Voting Simulation Program User’s Guide

Original Code by Duncan Buell

User’s Guide and code modifications by:

Michael Cantwell

Robert Carff

Anthony Frazier

Carson Kaylor

Sebastian Martin

-Input-

The program will need 3 input files to read from, which need to contain certain information properly formatted.

Executing the program will require execution of the file name, along with 4 filename arguments: **configfilename**, **pctfilename**, **outfilename**, **logfilename**. (the name of the configuration file, the name of the precinct file, the name out of the output file, and the name of the log file).

All of the data in the first input file (**configfilename**) will need to be included in lines. Each piece of data will need to be separated by a delimiter such as a space or tab.

The first value of the first line will need to be seed value, which is a number that will be used for random number generation.

The second value (still in same line) will need to be the number of hours (as an int) that the election day will last. This value is converted to seconds by multiplying the number of hours by 3600.

The third value needs to be the mean time to vote in seconds as an int value.

The fourth value needs to be the minimum number of voters per precinct that is expected for this simulation as an int value.

The fifth value needs to be the maximum number of voters per precinct that is expected for this simulation as an int value.

The sixth value is what is considered to be a wait time in minutes that is too long, also as an int value.

The seventh/last value of this line must be the number of iterations wanted for each precinct, which is the amount of times the simulation will run for each precinct.

The next line of the input file will be the percentage of voters per hour. The first value of this line will be the percentage of voters at the very start of the day as a double value, at hour zero. This value should be zero. After this value, there should be an equal number of double values as hours in the election. Each of these double values should be the percentage of voters that voted during each hour time frame. The first of these values (second value of the second line) should be the percentage of voters that voted from hour zero to hour one. The next value should be the percentage of voters that voted from hour one to hour two and so on. These values are then pushed to the arrival\_fractions\_ vector.

A second input file is then opened. The name of the input file is declared in configuration.cc. It is named “dataallsorted.txt”. This input file should have a list of all of the sorted service times, each value its own line. The service time is the amount of time at the voting poll. These values are then pushed to the actual\_service\_times\_ vector.

The third input file (**pctfilename**) will also require all of its data to be formatted in lines, with each piece of data in each line separated by a delimiter.

The first value in each line will need to be the precinct number as an int (Ex. 001).

The second value in each line will need to be the precinct name as a string (Ex. XXX00100).

The third value in each line will need to be the precinct turnout as a percent in double form. This value is the percent of voters from the precinct that came to the polls (Ex. 20.2).

The fourth value in each line will need to be the number of voters that turned out in that precinct as an int (Ex. 10101).

The fifth value in each line will need to be the number of expected voters for that precinct as an int (Ex. 100).

The sixth value in each line will need to be the number of voters expected per hour as an int (Ex. 10).

The seventh value in each line will be the number of polling booths an int (Ex. 8).

The eighth value in each line will be the precinct minority percent as a double (Ex. 10.3).

The next three values will be int values. They are stat1, stat2, and stat3 (Ex. 0).

-Output-

The output file will include all of the important information from the simulation that was just ran.

The first chunk of text that is seen contain important information about the total simulations.

The header will start with the “RN seed” this is the number that was used to randomly generate the voters. Then the length of the election day is stated in seconds, and then in hours. This is how long each precinct was open in the simulation. Then the “Time to vote mean:” is printed, this is the average time it took per voter to vote. The “Min and max expected voters” is the range of voters that can be expected at a certain precinct. After this the “too long” time is printed, this is the user assigned time where the program recognizes it has having taken too long for a voter to vote. The number of iteration is printed out after this. This is the amount of times that the simulation will run. The Max service subscript is the longest time it took for a voter to be able to vote, this is printed in seconds.

After that a table is printed out. The left side of this table, in the format of “#- #” shows the offset, this will always start at “0- 0” and then jump to “6- 7”, from there on out each side will increase by one until there are no more arrival fractions to print. The offset counts for times of the day for when the precinct is open. The right side of the table will include the arrival fractions during the timeframe of the offset.

After this header the actual simulation data is printed out.

Each separate precinct will begin with “SIM: RunSimulation for pct.”

After this a line will be printed out, also starting with “SIM:”

Example :” SIM: 1 XXX00100 20.20 10101 100 235 8 10.30 HH 0 HH”

Behind this there will be a number, stating the number of simulation ran, in this case “1”

Then the precint number is printed, in this case “XXX00100”

Behind this The voter turnout is printed, in this case “20.20”

Then, the number of voter in the precinct is printed out, in this case “10101”

After this the expected number of voters for the precinct is printed, here it is “100”

Then, the amount of voters coming within a single hour is printed, in this case “ 235”

Behind this the amount of precinct stations is printed out, being “8” here

Then the minority of the precinct is printed out, here it is “10.30”

After that, there will always be an “HH” printed out, this is for separation

Behind that, the Stations to histo is printed out, here it is “0”

Then, another “HH” is printed out for separation

After this line, the information is repeated and printed out again.

Below this, the data for the simulation is printed out, Each precinct will run a certain amount of times, depending on the amount of iterations inputted by the user.

Example line: “OnePct: 0 1 XXX00100 100 1 stations, mean/dev wait (mins) 0.78 1.60 toolong 0 0.00 0 0.00 0 0.00”

The first number in this line is the current iteration, here it is “0”

Next, the number of simulation is printed out, here it is “1”

Next, the Precinct number is outputted, in the example it is “XXX00100”

Then, the precinct number is outputted again in its shortened form, here it is “100”

After that the expected amount of voters is printed, here it is “100”

Next, the station count is printed, following that the phrase “stations, mean/dev wait (mins)” is printed for formatting and reading of the output.

After that the average wait time is printed out in seconds, here it is “.78”

Then, the average deviation is also printed out in seconds, it is “1.60” in the example.

After that the phrase “toolong” is printed, followed by the too long count, in this case “0”

The next number is the maximum amount of time that each voter should wait, in this case “0.00”

The next 4 numbers are the same, except for too long +10, and too long +20 respectively.

At the end of each precinct simulation the line “OnePct: toolong space filler” is printed as a line filler.

At the end of the entire program the line “SIM: PRECINCT COUNT THIS BATCH #” will print, with the # determining the amount of precincts simulated.

From this output a user can see that the number of stations significantly affects the mean wait times and deviations. Increasing the number of stations will greatly decrease the mean wait time and deviation. When a precinct with a certain number of voters and a certain number of stations is compared against itself with the same number of voters and a greater amount of stations, the output displays that the version of the precinct with less stations has a much greater mean wait time and deviation compared to the version with more stations.