Section 4 Activity

Introduction

Today we will practice using logical statements, for loops, functions, and simulating results. Since we have not yet dived into specific machine learning algorithms, we will borrow from gambling and probability by looking at poker.

Problem Statement

Our goal is to build a program that will:

1. Create a single hand of 5-card poker.
2. Determine the value of that hand from 1-10
3. Save the value to use later.
4. In such a way that we can arbitrarily repeat this process.

Poker Basics

We will limit ourselves to simply evaluating a hand without comparing within kinds of hands. In other words, we care whether a hand contains a pair, not that the pair is the pair of kings. The hands of poker rankings are here though we will ignore 5 of a kind and consider the Royal Flush to be the best possible hand.

With everything we have learned in class, we can write our evaluation function, simulate large numbers of hands, and compare our simulation results to theory. If you squint a little bit, you can consider this problem a type of classification.

Questions

Once you have built a program[1], use it to simulate 100, 1000, and 10000 hands of poker. This means that you will have three simulation runs. Before running your simulation, set your RNG seed to 355.

Based on your results, answer the following questions.

1. Estimate the probability of each hand using your collected data. How do those values compare to the exact values in Wikipedia's poker probability entry? (For the small runs, it is expected that you will not observe some hands. Those hands would have a "zero" percent empirical probability of showing up).
2. Based on your results, what is the smallest number of iterations you would want to use to estimate the probability of a given hand type? Do you need more than 10,000?
3. From the results of your third simulation run, identify the index of all the hands that were flushes. Take their sum. What is this value?
4. The worst hand in poker is 7 high (for example 2,3,5,6,7 from different suits). Conceptually, do you think this hand is more or less likely than a royal flush? What steps would you take (no coding) to simulate the probability of this kind of hand? If extra time is remaining once you finish, try amending your evaluation function to capture this possibility.

Hints

1. Start Small

From a practice standpoint, you can get all the same skill practice by writing an evaluation function to determine solely if the poker hand in question is a pair or not. Everything else is just additional features. Once you get this mini-program working, move on to expand to additional possible hands.

You'll probably want to have some function that samples 5 cards from a "deck", another function to compare those five cards against the condition that the hand is a pair, a loop to do it repeatedly, and some variables to save the result of each iteration.

You can (and should!) ignore suits at this stage.

[1] You can do this exercise solely on a limited version of poker for which there are no suits (and thus no flushes to concern yourself with). This turns the problem into a question about a vector of 52 elements (13 each repeated 4 times), from which you take 5. The values will of course not compare in many respects to the Wikipedia entry. This is the suggestion in the first and second hints.

2. Expand the evaluation function to cover more hands

Before including information about suits, the next thing you can try is to get the limited version that does not include suit. Here the only real change to be made is beefing up your evaluation function.

3. One way (though not the only way) to think of a Deck of Cards

Once you include suits, one way to view a deck of cards is as follows:

We have 52 observations, each with a value of the card rank and a value of the card suit. Each time we deal a card, we get information about one of the observations, and we do not put the observation back.

Face cards can be counted as numbers greater than 10. Aces are usually counted as 1.

4. Some potential test cases

(Aces = 1)

Hand 1: 1S, 2S, 3D, 4S, 5C

Hand 2:1S, 1C, 3S, 4S, 5S

Hand 3: 1S, 1C, 1H, 5S, 5D

Hand 4: 1S, 10S, JS, QS, KS

5. Run the following code.

The lines here provide examples of one way to get the distribution of card ranks.

```
example = c(1,1,2,2,3)
table(example)
length(table(example))
table(example) == 2

any(c("apple", "banana", "carrot") %in% c("carrot"))
```

6. An algorithm to rank poker hands

If you'd like to proceed to the full evaluation code immediately and since the goal of this activity is to write code, not come up with a new way to evaluate poker hands, the following works as an evaluation algorithm.[2]

1. Get the distribution of card ranks (e.g., number of 2s as opposed to suits).
2. Sort the distribution in some way.
3. Test if there are four of one rank. If yes, then the hand is four of a kind.
4. Test if there are three of one rank and a pair of another. If yes, then the hand is a full house.
5. Test if there are just three of one rank. If yes, then the hand is three of a kind.
6. Test if there is a pair of one rank and a pair of another rank. If yes, then the hand is two pairs.
7. Test if there is a pair of one rank. If yes, then the hand is a pair.
8. Test to see if all the cards of the same suit. If yes, check if the cards are A, K, Q, J, and 10. If they are, the hand is a royal flush. Otherwise, if you have a flush, keep note of it.
9. Test to see if the cards are straight. If you follow this evaluation order (and so have ruled out pairs and above), you can subtract the highest value from the lowest value and test if it equals 4. If yes, and the hand is also a flush, then it is a straight flush. If yes, and the hand is not a flush, then the hand is a straight. If no, and the cards are all the same suit, then the hand is a flush.
10. Anything else at this point must be High Card.

[2] Algorithm taken from here if you want to click through to the original