

# Assignment 1a: Liar's Dice

CS109L - Spring 2015

Turn in Deadline: Thursday May 21st at 11:59 PM

Redo Deadline: Thursday June 4th at 11:59 PM

## 1 Overview

Liar's Dice was brought to Spain by the Spanish conqueror Francisco Pizarro during the 16th century. Today, the game is perhaps best known for its appearance in the motion picture *Pirates of the Caribbean: Dead Man's Chest*, but historical records also show that the game was popular on real pirate ships. Why? The game is easy to learn, requires little equipment, and can be played as a gambling or drinking game, or both. Playing the game well requires the ability to deceive and detect and an opponent's deception.

There are at least three different versions of Liar's Dice.<sup>1</sup> In all of them, dice are rolled in a concealed fashion and *bids* are made about the result of the roll. Players in turn must then either raise the current bid or challenge the current bid. In the *common hand* version of the game, each player has a set of dice which is visible to him/her, but bids are made about the collective pool of dice (their hand plus all other players' concealed hands).

For your second assignment, you will implement the game of common hand Liar's Dice. Specifically, you will design a computer player strategy that will allow you to play a game of Liar's Dice against the computer.

## 2 Game Rules

Five six-sided dice with traditional dot faces are generally used per player, with dice cups used for concealment. Each round, the players roll their dice while keeping them concealed from the other players. One player begins bidding, picking a quantity of face values two through six. The quantity states the player's opinion on how many of the chosen face have been rolled in total on the table. A one (*ace*) is often wild and counts as the stated face of the current bid; however, the game can also be played without wilds. In a five-dice, four-player game with wilds, the lowest bid is "one two" and the highest bid is "twenty sixes".

Each player has two choices during his turn: make a higher bid, or challenge the previous bid as being a lie. Raising the bid means either increasing the quantity, or the face value, or both, according to the specific bidding rules used. Different bidding rule sets are described below. If the current player thinks the previous player's bid is wrong, he challenges it, and then all dice are revealed to determine whether the bid was valid. If the number of the relevant face revealed is at least as high as the bid, then the bid is valid, in which case the bidder wins. Otherwise, the challenger wins. A challenge is generally indicated by simply revealing one's dice, though it is customary to verbally make the challenge by saying something to the effect of "You're a liar".

---

<sup>1</sup>Other related games such as Liar's Poker use playing cards instead of dice, but the basic rules are almost the same.

For example, if a bid of “seven fours” is challenged, the bid is successful (and the player who made it wins) if there are seven or more fours (including aces if they are wild). If the bid fails (the bidder is a liar and the challenger wins) if there are fewer than seven fours (including aces if they are wild).

The starter code you receive for the assignment does not implement wildcards, but you may feel free to change this if you wish.

## 2.1 Bidding Rules

The most common systems for bidding are listed below, in order of the amount of restriction they place on bidders. All variants are described in relation to the face value and quantity of the previous bid:

1. **The player may bid an increased quantity of the same face, or any quantity of a higher face.** Given a bid of “three twos”, the minimum raise is either “four twos” or any quantity of “threes”. This is very common as it generally gives players many options, allowing for the most information to be gained. Using this ruleset, face value of the bid cannot be lowered except with a popular rule variant; the next ruleset, however, is similar and allows for reduction in face.
2. **The player may bid an increased quantity of any face, or the same quantity of a higher face.** Given a bid of “four fours”, the minimum raise is five of any face, or “four fives”. This is the most common ruleset in packaged games as it is very similar to Poker’s emphasis on quantity over rank in hand rankings (three twos is better than a pair of kings). It also allows a player to re-assert a lower face value believed to be predominant.
3. **The player may bid any quantity of any face, as long as either the quantity or face is higher than the higher of the two numbers of the previous bid .** Given a bid of “five threes”, the minimum bid must have a six, either six of any face or any quantity of “sixes”.

Raises must be at least the minimum; however, the current player may raise the bid to any legal bid. Given “four fours”, a player may (in any of these systems) call “seven sixes”. Such *bid jumping* has strategic value, but a large increase such as in this example has a high probability of being incorrect, and so is likely to trigger a challenge. It is also considered bad form, as bid jumping reduces the amount of information that can normally be gained by more conservative raises, and makes for shorter games.

The starter code you receive for the assignment implements the first bidding scheme listed above, but you may feel free to change it to one of the other two options given.

## 3 Strategy

Playing Liar’s dice involves many subtleties and interpersonal skills similar to other bluffing games such as Poker. However, there are some universal elements of strategy.

