Userguide for ALEX4 (simulation of electoral systems)

Marie-Edith Bissey, Vito Fragnelli and Guido Ortona

September 7, 2012

Contents

1 INTRODUCTION

The program is designed to compare electoral systems, on the basis of the voters' preferences for the parties and the candidates (simulated, or from surveys). The user can enter the parameters into the program (distribution of first preferences, size of the districts, etc.), or load them from a file. The program computes the resulting parliament, representativeness indexes and governability indexes according to the chosen electoral system. The program can implement uninominal and plurinominal systems.
The program has been developed using Java version 1.4, and tested on Microsoft Windows systems (2000 and XP). It is freeware, released under the GNU/GPL license.

2 INSTALLATION OF THE PROGRAM

The program is contained in the `ALEX4` directory. Unzip it (if necessary) and copy this directory on the computer. (*If the directory is copied from a CD, the files it contains may be 'read-only', which may prevent the program to function correctly. To remove this, select the directory, click on it with the right button of the mouse, select* `Properties` *and de-select* `read-only`*, applying the command on all files and subdirectories.*)
The program is written in Java, so at least Java Runtime needs to be installed on the computer for it to work. The oldest version of Java required for the program to run correctly is version 1.4.0. Java Runtime can be downloaded from the following URL: `http://java.sun.com`
In order to be able to launch the program from any directory, it is necessary to include the executable `javac` in the path. In some cases, you can choose it during the installation of Java. Otherwise you can follow the following steps (Windows 2000 and XP):

- double-click the System icon in the Control Panel
- click on Advanced, and then ENVIRONMENT VARIABLES
- select the PATH variable and click on Modify
- add the following to the existing string

  ```
  ;PATH_TO_JAVA/bin
  ```

  where `PATH_TO_JAVA` is the path to reach `javac` on the computer. For instance:

  ```
  c:\Programm Files\JRE\java\bin
  ```

3 USING THE PROGRAM

3.1 Starting the program

There are two ways to start the program:

- Double click on alex4.bat in the `ALEX4` directory
- From a MS/DOS window, use the `cd` command to go to the `ALEX4` directory and type: `java ALEX4` at the prompt (Remember: Java is case sensitive!).

This starts the program in the default way, which uses at most 64Mb of the computer's memory. For some simulations with many voters, parties and/or candidates, you may need more memory to run the program. In this case you can use the executables `alex4_128.bat` and `alex4_256.bat` which will use 128Mb and 256Mb of memory respectively.
Alternatively, you can type at the MSDos prompt: `java -Xmx128m ALEX4` to start Java with 128Mb of memory.

3.2 The desktop

When started, ALEX4 opens a window with several menus.

3.2.1 The File menu

This is the main menu of the program, from which the user can ask for new preferences to be simulated, or load voter's preferences, (for instance saved from a previous simulation, or computed from external sources, see page 34 for a description of how to create a preference files).
From this menu, the user can also save the parliaments, when they have been created.

3.2.2 The Edit menu

ALEX4 was developed with multilingual support. It is possible to change the language used by the program from the Edit menu. The user can also set the current language as the default. At the moment, the program can function in Italian, English and French. See page 33 for instructions on how to add a new language to the program.

3.2.3 The Simulations menu

When the program is first launched, this menu repeats the File menu, allowing to create or to load a simulation of voters' preferences. It is possible to open or create severals simulations at the same time, in order to confront the resulting parliaments.

3.3 Creating a parliament from simulated preferences

NOTE: the program is set up to use the characteristics of its current language. Therefore, if for instance the program's language is French or Italian, the user will have to use a comma for decimal numbers, whereas if the program's language is English, a dot will be required. This is independent from the actual language of the computer's operating system.

- From the File or the Simulations menu, choose Simulate new preferences
- The following parameters can be set in the first window:
  - number of voters in each uninominal district
  - number of uninominal districts
  - number of seats per plurinominal district (that is the size of plurinominal districts)
  - number of parties
  - probability to choose a first adjacent party (see section 4)
  - probability to choose a second adjacent party (see section 4)
  - probability to choose the preferred candidate (see section 4)

  To simulate plurinominal systems, the number of seats per plurinominal district *must* be greater than one. ***Remember to press RETURN to save each new parameter value in the program.*** Clicking OK, the user sees a confirmation window.

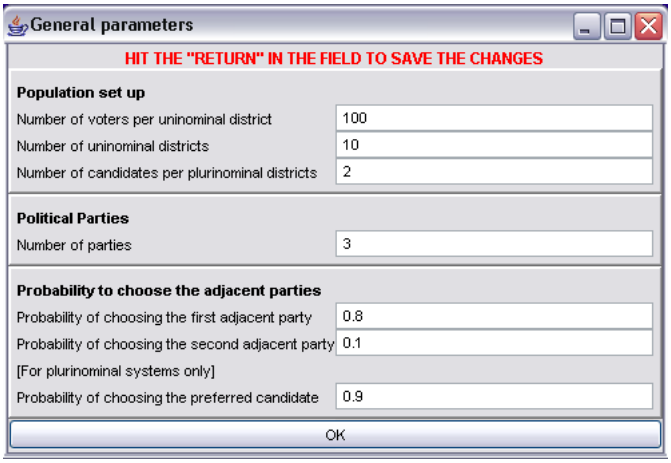

Figure 1: Setting up the number of voters, parties, districts and the probabilities

- The next window allows to set up the characteristics of the parties: their name, their share of votes in the population, their position on a left/right axis

and whether the party is "major", i.e. it is concentrated in some districts and with what weight (see section 4.3). Clicking OK, the user sees a confirmation window.

- The program starts creating for each voter his preference ordering for the parties. Then it creates the uninominal districts (by grouping voters), and the plurinominal districts (by grouping uninominal districts) if requested.

- Once the districts and the preferences are created, the user is asked to give a name to the simulation, so that s/he can identify it in the Simulations menu. Then the program opens a window with the list of the available electoral systems. To see the parliament resulting from a system, the user must click on the button next to its name. If the parameter "number of seats for plurinominal districts" is greater than 1, the plurinominal systems are displayed in the right hand column. If this parameter is equal to 1, only the left hand side column is displayed (uninominal systems).
  A window opens with the characteristics of the parliament: number of seats, graphical representation, representativeness indexes, and the possibility to create a government and compute the governability indexes. An example of a results window is shown in figure 7.
  Some systems need some extra parameters. A dialog window appears when the user clicks on the name of the system to create the parliament. To close the window, click on the X (upper right). From this window, and from the File menu, it is possible to save the characteristics of the parliament.
  There is no limit on the number of parliament windows that can be opened in the program, therefore it is possbile to open a window for each available parliament.

3.4 Loading a saved simulation

Simulations can be saved into files which can be reloaded into the program. The program saves all the general details of the simulation (described below), the voter's preference ordering for parties and eventually candidates, the composition of the uninominal districts and if present the composition of the plurinominal districts.
The simulation is saved in an ascii file. The format of the file is shown below on page 34.
The file can be edited in a text editor for changes.

