Shannon's Mind Reading Game

Himadri Mandal (BS2327)
Ayan Ghosh (BS2321)
Drishti Singhla (BS2325)
Siddhartha Bhattacharya (BS2345)
Aman Verma (BS2309)
Mrittika Giri (BS2332)

Advisor/Instructor
DR. ARNAB CHAKRABORTY
Associate Professor, Applied Statistics Unit

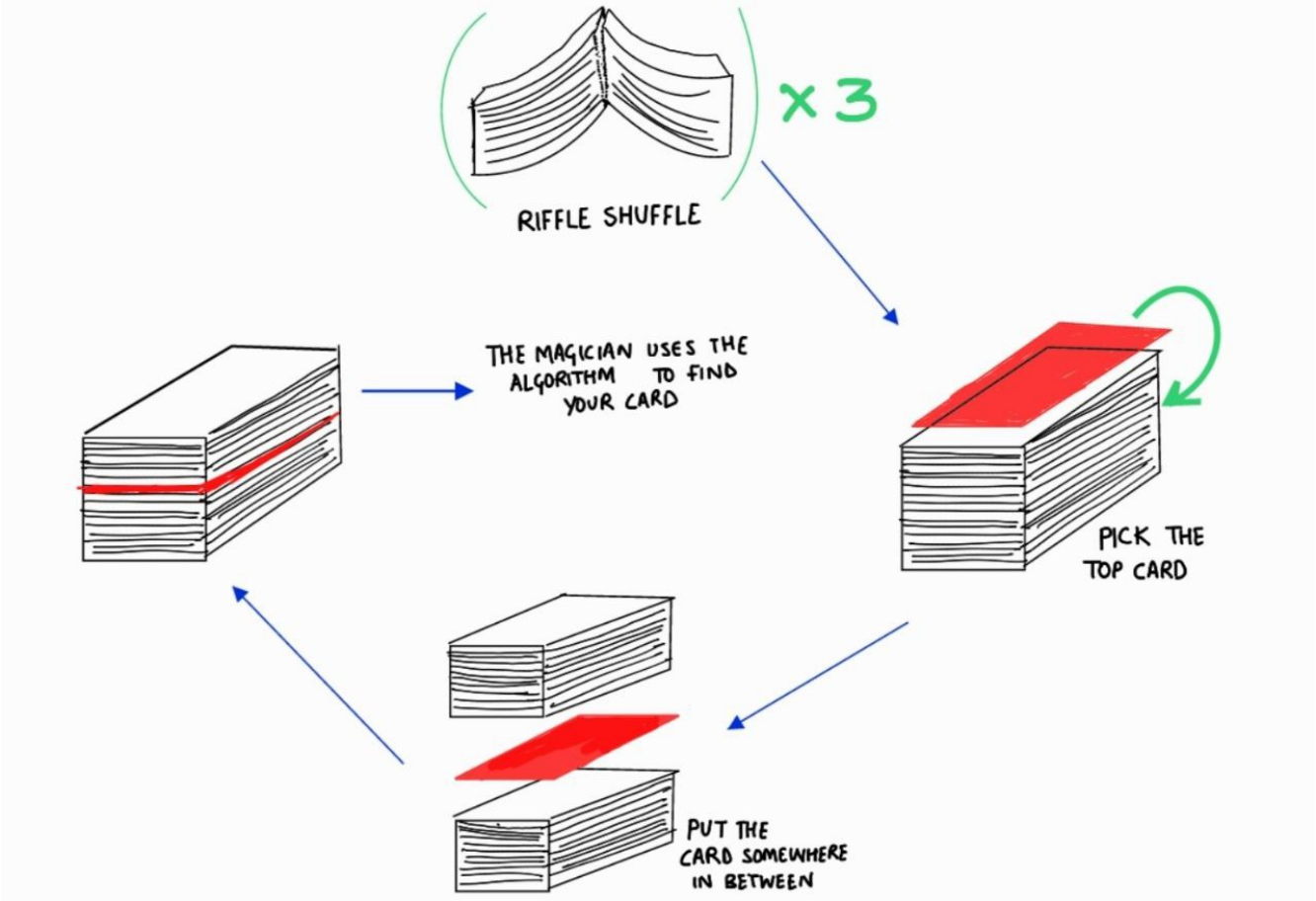
Shannon's Mind Reading Game

Abstract

Claude Shannon, famously known as the father of information theory, is credited with inventing this magic trick. We characterize the setup, and then implement the algorithm used by Claude Shannon. To do this, we appropriately choose the models which represent the transitions. This project empirically confirms the success of this trick.

