Card Shuffling Optimality

Travis Trail

May 17, 2016

Keywords:

riffle shuffle, simulation, Monte Carlo, rising sequences

Abstract

The aim of this paper is to determine how many times an ordered deck of cards would need to be shuffled for it to be said that its cards are in random order. We will simulate 10000 shuffled decks, each having been shuffled n times, and count the number of rising sequences in each deck. We will then plot the distributions of these rising sequences, and compare them to the distribution from a randomized deck of cards to determine how many shuffles are required to answer the question. Depending on how you interpret the question, I argue that there are at least two acceptable answers: 7 shuffles, and 12 shuffles.

1. Defining the problem. In order for us to answer the question, "How many shuffles does it take to randomize a deck of cards?", we first need to know what the benchmark of a sufficiently shuffled deck is. One such way is to make use of a random number generator to randomly assign card locations in a 52 card deck. We can then use a simulation to examine the distrubtion of the number of rising sequences found in each of the randomized decks. We can use this distribution to identify a target mean and standard deviation which we can compare our results to. Equipped with these statistics, we will be able to identity after how many shuffles a deck's rising sequences count reaches our target distribution.

Depending on how we interpret the question, I argue that there are at least two acceptable approaches to answering it. We can answer how many shuffles are sufficient to guard against a skilled individual being able to take advantage of any patterns in the deck resulting from insufficient shuffling; and we can also answer how many shuffles are needed in order for a deck's distribution of rising sequences to be indistinguishable from that found in randomized decks.

2. Breaking down the problem. There are many ways in which one could approach this problem. One way is to simulate a large number of decks which have been shuffled 'n' times, and to count the number of rising sequences found in each deck (Trumbo, 3). We can then look at the distribution of rising sequences for a deck shuffled 'n' times, and compare that against the distribution we get from decks that have been placed in random order by a RNG.

3. Shuffling the deck. We begin by creating a function for shuffling a deck of cards. The first step of this function leverages the binomial function (Trumbo, 7) to simulate "cutting the deck" into two heaps of approximately the same size. The aim here is to simulate most real world situations where someone cuts the deck nearly in half before shuffling. The left heap is

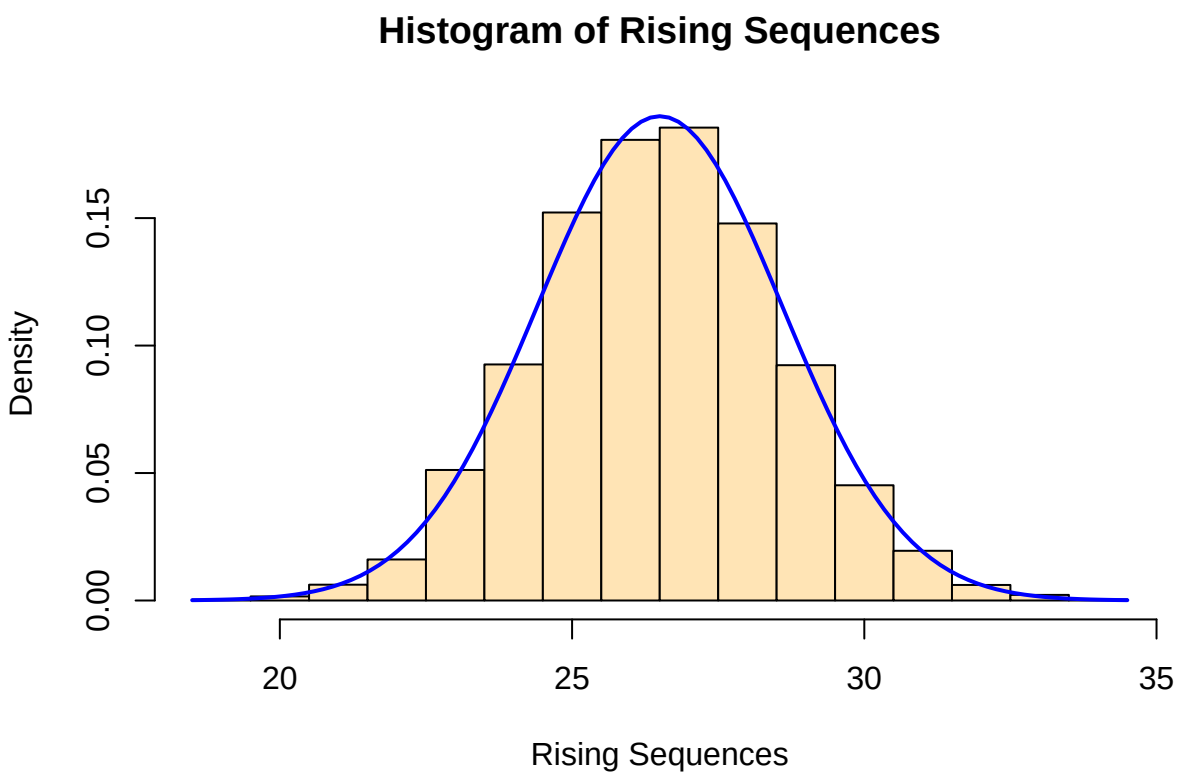
then inserted into sorted random locations in the right heap. They are not woven perfectly, as this would require a skilled shuffler. Most people will insert the cards in somewhat random locations. This process can then be repeated 'n' number of times in order for a deck to be shuffled 'n' times.

```
# This function simulates shuffling a deck of cards by first cutting it
# approximately in half, using a binomial distribution to determine the
# number of cards in each heap, then placing the left heap into ordered
# random slots into the right heap. It then outputs a shuffled deck.
# (Trumbo, 7)

shuffle <- function(deck) {
  cut.no = rbinom(1,48,.5) + 2          # Cut deck into two heaps
  left.heap = deck[1:cut.no]
  right.heap = deck[(cut.no+1):52]
  ix = sort(sample(1:52, cut.no))       # New deck indices for left heap
  new.deck = numeric(52)                # Initialize new deck
  new.deck[ix] = left.heap              # Fill in cards from left heap
  new.deck[new.deck==0] = right.heap    # Fill in cards from right heap
  return(new.deck)                      # Returns the shuffled deck
}
```