3.5 Loading a saved parliament

It is also possible to save parliaments (i.e. the result of the application of a particular voting system on a population of voters and districts). The parliament is saved in an ascii file which contains the general parameters of the simulation (described below), the seats obtained by each party with the current voting system (`allocSeats_party`), the votes obtained by each party with the current voting

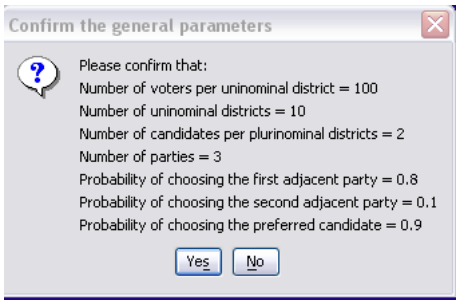

Figure 2: Confirming inserted values

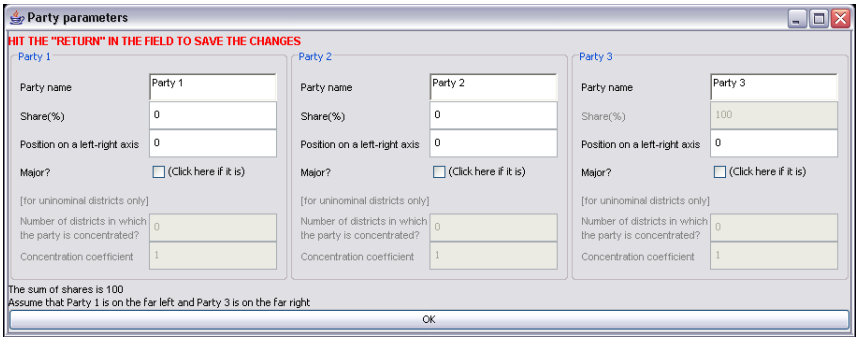

Figure 3: Setting up the characteristics of the parties

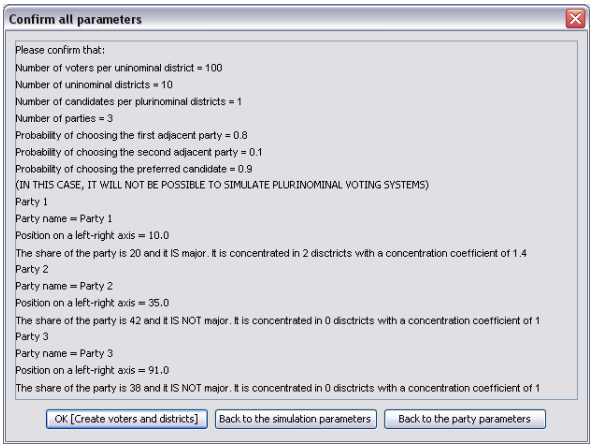

Figure 4: Confirming the parameter values inserted for the parties

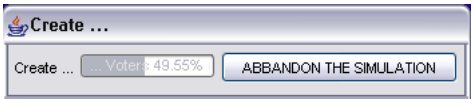

Figure 5: Starting creating the districts

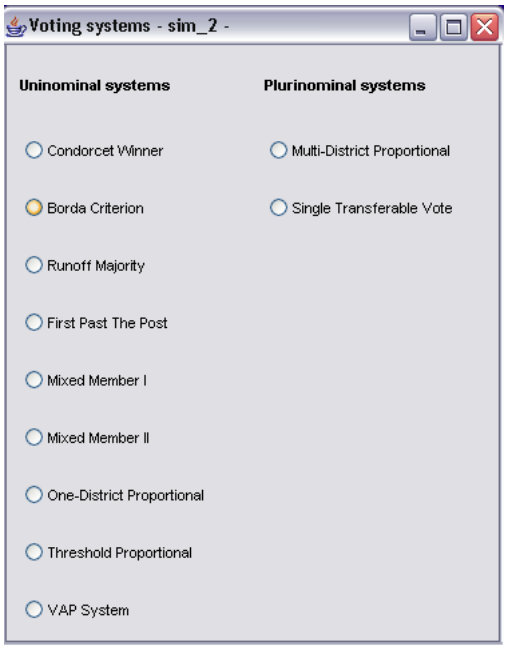

Figure 6: List of available parliaments

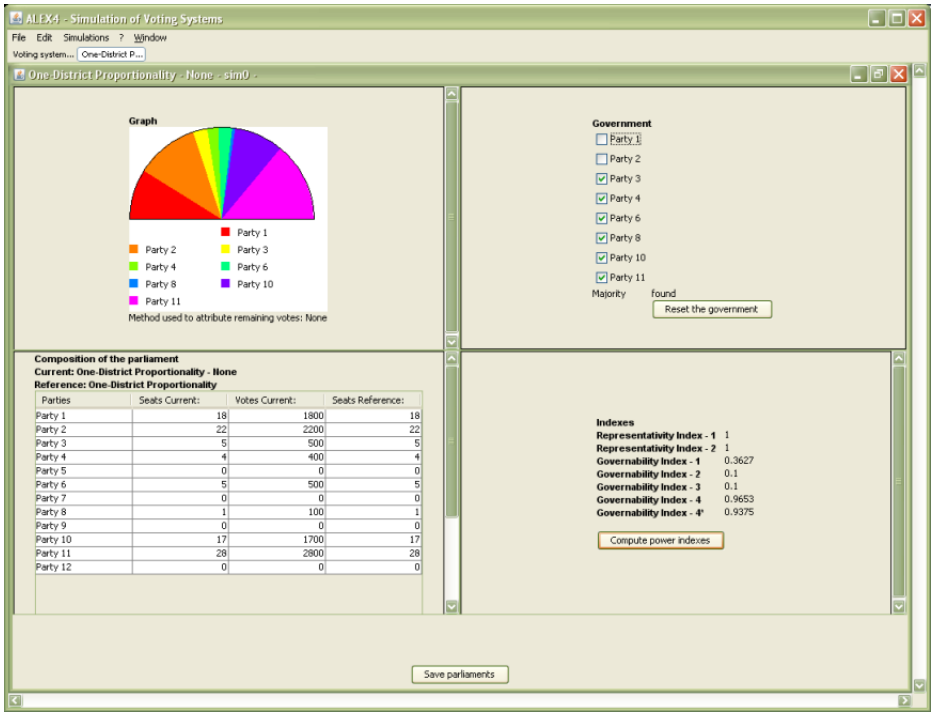

Figure 7: Example of results: the One-District Proportional system

system (`votes_party`), the seats obtained by each party with the "one-district proportional" system (`refAllocSeats_party`), the votes obtained by each party with the "one-district proportional" system (`refVotes_party`, which correspond to the first preferences of the voters in the population).
When a parliament is loaded into the program, the "parliament" window is displayed (figure 7 on page 8).
The program allows one to save more than one parliament in a given file (i.e. you may want to save together all the parliaments regarding a given simulation). If you load such a file in the program, one "parliament" window will be displayed for each parliament saved in the file.

4 DESCRIPTION OF THE PROGRAM

4.1 Voters' preferences for the parties - when the position on the left/right axis has not been specified

The program uses the following procedure to assign votes to parties: the first n_1 voters (corresponding to the share of Party 1 in the population) have Party 1 in the first place of their preference order, the next n_2 voters (corresponding to the share of Party 2 in the population) have Party 2 in the first place of their preference order, etc.
The parties are assumed to be ordered on a left/right scale, with Party 1 representing the far left, and party n representing the far right. The parties are saved in an array in increasing order of their index, independently of the name given by the user.
Once the first preference of the voter is known, his complete preference order for all parties is created. To do this, the program uses the probabilities defined in the first window: the probability to choose an adjacent party (or p_1), and the probability to choose a second adjacent party (or p_2). If these probabilities do not sum to 1, the remainder is the probability to choose another party at random.
To sum up, to choose the second party in the voter's preference order, the program follows these steps:

1. the first preferred party (or the one preferred at the moment) is deleted from the list of available parties and is saved in the first position in the voter's preference array.

2. to choose the second preferred party, the program chooses randomly with probability p_1 a party next to the one currently preferred, with probability p_2 a party second next to the one currently preferred, with probability $p_3 = 1 - p_1 - p_2$ another party at random.
   Practically, the choice of a party is done through giving each party a range of numbers (from 1 to 100). For instance, if the preferrend party is in the middle of the array of parties, the first party on the left of it is given the numbers from $S_1 = 50 - (p_1/2) * 100$ to 50. The first party on the right of it is given the

numbers from 51 to $D_1 = 50 + (p_1/2) * 100$. The second party on the left of the preferred party is given the numbers from $S_2 = S_1 - (p_2/2) * 100$ to S_1. The second party on the right of the preferred party is given the numbers from D_1 to $D_2 = D_1 + (p_2/2) * 100$. The remaining numbers are given to the other available parties. Then a number between 1 and 100 is chosen randomly. The party chosen to be second in the voter's preference order is the one corresponding to this random number.
This party become the party currently preferred.

