TECHNIUM

Technium Vol. 2, Issue 3 pp.43-48 (2020)
ISSN: 2668-778X
www.techniumscience.com

"DRAWS": an application for small-scale electronic draws

Nicolaos Charitos, Evi Papaioannou

University of Patras, Greece
papaioan@ceid.upatras.gr

Abstract. We present "Draws", an application for conducting small-scale draws. The efficiency of the procedure and the true random and unbiased nature of the computed results stems from the theoretical soundness of the exploited shuffling algorithms, namely Durstenfeld-Knuth shuffle and its predecessor, Fisher-Yates shuffle. Using the application does not require advanced computer skills or sophisticated equipment. The application could highly facilitate and enhance administrative procedures offering speed and transparency.

Keywords. Electronic draws, shuffling algorithms, true randomness and efficiency, software application, java, transparency in administration.

1. Context and motivation

The constitution of administrative bodies by drawing candidates out of a predefined pool is a frequent issue arising during several administrative procedures which require the formation of committees or working groups. Usually, pools of candidates as well as required administrative bodies are composed of a relatively small number of members (which rarely exceeds 50). A basic requirement, common in all these cases, is the need for transparency in the selection process, which implies draws generating true random and unbiased results. A traditional implementation approach, often adopted in practice, consists in picking "tickets out of a hat" in public for preserving transparency. However, the transparency and the true randomness of this process are often questioned. In addition, manually conducted draws are time-consuming, thus making inefficient solutions.

There are, indeed, sophisticated software systems and platforms available for electronic election-like procedures, which could perhaps be adjusted to also perform electronic draws. There are also advanced lottery systems, implementing particular cryptographic protocols, used in the context of lottery games. However, apart from their inherent complexity, such systems require advanced computer and programming skills to configure and use. They may also require sophisticated equipment and fees not affordable by small-scale organizations (like for example, educational institutes, public agencies, ministries, small companies, etc).

Motivated by experiences in our professional environment (a higher education institute) as well as by similar experiences of colleagues in other sectors ranging from public administration agencies to small-scale companies, we designed and implemented an application for conducting small-scale draws, guaranteed to generate true random and unbiased results in the blink of an eye. “Draws” can be very easily installed on any machine (desktop computer or laptop featuring average specifications), requiring elementary computer skills. It offers a user-friendly graphical user interface (GUI), which is simple and light-weighted and computes results which are provably true random and unbiased. It is worth mentioning that “Draws” is an efficient application – both in terms of true randomness of computed results and execution time – since it has been built exploiting provably efficient shuffling algorithms, i.e., Fisher-Yates shuffle and Durstenfeld-Knuth shuffle [1, 3].

Technium Vol. 2, Issue 3 pp.43-48 (2020)
ISSN: 2668-778X
www.techniumscience.com

2. The application environment

"Draws" application is composed of a single executable file, an installer, which when executed installs the application on a Windows machine, without requiring the installation of additional software or any other modification. Once installed, "Draws" appears under "All Programs" list and can be optionally called via a desktop shortcut. The application receives a list of candidates and the number of required random selections. It computes the requested random selection and returns the results in the form of a text file, which also contains the exact date and time of the draw. The date and time of each draw is automatically appended to the filename of the produced text file and at the end of its content, as well. "Draws" performs the random selection of the requested members by first shuffling the list of candidates and then extracting a prefix of the shuffled list of size equal to the number of the requested selections.

The GUI of our application is depicted in Figure 1. Functionality boxes of the upper row perform an electronic draw utilizing the Fisher-Yates shuffle, while functionality boxes of the bottom row perform an electronic draw utilizing the Durstenfeld-Knuth shuffle. For each shuffling method, two options are available depending on how the input list is provided: via a file, or manually, i.e., by typing in the list members. The last functionality box of each line allows conducting the electronic draw without a graphical environment, i.e., via the command line environment.

Figure 1: "Draws" GUI

2.1. Providing the list of candidates via a file

By choosing to perform the electronic draw using an input file for providing the list of candidates, a new pop-up window appears, as depicted in Figure 2. The window on the left of Figure 2 can be used for performing an electronic draw exploiting the Fisher-Yates shuffle, while the window on the right part of Figure 2 can be used to perform an electronic draw exploiting the Durstenfeld-Knuth shuffle.

