Mini-project in TSRT04: Allocation of Seats in Elections

2022-03-10

1 Problem Formulation

In the elections to parliaments, the citizens vote for different parties that will represent them. When all the votes have been counted one can see how the votes have been distributed over the different parties. Since each parliament only has a limited number of seats (e.g., 349 in the Swedish national parliament and 20 Swedish seats in the European parliament), there is a need for a method to decide how many seats that should be allocated to each party. This known as "allocation of seats".

Sweden and many other countries have elections with proportional representation, which means that the percentage of seats that a party is allocated should be as similar as possible to the party's percentage of all votes.[1] Since the number of seats is much smaller than the number of votes, rounding errors are unavoidable. There many known methods to allocate seats and these are designed to give different types of rounding errors. Some methods give reward to parties that have a large percentage of all votes, while some methods will emphasize smaller parties.

In this mini-project, you should compare three different methods to allocate seats: D'Hondt method, Sainte-Laguë method, and modified Sainte-Laguë method. The goal is to write a function that, given a certain election result and number of seats, computes the allocation of seats with the different methods and plot the results along with the real distribution of votes. You will test your function on the latest Swedish election to the European parliament and also import results from another election.

1.1 Presentation

This is one of the mini-projects in the course TSRT04. To pass the course you need to solve one of the mini-projects according to the guidelines and present it to a teacher at one of the lab exercises that offers examination. The examination will be in English so please write your code and comments in English (a good exercise since English coding is common at companies). The problem should be solved in groups of two students, or individually. Since it is an examination assignment it is not allowed to share or show MATLAB code or notes to other students. It is, however, permitted to discuss the project orally with other groups; for example, to share some good advice!

- Write a MATLAB function `electionresults` that computes the allocation of seats using three different known methods. The function should also plot charts of the results and thereby illustrate how the rounding errors behave with the different methods. The function `electionresults` should call other functions, that you've written yourself, to solve certain subproblems.
- The solution should be demonstrated and the code should be showed to the teachers at Lab 3 or Lab 5. We will check if you have followed the guidelines, if you have a reasonable amount of comments in the code, if the plots are easy to understand, and if you can describe

[1] Not all electoral systems are proportional. USA and several other countries use the *winner takes it all* principle in each election district. The party that receives the largest number of votes in an election district receives all the seats that are dedicated for that district. There are some occasions, for example in Eurovision Song Contest, when Board count is used; that is, that the voters are ranking the candidates and the winner is the one with the highest overall ranking.

what the different parts of the code are doing. There is a style guide available on the course homepage, which elaborates on how a good solution should look like. Please make sure to read it and follow the recommendations!

- When the teachers have told you that you have passed the project, the code should be uploaded to the study room under Submissions. The examiner will then send the code to Urkund for control of plagiarism. The code should be saved in a text document called coursecode_year_studentid1_studentid2.txt (e.g., TSRT04_2022_helan11_halvan22.txt). This document should contain all functions and scripts that were examined, including a short example of how to call the function to solve the project.

2 Suggested Solution Outline

Begin by reading through the whole document so that you understand the problem specification and our suggested way to divide the problem into smaller subproblems.

The steps below provide one way to split the main programming problem into subproblems that can be solved one at a time. We have chosen the subproblems to make it possible to verify each part before you continue with the next part. Make sure to take these opportunities! Although the solution outline suggests that you write small functions that solve subproblem, this doesn't mean that it is good or efficient to use (call) all these small functions to solve the original problem. Some functions take care of such small pieces that it is better to copy the code into a new larger function, than to call the small function.

Note that this solution outline is intended for students with no or little previous programming experience. An experienced programmer would probably, based on previous experience, divide the problem into different and/or fewer parts. If you consider yourself an experienced programmer and think that you have a better approach to solve the problem, you are allowed to solve it in your own way. However, if you need help, the teaching assistant has more experience of the outlined solution. If you don't follow the solution outline you must at least make sure that the solution gives the same functionality.

2.1 Theory: Three methods to allocate seats in elections

You will test three methods to allocate seats. These methods are all sequential, which means that one seat is allocated at a time. This section describes how it works. A seat is given to the party that currently has the largest *quotient*. The quotient is initially equal to the number of votes that the party has received (or proportional to it). Each time a party gets a new seat, the party's quotient will be reduced so that the next seat is probably given to another party. The difference between different methods is how the reduction is computed.

