1 Problem Statement

Suppose you want to have an election where people vote for parties rather than specific candidates. You also want a proportional voting system, but with a fixed number of seats per district. How do you "most fairly" proportion the seats? (For example, suppose you have 5 seats and 3 parties with 50% for party A, 30% for party B and 20% for party C. The optimal distribution would obviously be 2.5 for party A, 1.5 for party B, and 1 for party C, but this obviously cannot be done.)

2 Answer

There is no "actual" answer with how I phrased the question because "most fairly" is a term always up for debates. The most fair way would be to use the exact proportions, but that would require changing the number of seats available and so is usually a non-starter practically speaking.

The d'Hondt or Jefferson method is one way of dividing up the vote with a whole number of seats to each party. It essentially works by awarding a seat to the most popular party, then removing a portion of that vote from the total vote and reawarding a seat to the most popular party, etc. until all seats have been filled. The real question is how do we reapportion the vote after we have awarded a seat.

One way of doing so is to propose a rule

$$Q_p = \frac{v_p}{as_p + b} \tag{1}$$

where Q_p is the quotient for party p, v_p is the total number of votes for party p in the election, s_p is the number of seats allocated to party p already, a is a constant that weights subsequent rounds of voting and b weights the intial vote totals.[1] I will simply put $b = 1$ for our examples. We then assign a seat one-by-one to the party with the highest quotient.

Note that $a = 0$ is not proportional and is equivalent to a "first past the post" situation. We award all of the seats to the party that got the most votes, as that will translate to the largest quotient.

The $a = 1$ choice leads to the d'Hondt method. In this case, each time a party wins a seat you divide its total by one more than the previous number of seats won. We can then build a table where we divide by each number up to the number of seats available and we can then find the winners by finding the largest Q_p in the table. As an example consider Table 1. This simply lists all the possible quotients one could get in a 4 seat race with 4 parties. The actual proportions of the vote are about 37.7% for Party A, 34.6% for Party B, 15.1% for Party C, and 12.6% for party D. So the we'd prefer to give 1.5 seats to A, 1.4 seats to B, 0.6 seats to C, and 0.5 seats to D. Thus A and B are getting about half an extra seat than pure representation would give.

You might think this is unfair. You can then alter a. People generally like to keep whole numbers so that it is easy to create a table. When $a = 2$ you get the Webster-Saint-Laguë method. You now make the denominator go up by two each time. We can create the table of possibilities as we did for Table 1, but now with different divide by entries near the top. We see that this helps smaller parties at the expense of sort of mid-popularity parties.

[1] Usually $b = 1$ is chosen, though some choose $b > 1$ to tilt the seats given toward bigger parties.

	round 1	round 2	round 3	round 4	
p	1	2	3	4	p won
A	$\boxed{60}$	$\boxed{30}$	20	15	**2**
B	$\boxed{55}$	$\boxed{27.5}$	18.3	13.75	**1**
C	24	12	8	6	**1**
D	20	10	6.67	3.33	**0**
All	159	79.5	53	39.75	4

Table 1: We can without loss of generality put the parties in order from most votes (party A) to least votes (party D) initially. The numbers below the rounds are $s_p + 1$ (round number in this case, too), the number we divide the total vote count for the party by. Then the upper left entries that are largest tell us how many seats are allocated. Only parties A and B get any seats.

	round 1	round 2	round 3	round 4	
p	1	3	5	7	p won
A	$\boxed{60}$	$\boxed{20}$	12	8.57	**2**
B	$\boxed{55}$	18.3	11	7.857	**2**
C	$\boxed{24}$	8	4.8	3.43	**0**
D	20	6.67	4	2.857	**0**
All	159	79.5	39.75	19.875	4

Table 2: We can without loss of generality put the parties in order from most votes (party A) to least votes (party D) initially. The numbers below the rounds are $2s_p + 1$, the number we divide the total vote count for the party by. Then the upper left entries that are largest tell us how many seats are allocated. This time Party C gets a seat at the expense of Party B.