Technium Vol. 2, Issue 3 pp.43-48 (2020)
ISSN: 2668-778X
www.techniumscience.com

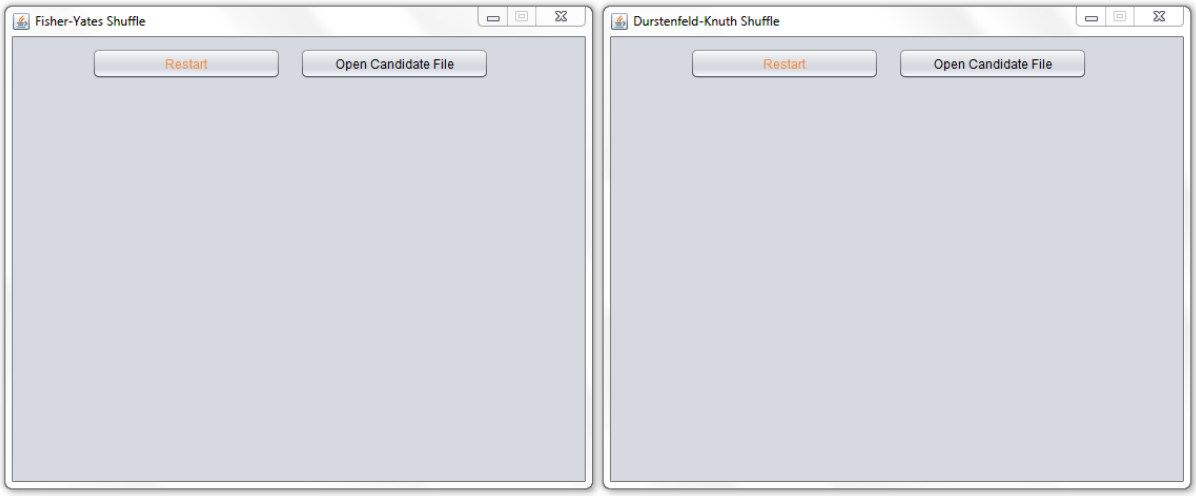

Figure 2: Performing an electronic draw by providing the list of candidates via a file

For the purpose of demonstrating the application components, we use a test input file composed of approximately 10000 entries, which are recorded one per line. We request the random selection of two groups of 20 and 40 members to be performed via Fisher-Yates shuffle and Durstenfeld-Knuth shuffle, respectively. By selecting "Shuffle list", we complete the draw. We are then prompted to save the computed results locally, in a text file. Full information about the stored files containing the draw results appears and the procedures terminate (see Figure 3).

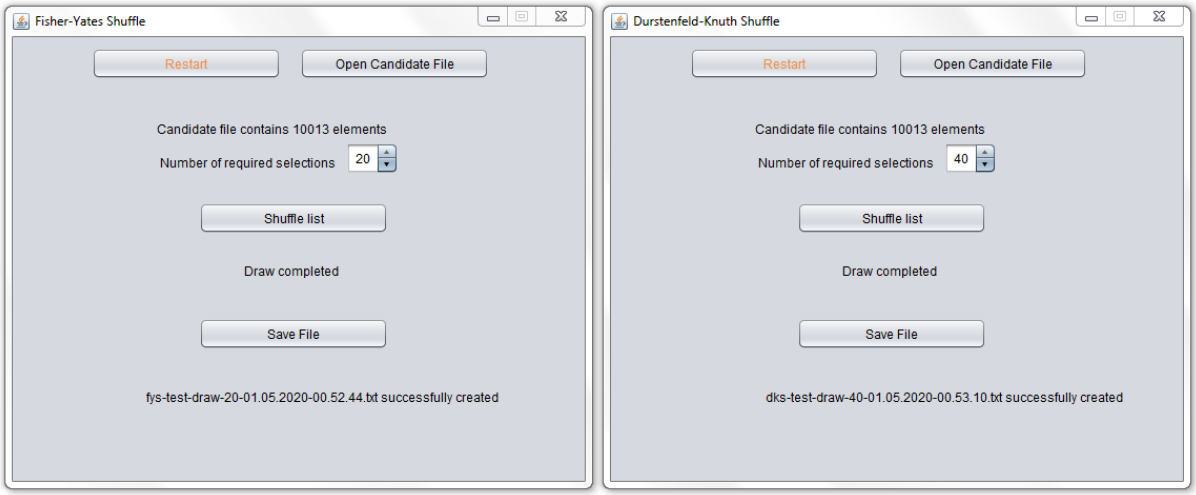

Figure 3: Completion of electronic draws by providing the list of candidates via a file

2.2. Typing in the candidates

By choosing to perform the electronic draw by typing in the candidates, a new pop-up window appears, as depicted in Figure 4. As before, the window on the left of Figure 4 can be used for performing an electronic draw exploiting the Fisher-Yates shuffle, while the window on the right part of Figure 2 can be used to perform an electronic draw exploiting the Durstenfeld-Knuth shuffle. Up to 50 candidates can be typed in, one per line. Next to each line, options to remove current line, add new line or complete the list are provided.