We need some mathematical notation to describe the methods. The number of parties is denoted by n. The number of votes that party $1, 2, \ldots, n$ have received is denoted by $r_1, r_2, \ldots, r_n$. Let $m_1, m_2, \ldots, m_n$ be the number of seats that a party has received so far in the sequential method. These variables are initially equal to zero ($m_1 = m_2 = \ldots = m_n = 0$), but will increase as the seats are allocated.

When a seat is to be allocated, the quotients are computed as

$$z_i = \frac{r_i}{f(m_i)} \quad i = 1, 2, \ldots, n, \tag{1}$$

where the function $f(\cdot)$ differs between different methods. It is

$$f_{\text{hondt}}(m_i) = m_i + 1 \tag{2}$$

in *D'Hondt method*, which means that we will divide the number of votes with the number of seats that the party has received plus one. The function is

$$f_{\text{sainte}}(m_i) = 2m_i + 1 \tag{3}$$

in *Sainte-Laguë method*, so the division is by twice the number of seats plus one. Sweden uses the *modified Sainte-Laguë method* where the quotient for parties without seats is reduced by a factor 1.4 (instead of 1):

$$f_{\text{mod}}(m_i) = \begin{cases} 1.4, & \text{if } m_i = 0, \\ 2m_i + 1, & \text{if } m_i > 0. \end{cases} \tag{4}$$

The party with the largest quotient in (1) gets the next seat. If this is Party j then one should increase its seats: m_j is replaced by $m_j + 1$. The quotient is now updated as in (1) and then the next seat is allocated in the same way, to the party that now has the largest quotient. This sequential process continues until all seats have been allocated.

There is often also a "threshold" that requires a party to have a certain percentage of all votes, otherwise the party cannot get any seats. The threshold in Sweden is 4% in elections to the national parliament and European parliament, while the threshold is 3% for county councils. There are no thresholds for city councils in Sweden.

2.2 Visualisation of Election Results

Begin by writing a function that plots a chart that illustrates the election results. Let the argument of the function be a matrix with the number of votes. Each column represents a party and each row represents a certain election district (e.g., Linköping in the county of Östergötland). Each element is thus the number of votes for a given party in a given election district. Your function can, for example, present the election results as a pie or bar chart—preferably by using percentages.

Test your function on the Swedish results from the election to the European parliament in 2014. This data is available in the file `euelection2014.mat` on the course homepage. The columns represents the parties M, C, FP, KD, PP, MP, SD, FI, S, V and Others. Does you chart coincide with the final election results? You can find it at the homepage of Valmyndigheten:

`https://www.val.se/valresultat/europaparlamentet/2014/valresultat.html`

To make the results more readable, the function can have an argument that contains the names of the parties that should be used in the plots. It is possible to store text in matrices in MATLAB, but it is a bit tricky. It is better to write it using a cell structure:

```
>> parties = {'M','C','FP','KD','PP','MP','SD','FI','S','V','Other'};
```

When the party names are stored in this way, it is quite easy to use it in pie and bar charts, but the procedure is a bit different between the two charts. Check the documentation of each function.

2.3 Allocate a Seat With the D'Hondt Method

Write a function that allocates a seat with the simplest of the three methods: D'Hondt. The function takes the number of votes per party ($[r_1 \ \ldots \ r_n]$) and the number of seats that each party has received so far ($[m_1 \ \ldots \ m_n]$) as arguments. The function should return the index (a number from 1 to n) of the party with the largest quotient, according to (1) and (2).

Example:

- The voting distribution [6 19 5 10 4] and seat distribution [0 2 0 1 0] give the quotients [6 6.33 5 5 4]. Party 2 has the largest quotient and thus the function should return the index 2.

2.4 Allocate t Seats With the D'Hondt Method

Write a new function that allocates a certain number of seats, t, with D'Hondt method. The function takes the number of votes per party ($[r_1 \ \ldots \ r_n]$) and t as arguments. The function should return a vector with the number of seats that each party receives. You can either allocate the seats, one after the other, by calling the function from the previous step. Or you can create a new function that takes care of the whole procedure. Hint: What type of programming structure is ideal for sequential methods? Verify that the elements in the final vector sum up to t.

Example:

- The voting distribution [6 19 5 10 4] and $t = 10$ should give the seat distribution [1 5 1 2 1]. The quotient would be [3 3.17 2.5 3.33 2] if an eleventh seat is to be allocated. Verify that $1 + 5 + 1 + 2 + 1 = 10$, which equals the value of t.