For the rest of the preference order, the program uses the same method, using the party currently preferred as a starting point.
The method to choose the parties and attribute the numbers is adapted when the party currently preferred is at the ends of the array of parties.

4.2 Voters' preferences for the parties - when the position on the left/right axis has been specified

If the user has specified the position of the parties on the left/right axis, this information is used to create the voters' preferences for the parties. The procedure is as follows:

1. The first n_1 voters (corresponding to the share of Party 1 in the population) have Party 1 in the first place of their preference order, the next n_2 voters (corresponding to the share of Party 2 in the population) have Party 2 in the first place of their preference order, etc.

2. For each voter, the parties, excepting the voter's first preference, are ordered according to their distance on the left/right axis with respect to the first party. The voter's first preference becomes the "reference party".

3. With probability p_1 the next party in the voter's preference ordering is the closest to the reference party. With probability p_2 the next party is the second closest to the reference party. With probability $p_3 = 1 - p_1 - p_2$ the next party is one other party at random. If more than one party are at an equal distance to the reference party, we choose one of them at random.

4. The party just chosen becomes the reference party and the procedure iterates until the voter's preference ordering is complete.

4.3 Voters' preferences for the candidates

The procedure described in this section is employed for the simulation of the Single Transferable Vote system (see 4.6.14).

1. we assume that each party presents a number of candidates equal to "Number of seats per plurinominal district"; therefore each party has a list of (at

least 2) candidates. The candidates are inserted in the voters' preferences according to their position in the list: the candidate in first position is chosen before the candidate in second position, etc. When creating the voters' preferences for the candidates, the program takes into account their preferences for the parties, computed as described above.

2. the necessary parameters, computed using the probabilities defined in the first window of a simulation are as follows:

   - P_p: probability of choosing a *candidate* from the preferred party (defined by the user at the beginning of a simulation: "Probability of choosing the preferred candidate")
   - $(1-P_p)$: probability of choosing a *candidate* from a party which is not the preferred one
   - P_1: probability of choosing the *party* next to the preferred one (defined by the user at the beginning of a simulation: "probability to choose the first adjacent party")
   - P_2: probability of choosing the *party* in third place of the voter's preference order (defined by the user at the beginning of a simulation: "probability to choose the second adjacent party")
   - P_3: probability of choosing one of the other *parties* in the voter's preference order $(1 - P_1 - P_2)$
   - $(1-P_p).P_1$: probability to choose a *candidate* from the party in second position in the voter's preference order
   - $(1 - P_p).P_2$: probability to choose a *candidate* from the party in third position in the voter's preference order
   - $(1 - P_p).P_3$: probability to choose a *candidate* from another party in the voter's preference order

3. the parties are given number from 1 to 100 according to the following rules:

   - the preferred party is given numbers from 1 to $n_1 = 100.P_p$
   - the second preferred party is given numbers from $(n_1 + 1)$ to $n_2 = n_1 + 100.[(1 - P_p).P_1]$
   - the third preferred party is given numbers from $(n_2 + 1)$ to $n_3 = n_2 + 100.[(1 - P_p).P_2]$
   - the other parties are given numbers from $(n3 + 1)$ to 100

4. the voter's preferences for candidates are created extracting random numbers from 1 to $N = 100$ and choosing the candidates from the party corresponding to the chosen number, as explained below. To make the search faster, the extracted number is removed from the range. When all candidates from the party have been chosen, the numbers attributed to this party are removed

from the range.
For this system to work, we need to be sure that each range contains enough numbers to represent all candidates for the party. If the range is too small, all the ranges are recomputed so that the range becomes big enough, and the proportions of all ranges remain intact. At this point, the upper limit N of the range may become greater than 100.

5. the voter's first preference is the first candidate of the preferred party. This candidate is deleted from the list of candidates for the party. The second candidate in the party's list becomes 'first'.

6. a number between 1 and N is chosen at random: the chosen candidate is the one at the first position of the party associated with this number. The candidate is deleted from the list of available candidates for the party.
If the number extracted corresponds to the "other parties", the party to consider is randomly chosen among these "other parties".

7. the process is repeated until the voter's preference orders for the candidates is completed.

4.4 Creating uninominal districts

NOTE: in what follows, 'district' stands for 'uninominal district'.
The districts are created after the voters' preferences have been computed. Each district contains E voters, with E being equal to the value of the parameter "Number of voters per uninominal district" defined by the user at the beginning of a simulation (therefore, all the districts have the same size). In the program, it is possible to set up the concentration of some parties in some districts (for instance, think of districts in which a large majority of the voters vote for a given party, whereas few voters vote for this party in other districts). This feature is defined in the third window of a simulation, when the user sets up the characteristics of the parties. A party that is concentrated in one or more districts is called **major party**. When a party is said to be major, the user must declare the number of districts in which it is concentrated, and the weight of the concentration $T > 1$, which allows the program to compute how many voters with this party as first preference are assigned to the district. In other words, let us define:

- K the number of districts in which the party is concentrated
- F the total number of voters with this party as first preference
- N the size of the parliament (number of uninominal districts)

then each district in which the party is concentrated will contain $T * F/N$ voters with this party as first preference. T must be greater than 1 to make sure that the voters are *concentrated* in some districts (thereafter called *major districts*).

The creation of the uninominal districts is performed according to the following steps:

1. Separate the major parties from the 'normal' parties, creating 2 lists. First, the districts containing major parties are created. The districts containing normal parties are created in a second stage.

2. For each major party:

   (a) separate all voters with this party as first preference from the other voters, creating a new list, and deleting them from the list of voters

   (b) find out (from the characteristics of the party) the number of districts in which the party is concentrated

   (c) find out (from the characterisitics of the party) the concentration coefficient (variable `coefficient`)

   (d) find out the share of votes of the party in the population

   (e) find out how many voters have this party as first preference (variable `total`, size of the list created above)

   (f) find out how many voters are inserted in the 'major' districts:

   ```
   numberOfVotersToInsert=
      coefficient*total/numberOfUninominalDistricts
   ```

   (g) create the major districts for the party, inserting the number of voters computed above

   (h) if after this procedure some voters with the party as first preference have not been inserted in a district, they are put back into the main list of voters.

   When all major districts have been created, if some of them are not full (the number of voters inserted in the major district is smaller than the size of the district), they are completed with voters chosen randomly in the main list of voters.

3. For each normal district: the required number of voters is chosen at random from the main list of voters.

4.5 Creating plurinominal districts

NOTE: in what follows, 'district' stands for 'uninominal district'.
The plurinominal districts are created grouping uninominal districts according to the following rules:

1. the number of plurinominal districts that must be created is obtained dividing the number of uninominal districts (which is equal to the size of the parliament) by the number of seats per plurinominal district.
   NOTE: to differentiate between uninominal and plurinominal districts, the 'name' of the plurinominal districts are written with roman numbers in the output from the program.

2. create a list of plurinominal districts with their names

3. separate the uninominal districts in 2 lists: major and normal

4. starting with the major districts (which contain concentrated parties)

   (a) for each major party, the uninominal districts where this party is concentrated are extracted from the list of major uninominal districts. Then a plurinominal district is taken at random in the list and the uninominal districts where the party is concentrated are included in it, until either the plurinominal district is complete or all the districts where the party is concentrated have been inserted in a plurinominal district.

   (b) if the plurinominal districts are not full with the major districts, they are completed with *non major* uninominal districts, taken randomly.

5. if there are no major uninominal districts, or all the major districts have been used, the remaining plurinominal districts are created taking at random the requested number ("number of seats per plurinominal district") of uninominal districts.

4.6 The parliaments

NOTE 1: If there are ex-aequo winners in a district (parties or candidates with the same number of votes), the effective winner will be the party or candidate who has the highest share in the whole population. In the case where the possible winners also have the same share of votes in the population, the effective winner is chosen at random among them.
NOTE 2: in what follows, 'district' stands for 'uninominal district'.
NOTE 3: when the user clicks on the name of an electoral system, s/he is asked to specify the majority level (in percent) to be used when computing the indexes and forming the government. The default value is 50%. When indexes are computed, the majority level in seats or votes is the integer that is closest to the percent computation. When the majority level is 50%, he majority level in seats or votes is the integer that is closest to the percent computation +1.

