to (f_{ij}).

					$\sum$	target (s_i)
	40	30	20	10	100	150
	35	50	100	75	260	300
	30	80	70	120	300	400
	20	30	40	50	140	150
$\sum$	125	190	230	255		
target (h_i)	200	300	400	100		

Iterative proportional fitting: example

Iterative proportional fitting is an algorithm for computing the fair share matrix.
First rescale rows, then columns, then rows, etc.

Converting (v_{ij}) to (f_{ij}).

					Σ	target (s_i)
	60.00	45.00	30.00	15.00	150.00	150
	40.38	57.69	115.38	86.54	300.00	300
	40.00	106.67	93.33	160.00	400.00	400
	21.43	32.14	42.86	53.57	150.00	150
Σ	161.81	241.50	281.58	315.11		
target (h_i)	200	300	400	100		

Iterative proportional fitting: example

Iterative proportional fitting is an algorithm for computing the fair share matrix. First rescale rows, then columns, then rows, etc.

Converting (v_{ij}) to (f_{ij}).

					Σ	target (s_i)
	74.16	55.90	42.46	4.76	177.44	150
	49.92	71.67	163.91	27.46	312.96	300
	49.44	132.50	132.59	50.78	365.31	400
	26.49	39.93	60.88	17.00	144.30	150
Σ	200.00	300.00	400.00	100.00		
target (h_i)	200	300	400	100		

Iterative proportional fitting: example

Iterative proportional fitting is an algorithm for computing the fair share matrix. First rescale rows, then columns, then rows, etc.

Converting (v_{ij}) to (f_{ij}).

and so on...

Iterative proportional fitting: example

Iterative proportional fitting is an algorithm for computing the fair share matrix. First rescale rows, then columns, then rows, etc.

After three iterations:

					Σ	target (s_i)
	64.61	46.28	35.42	3.83	150.13	150
	49.95	68.15	156.49	25.37	299.96	300
	56.70	144.40	145.06	53.76	399.92	400
	28.74	41.18	63.03	17.03	149.99	150
Σ	200.00	300.00	400.00	100.00		
target (h_i)	200	300	400	100		