2.5 Allocate Seats Also With the Sainte-Laguë Method and Modified Sainte-Laguë Method

The next step is to improve the function from the previous step to also take care of the Sainte-Laguë method and the modified Sainte-Laguë method). You can, for example, introduce a new argument that choose which of the methods that should be used. Then you only have to update the rows in the code where the quotients are computed, so that the formula depends on which method that has been chosen (see (1)–(4)).

Example:

- The voting distribution [6 19 5 10 4] and $t = 5$ should give different seat distributions and different quotients for the sixth seat. D'Hondt method should give the seat distribution [1 3 0 1 0] and quotients [3 4.75 5 5 4]. Sainte-Laguë method should give the seat distribution [1 2 1 1 0] and quotients [2 3.8 1.67 3.33 4]. Modified Sainte-Laguë method should give the seat distribution [1 3 0 1 0] and quotients [2 2.71 3.57 3.33 2.86].

2.6 Threshold for Small Parties

There is often a threshold that protects the method from small parties, which then need to achieve a certain percentage of the votes to be allowed to get any seats. Add this threshold as an argument wherever needed. Hint: A simple way to handle the threshold is to check which parties that are under the threshold and put their quotients to zero. Think about how you can verify that the threshold works as it is supposed to do.

2.7 Visualize Allocation of Seats

Extend the function that you wrote in Part 2.2 so that it solves the complete problem, as it was described in the beginning of the document (go back and read to Section 1 again!).

The suggested input arguments to your final function are the election results (`votes`), party names (`parties`), threshold (`minpercent`), and the number of seats to be allocated (`totalseats`). You can name the function in any way that describes its purpose for a potential user (it should not be called `problem2.7.m` or similar). One possibility is to name and define the function in the following way:

```
>> [seats,quotients] = electionresults(votes,parties,minpercent,totalseats)
```

where `seats` contains the final allocation of seats for each of the methods and `quotients` contains the final quotients.

Your final function should compute the seat allocation with the three methods and plot the result as charts. You should create one figure with bar charts and one figure with pie charts. For each

chart type, use subplots to demonstrate all the methods in a single figure. Do not forget to add the party names to the figures and to write descriptive names on the axis and by using `legend`. As always, the goal is that the figures should be easy to read and understand!

You can think about how the different methods handle large and small parties. You can base your observations on the results from the Swedish election to the European parliament. You can change the number of seats (which originally was 20) and/or remove the threshold (which originally is 4%). Another interesting question is: how many seats seem to be needed, if the rounding errors should be negligible?

2.8 Apply the Function on Another Election

Finally, you should import the results from some other election and apply your function. You can find such statistics at `http://www.val.se/valresultat/`

You can, for example, use results from earlier election to the European parliament, from your home city council, or from some other country. Check how many seats that exist or test the results for different number of seats. You should be prepared to show this to the teaching assistant when you present the project.

2.9 Preparations for the Presentation

To prepare for the project presentation, you need to read through the section "Presentation" in the beginning of this document and also read the Examination and Coding Guide that you find on the course homepage. At these places you will find that your need to have at least two functions in your code, that functions and variables need to have descriptive names, that there should be a reasonable amount of comments in the code, that all figures should be self-explaining, that comments should be in English, etc. If there is any requirements in the Examination and Coding Guide that you do *not* fulfill, then you need to take care of this *before* you present the project, otherwise you might have to wait a long to get a second chance to present.

Finally: Don't forget to submit the code to Urkund when you have passed to project assignment. You will find the instructions in the section "Presentation" in the beginning of this document.

3 Voluntary Extra Assignment

This section describes a voluntary extra assignment. If you are experienced programmers and feel that the project was too easy, we recommend that you also do some of these extra assignments, to learn MATLAB properly.

Coalitions in elections: The rounding errors in the seat allocation can sometimes have critical consequences. Two small parties can get 3% of the votes and therefore not reach the threshold. However, if the parties would have formed a coalition in the election and received 6% of the votes, then they may have received a seat.

Your extra assignment is to write a function that investigates all combinations of two parties that form a coalition. A coalition means that you sum up the parties' votes and treat them as a single party. Under the motto "together we're strong", you should determine which coalitions that receives more seats than the parties would have received separately.

A natural next step would be to look for coalitions with three or four parties that try to increase the joint number of seat. You can also write a function that looks for bad coalitions, where the parties would get fewer seats by collaboration. Finally, you should keep your eyes open for critical situations where a number of parties only would get a majority of the seats if they collaborate—or would lose their majority of the seats by forming a coalition.

Choose some interesting examples, preferably from reality, and plot these results in a suitable way.