4.6.1 One-district Proportionality

All districts are aggregated: the seats in the parliament have the same proportions as the shares of votes in the population. In other words, the parliament represents

exactly (but for rounding) the shares inserted when setting up the simulation.

4.6.2 Threshold Proportionality

When the user clicks the button next to this system, a dialog window appears, asking him to set-up the threshold level. All the parties who have a share of votes (strictly) smaller then this threshold level will be excluded from the parliament. The seats are divided among the remaining parties proportionaly with their share of votes in the population.

4.6.3 First Past The Post (Plurality)

The winner *in each district* is the party with the most votes in the district.

4.6.4 First Past The Post (Plurality) - with strategic vote

In this case, the voter has a positive probability of voting for the main party of a coalition, instead of his preferred party. This probability is equal to:

$$p = (1 - kD/100)$$

where $0 \leq D \leq 100$ is the distance between the main party of a coalition and the voter's preferred party.
When this system is chosen, the user is asked to give a value for k. The user is then asked to define the coalitions of parties and, if they were not defined at the beginning of the simulation, the parties' positions on the left/right axis.
The probability of voting for the preferred party is a multiple, k, of the relative distance on the left-right axis of the main party of the coalition, $D/100$ (obviously, if $kD/100 > 1$ the value is truncated to 1). $p = 1 - kD/100$ is consequently the probability of voting for the main party of the coalition.
In each district, the main party of the coalition is the one with the highest share of votes *in the district*. In other words, in each district, it is the party which appears most often as the voters' first preference. As a consequence, the main party in a coalition may change from district to district.
As with the previous system, the winner *in each district* is the party with the most votes in the district.

4.6.5 Runoff Majority

In each district, all the parties minus the 2 who have the most votes are excluded. A second round is performed with these two only, and the one who has the largest number of votes wins. If in the first round one party has more than 50% of the votes, it wins without the need for a second round.

4.6.6 Mixed Member I

In this system, part of the parliament is elected with the First Past The Post system, and the other with the One-district Proportional system. The percentage of seats elected with the One-District Proportional system is a parameter specified by the user when the Mixed Member I system is chosen.
Practically, the program computes a full parliament with the First Past The Post system, and another parliament with the One-district Proportional system. The final parliament is a weighted average of these two parliaments, the weights being determined by the parameter set up by the user.

4.6.7 Mixed Member I - with strategic vote

As the previous one using the First Past the Post system with strategic vote.

4.6.8 Mixed Member II

In this system, part of the parliament is elected with the First Past The Post system, and the other with the One-district Proportional system. However, the votes used for the First Past The Post part of the parliament are lost for the One-district Proportional one (this is the difference with the previous system).
In the program, the part of the parliament elected with the First Past The Post system is determined by the user. As with the previous system, the program computes first a full parliament with the First Past The Post System, then a parliament with the One-district Proportional system *with the votes not used by the First Past The Post system*. The final parliament is, as before, a weighted average of these two parliaments.

4.6.9 Mixed Member II - with strategic vote

As the previous one using the First Past the Post system with strategic vote.

4.6.10 Borda Criterion

This system uses the voters' complete preference order for parties. For each voter, the program gives points to each party, according to its place in the preference order. Here, the points correspond exactly with the position of the party in the array of preferences of the voter (that is 0 for the most preferred party, 1 for the second preferred party, ..., $N - 1$ for the least preferred party); where N is the total number of parties. The points obtained by each party are then summed up for all voters in the population. The winner is the party with the *smallest* sum.

4.6.11 Condorcet Winner

In each district the Condorcet Winner is chosen. This is the party that beats all the others when confronted in pairs.

NOTE: in case of cycle, the party with the highest number of votes in the district wins. The program reports under the graph of the parliament the number of cycles found during the computation of this system.

4.6.12 VAP System

NOTE: the VAP System is a new electoral system elaborated at the department POLIS of the Universita' del Piemonte Orientale. For further details, see G. Ortona "A weighted-voting system that performs quite well", POLIS Working paper n. 4, 1999 `http://polis.unipmn.it/pubbl/index.php?paper=440`.
In this system, we consider a 'real' parliament (defined in terms of seats) and a 'virtual' parliament (defined in terms of votes for the parties in the parliament, see below). The real parliament is elected with the One-district Proportional system. The virtual parliament depends on the composition of the government. As a consequence, the first output of this system is identical to that of the One-District Proportional system.
To obtain the virtual parliament (in terms of votes of the parties in government), the user must decide which parties compose the government (upper right panel). When the parties in government reach the majority, the virtual parliament is computed according to the following rule: the votes of the MPs from the bigger parties in the government are given a weight $a > 1$. These parties are defined *crucial*. The rationale for this weight is to allow the government to keep the majority even if some small parties leave the governing coalition. Therefore, the smaller parties lose their blackmail power. The government loses the majority only if big parties leave the coalition.
a is computed as follows:

$$a \sum_{i=1}^{m} X_i = (a \sum_{i=1}^{m} X_i + T - \sum_{i=1}^{m} X_i)/2 + y \tag{1}$$

that is

$$a = \frac{T - \sum_{i=1}^{m} X_i + 2y}{\sum_{i=1}^{m} X_i} \tag{2}$$

where X is the number of seats in the parliament of the m biggest parties in the government, and T is the size of the parliament. This way, the government keeps a majority of y even if small parties leave it.
X and y are parameters. In the program, they have the following values:

- $y = 1$, that is the government keeps a majority of 1 vote if the small parties leave
- $X = 18\%$ that is a crucial party must have at least 18% of seats in the parliament, and be part of the government. Note: X can be specified by the user when the virtual parliament is created, 18% is only the default value of the parameter.

Once it is created, the virtual parliament substitutes the real one in the parliament graph and in the summary table below.

4.6.13 Multi-District Proportionality

Hare method The program:

1. computes the quota (number of votes necessary in order to obtain a seat) using the following formula: `hare quota = number of electors in the plurinominal district/number of seats per plurinominal district`
2. gives each party as many seats as the number of full quotients (`n=number of votes/quota`) obtained
3. substracts from the votes for each party the number of votes used to give the seats to the party (`n*quota`): we get the remaining votes for each party
4. checks whether all the seats available for the district have been attributed to a candidate
   - SI' (yes): the process terminates
   - NO: attributes the remaining seats to the parties with the most remaining votes

Imperiali Method Like the Hare method, but using the following formula to find the quota: `imperiali quota = number of electors in the plurinominal district/(number of seats per plurinominal district + 2)`
NOTE: when there are few parties (less than 5), or the plurinominal districts are small (parameter "number of seats per plurinominal district" smaller than 10), the share in the Imperiali method is very small. As a consequence, the method will favour small parties, and under-represent large parties.

Sainte-Laguë Method The program:

1. creates a matrix containing all the parties, their respective votes and the factors as shown in the example of table 1 (assuming that the number of seats per plurinominal district is equal to 5):
2. uses as factors a series of odd numbers starting with 1 (that is 1, 3, etc).
3. takes from the matrix the best n results (where n=number of seats per plurinominal district) and gives the parties as many seats as they had best results.

D'Hondt Method As the Sainte-Laguë method, but using a series of factors starting with 1 (that is 1, 2, 3, etc) and ending with n, the number of seats per plurinominal district.

4.6.14 Single Transferable vote

With this system, the seats for each party, in each plurinominal district, are assigned according to a quota value. If some seats are not assigned with this method, the votes unused by the elected candidates are transferred to the next candidates in the elector's preference ordering. The candidates with the highest number of votes (obtained + transferred) are elected.