Technium Vol. 2, Issue 3 pp.43-48 (2020)
ISSN: 2668-778X
www.techniumscience.com

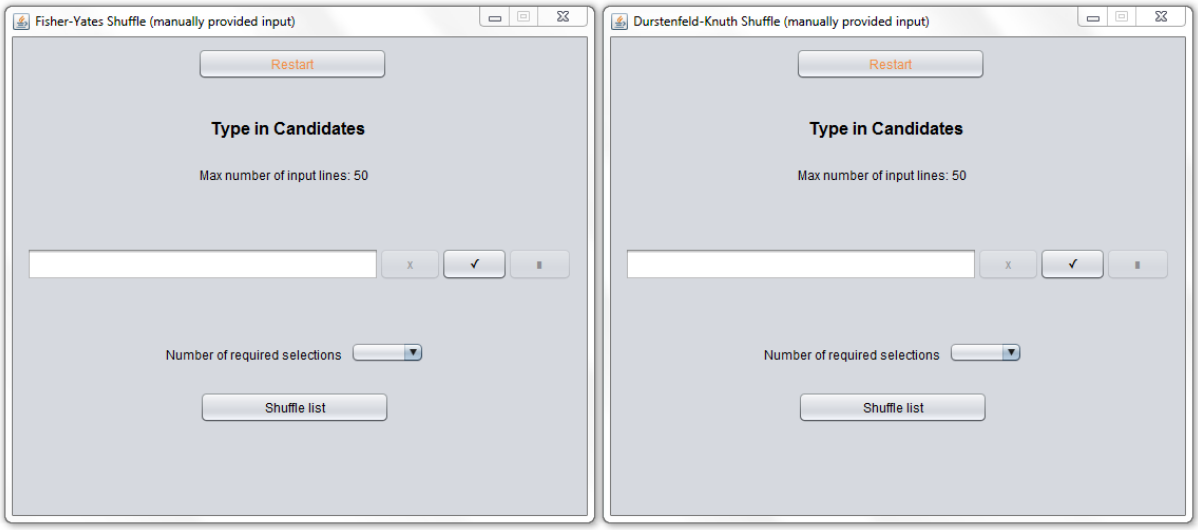

Figure 4: Performing an electronic draw by typing in candidates

For the purpose of demonstrating these application components, we type in 7 candidates, i.e., candidate1 to candidate7. We finalize the lists (by pressing the small box next to the last line) and then request the random selection of two groups of 4 and 5 members to be performed via Fisher-Yates shuffle and Durstenfeld-Knuth shuffle, respectively. By selecting "Shuffle list", we complete the draw. We are then prompted to save the computed results locally, in a text file. Full information about the stored files containing the draw results appears and the procedures terminate (see Figure 5).