4. Rising sequences found in a randomized deck; our benchmark. Plotted below is a histogram of the number of rising sequences found in 10,000 simulations of a randomized deck of cards. As we can also see below, the mean is 26.5 rising sequences with a standard deviation of 2.1, which makes sense intuitively as 26.5 is the mid-point between 1 and 52, the number of cards in a deck. A rising sequence occurs in a deck for each sequence of cards existent within the deck from front to back. A rising sequence can have cards between it's group members, they do not have to be adjacent in the deck. Members of the sequence can be found in order from the top of the deck to the bottom.

```
# Count rising sequences (Trumbo, 7)

m=10000; x=numeric(m)                   # Initialize vector
for (i in 1:m) {
  deck = sample(1:52, 52)               # Create a randomized deck
  loc = match(1:52, deck)               # Match cards with deck location
  back = diff(loc) < 0                  # Find cards which have moved back
  x[i] = sum(back) + 1                  # Add them up as rising sequences
}

# Plot Histogram of the number of Rising Sequences
cutpts = seq(min(x)-1, max(x)) + 0.5
hist(x, breaks=cutpts, prob=T, col="moccasin",
```

```
     xlab = "Rising Sequences", main="Histogram of Rising Sequences")
curve(dnorm(x, 26.5, 2.1), add=T, col="blue", lwd=2)
```




```
mean(x); sd(x)                          # Print the mean and sd of sequences
```

```
## [1] 26.4936
```

```
## [1] 2.096807
```

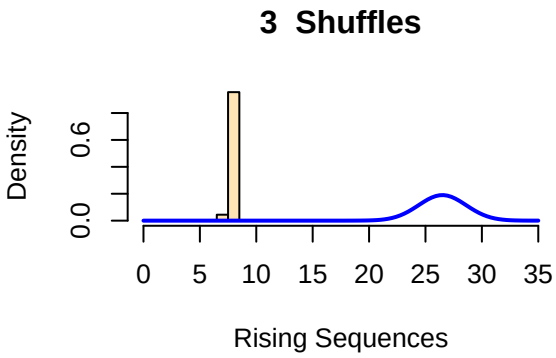
5. Counting the number of rising sequences in a deck. Here we write a function to count the number of rising sequences in a deck of cards which has been shuffled n times. (Trumbo, 86)