The types of quota 3 methods can be used to compute the quota:

N.B. quota : (number of electors in the plurinominal district) divided by (number of seats per plurinominal district + 1)

Droop quota : [(number of electors in the plurinominal district) divided by (number of seats per plurinominal districts + 1)]+1

Hare quota : (number of electors in the plurinominal district) divided by (number of seats per plurinominal district)

Creating the parliament

1. from the elector's preference ordering with respect to the candidates, the program sums up the votes obtained by each candidate (we assume that the elector votes for his preferred candidate).
2. some candidates have a total of votes equal or greater than the quota?
   (a) YES
      i. the candidates with more votes than the quota are inserted in a list of elected candidates
      ii. these candidates are eliminated from the list of candidates to be elected for their party
      iii. the surplus of votes of the elected candidates is transferred as described below.
   (b) NO
      i. the candidate with the smallest number of votes is eliminated from the list of available candidates
      ii. the voters who had the candidate just eliminated as first preference give their votes to the next candidate in their preference order

This process is repeated until:

- the requested number of candidates is elected (parameter "number of seats per plurinominal district")
- the number of candidates not yet elected is equal to the number of seats left. In this case, the remaining candidates are elected, even though their votes are less than the quota.

Criteria for transferring the surplus

- for each elected candidate (number of votes greater than or equal to the quota), the surplus is computed using the following formula: `surplus=sum of votes given to the candidate - quota`
- the surplus is transferred, starting by the largest, in the following way:
  1. from the candidate's voters, a number of voters equal to the surplus to share out is extracted at random
  2. the votes of these voters are transferred to the candidate indicated as next preferred in their preference ordering, and not yet elected
  3. if doing this we find that a candidate's votes becomes equal to or greater than the quota, this candidate is elected and eliminated from the list of candidates available to be elected.

4.7 Representativeness and governability indexes

NOTE: for the VAP system, these indexes are computed using the virtual parliament (that is the one in terms of votes).

4.7.1 Representativeness index - 1

The representativeness index of the electoral system j is given by:

$$R_j = 1 - \frac{\sum |S_{j,i} - S_{pp,i}|}{\sum |S_{u,i} - S_{pp,i}|} \tag{3}$$

where

- $S_{j,i}$= number of seats obtained by the party i with the system j
- $S_{pp,i}$= number of seats obtained by the party i with the one-district proportional system

Hence the first sum is the sum of the seats 'displaced' with respect to the proportional system. This number is an indicator of the lack of proportionality.

- $S_{u,i}$= number of seats that the party has in the case of maximum lack of proportionality, defined as the case in which the largest party in the proportional system has all the seats, and the other parties have none.

4.7.2 Representativeness index - 2

$$R = 1 - \frac{X}{T} \tag{4}$$

where

- X = number of seats in eccess with respect to one-district proportionality
- T = total number of seats in the parliament

This is a "naive" index, easy to understand and therefore useable in surveys. If for instance there are 100 seats in the parliament and three parties A, B, C with 50, 30 and 20 seats respectively with the one-district proportional system, and 80, 20 and 0 seats respectively with the current system, the index is equal to 0.7 (parties B and C lost 30 seats overall and party A gained 30).
The two indices of representativeness are highly correlated, but they have a different meaning. That of R2 is obvious; R1 has two differences:

a) the seats assigned differently from pure proportionality are counted twice, as they are unduly gained from one side and unduly lost from the other. If a party gains 10 seats and another one loses 10, the loss of proportionality (in the numerator) is 20, while in R2 is 10. It is possible to make R1 analogous to R2, simply dividing the numerator by 2;

b) the standardization (and the value of the denominator) are case-specific. As a result, for a given share of seats allocated differently from pure proportionality, the value of r1 increases if the relative decrease of parties that lose seats decreases. For instance, if under pure proportionality A has 70 seats and B 30, and under X they are respectively 90 and 10 (hence with 20% of the seats misallocated and a relative decrease of B of 2/3), the value of R1,x is 0.33; while if the figures are 60 and 40 under pure proportionality and 80 and 20 under X (again with 20% of the seats misallocated, but with a relative decrease of B of 50%), the value of R1, x rises to 0.5.

4.7.3 Representativeness indexes based on power

The program presents two indexes of representativeness based on power indexes according to section 4.8. These indexes appear in the "Power indexes" window (figure 9 on page 27), and are computed for each power index required by the user.

The "Ortona" index Consider the value S_o:

$$S_o = \sum_{i \in P} |s_i - \phi_i| \tag{5}$$

where

- P is the set of all the parties in the simulation
- s_i is the share of votes of party i in the simulation
- ϕ_i is the power index for party i computed according to the number of seats party i has in the parliament. ϕ_i therefore depends on the current voting system, and the power index that is used.

The representativeness index R_o is computed as follows:

$$R_o = 1 - \frac{S_o}{2} \tag{6}$$

The "Fragnelli" index Consider the value S_f:

$$S_f = \sum_{i \in P} |\phi_i - \psi_i| \tag{7}$$

where

- P is the set of all the parties in the simulation
- ϕ_i is the power index for party i computed according to the number of seats party i has in the parliament. ϕ_i therefore depends on the current voting system, and the power index that is used.
- ψ_i is the power index for party i computed according to the number of votes party i has obtained. ψ_i therefore depends on the current voting system, and the power index that is used.

The representativeness index R_f is computed as follows:

$$R_f = 1 - \frac{S_f}{2} \tag{8}$$

4.7.4 Gallagher index

This index measures the disproportionality of the parliament, confronting the seats obtained by each party with the actual votes they received. It is computed as follows:

$$G = \sqrt{\frac{\sum_{i=1}^{n} (v_i - s_i)^2}{2}} \tag{9}$$

where

- i indexes the parties in the simulation
- v_i is the percentage of the total votes obtained by party i
- s_i is the percentage of seats obtained by party i in parliament.

4.7.5 Governability index - 1

This index is made up of two sums:

$$G_j = A + B \tag{10}$$

- $A = 1/(C+1)$ where C is the number of parties in the government such that if they leave the government, it loses the majority (these parties are said to be *crucial*)
- $B = \frac{n}{[(m/2)+1]} * [\frac{1}{C} - \frac{1}{C+1}]$ where
  - n=number of seats *above the majority level* (half the number of seats plus one if the number of seats is even, half the number of seats plus 0.5 if it is odd)
  - m=total number of seats
  - C=total number of crucial parties

Suppose for instance that the majority is made of two parties, both crucial. The A component of the index states that its value is comprised between 1/3 and 1/2. The exact value is determined by the component B. If the majority has just half of the seats plus one, the value of B will be zero, and $A + B$ will be equal to A, i.e. to 1/3. At the opposite, if the majority has all the seats, the value of $A + B$ will be very close to 1/2 (actually it will tend to 1/2 as the total number of seats increases).

4.7.6 Governability index - 2

$$G = \frac{S}{N} \tag{11}$$

where

- S = share of seats in the parliament obtained by the parties in the government
- N = number of parties in the government

This is a "naive" index, that simply expresses the direct relationship between governability and S, and the inverted relationship between governability and N. It is easy to understand and can therefore be used in surveys.

4.7.7 Governability index - 3

$$G = \frac{S}{N} . \left(\frac{N}{P}\right)^a \tag{12}$$

where

- S = share of seats in the parliament obtained by the parties in the government
- N = number of parties in the government in the system x
- P = number of parties of the minimal majority in the one-district proportional system that includes all the parties of the majority of the system x

- a = parameter with values between 0 and 1, set up by the user

This index introduce the *factions* within the parties. Assume that there are 100 seats in the parliament, that the current government (system x) is made up of two parties with 60 seats, and that in the one-district proportionality these parties would reach the majority only in alliance with 3 other parties. In this case, S/N is equal to 0.3 and N/P to 0.4. Now assume $a = 0$. In this case, S/N is not changed, which means that the factions have no influence. If instead $a = 1$, the index becomes S/P (0.12 in the example), which means that the 'real' parties are not the ones in the government witht the system x but those in the government with the one-district proportional system. In the system x, those parties become factions of bigger parties in order to get votes, but still maintain fully their independence. The weight of the factions increase with a: if for instance $a = 0.8$, the index is equal to 0.144 instead of 0.3.
For further details on the meaning of the representativeness index - 1 and the governability index - 1, see Marie-Edith Bissey, Mauro Carini and Guido Ortona "ALEX3: a simulation program to compare electoral systems", Journal of Artificial Societies and Social Simulation, vol.7, n.3 (2004), downloadable from `http://jasss.soc.surrey.ac.uk/7/3/3.html`.
In practice, when the user has chosen the government for system x, the program will check whether the parties in the government also have the majority of seats in the one-district proportional system. If it is the case, the user will be prompted for the value of the parameter a. If it is not the case, the user will be prompted for the value of the parameter a and for the government in the one-district proportional system. The parties currently in the government in system x will appear already checked. The user only needs to add the new parties in the government, until the majority is reached.