How one decides on a and b is completely determined by subjective preferences for what is fair, but we can at least come up with some metrics and say that if you care about this metric then you should choose this a and this b.

For example, if we care about the seat to vote proportionality, then it turns out that d'Hondt minimizes the amount of overrepresentation possible (the "advantage ratio" defined below). Here the seat share (number of seats divided by total seats) of party p is given by S_p and the vote share (number of votes divided by total votes) of party p is given by V_p. We'd like the advantage ratio

$$A_p = \frac{S_p}{V_p} \tag{2}$$

to be as close to one as possible. Note that

$$\sum_p S_p = 1 \tag{3}$$

$$\sum_p V_p = 1 \tag{4}$$

If we define the largest advantage ratio to be δ then

$$\delta = \max_p(A_p) \tag{5}$$

It can be shown[2] that the d'Hondt method minimizes δ to the smallest possible value given our restrictions on the number of seats. This proof uses $\delta \geq 1$ and that $S_p/\delta \leq V_P$ for all p. For then we find the "residual votes" (those beyond proportionality) by

$$T_p = V_p - S_p/\delta \tag{6}$$

if $T = \sum_p T_p = 1 - 1/\delta$ (remember the sums above) is the sum of this, we can use the normalized residual votes as

$$R_p = \frac{V_p}{T} - \frac{S_p}{T\delta} \tag{7}$$

or

$$V_p = \left(1 - \frac{1}{\delta}\right) R_p + \frac{S_p}{\delta} \tag{8}$$

One is then left to show that the d'Hondt method actually minimizes δ.

An especially easy way to see this is when we divde the seats for each party s_p into "proportional" s_{pP} and "residual" s_{pR} (let the total number of seats be s). Then if the total number of votes is v and the votes for the party are v_p we have

$$s_{pP} = \text{floor}\left(\frac{v_p}{v}\right) \tag{9}$$

[2]See Medzihorsky, Juraj. "Rethinking the D'Hondt method." Political Research Exchange 1, no. 1 (2019): 1625712.

The total number of "residual" seats s_R will be

$$s_R = s - \sum_p s_{pP} \tag{10}$$

if we then desire that

$$A_p = \frac{s_p}{s}\frac{v}{v_p} \tag{11}$$

to be as small as possible, we form $A_{p'} = \frac{s_{p'}}{v_p}$ where $s_{p'}$ is s_p plus the number of s_R assigned to p. We can do this initially with $s_{p'} = s_p$ and then take $s_R \to s_R - 1$ (until s_R goes to zero) and give the "residual" seat to the current party with the lowest $A_{p'}$. But this is exactly what we do with d'Hondt! This is because in d'Hondt we give to the highest quotient Q_p. That is when we test the new advantage A'_p we add 1 to each $s_{p'}$ and find the smallest. Q'_p is given by

$$Q_{p'} = \frac{v_p}{s_{p'} + 1} = \frac{1}{A_{p'}} \tag{12}$$

and so the largest $Q_{p'}$ corresponds to the smallest $A_{p'}$.

For a generic rule for each new seat awarded we give it to the one that maximizes

$$Q_{p'} = \frac{v_p}{as_{p'} + b} = \frac{1}{a}\frac{v_p}{s_{p'} + \frac{b}{a}} = \frac{1}{a}\frac{1}{A_{p'} - \frac{b-a}{av_p}} \tag{13}$$

and so we minimize the value of the modified advantage

$$A_{mp} \equiv A_p - \frac{b-a}{av_p} \tag{14}$$

Specifically for $b = 1$, and $a = 2$, we minimize

$$A_{mp} \equiv A_p + \frac{1}{2v_p} = \frac{s_p + \frac{1}{2}}{v_p} \tag{15}$$

This means that parties with the most seats are further penalized.

Some also use a different factor of b on the first column of the table only to prevent small parties from getting too many seats.