Perhaps foremost, **a bid gives others at the table information.** Players, through subsequent bids, reveal the players’ confidence in the quantity of each face value rolled. A player with two or three of a certain face value under his or her own cup may make a bid favoring that face value. Players can thus use these bids to build a mental picture of the unknown values, which either strengthens or weakens their confidence in a bid they are considering. Others may consider a bid as evidence it is true, and if their own dice support the same conclusion, may increase the bid on that face value, or if their dice refute it may bid on a different face, or challenge the previous bid.

Conversely, **bids can also be bluffs**. Bluffs in Liar’s Dice can be split into two main categories; early bluffs and late bluffs. An early bluff, for instance “three threes” when the player making the bid has no threes under his cup, is likely to be correct by simple probability (depending on the number of players); however, other players may believe the bidder made that bid because his or her dice supported it. Thus, the bluff is false information that can lead to incorrect bids being made on that face value. Players will thus attempt to trick other players into overbidding by use of early bluffs to inflate a particular face value. A late bluff, on the other hand, is usually less voluntary. The player is often unwilling to challenge a bid, but as a higher bid is even more likely to be incorrect it is even less appealing. A late bluff is thus a critical part of the game. Convincing bluffs, as well as reliable detection of bluffs, allow the player to avoid being challenged on an incorrect bid.

As with any game of chance, **probability is highly important**. The key element is the *expected quantity*, or the quantity of any face value that has the highest probability of being present. For six-sided dice, the expected quantity is one-sixth the number of dice in play, rounded down. When wilds are used, the expected quantity is doubled as players can expect as many aces, on average, as any other value. Because each rolled die is independent of all others, any combination of values is possible, however the “expected quantity” has a greater than 50% chance of being correct, and the highest probability of being exactly correct, or *spot on*. For example, when 15 dice are in play and wilds are used, the expected quantity is 5. The chances of a bid of 5 being correct are about 59.5%; in contrast, the chances of a bid of 8 being correct are only about 8.8%.

However, a high bid is not necessarily incorrect, because **bids incorporate information the player knows**. A player who holds a preponderance of a single value (for instance, four out of the five dice in his hand are threes) may make a bid, with fifteen dice on the table, of “six threes”. To an outside observer who sees none of the dice, this has an extremely low probability of being correct (even with wilds). However, since the player knows the value of five of those dice, the player is actually betting that there are two additional threes among the ten unknown dice. This is far more likely to be true.

## 4 Dice Probability

For a given number of unknown dice  $n$ , the probability that exactly a certain quantity  $q$  of any face value are showing,  $P(q)$ , is

$$P(q) = \binom{n}{q} \cdot (1/6)^q \cdot (5/6)^{n-q}$$

or, in other words, the number of dice with any particular face value follows the binomial distribution. If the binomial distribution is the probability of the number of “successes” in  $n$  independent Bernoulli trials with the same probability of “success” on each trial, the multinomial distribution is the analog for a categorical distribution where each trial results in exactly one of some fixed finite number  $k$  of possible outcomes, each with probabilities  $p_1, \dots, p_k$  such that  $p_i \geq 0$  for  $i = 1, \dots, k$  and  $\sum_{i=1}^k p_i = 1$ . For  $n$  independent trials, the random variable  $X_i$  indicates the number of times outcome number  $i$  was observed, so  $X_i \in \{0, \dots, n\}$  and  $\sum_i X_i = n$ . The vector  $X = (X_1, \dots, X_k)$  follows a multinomial distribution with the following probability mass function:

$$P(X) = \frac{n!}{x_1! \cdots x_k!} p_1^{x_1} \cdots p_k^{x_k}$$

So the probability of getting two threes and four fives in a roll of six die is

$$P(X_1 = 0, X_2 = 0, X_3 = 2, X_4 = 0, X_5 = 4, X_6 = 0) = \frac{6!}{0!0!2!0!4!0!} (1/6)^0 (1/6)^0 (1/6)^2 (1/6)^0 (1/6)^4 (1/6)^0$$

## 5 The Assignment & Requirements

Your goal for this assignment is to model and program a intelligent computer strategy for Liar's Dice. Although the implementation details are left entirely up to you, you are strongly encouraged to use either or both of the methods from the Spam Filter and Bayesian Prediction examples. You may spend as much or as little time as you want working on this assignment, but for credit it should be clear that you've put in at least a couple of hours. The only requirement for all implementations is that your computer cannot cheat. In other words, a computer player should only make decisions based on its own hand and the information that is available to all players, such as what calls have previously been made.