4.7.8 Governability Index - 4

Proposed by Vito Fragnelli. From *The propensity to disruption for evaluating a parliament*, Vito Fragnelli, published in Homo Oeconomicus, 2009.
This governability index assess the propensity to disrupt of a party in government, and uses the Owen Power Index.

$$g^{\Omega} = 1 - \frac{\sum_{i \in S} \Omega_i(S \backslash \{i\})}{n} \tag{13}$$

where

- S is the set of parties in government
- $\Omega_i(S \backslash \{i\})$ is the Owen index of party i when i has left the government so the a priori unions structure K considers the coalition $S \backslash \{i\}$ and all the other players as singletons, i.e. $K = \{S \backslash \{i\}, \{i\}, \{i_1\}, ..., \{i_{n-s}\}\}$, where $s = |S|$

- n is the total number of parties in the simulation

In this case, all the parties not in the government are considered as singletons.

4.7.9 Governability Index - 4*

Proposed by Vito Fragnelli. From *The propensity to disruption for evaluating a parliament*, Vito Fragnelli, to appear on Homo Oeconomicus, 2009.
This governability index assess the propensity to disrupt of a party in government, and uses the Owen Power Index.

$$g^{\Omega^*} = 1 - \frac{\sum_{i \in S} \Omega_i(S \backslash \{i\}, (N \backslash S) \cup \{i\})}{n} \tag{14}$$

where

- S is the set of parties in government
- $\Omega_i(S \backslash \{i\}, (N \backslash S) \cup \{i\})$ is the Owen index of party i when i has left the government and formed a coalition with all the parties not in the government, so the a priori unions structure K considers the coalition $S \backslash \{i\}$ and the coalition $(N \backslash S) \cup \{i\})$, i.e. $K = \{S \backslash \{i\}, (N \backslash S) \cup \{i\})\}$
- n is the total number of parties in the simulation

So the difference with the previous index g^{Ω} is that all the parties not in the government form a coalition instead of being singletons.

4.8 Power indexes

Since their computation may take some time, these indexes are produced only if the user requests them by clicking on the "Compute power indexes" button at the bottom of the "Indexes" part of the output window. When the button is clicked, the user needs to specify which index s/he wants and its relative parameters, as shown on figure 8. The resulting output is shown on figure 9.

4.8.1 Shapley-Shubik Index

For each party i in the simulation:

$$\phi_i(v) = \sum_{S \subseteq N, i \in S} \frac{(s-1)!(n-s)!}{n!} [v(S) - v(S \backslash \{i\})] \tag{15}$$

where

- S is a subset of the parties (N) in the simulation
- s is the number of elements in set S

Factor Sainte Laguë	party A	party B	party C	party D	party E
/1	**350**	**310**	**140**	**120**	80
/3	**116**	103	46	40	26
/5	70	62	28	24	16
Total sum of seats	2	1	1	1	0

Table 1: Matrix for the Sainte Laguë method

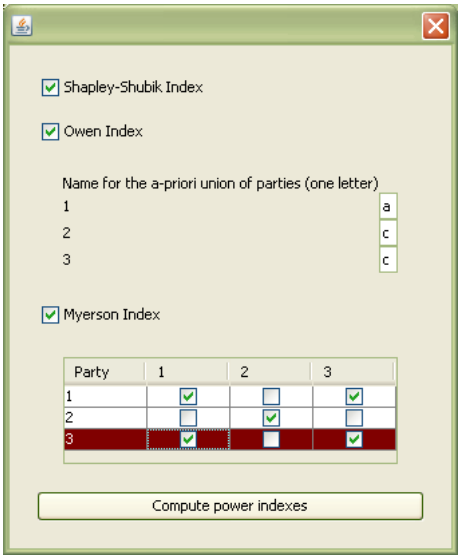

Figure 8: Setting up the parameters of power indexes

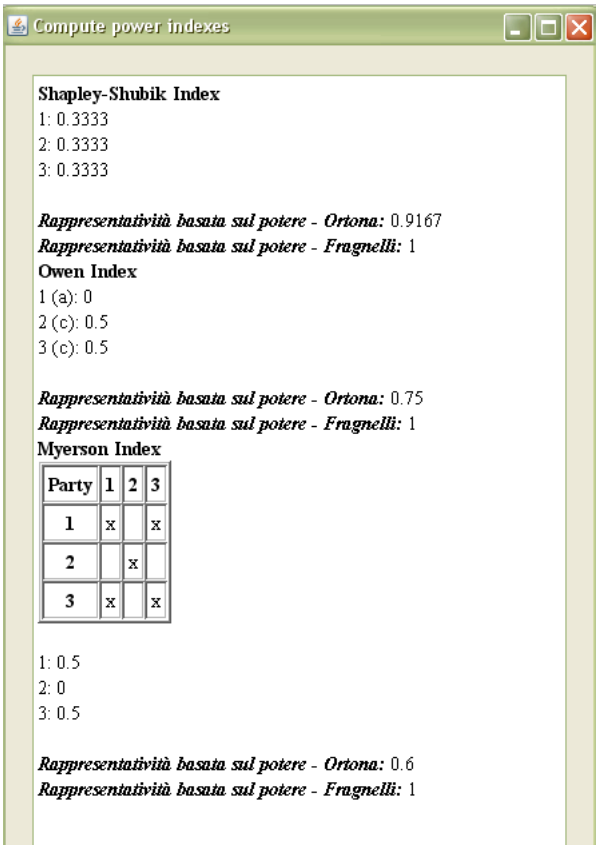

Figure 9: Results of the computation of power indexes

- n is the number of elements in set N, that is the total number of parties in the simulation
- $v(.)$ is the characteristic function which, in this case, takes the value 1 if the set of parties on which it is computed reaches the majority of seats in parliament, and the value 0 if this set of parties does not reach the majority of seats in parliament.

4.8.2 Owen Index

The user must specify the a-priori unions between parties, assigning a letter of the English alphabet to each union. The default is that each party is in a separated union (case in which the Owen index reduces to the Shapley-Shubik index).
Consider a set of a-priori unions $K = \{T_1, \ldots, T_k\}$. For each party i in the simulation, belonging to union T_j:

$$\Omega_{i,i\in T_j} = \sum_{H\subset K, T_j\notin H} \frac{h!(k-h-1)!}{k!} \sum_{S\ni i, S\subset T_j} \frac{(s-1)!(t_j-s)!)}{t_j!}[v(H\cup S) - v(H\cup S\setminus\{i\})] \tag{16}$$

where

- H is a subset of the a-priori unions set K which does not contain union T_j
- h is the number of a-priori unions in H
- k is the number of a-priori unions in K
- S is a subset of parties of the union T_j which contains party i
- s is the number of parties in S
- t_j is the number of parties in T_j
- v is the characteristic function as described above.

4.8.3 Myerson Index

The user must specify the connections between the parties. The parties appear in a double entry table. Clicking on the cell at the intersection of two parties indicate that they are connected together. Let us call G a particular set of connections between parties.
In the Myerson Index, the characteristic function is computed as follows for a set of parties S:

$$v_G(S) = \begin{cases} v(S) \text{if } S \text{ is connected} \\ \sum_{T\in S/G} v(T) \end{cases} \tag{17}$$

where

- v is the characteristic function as described above
- $T \in S/G$ is a subset of the connected elements of S according to the set of connections G

The Myerson Index is the Shapley-Shubik Index computed using the game defined by the characteristic function v_G:

$$\phi_i(v_G) = \sum_{S \subseteq N, i \in S} \frac{(s-1)!(n-s)!}{n!} [v_G(S) - v_G(S \backslash \{i\})] \tag{18}$$

If every party is in connected to all the others, the Myerson Index reduces to the Shapley-Shubik Index.