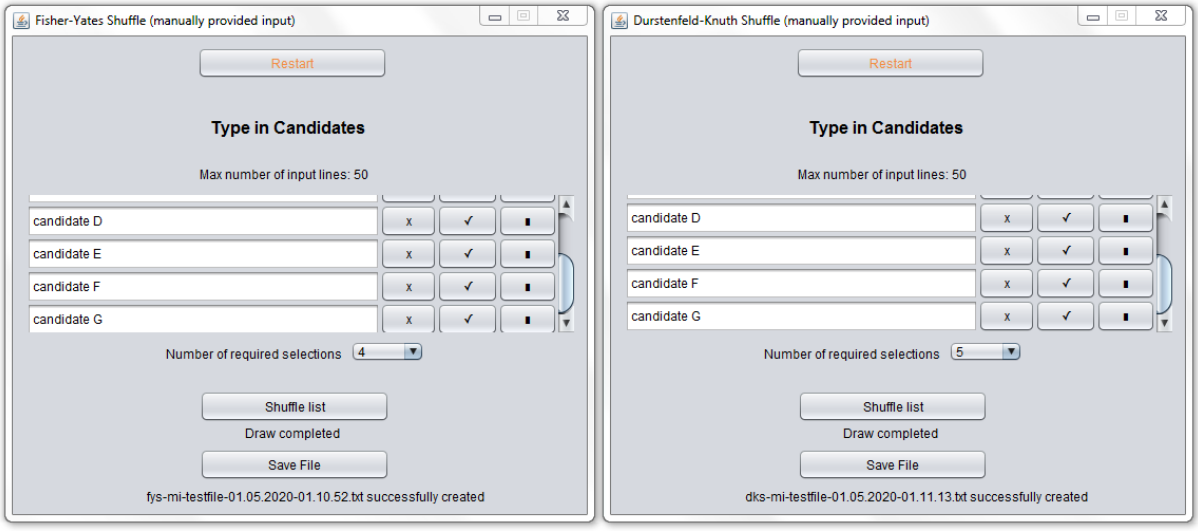

Figure 5: Completion of electronic draws by typing in candidates

2.3. Running "Draws" via cmd

By selecting to run "Draws" via the command prompt environment, a new pop-up window appears, as depicted in Figure 6. As before, the window on the left of Figure 6 shows an electronic draw performed via the Fisher-Yates shuffle, while the window on the right part of Figure 6 shows an electronic draw performed via the Durstenfeld-Knuth shuffle. Following the instructions on screen, we type in the candidates and END the list. Then, we are asked to submit the required number of random selections. Draws complete and the computed random selections appear on screen. An option for exiting "Draws" or proceeding with another draw is provided.

Technium Vol. 2, Issue 3 pp.43-48 (2020)
ISSN: 2668-778X
www.techniumscience.com

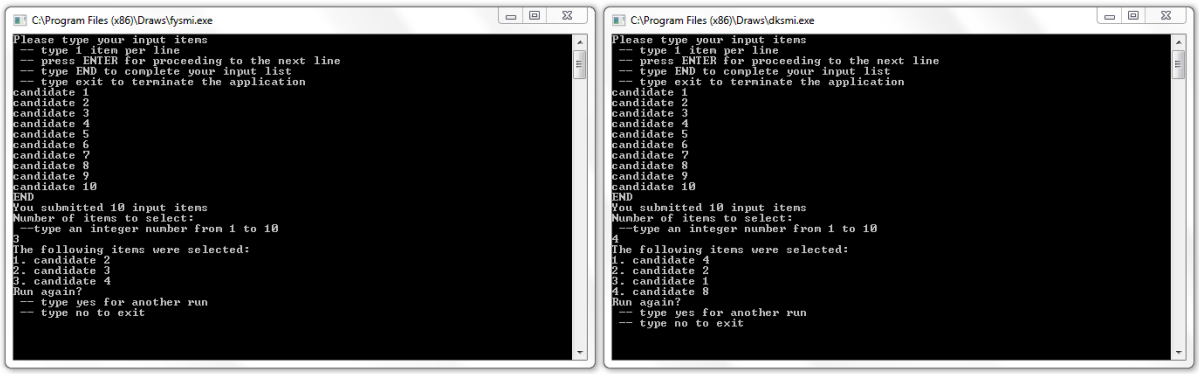

Figure 6: Performing an electronic draw by typing in candidates

3. Development framework

"Draws" has been built on Java via the Apache Net Beans IDE 11.3 (https://netbeans.apache.org/). Java Swing (https://netbeans.org/features/java-on-client/swing.html), a GUI (Graphical User Interface) widget toolkit for Java has been used for the implementation of the "Draws" user interface. Launch4j 3.12 (https://sourceforge.net/projects/launch4j/files/launch4j-3/3.12/) has been used for the generation of executable files. Inno Setup version 6.0.4 (https://jrsoftware.org/isinfo.php) has been used the creation of script-based installers.

For developing "Draws", we used a DELL Inspiron 7559 laptop running Windows 10 Pro 64-bit on an Intel (R) Core(TM) i7-6700HQ CPU @ 2.60GHz (8CPUs) with 8GB RAM. For testing "Draws", we used a Lenovo 80MJ laptop running Windows 10 Pro 64-bit on a Intel(R) Celeron(R) CPU N2840 @ 2.16 GHz (2CPUs) with 4GB RAM. We also used the free version of REPL.IT collaborative online IDE (https://repl.it/) which allows using only a browser for building applications online without requiring any additional, local, software installation.

"Draws" installer is publicly available to download at https://bit.ly/2zM9J8M.

4. Theoretical framework

As depicted in Figure 7, "Draws" receives as input a list of candidates and the number of required random selections. It shuffles this list, i.e., it produces a truly random and unbiased permutation of the input list. It computes a prefix of the shuffled list of size equal to the number of required selections. Then, it outputs the computed prefix as the result of the draw.

