

A Look at Randomness in Microsoft Windows Solitaire, or, Using Electronic Games of Chance for Statistics Projects

KEYWORDS:

Teaching;
Computer games;
Random numbers;
Classroom projects.

Ronald D Fricker Jr

RAND, California, USA.
e-mail: Ron_Fricker@rand.org

Summary

The popular computer game Solitaire is used as a vehicle for investigating random number generation. This can form a very good basis for classroom projects.

◆ INTRODUCTION ◆

ELECTRONIC games pervade the computer world and users tacitly assume that these games accurately mimic their real-world counterparts. But is such an assumption well founded? Electronic games of chance, such as poker or solitaire, must be based on pseudo-random number generators, of which some are better than others. Besides the random number generators the games are also based on other encoded rules and mathematics. If the emphasis during game program development was on clever graphics and not on accurate programming and good probability theory, the result could be a game that looks pretty but is a poor representation of the true game.

Because these electronic games are now distributed throughout the home and classroom they can be used as a rich source of data for students' statistics projects and papers. The idea is for students to collect data from the program and analyse whether the electronic game is an accurate representation of the real game. This can be done at many levels of sophistication; here I use one simple, illustrative example.

Consider the ubiquitous solitaire game distributed with the Microsoft Windows operating system. Virtually every IBM-compatible computer comes with this game, making for millions of copies installed around the world. The beauty of the electronic version, from a player's point of view, is that all of the drudgery of shuffling and laying out the cards is eliminated. Upon completion of a game one simply chooses "deal" from a pull-down menu and, presto, the cards are shuffled and neatly laid out for new play.

The type of solitaire distributed with Windows is known

in card playing circles as *Klondike* solitaire. There are many references for those unfamiliar with the game; see for example Arnold (1988), or *Official Rules of Card Games* by The United States Playing Card Company, or almost any general text about card games. For the purposes of this discussion and analysis the details of Klondike solitaire are not particularly important, it is only necessary to know that before play the deck is thoroughly shuffled and then laid out in the pattern as shown in figure 1 (where the cards laying face-up are a random outcome of the shuffle). The object of the game is to move all the cards from the stock and tableau up to the four places at the top, stacking them by suit in the order Ace, 2, ..., 10, J, Q, K.

◆ DISCUSSION ◆

My interest in whether Windows solitaire works properly arose one day after playing *many* solitaire games in a row. A pattern seemed to exist: if I won a game I was more likely to win the next game and if I lost a game I was more likely to lose the next game. In fact the pattern was stronger than that, because if I won with a very high score then the next game was likely to be won with a high score and if I lost very badly then the next time I would lose badly.

My first inclination was to explain these streaks by rationalising that I was paying closer attention to the game for a time and then I was getting sloppy for a time. But careful attention to my play did not seem to change the winning and losing streaks. I next tried to rationalise the outcomes by appealing to the fact that streaks do occur by chance, that randomness is "lumpy", but the statistician in me wanted to check this. With a compiled computer program like solitaire and my novice computer programming skills, the question was how?

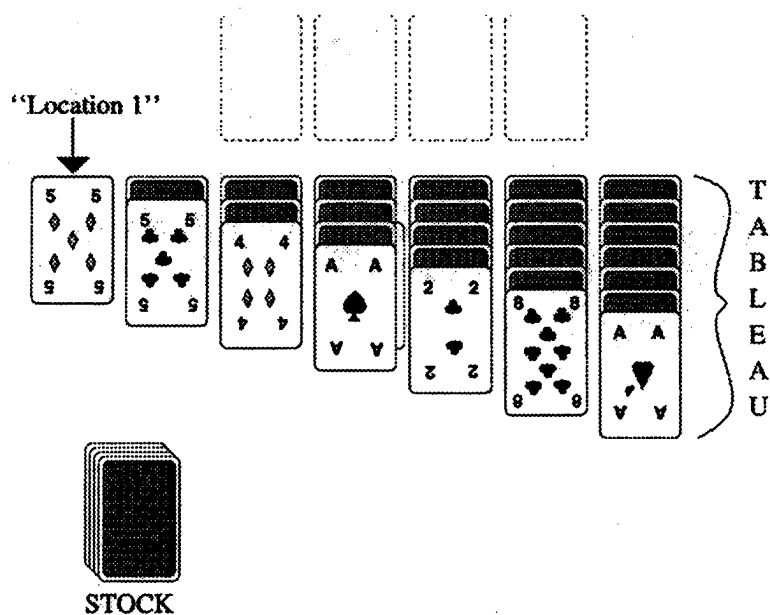


Fig 1. Initial layout of the cards for solitaire

Because the program was compiled I had no way of looking at the actual computer code in order to analyse the random number generator and the coded game rules. I could look at and manually record the cards as they were dealt on the monitor, but even if I was inclined to the tedium of manually recording the entire deck of cards many times, it was impossible to do for this game. This is because the entire deck is only revealed if the game is won; if it is lost there is no way of knowing the locations of the unplayed cards. Thus only a subset of the deck for a given game was available. Since I was not interested in mounting a detailed data collection effort nor an overly sophisticated statistical analysis I had to find another approach.