4.8.4 Deegan-Packel Index

This index is based on the set of all *minimal winning coalitions* among the parties, that is all the smallest groupings of parties that have the majority of seats in the parliament.
Let us call this set $N = \{S_1, \ldots, S_k\}$, containing k elements.
The Deegan-Packel index is therefore, for each party:

$$DP_i = \frac{1}{K} . \sum_{j: i \in S_j} \frac{1}{s_j} \tag{19}$$

where

- $j : i \in S_j$ indicates all the minimal winning coalitions in N that contain party i
- s_j is the number of parties in the minimal winning coalition S_j.

4.8.5 Holler Index

This index is based on the set of all *minimal winning coalitions* among the parties, that is all the smallest groupings of parties that have the majority of seats in the parliament.
Let us call this set $N = \{S_1, \ldots, S_k\}$, containing k elements.
To obtain the Holler index, we first compute for each party i:

$$\theta_i = \sum_{j: i \in S_j} 1 \tag{20}$$

which basically counts the number of times the party is in a minimal winning coalition.
Then the Holler index is:

$$h_i = \frac{\theta_i}{\sum_{k \in N} \theta_k} \tag{21}$$

where $\sum_{k \in N} \theta_k$ is the sum over all parties of the θ values computed above.

4.8.6 ALEX1 index

Let a *contiguous winning coalition* be a coalition that

- (a) enjoys the majority requested to be winning and
- (b) is made of parties adjacent on the left-right axis.

In each of such coalitions the Alex1 power of the party i is 0 if that party is not crucial, that is if the coalition is winning also without i; and is $(S_i/S_c)^a$ if i is crucial, where

- S_i are the seats of party i;
- S_c are the seats of the coalition;
- a is a parameter to be inputed, ranging 0 to 1.

Then, the total Alex1 power of i, $PA1_i$, is the sum of the powers of i across all the contiguous winning coalitions. The index, ranging 0 to 1, is obtained dividing the total Alex1 power by the sum of the total Alex1 powers of all the parties:

$$A_1(i) = \frac{PA1_i}{\sum (PA1)} \tag{22}$$

The logic behind the index is the following. A non-crucial party enjoys no power. The power of a crucial party depends also, and realistically, from its share of seats. If $a = 1$ it is determined only by this share; if $a = 0$ it is determined only by its being crucial, and the index collapses to Banzhaf's. Note that a is the elasticity of the power of i with respect to its share of seats, and as such it may (in principle) be assessed empirically.

4.8.7 ALEX2 index

All the notions and notations are as for Alex1. Let the *Alex2 power* of party i in a given contiguous winning coalition be $bPi + (1-b)S_i/S_c$ where $P_i = 0$ if i is not crucial and 1 if it is, and b ranges 0 to 1.
The *total Alex2 power* of i is the sum of its powers across all the contiguous winning coalitions, and the index for i is obtained, as before, by dividing the total Alex2 power of i by the sum of the total Alex2 powers of all parties.
This index is the simplest formulation of the hypotheses that the power of a party depends from its being crucial and from its share of seats, and that non-crucial parties have some power too (why staying in a governing coalition if not)? Yet, it introduces a parameter b more difficult to assess empirically than a in Alex1. Obviously, if $b = 1$ the index collapses to Banzhaf, and if $b = 0$ its value is determined only by the share of seats.

4.8.8 FP indices

Analogously to the Alex indices, it is assumed that the feasible coalitions include only contiguous parties. Let W^c be the set of contiguous winning coalitions, where a coalition S_i belonging to W^c is contiguous if for all k, h belonging to S_i if there exists j belonging to N with $k < j < h$ then j belongs to S_i.
Starting from this idea the family of power indices FP assigns (see Fragnelli et al., 2009) to each party j belonging to N:

$$FP_j = \sum_{S_i \in W^c, j \in S_i} \alpha_i \beta_{ij} \tag{23}$$

where $\alpha_i \geq 0$ represents the relative probability of coalition Si to form, with the condition:

$$\sum_{S_i \in W^c} \alpha_i = 1 \tag{24}$$

and $\beta_{ij} \geq 0$ is the power share assigned to party j in S_i, with the condition for each coalition S_i belonging to W^c:

$$\sum_{j \in S_i} \beta_{ij} = 1 \tag{25}$$

The choice of parameters α_i differentiates the probabilities of the coalitions to form and the choice of parameters β_{ij} differentiates the relevance of the parties inside each coalition. We address to Fragnelli et al. (2009) for possible methods to compute the values of the parameters α_i and β_{ij} that account for the ideological distances, number of parties in the majority, their number of seats or via a suitable analysis of historical data.
We remark that the definition of the FP family allows considering even a subset of contiguous winning coalitions, but this is equivalent to assign a null probability to the remaining contiguous coalitions.

Fragnelli, V., Ottone, S. and Sattanino, R. (2009). A new family of power indices for voting games. Homo Oeconomicus, 26, 381-394.

5 EXPANDING THE PROGRAM

5.1 Adding a new electoral system

[To add a new electoral system or otherwise make changes to the program, you need to have Java Standard Edition installed on your computer, not just Java Runtime!]
The program is written using the object-programming capabilities of Java, therefore any new electoral system has to be extended from a preceeding one. Only the methods that differ between the two systems have to be written. All the electoral

systems are extended from the class `Parliament` in the package `parliaments`. Uninominal systems are grouped in the package `parliaments.uninominal` and the plurinominal are in the package `parliaments.plurinominal`. Each class defining a new system has a name identical to its reference key in the language file in the following directories:

- `languages\uninominal_**.properties`

or

- `languages\plurinominal_**.properties`)

This identity allows the program to load automatically any new system in the list of the available parliaments. No other class needs to be altered.
To create a new electoral system:

1. create a new class in the package (and the directory) `parliaments uninominal` or `parliaments\plurinominal`, with a name that represents the system, and is identical to its reference key in the language file

2. this new class **must** at least extend the `Parliament` class. It is possible to extend another class of the package to take advantage of its characteristics (all the existing classes in the program already extend `Parliament`).

3. the `Parliament` declares **3** `abstract` functions, that must be defined for each new class that extends it:

   - `public abstract HashMap findAllocationOfSeats();` defines the way the seats are allocated in the parliament. It is the most important class of the electoral system. The two other classes that follow allow to identify the parliament in the various menus and output files.
   - `public abstract String getParliamentName();` defines the name of the parliament using the language file and adding to it the name and/or values of extra parameters (eg. the rounding method for the VST, or the threshold value of the threshold proportional)
   - `public abstract String getParliamentKey();` defines the key with which the parliament is saved in the list of the simulations (object `SimulationRepository` of the `VotingObjects` package). This key is used to save the parliament in a file, or to re-load it if the display window has been closed. Again in this case it is advisable to include in this string the names and/or values of the extra parameters of the system.

   In addition, it is possible to define extra parameters (eg. the rounding method for the plurinominal systems) in the class constructor.

To extract a name from the language file, the program uses the `getString` function from the `Language` object, which takes as arguments the name of the file and the requested key:

```
language.getString("uninominal","ThresholdProportional");
```

The `language` object is an instance of the `Language` class of the `globals` package. This is an object of type Singleton (there can be only one instance of this object in the program). Using this type of object allows to manage the display language inside the program, and to modify it 'on the fly'. The name of the file to pass as argument is therefore only the part before the extension `.properties`.

4. add the name of the class as key string in all the language files, and the name of the parliament in the considered language. NOTE: even if you do not translate the name of the parliament in other languages than your own, it is recommanded to insert the parliament key in all language files.

5. compile the program using the `make` command at the MS/DOS prompt.

The new electoral system will automatically appear in the list of available electoral systems that is shown once the districts have been created. NOTE: a plurinominal system will only appear if the parameter "number of seats per plurinominal district" is greater than one in the simulation.

