EVERYDAY I’M SHUFFLING: A PROBABILISTIC ANALYSIS OF CARD SHUFFLING

RILEY NOLAND

1. INTRODUCTION

Link to video of code: https://www.youtube.com/watch?v=VLzroRVPgHk&feature=youtu.be

What makes a card shuffle random? This seemingly unassuming question has captivated many mathematicians throughout the ages and given rise to the development of new and sophisticated areas of mathematics that have seen widespread use. In this project, we investigate the efficiency of different card shuffles at creating perfectly random card orderings. We explore a variety of different shuffles, ranging from the simple and intuitive insertion shuffle to the novel and almost completely unexplored smoosh shuffle.

We analyse the complexities of these different shuffles with a combination of theory, heuristic, and computational simulation to make a compelling case for the efficacy of different shuffles. Along the way, we make several notable contributions in the form of interesting explanations for known results as well as the development of novel simulation models for the smoosh shuffle, allowing us to monitor its randomness via application of the Kullback-Liebler Divergence Test. Along the way, we also explore some intriguing properties of common shuffles that allow us to exploit mathematical structure that gives rise to amazing magic tricks.

2. BACKGROUND AND SETUP

In this paper, we model a standard deck of cards as a sequence of numbers $\pi = [1, 2, \ldots, 52]$ where each number represents a card in its canonical lexicographic order (see Figure 1). We use $\pi[i]$ to denote the $i$-th card in the ordering. For example, $\pi[3]$ is the 3 of Hearts.

2.1. Card Shuffling. We define a shuffle abstractly as any algorithmic procedure that takes a list of cards as input and applies some steps to modify the order. More formally, a shuffle algorithm takes as input an ordering $\pi$ of the cards, and outputs a new ordering
2 RILEY NOLAND

Figure 1. A standard deck of cards in canonical order.

\( \pi' \). The algorithm is often non-deterministic, resulting in different output orderings on different runs.

2.2. Randomness of Shuffles. We are often interested in understanding how random different shuffles are. From the perspective of a casino or game player, it is really important that shuffling the cards doesn’t result in orderings that are predictable or can be exploited to some unfair advantage.

More concretely, we can imagine each shuffle algorithm as a probabilistic procedure that takes as input the canonical ordering \( \overset{\sim}{\pi} \) and outputs orderings according to a random variable \( X \) representing the ordering of the cards after the shuffle. Then, \( P(X = \pi) \) is the probability of the specific ordering \( \pi \) being output by the shuffling algorithm.

For a standard deck of cards, there are 52! possible unique orderings. For a shuffle to perfectly produce a random deck of cards, we want any one of these 52! arrangements to be equally likely after the shuffle. In other words, we want \( P(X = \pi) = 1/52! \) for any ordering \( \pi \).

In reality, there is often a trade off between randomness and efficiency of a shuffle. For example, there are easy shuffles that are provably random, but would require an infinite number of iterations to produce truly random orderings. In this work, we will explore this intriguing dilemma, investigating which shuffles are both efficient and highly random.

3. A Simple Case Study: Random Insertion Shuffle

Before diving into more complex, real world shuffles, it is instructive to start our analysis in a simpler setting. In this section, we consider a simple variant of a popular shuffle that lends itself to mathematical analysis and serves as an interesting introduction.

In this shuffle, we start with a deck of cards in standard order \( \overset{\sim}{\pi} \) and repeat the following: we remove the top card from the deck and insert it into any of the 52 positions in the deck with equal probability. We repeat this procedure until the card that was originally on the bottom of the deck rises to the very top (and we repeat once more after that). This algorithm is described in Algorithm [1].
Algorithm 1 Random Insertion Shuffle

Inputs:
- An ordering $\pi$ of the deck of cards of length $n$.

Algorithm:
1. While original bottom card is not at top:
   i) Remove top card from deck.
   ii) Insert the card randomly into one of the 52 possible locations in the deck.
2. Insert the top card randomly into deck one last time.

3.1. Randomness. It is easy to see that once the bottom card has risen to the top of the deck, the entire deck is in a completely random order. This is because each card was inserted into a position below the original bottom card with equal probability. Therefore, all possible orderings are equally likely.

3.2. Efficiency. Although the shuffle results in a truly random deck, we have no sense of how long it takes on average. To analyse this, we consider the probabilities.

Whenever each step of this procedure is completed, note that the bottom card can only either stay in its position, or move up one position in the deck. Initially, when the bottom card is at the bottom, the probability of it shifting up in the deck (by placing the top card below it) is $1/52$. Once this happens, the probability of it shifting up in a step is now $2/52$, because the top card can be placed in one of the two spaces below the original bottom card.