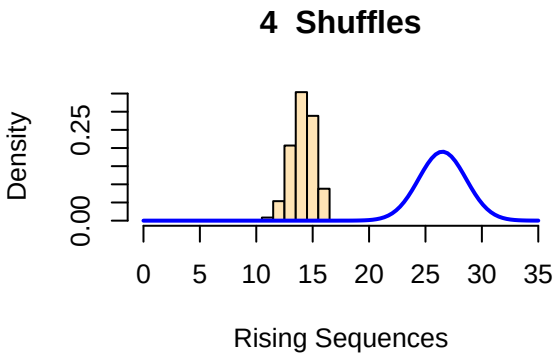
```
# This function simulates shuffled decks of cards and returns
# the number of rising sequences in each deck. It's only parameter,
# n, is the number of shuffles it will simulate.

ris.seq <- function(n) {
  m=10000; x=numeric(m)             # Initialize x (rising sequences)
  for (i in 1:m) {
    deck = 1:52                     # Initialize deck
```

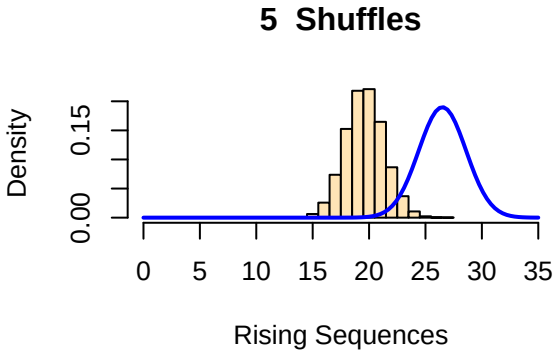
```
    # Shuffle deck n times
    for (j in 1:n) { deck = shuffle(deck) }

    # Count rising sequences in shuffled deck
    loc = match(1:52, deck)        # Locate card indices in deck
    back = diff(loc) < 0           # Identifies cards moving backward
    x[i] = sum(back) + 1           # Sums backward cards as rising sequences
  }
  return(x)                        # Returns number of rising sequences
}
```

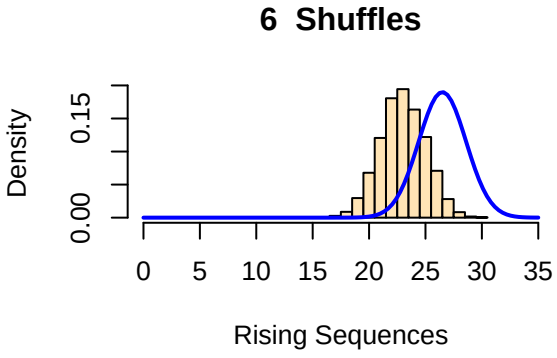
6. Distribution of rising sequences from a deck shuffled n times. Here, we can see the simulated distribution for the number of rising sequences found in 10,000 decks, each shuffled n times, represented by the tan colored histograms below. The blue normal curve represents our target distribution from that of a randomized deck of cards. As we can see, for $n = 3$, there are only a few rising sequences, however there is a steady increase in the number of rising sequences found as we shuffle more and more times. As the number of shuffles increases, we can see the distribution approaching our target distribution of a randomized deck.

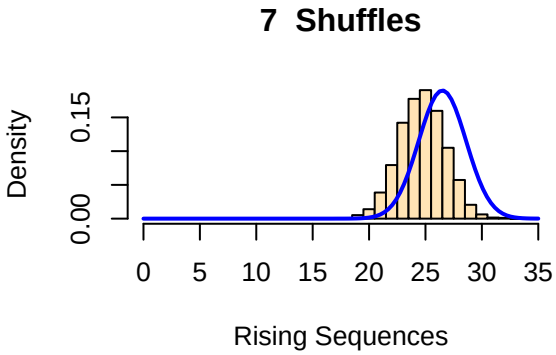
```
par(mfrow=c(2,2))                      # Plot results in quadrants
rs.stats = matrix(nrow=2, ncol=12)     # Initialize matrix for statistics
row.names(rs.stats) = c("Mean", "StdDev")
colnames(rs.stats) = c("3","4","5","6","7","8","9","10","11","12","13","14")

# Plots a histogram for each of our 'n' shuffled decks.
for (i in 3:14) {
  x <- ris.seq(i)
  cutpts = seq(min(x)-1, max(x)) +0.5
  hist(x, breaks=cutpts, prob=T, col="moccasin", xlim=c(0,35),
       main = paste(i, " Shuffles"), xlab = "Rising Sequences")

  # Add in our blue target distribution curve
  curve(dnorm(x, 26.5, 2.1), add=T, col="blue", lwd=2)
  rs.stats[1,i-2] = round(mean(x), 2)      # Store means
  rs.stats[2,i-2] = round(sd(x), 2)        # Store sds
}
```