5.2 Adding a new index

The steps to follow are similar to those indicated for the electoral systems. The indexes are in the package `indexes` and all extend the `Index` class, which declare the abstract function `computeIndex()`, that must be defined for each index. Once the program has been recompiled using the `make` command, the new index will automatically appear in the bottom right list of indexes within the parliament window. As for the electoral system, the name of the new index class must be the same as its reference key in the language files `languages\index_xx.properties`.

5.3 Adding a new language

You can add a new language to the program without recompiling it. It is enough to create the following 5 files (and to translate their contents):

- `indexes_xx.properties` contains the reference keys and names of the indexes that appear in the bottom right part of the parliament window

- `uninominal_xx.properties` contains the reference keys and names of the uninominal systems

- `plurinominal_xx.properties` contains the reference keys and names of the plurinominal systems
- `labels_xx.properties` contains the reference keys and names/messages that appear in the various windows of the program
- `messages_xx.properties` contains the reference keys and text of the error messages of the program

NOTE 1: In the file names indicated above the sequence **`xx`** must be substituted by the 2 letters that identify the requested language (for instance `it` (Italian), `fr` (French), `en` (English)). A list of the languages accepted by Java can be seen at the following address: `http://ftp.ics.uci.edu/pub/ietf/http/related/iso639.txt`.
NOTE 2: Up to now the languages used in the program use the latin alphabet. I do not know how (and whether) the program would work with a cyrillic or an Asian alphabet.
NOTE 3: I would be glad if you could send me your translation of the program (files `*.properties` and `readme` if you did that one too) at the following address: `marieedith.bissey@sp.unipmn.it` so I can include them (with all due reference to the translator) in further versions of the program.

5.4 Creating a file of preferences

As explained above, the program allows to save a simulation, as well as the parameters of the simulation. For instance, if the simulation data are saved while one saves a parliament, it is possible to re-use this file to re-create a new simulation with the same parameters. The program simulates automatically the missing data (voters' preferences and district composition).
The directory `savedSimulations` contains some examples of files saved from the program (`exampleUninominal.out` and `examplePlurinominal.out`). These are simple ascii files, so it is not too complex to create one of them to be loaded up in the program, in order to introduce preferences and district composition obtained from surveys.

5.4.1 The least a file must contain

The least a file must contain are the following parameters, in a format identical to the one reported below:

```
nameRepository : sim_1 // name of the simulation
descriptionRepository :
numberPlurinominalDistricts : 5 :// Number of plurinominal districts
numberUninominalDistricts : 10 :// Number of uninominal districts
probFirst : 0.8 :// first adjacent party
probSecond : 0.1 :// second adjacent party
```

```
probPreferred : 0.9 :// preferred candidate
totalNumberVoters : 1000 :// total number of voters
numberParties : 3 :// Number of parties
numberVoters : 100 :// Number of voters per uninominal district
numberCandidates : 2 :// Number of seats per plurinominal district

party_1_name:1 : // party name
party_1_share:25.0 : // share (\%)
party_1_major:false :// major?
party_1_concentration:0: // concentrated in how many districts?
party_1_coefficient:1.0 :// Concentration coefficient

party_2_name:2 : //  party name
party_2_share:40.0 : //share (\%)
party_2_major:false :// major?
party_2_concentration:0: // concentrated in how many districts?
party_2_coefficient:1.0 ://  Concentration coefficient

party_3_name:3 : // party name
party_3_share:35.0 : //share (\%)
party_3_major:false :// major?
party_3_concentration:0: // concentrated in how many districts?
party_3_coefficient:1.0 ://  Concentration coefficient
```

The strings on the left of the ":" allow to identify the variables to insert in the program, therefore they must not be altered. The decimal values must have a dot to separate the decimal part of the number (not a comma). The strings on the right of the ":", preceded by "//" are comments and can be omitted. The indexes used as the parties name must start from 1 to the number of parties in the system.
With these values, the program is able to simulate the voters' preferences for the parties (and for the candidates if requested), to establish the composition of the districts and to compute the parliaments.

5.4.2 Other contents

A file saved from the program contains, in addition to the parameters described above, the voters' preferences for the parties and the candidates, and the composition of the uninominal and the plurinominal districts. If one of these elements is present in the file in the expected format and is complete, the program will use it. Otherwise, it will be simulated.
Please look at the example files to see which format you must use. As for the previous section, each row must contain 3 elements separated with ":":

```
chiave_di_riferimento:valore:commento
```

Voters' preferences for parties

```
voter_1000_partyPreferences:[3, 2, 1]:
voter_999_partyPreferences:[3, 2, 1]:
voter_998_partyPreferences:[3, 2, 1]:
voter_997_partyPreferences:[3, 1, 2]:
voter_996_partyPreferences:[3, 2, 1]:
....
```

In this case, there are 1000 voters in the simulation, and 3 parties (identified as 1, 2, 3). For each voter, the reference key is: `voter_xxx_partyPreferences:[a,b,c]:` where `xxx` is the index of the voter, in our case from 1 to 1000, the list of parties is written between square brackets, the indexes of the parties are comma-separated lists, the most preferred parties is the first one written in the list (on the left). Please note that even if there are no comments, the second ":" must appear anyway.

Voters' preferences for candidates A candidate is identified by its party, and its position in the party's list of candidates. For instance, if there are 3 parties and 2 seats per plurinominal district, a party list will contain 2 candidates, and there will therefore be a total of 6 candidates in the simulation. In this case, the list of candidates will be as follows:

```
candidate1_1:1_1:// candidate in first position in the list of party 1
candidate1_2:1_2:// candidate in second position in the list of party 1
candidate2_1:2_1:// candidate in first position in the list of party 2
candidate2_2:2_2:// candidate in second position in the list of party 2
candidate3_1:3_1:// candidate in first position in the list of party 3
candidate3_2:3_2:// candidate in second position in the list of party 3
```

Voters' preferences for the candidates are written as follows:

```
voter_1000_candidatePreferences:[3_1, 3_2, 2_1, 2_2, 1_1, 1_2]:
voter_999_candidatePreferences:[3_2, 3_1, 2_1, 2_2, 1_1, 1_2]:
voter_998_candidatePreferences:[3_1, 3_2, 2_1, 1_1, 2_2, 1_2]:
voter_997_candidatePreferences:[3_1, 3_2, 1_1, 2_2, 2_1, 1_2]:
voter_996_candidatePreferences:[3_2, 2_1, 3_1, 2_2, 1_1, 1_2]:
```

where in the list of candidates, the number before the _ is the index of the party, and the number after the _ is the position of the candidate in the list.

Composition of uninominal districts Uninominal districts are essentially groups of voters. The size of the districts is defined by the `numberVoters` parameter and is the same for all the uninominal districts in the simulation. In the file, the composition of a plurinominal district is written as follows

```
districtUninominal_1_nameOfDistrict : 1 :// Name of the uninominal district
districtUninominal_1_concentratedMajorParty : false :// does it contain a concentrated party?
districtUninominal_1_nameMajorParty : 0 : // name of the major party (if there is one)
districtUninominal_1_listOfVoters :[956 , 154 , 550 , 76 , 935 , 113 , 407 , 507 , 818 , 530 , 41 , 600 ,(...), 759 , 640 ]
```

The first 3 rows describe the district: its 'name' (or index), whether it contains a concentrated party, and in this case, the party index. If it does not contain a concentrated party, as is the case above, the party index is 0. The fourth row contains the voters that belong to the district, identified by their index. In the example, the district contains the voters 956, 154, etc. The list of voters must contain `numberVoters` elements.

Composition of plurinominal districts Plurinominal districts are made of uninominal districts.

```
districtPlurinominal_1_name : 1 :// Name of the plurinominal district (roman= I)
districtPlurinominal_1_listOfDistricts : [2, 4]
```

The first row defines the name of the plurinominal district (as before, an index). In the results files, this plurinominal district will appear with a name written in roman number, to distinguish it from the uninominal districts. The list of uninominal districts contained in the plurinominal district contains `numberCandidate` elements, the uninominal districts are identified by their index.