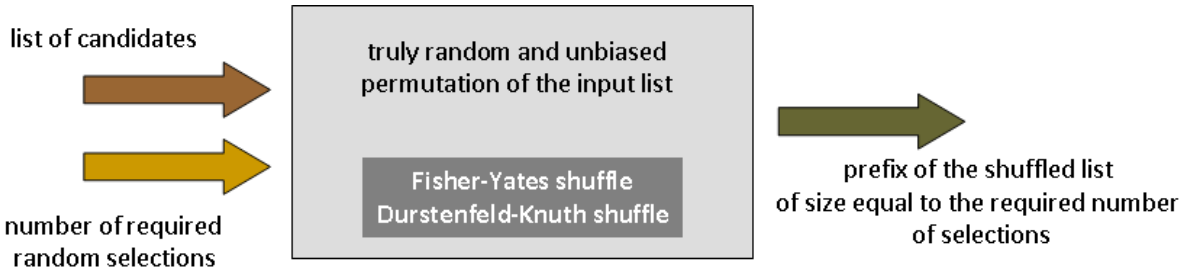

Figure 7: The algorithms behind "Draws"

The focal point for guaranteeing the integrity of the draw is to guarantee that the computed permutation of the input list is truly random and cannot be biased. To do so, we utilized two efficient algorithms perform shuffling of given lists, namely Fisher–Yates shuffle [2, 3] and Durstenfeld-Knuth shuffle [1, 3]. Ronald Fisher and Frank Yates first presented an algorithm for computing – via pencil and paper – truly random and unbiased permutations of given finite sequences in their book "Statistical tables for biological, agricultural and medical research" [2], in 1938. Then, in 1964, Richard Durstenfeld in 1964 [1] suggested a "computerized" version of the Fisher-Yates shuffle, which improves the previous version via a small but important implementation trick. Donald E. Knuth

later popularized these shuffling algorithms in his legendary book "The Art of Computer Programming" [3].

Fisher-Yates method randomly shuffles the elements of a given input list. To do so, it maintains two lists and iteratively removes items from the input list and inserts them into the output list. The output list is initially empty and gets populated as elements from the input list – one at a time - are randomly selected to move. Moved elements are placed on the output list, starting form its end and proceeding towards its start. The algorithm ends when all items of the input list have been moved to the output list. The output list contains a truly random and unbiased permutation of the elements of the input list provided that, at each step of the process, the selection of the element to move was made in a truly random and unbiased way. Fisher and Yates [2] provide extensive evidence on how the selection of the element to move at each step is indeed obtained in a truly random and unbiased manner. However, iteratively scanning the input list for locating the randomly selected item to move implies a time complexity of $O(N^2)$, where N denotes size of the input list. Durstenfeld-Knuth shuffling algorithm, on the other hand, avoids this drawback as well as the overhead of using a second (output) list, by implementing a clever approach for shuffling the elements of the input list in-place. To do so, it starts from the last element of the input list and swaps it with a randomly selected element (including the last element itself). Swapped elements get final positions and are never visited again. The process is repeated for the remaining – not yet swapped – elements of the input list. In this way, Durstenfeld-Knuth shuffling algorithm has a decreased (linear) O(N) time complexity, which can be of significant practical importance in cases of highly populated input lists. As far as our application is concerned, given the relatively short input lists, the potential impact of the theoretical complexity bounds remains unrevealed in practice. Using simple probabilistic analysis, it can be observed that the probability that an arbitrary not-yet-swapped element is moved to a not-yet-visited position remains 1/N throughout the process. This implies that all N! permutations of the elements of the input list are equally likely, thus yielding the correctness of the Durstenfeld-Knuth shuffling algorithm.

5. Concluding remarks

We designed and implemented "Draws", a simple, user-friendly application for conducting small-scale electronic draws in the context of administrative procedures requiring the formation of administrative bodies by drawing candidates out of a predefined pool in a truly random and unbiased manner. The application has been built on a theoretically sound basis and can be used by non-experts without the need for sophisticated equipment or extra financial cost.

Collecting usage statistics and user feedback, extending the use of graphics and implementing additional functionalities fall within our future plans for "Draws". Ongoing work addresses the improvement and parameterization of the format of the output file as well as the development of advanced functionalities for the input and output files (e.g., automatic check for inconsistencies, lock option).

"Draws" is the result of fruitfully mixing computer science and technology for assisting society. Towards the objective of simpler, less time-consuming and more transparent administrative procedures, "Draws" can, perhaps, make a useful tool.

References

[1] R. DURSTENFELD: Algorithm 235: Random permutation. *Communications of the ACM*, **7** (7), 420 (1964)

[2] R. A. FISHER, F. YATES: Statistical tables for biological, agricultural and medical research (3rd edn). London, UK, 1938.

[3] D. E. KNUTH: Seminumerical algorithms" in The Art of Computer Programming (vol. 2). Reading, Massachusetts, USA, 1969.