1 Problem Statement

Magician and **User** play a game. The **Magician riffle shuffles** an ordinary deck thrice, and sends it to the **User**. **User** picks a card from the top, and then inserts it back into the deck randomly. The **User** sends the deck back to the **Magician**: who finds out the picked card.



2 Setup

- Pick a bijection
$$\Psi : \textbf{Deck} \to \{1, 2, \cdots, 52\}$$
We henceforth only talk about $\{1, 2, \cdots, 52\}$, as the algorithm used doesn't depend on the specific type of the card in consideration.
- **Riffle Shuffle:** A specific method of shuffling cards by dividing a deck into roughly equal halves, and then interweaving the two halves by successively interleaving the cards from each half, in a randomized order.
- Any arrangement of the $\Psi(\textbf{Deck})$ is a permutation of $[52]$. Consider any permutation σ, we define
$$\textbf{Index}(\sigma) = \{\sigma(1), \sigma(2), \cdots, \sigma(52)\}$$
- **Rising Sequence:** Given a permutation σ, $S_a = \{a, a+1, \cdots, a+k-1\} \subseteq \sigma(\Psi(\textbf{Deck}))$ is called a rising sequence of size k if
$$\sigma(a-1) \not< \sigma(a) < \sigma(a+1) < \cdots < \sigma(a+k-1) \not< \sigma(a+k)$$

3 Idea

The idea is to notice that the final deck has a unique Rising Sequence of size 1 pretty often.
To implement this, we would have to fix appropriate models for the steps involved: splitting the decks, interweaving them, randomly putting the card in at some index. The models we use are:

- **Deck splitting:** We split a deck 100 times and counted the size of the smaller deck. Then, we flipped a coin for each entry x and picked x or $52 - x$ randomly, in the final dataset, to ensure there are no **IID** issues. Atlast, we fit a normal distribution over this dataset to get:
$$\mathcal{N}(\mu, \sigma^2) = \mathcal{N}(25.78, 2.3046^2)$$
- **Interweaving them:** We maintain two arrays **Right**, **Left**. The probability that the next card falls from the **Right** is $\dfrac{|\textbf{Right}|}{|\textbf{Right}| + |\textbf{Left}|}$ and from the **Left** is $\dfrac{|\textbf{Left}|}{|\textbf{Right}| + |\textbf{Left}|}$. This is justified by drawing from domain knowledge that the weights of the hands are linearly related to the number of cards.
- **Randomly putting it in somewhere:** We use $\text{Binom}(52, \frac{1}{2})$, as humans are, probably, more likely to put the card somewhere close to the middle.

4 Implementation

The following algorithms were used in the project.

Algorithm 1 Riffle Shuffle the deck of cards

```
procedure RiffleShuffle(Cards)
    Deck := {}
    split := floor(rnorm(1, mean = 25.78, sd = 2.3046))
    LeftHand := Cards[1 : split]
    RightHand := Cards[split + 1 : 52]
    for i in {1, 2, ··· , 52} do
        hand := sample((L, R), prob = (size(LeftHand), size(RightHand)))   ▷ hand the Card falls from
        if hand == L then
            Deck ← LeftHand[1]
            pop(LeftHand)                                                  ▷ remove the added Card
        else
            Deck ← RightHand[1]
            pop(RightHand)
        end if
    end for
    return Deck
end procedure
```

Algorithm 2 Count number of Rising Sequences of size 1 and check if it's unique

```
procedure RisingSingleton(S)
    Index := {}
    for i in {1, 2, ··· 52} do
        Index[S[i]] = i                                                    ▷ Define the Index array.
    end for
    Sum := 0, SingletonCard := 0
    for i in {1, 2, ··· , 52} do
        if Index[i] ≥ Index[i + 1] and Index[i] ≤ Index[i − 1] then        ▷ Singleton Rising Sequence
            Sum = Sum + 1
            SingletonCard = i
        end if
    end for
    return (Sum == 1), SingletonCard
end procedure
```

5 Experimental Setup

The code simulates the magician's guessing by repeating the process 10,000 times.

- A deck of cards, numbered 1 to 52 in ascending order, is defined.
- The `RiffleShuffle` function is called three times, shuffling the original deck.
- The `Switch` function is used to place the top card somewhere within the deck.
- The `RisingSingleton` function identifies a potential unique rising sequence, and the count of unique singleton rising sequences is recorded.