Let $X_i$ be the random variable representing the number of trials till the $i$-th success, where a success is the event that original bottom card moves up a position in the deck. We can see that each $X_i \sim \text{Geom}(i/52)$.

Let $X$ be the number of steps till the bottom card reaches the top (plus one final insertion). Then clearly $X = 1 + \sum_{i=1}^{52} X_i$, and therefore, the expected value of $X$ is:

$$
\mathbb{E}[X] = 1 + \sum_{i=1}^{52} \mathbb{E}[X_i] = 1 + \sum_{i=1}^{52} \frac{52}{i} \approx 236.
$$

This tells us that, on average, it takes 236 iterations of this algorithm to randomly shuffle a deck of cards. We can confirm this theoretical number by simulating the shuffle.
Around 236 iterations of Random Insertion Shuffle needed to get random deck over 10,000 trials and observing the distribution of $X$. Figure 2 shows a histogram of the distribution of $X$ over 10,000 trials.

4. Understanding the Riffle Shuffle

Although instructive, the Random Insertion Shuffle is not an interesting model for shuffling in the real world. A more sophisticated shuffle to analyse is the Riffle Shuffle which is ubiquitous in real life, albeit much harder to analyze.

A Riffle Shuffle consists of splitting an ordering of the cards $\pi$ into two piles of roughly equal size: $\pi_R$ and $\pi_L$. Cards are then alternatively interleaved from $\pi_R$ and $\pi_L$ (see Figure 3) to generate a new ordered deck $\pi'$. Cards are typically cut and then interleaved multiple times in any given Riffle Shuffle.

When the riffle shuffle is performed in the real world, the deck $\pi$ is rarely split into two exact equal halves. Similarly, cards from the two piles are not usually interleaved perfectly from $\pi_R$ and $\pi_L$, often falling in clumps.

To simulate these effects in code, we need a mathematical model to imitate this behaviour. Fortunately, in 1995, Claude Shannon and some collaborators were investigating this exact shuffle and came up with a compelling mathematical representation.
Figure 3. Interleaving cards from $\pi_R$ and $\pi_L$

called the Gilbert-Shannon-Reeds model. This model has been experimentally verified to be an accurate representation of how people riffle shuffle in real life [Bayer and Diaconis, 1992].

4.1. **Gilbert-Shannon-Reeds model.** Consider an input deck of ordered cards $\pi$. Let $K$ be the random variable modelling the index at which the deck is cut. Then $K \sim \text{Bin}(52, 0.5)$; that is, $K$ is distributed as a discrete normal distribution centered at the halfway point of the deck.

We cut the deck $\pi$ at index $K$ to give two heaps $\pi_L$, of length $K$, and $\pi_R$ of length $52 - K$. The two decks are riffled in such a way that cards drop from the left of right heap with probability proportional to the number of remaining cards in each heap [Bayer and Diaconis, 1992]. Therefore, if there are $m$ cards remaining in the left heap and $n$ cards remaining in the right heap, then the probability that the next card will drop from the left heap is $m/(m + n)$ and the probability that the next card will drop from the right heap is $n/(m + n)$. This is continued until both heaps are empty, resulting in a new deck with ordering $\pi'$.

4.2. **Randomness and Efficiency.** While theoretically interesting, the Gilbert-Shannon-Reeds model is much harder to analyze in generality than the simple insertion shuffle.

Let us define $\text{RiffleShuffle}_k(\pi)$ to be the algorithm that takes the standard ordered deck of cards as input and applies the riffle shuffle $k$ times. Ideally, we want to
show that for some fixed number $M$, if $k \geq M$, then riffle shuffling $k$ times results in any ordering $\pi$ with equal probability. In other words, we want:

$$P(\text{RiffleShuffle}_k(\hat{\pi}) = \pi) = \frac{1}{52!}.$$ 

Unfortunately, it is incredibly difficult to verify this theoretically or experimentally because there are too many possible orderings to consider. Instead, we can consider some heuristics and statistics about the deck that are necessary but not sufficient for true randomness.

4.3. **Heuristic.** Let’s start with a simple heuristic analysis by considering the probability of the initial bottom card staying at the bottom of the deck after $k$ riffle shuffles.

During the first riffle shuffle, there is a $\frac{1}{2}$ chance that the bottom card is on the bottom of the new ordering. Similarly, during the second riffle, there is again a $\frac{1}{2}$ chance that the original bottom card stays at the bottom. In general, after $k$ riffle shuffles, the probability of the bottom card staying at the bottom is $\frac{1}{2^k}$. Compare this with any random ordering of the 52 cards. The probability that the original bottom card is at the bottom of any random ordering is $\frac{1}{52}$. Thus, for our riffle shuffle to be random, we at least need the probability of the bottom card staying at the bottom to be about $\frac{1}{2^{52}}$. This tells us that we need at least $k \geq 6$ to generate random decks.