◆ ANALYSIS AND RESULTS ◆

The approach I settled on was to invert the investigation. Instead of trying to determine whether the game was “proper” (i.e., an accurate representation of the real world), I decided to see if there was a smaller subset of the data from which I could check whether the game was improper in some particular way. This is a considerably easier task, though one in which failure to find a problem is not a guarantee that the game is proper.

Since my experience playing many games in a row led me to question whether the deck was well shuffled between games, I narrowed my attention to looking for a trend between deals. One way to do this is to focus only on the first face-up card that is dealt to the far left in the tableau (“location 1” in figure 1). If the deck is properly

shuffled then the card that appears in this spot on one deal should only re-appear in location 1 on the next deal 1 out of 52 times on average.

It is easy to deal the deck in Windows solitaire, so it is relatively simple to do the following operation many times: deal the deck (from the pull-down menu), note the card that appears in location 1, re-deal the deck (from the pull-down menu again), and check to see if the new card in location 1 matches the old card. In fact, if this is done 104 times one should expect to see two matches on average. (The number of matches X is a binomial random variable with probability of success $p=1/52$ and number of trials $n = 104$, so $E(X) = np = 2$.) I did ten sets of 104 comparisons, re-starting the game before each set to re-seed the random number. (That the game re-seeds each time it is started is an assumption on my part based on the observation that each time the game is started it re-deals the deck. If the game is properly random then collecting data in sets is immaterial. But if there is some sort of seed-dependent error, then using different seeds should help detect this).

The results are shown in Table 1. It doesn't take a fancy analysis to see that something appears wrong. Out of 1,040 paired comparisons there were a total of 114 matches - almost 11%. This is significantly larger than the expected 20 matches. To see how far off these data are consider the best set, the third: If X is binomial with $n = 104$ and $p = 1/52$ then $P(X \geq 5) = 0.051$. So, a standard hypothesis test (i.e., using a p -value of 0.05) nearly rejects the best set of the 10. Aggregating the 10 sets so that $n = 1,040$ gives $P(X \geq 114) = 0$ (to very many decimal places!).

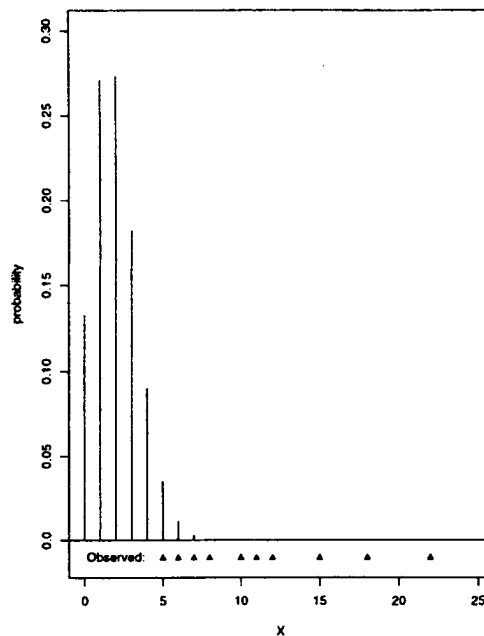


Fig 2. Plot of the probability function of a $B(104,1/52)$ distribution along with the observed values from the 10 sets of data. The data were collected from the solitaire game included with Windows version 3.1.

Set	Number of Matches	Set	Number of Matches
1	8	6	12
2	7	7	11
3	5	8	15
4	10	9	18
5	6	10	22

Table 1. Solitaire data. Each “set” consisted of 104 pairs of deals. A “match” occurred if the card in location 1 of both deals was the same.

A picture is worth a thousand words. Figure 2 plots the $B(104,1/52)$ probability function. The height of a vertical line represents the probability that the number of matches in a set equals x . From the plot it is clear that the probability that X achieves a value greater than 8 is essentially zero. Most of the observed values (the black triangles) are improbably large compared to the $B(104, 1/52)$ distribution.

It is interesting to contrast this result with comparisons made with data taken only when the solitaire program is first started. That is, in the previous comparisons the program was started for the first deal and then subsequent decks were dealt using the “deal” option from the pull-down menu. An alternative is to boot the program by double clicking on the solitaire icon, record the card in location 1, quit the program, reboot it, compare the card in location 1 with the previously recorded card, and then

quit again. I did this operation 208 times (it’s much more time consuming than the previous method) and garnered only 4 matches, exactly the expected amount.

◆ CONCLUSION ◆

The simple statistical analysis here gives support to my intuition that “shuffles” in sequential games (using “deal” from the pull-down menu) do not randomise the electronic deck of cards in this version of Windows solitaire. In comparison, there is no evidence here that the initial shuffle when the program is first booted is not random.

For statistics instructors and courses, this article has illustrated how electronic games can be used as a source of data for student projects, where students analyse whether the electronic version is an accurate representation of the real thing. The strength of this type of project is that it is in many ways representative of real-world problems: the game is a complex system from which the student must specify a question, collect data, and then define and conduct the analysis. Windows solitaire is certainly a prime candidate for additional analysis, and there is a plethora of other electronic games of chance that could be scrutinised.

Reference

Arnold P. (1988). *The Book of Card Games*, London: Christopher Helm.