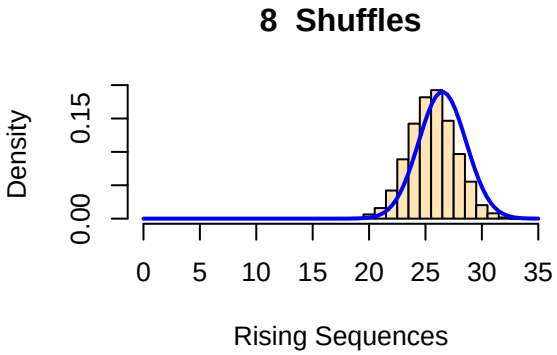

3 Shuffles
Density
0.0
0.6
0 5 10 15 20 25 30 35
Rising Sequences

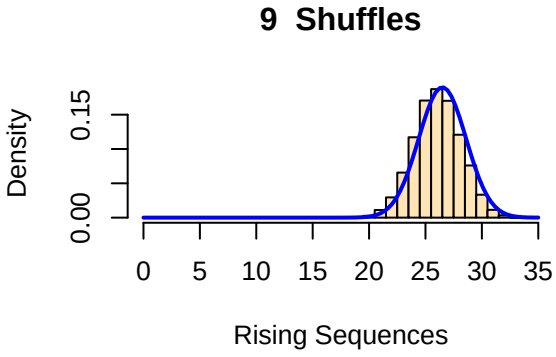

4 Shuffles
Density
0.00
0.25
0 5 10 15 20 25 30 35
Rising Sequences

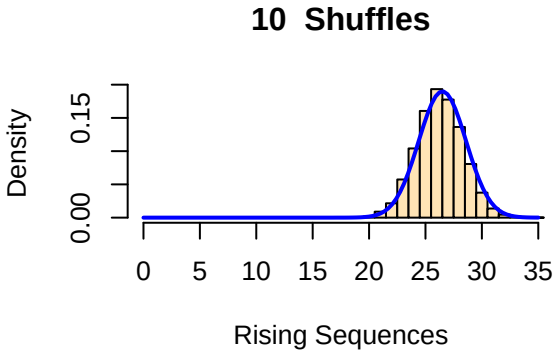

5 Shuffles
Density
0.00
0.15
0 5 10 15 20 25 30 35
Rising Sequences

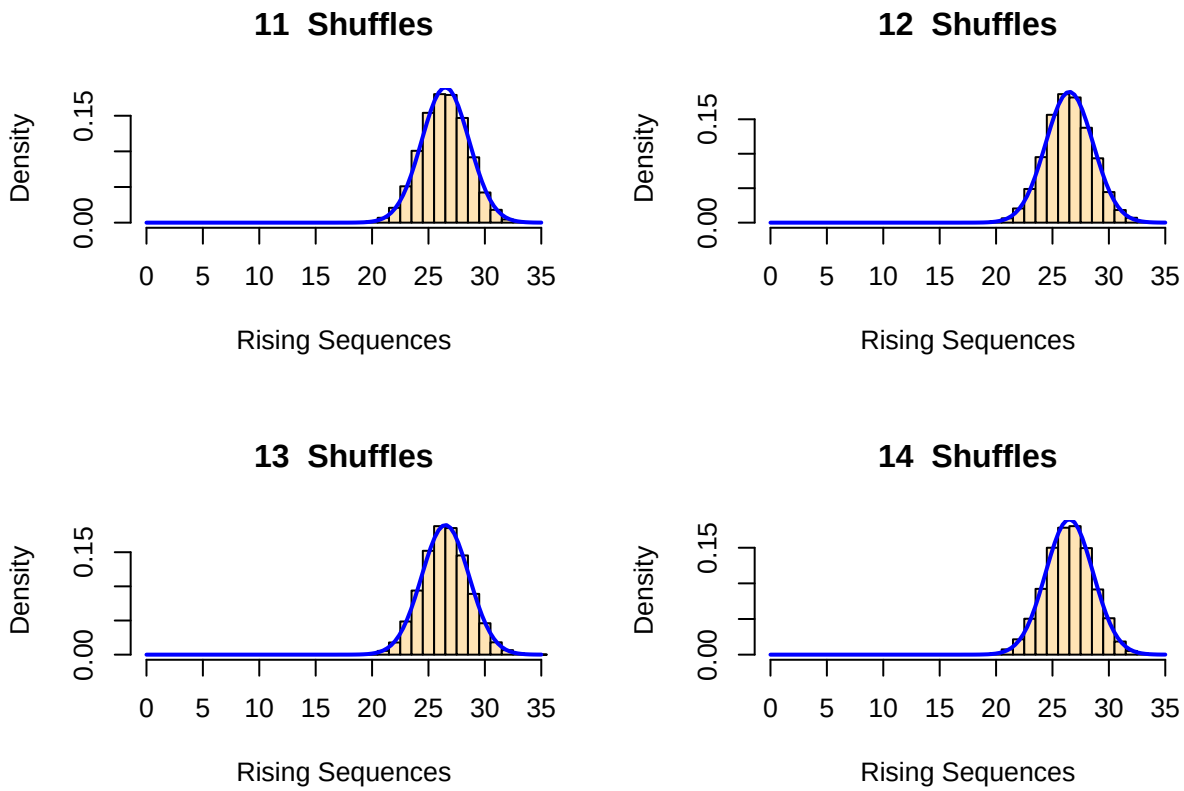

6 Shuffles
Density
0.00
0.15
0 5 10 15 20 25 30 35
Rising Sequences


7 Shuffles
Density
0.00
0.15
0 5 10 15 20 25 30 35
Rising Sequences


8 Shuffles
Density
0.00
0.15
0 5 10 15 20 25 30 35
Rising Sequences


9 Shuffles
Density
0.00
0.15
0 5 10 15 20 25 30 35
Rising Sequences


10 Shuffles
Density
0.00
0.15
0 5 10 15 20 25 30 35
Rising Sequences




```
# Plot statistics for rising sequences after n shuffles
rs.stats[,1:10]
```

```
##            3     4     5     6     7     8     9    10   11    12
## Mean    7.95 14.12 19.62 22.98 24.71 25.61 26.06 26.26 26.4 26.44
## StdDev 0.21  1.06  1.75  2.03  2.07  2.10  2.10  2.08  2.1  2.10
```

7. Summary How many shuffles are sufficient? This question depends on one's definition of sufficiency. It is clear from the distributions shown above, that shuffling a deck only a handful of times does not result in a deck that is anywhere near ranomly ordered. After 7 shuffles, it would be very difficult for anyone to be able to take advantage of any patterns in the deck. As we can see from the plot, the majority of the 'n shuffles' distribution has moved into the target distribution after 7 shuffles. This is sufficient for general game play.

However, we can still see from the histogram that this is still distinguishable statistically from our target distribution. After about 12 shuffles, the distributions appear to be fully overlapped; both having a mean of 26.5, and a sd of 2.10 rising sequences. The additional strides toward randomness achieved by shuffling 8 or more times are very minor. If we wish to use strict statistical measurements from a random deck having the mean number of rising sequences = 26.50 and sd = 2.10, then we would need to shuffle the deck about 12 times as shown in the table above.

Sources

1. Trumbo, Bruce. “How Many Shuffles Does It Take to Randomize a Deck of Cards?”, ppg 1-8. PDF Handout. CSUEB. Hayward, CA. 2016

2. Trumbo, Bruce. “4.3 - Shuffling Cards”, ppg 84-88. PDF Handout. CSUEB. Hayward, CA. 2016