4.4. **Simulation.** The Gilbert-Shannon-Reeds model for Riffle Shuffling $\hat{\pi}$ some $k$ number of times can be simulated in code. Let $\pi$ be the resulting ordering after the standard deck $\hat{\pi}$ shuffled by the Gilbert-Shannon-Reeds model. As discussed earlier, there are $52!$ possible permutations of $\pi$. Directly measuring how effective the Gilbert-Shannon-Reeds model is at producing a $\pi$ with a completely random ordering is not feasible. This is because doing so would require seeing if each possible permutation of $\pi$ appears equally likely among those orderings produced by the Gilbert-Shannon-Reeds model.

Given this limitation, we can instead measure other necessary statistics for randomness. As an example, consider the distribution of positions that the original top card moves to after riffle shuffling $k$ times. For a truly random shuffle, the top card should move to every position in the deck with equal probability.

To experimentally analyse this statistic, we can track the distribution of positions that the original top card ends up in over 5000 trials and compare it to the uniform distribution. To measure this successfully, we need a notion of “distance” between two distributions so we can quantify how far the distributions are from the uniform distribution.
4.5. **Kullback-Leibler Divergence.** A common and powerful notion for measuring the similarity between two distributions is the idea of Kullback-Leibler (KL) Divergence.

\[
KL(P||Q) = \sum_x p(x) \log \frac{p(x)}{q(x)}.
\]

The Kullback-Leibler Divergence of \(P\) and \(Q\) measures the similarity between distributions \(P\) and \(Q\). A lower value means the distributions are more similar.

We can track how many shuffles are necessary to at least randomize the position of the original top card \(\hat{\pi}[1]\). Within the Gilbert-Shannon-Reeds model simulation, we track the distribution of positions of the top card as a function of how many times \(k\) the riffle shuffle is carried out. For each \(k\), we measure the KL Divergence between the distribution of top card positions after \(k\) riffle shuffles and the uniform distribution.

In fact, we can run this analysis for the distribution of positions for all possible cards in the original deck, tracking the KL divergence between the distribution of their
Figure 5. The first python list represents a deck of cards—originally ordered from 0, 1, ..., 51—that was riffle shuffled 3 times. The bottom list contains 9 rising sequences, one of which contains only card positions and the uniform distribution. Figure 4 shows these results, which show an interesting finding. After around 6 riffle shuffles, all the card positions are randomised, as can be seen by the low KL divergence in Figure 4.

5. Real World Application: The Premo Magic Trick

Although we have explored the randomness of the riffle shuffle, showing that it is randomised after about 6 shuffles, it is interesting to study the structure of the deck when less than 6 riffle shuffles are used.

The key invariant of riffle shuffles is the notion of rising sequences. A rising sequence is a subset of a permutation of some deck of cards such that successive card values are in order. In Figure 4 we note that there are 9 rising sequences within the list of number above it (see Figure 5).

There is an interesting structural phenomena whereby the number of rising sequences within an ordering \( \pi \) doubles after every riffle shuffle. In the original deck \( \hat{\pi} \), there is one rising sequence. After riffle shuffling once, the resulting deck will contain 2 rising sequences. After another riffle shuffle, there will almost always be 4 rising sequences.

Using the phenomena of rising sequences, one can perform a great magic trick! A performer can ask a member of the audience to riffle shuffle an original sorted deck \( \hat{\pi} \) three times. The audience member can then pick the card on top, memorise it, and insert it anywhere into the middle of the deck. The magician then spreads the deck out on the table, and after a little pondering, immediately points to the audience’s card.

In reality, the magician is computing the rising sequences of the deck when laid out on the table. After three riffle shuffles, the deck will have 8 rising sequences. When the audience member picks the top card and moves it into the middle of the deck, the rising sequence structure of the deck is broken, and this card often creates a ninth
rising sequence consisting of only this one card. With enough practice, the magician can easily identify this lone rising sequence and magically declare the audience’s chosen card.

In our code, the Premo() function takes as a parameter a deck of cards shuffled by my riffle shuffle function. It then places the top card (call it $card_0$) randomly within the shuffled deck, and splits the deck into multiple lists—where each list represents a rising sequence. $Card_0$ is easily spotted within the lists of rising sequences because it creates a ninth rising sequence whose only card is $card_0$.

6. Going Beyond: The Smoosh Shuffle

An even more sophisticated shuffle to analyze than the Riffle Shuffle is the Smoosh Shuffle. Unlike the Riffle Shuffle, the smoosh shuffle has not been extensively studied, providing a rich field of research to explore. In this paper, we used aspects of Persi Diaconis’ and Soumik Pal’s research on smoosh shuffling in creating the python simulation [Diaconis and Pal, 2017].