We see from this that we could summarize these methods as those that minimize

$$A_{mp} = \frac{s_p + c}{v_p} \tag{16}$$

which leads to any number of quotients all proportional to each other. If we always choose $a = 1$ then $c = 1 - b$ which means we maximize

$$Q_{mp} = \frac{v_p}{s_p + c} \tag{17}$$

Then $c > 0$ favors parties with larger vote totals and $c < 0$ favors parties with smaller vote totals.

As a final reiteration, the idea is that to have the smallest modified advantage ratio, you start from 0 seats given calculate what the modified advantage ratio would be if you awarded the seat to each possible party (the first column in any of the tables), and then award it to the one that has the smallest modified advantage ratio. You then do this again for the second seat, etc., etc. This will lead to the smallest modified advantage ratio overall, because if you award all the seats, if you hypothetically switched any of the seats, you would automatically increase the modified advantage ratio (by definition). And if you changed two of the seats you would also increase the modified advantage ratio, etc. We shortcut this by calculating the inverse of the modified advantage ratio and award to the maximum instead of the minimum with the modified quotients Q_{mp}.

One can then construct a program that takes in vote totals and yields the generalized d'Hondt winner. The code is listed below.

dhondt.py

```
#/usr/bin/env python3
import numpy as np


# This program implements the d'Hondt or Jefferson method
# for an election.  This is a way of fairly
# proportioning multiple seats to parties.
#
# It takes a list of values and creates a table
# dividing each value by 1, 2, 3, etc.
# vnum is the number of possible victors
# it outputs the table z of
# the winning values
# and the number of seats won by each party
# via an array corresponding to each entry in a
#################################################################*
# function hondt:
#       input:
#           a : a list of integers with the number
#                  of votes for each party in the
#                  district (not candidate)
#               Ex. [5,25,2] would be 5 votes for
#                  party 1, 25 votes for party 2,
#                  and 8 votes for party 3
#        vnum : the total number of candidates to be
#                  selected from the district
#               Ex. 2, then for [5,25,2], all would come
#                      from party 2 since it has
#                      more than double the vote total
#                      of any other party
#################################################################
def hondt(a,vnum,c=1):
  # prepare and create table
  pnum=len(a)
  z=np.zeros((pnum,vnum))
  for i in range(pnum):
    for j in range(vnum):
      # use the formula where for the table
      z[i,j]=float(a[i])/float(j+c)
  # resorts values from least to largest
  flat=z.flatten()
  flat.sort()
  # put winners at the front
  flat=flat[::-1]
  # take only the winning values
  flat=flat[:vnum]
  # counter of winners from each party
  counter=np.zeros(pnum)
  zz=z.copy()
  zznum=zz.shape
  # set random seed for testing
#  np.random.seed(1)
```

```
53    for i in range(len(flat)):
54      # find maximum values
55      d=np.argwhere(zz-flat[i]==np.amax(zz-flat[i]))
56      # create lists of the indices for these
57      # possibly tied locations
58      c=[]
59      e=[]
60      for j in range(d.shape[0]):
61        c.append(d[j][0])
62        e.append(d[j][1])
63      # rename c to adder, as one of the
64      # locations to add at
65      adder=c
66      cshape=len(c)
67      # randomly eliminate if tie in last place
68      while (cshape+counter.sum())>vnum:
69        elim=np.random.randint(0,cshape)
70        adder=c[:elim]+c[elim+1:]
71        cshape=len(adder)
72      # replace value in table so not recounted
73      zz[c,e]=0.
74      # add to victor counter
75      counter[adder]=counter[adder]+1
76    return z,flat,counter,a
77
78
79  aa,maxes,partyseats,vote=hondt([60,55,24,20],4)
80  print(aa)
81  print(maxes)
82  print(partyseats)
83  print(vote)
84  print('')
85  aa2,maxes,partyseats,vote=hondt([60,55,24,20],4,0.5)
86  print(aa2)
87  print(maxes)
88  print(partyseats)
89  print(vote)
90  print('')
```