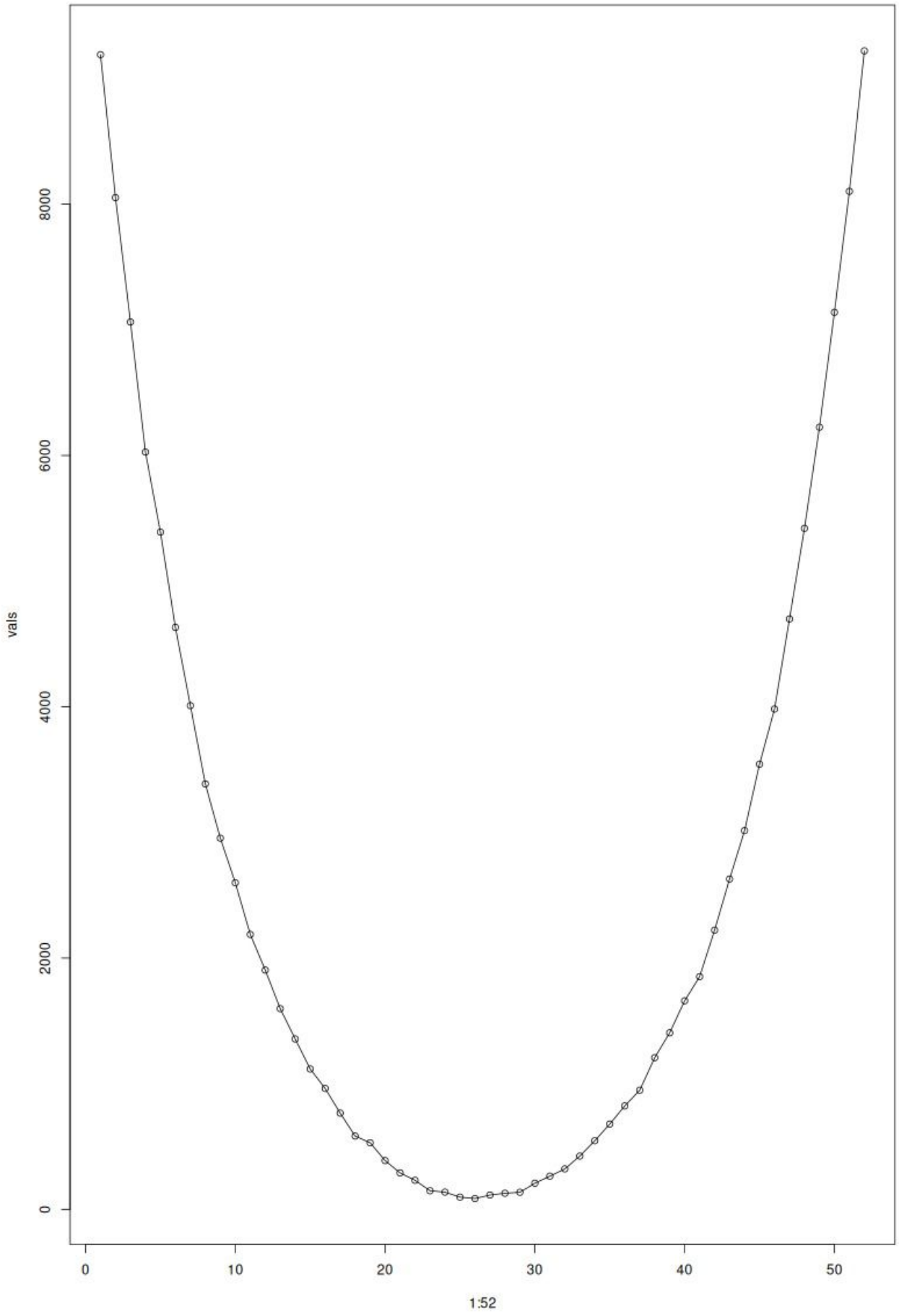
6 Results and Conclusion

The success percentage of $\boxed{\mathbf{91\%}}$, calculated as the ratio of unique single rising sequence, which match the picked card, to the total attempts, indicates the probability of identifying the top card. This percentage demonstrates the effectiveness of the method in predicting the chosen card in a card guessing scenario.

7 Beyond

The setup had us pick the *first* card. However, playing with the index where the card is picked from is something interesting. We did this, and calculated the success percentage for all possible starting indexes (~ 5200000 trials) This results in an interesting graph attached in the next page.

.


vals
0
2000
4000
6000
8000
0
10
20
30
40
50
1:52