Unlike the riffle shuffle, there doesn’t exist any provable algorithm for modeling the smoosh shuffle. To that end, I conducted personal research to develop my own model.

6.1. Modeling the Smoosh Shuffle. A smoosh shuffle consists of placing $\hat{\pi}$ in the center of some flat surface, and ”smooshing” the cards within $\hat{\pi}$ in a non-deterministic manner with one’s hands across the surface (See Figure 6).
Algorithm 2 The Two Dimensional Smoosh Shuffle

Inputs:
- An ordering $\pi$ of the deck of cards of length $n$.

Algorithm:

1. $\text{ShuffledDeck}$ initialized as empty list
2. Place $\pi$ in the center of a 15.75 inches x 24.5 inches table at position 7.35 inches x 10.5 inches. (*Note: the table can be thought of as a matrix of 7 x 7 where each cell has the dimensions of a single card, such that the center cell is 3 x 3).
3. Note this starting position of cell 3 x 3 in the matrix as $\text{StartingPosition}[\text{row}][\text{column}]$ where row = 3 and column = 3.
4. Place a single hand on top of this deck of cards. (*NOTE: your hand cannot be spread out such that it is larger than the dimensions of a single card. If your hand is spread out larger than dimensions of a single card, you will add additional parameters to randomizing cards via the smoosh shuffle that are not accounted for by The Two Dimensional Smoosh Shuffle)
5. For i in S number of steps/seconds:
   i) Uniformly choose one of four directions $D = \text{StartingPosition}[\text{row}+i][\text{column}], \text{StartingPosition}[\text{row}-i][\text{column}], \text{StartingPosition}[\text{row}][\text{column}+i], \text{StartingPosition}[\text{row}][\text{column}-i]$.
   ii) Check to make sure that direction $D$ is still within bounds of the 7 x 7 matrix (15.75 inches x 24.5 inches table).
   iii) If the direction $D$ is within bounds of the 7 x 7 matrix (15.75 inches x 24.5 inches table):
      a) For every j card in $\text{StartingPosition}[\text{row}][\text{column}]$ that single hand is currently on:
         1) If $P(X = 1)$ where $X \sim \text{Bernoulli}(0.5)$ then j-th card moves with hand to Direction $D$
         b) Update the position of single hand within 7 x 7 matrix to reflect hand moving to Direction $D$ on the i-th step.
   iv) else: end the i-th step
6. For each column $C$ in 7 x 7 matrix
   i) For each row $R$ in 7 x 7 matrix:
      i) For card j in matrix cell $[R][C]$:
         i) Append j-th card to $\text{ShuffledDeck}$
         ii) Reinitialize matrix cell $[R][C]$ to empty list ($[R][C]=[]$)
7. return $\text{ShuffledDeck}$

My algorithm The Two Dimensional Smoosh Shuffle mathematically models shuffling a deck of cards with the smoosh shuffle (see Algorithm 2). Let $\pi$ be $\tilde{\pi}$ shuffled by The Two Dimensional Smoosh Shuffle. Much like the Riffle Shuffle, directly measuring how
effective The Two Dimensional Smoosh Shuffle is at producing a $\hat{\pi}$ with a completely random ordering is not possible.

6.2. **Kullback-Leibler Divergence.** The same method used to quantify randomness for the Gilbert-Shannon-Reeds model was used for The Two Dimensional Smoosh Shuffle in order to attain 20 Kullback-Leibler Divergence tests. Importantly though, in The Two Dimensional Smoosh Shuffle we adjust how many seconds/number of steps the model’s ”hands” are smoosh shuffling by increments of 15. For example:

   The first Kullback-Leibler Divergence test’s parameter $P$ represents how many times $\hat{\pi}[1]$ appeared in every position of a deck of cards in $R$ samples after smoosh shuffling $\hat{\pi}$ for 15 seconds/steps. The second Kullback-Leibler Divergence test’s parameter $P$ represents how many times $\hat{\pi}[1]$ appeared in every position of a deck of cards in $R$ samples after smoosh shuffling $\hat{\pi}$ for 30 seconds/steps. The third Kullback-Leibler . . .

6.3. **Efficiency.** The Kullback-Leibler Divergence of $P$ and $Q$ was calculated and plotted for each of these 20 tests (see Figure 7). The Kullback-Leibler Divergence does not significantly move further towards 0 after smoosh shuffling for 100 seconds/steps. This suggests that at least 100 seconds/steps of smoosh shuffling are necessary to at least randomize the position of the top card.
References