The rest of this handout will guide you through the starter code and give you some implementation hints. Almost all of the code needed for the assignment has been written for you, so you can focus exclusively on the task at hand: getting your computer players to intelligently play the game.

All players (computers and human) are represented internally as lists with the following fields:

**\$name**: name of the player

**\$ndice**: the number of dice visible to the player

**\$type**: the type of player (human or computer)

**\$dice**: the players hand (represented as a vector of integers with values 1 through 6)

In this way, we have defined each player by a collection of attributes. Although it is up to you to decide exactly what additional attributes a computer player might need to play the game, the starter code includes a **\$confidence** attribute for all computer players that you can leverage as your computer determines whether or not to challenge the previous bid. Another attribute you could give your computer players is **\$skill**, which may be used as a weighting factor to add variability to the computer's difficulty.

The current bid is modeled by a list called **bid** with fields **\$face** and **\$num** corresponding to the face value and quantity of the bid. Following this convention, `bid <- list(face = 4, num = 12)` represents the bid "twelve fours".

To receive a "1" for the assignment, you must complete the **Computer Turn** function in **liarsdice.R** so your computers perform the following tasks:

1. If a computer player is the first to go in a round, it must decide how to make the first bid.
2. Otherwise, the computer player must decide how to "raise" and when to challenge the previous bid. One very poor choice for deciding how to raise is to have the computer make the next minimally higher bid. You can certainly do a lot better than this. Each computer can see what its own hand is and also knows the bidding history, so it should be able to make at least semi-intelligent decisions. You can even randomly have your computer lie. Check out the Strategy section for other ideas. A great implementation should be able to make a computer that is hard to beat. Consider this a fun challenge.

Most of the code has already been written for you, and the code segments you need to finish are clearly labeled with FIXMEs. To run the program, call `source("liarsdice.R")` followed by `game <- LiarsDice(players)`<sup>2</sup>. Both human and computer players are instantiated in **liarsdice\_init.R**. This file is sourced by **liarsdice.R** in addition to **liarsdice\_util.R**, which contains utility functions for gameplay, including those which specify the bidding rules; if you want to change the bidding scheme, you'll most likely want to look here. Two

---

<sup>2</sup>Note that you might need to change your working directory for this to work, which can be done with the `setwd` function.

additional parameters to the `LiarsDice` function are `debug` and `halt`. The former when set to `TRUE` will print debugging information such as what dice are on the table at the beginning of the round. The latter when set to `TRUE` requires a carriage return to be made between each players turn before gameplay proceeds. This slows down gameplay (especially for sequential computer turns) to a followable pace. If you would like to run a game exclusively with computer players, however, you will probably want to set `halt` to `FALSE` and then just look at sequence of bids made during the round after the game has ended. The return value of the `LiarsDice` function by default is a list of the bids made over the course of the game, so you can run `LiarsDice` using only computer players and then run analytics over the bid histories. This is a great way to see how your computers are performing.

## 5.1 Implementation Ideas

This assignment is as much about creativity as it is about using `R` to solve an interesting problem. As such, I'd like everyone to come up with his/her own implementation. That said, you are encouraged to try using the ideas shown in the Spam Filter and Bayesian Prediction examples. For these, you may potentially model the probability that the current bid is a lie as being dependent on a subset of the bids that came before it. If not the actual bids that came before it, perhaps instead the amount that previous bids were incremented from the ones that came immediately before them (so the bid “three fours” would be an increment of two quantity from “one four” and two face value from “three twos”). Although you will not be learning probabilities from a training data set, you can assign them manually or take a Bayesian approach and use priors and posteriors.

For other general ideas, your solution may include any of the following (this is a non-exhaustive list):

1. Statistical analysis \*
2. Simulation
3. Machine learning \*
4. Probability
5. Distributions

Items with \* denote those you may not know enough to use (at least not yet) but that would be helpful if you've already had some exposure.

## 5.2 Extensions

In the past, some students have been interested in changing the rules of gameplay. Sometimes the change is minor, like changing the order of bidding. Sometimes, the change is more substantial, like using a version of the game that isn't *common hand*. If you'd like to change anything about the starter code, just make sure to email me and ask first, then document it in the writeup file (mentioned below).

## 6 Submission

1. Zip up all your `.R` files into `[sunet]-1a.zip` (inserting your sunet (e.g. `hgkshin`) into `[sunet]`). As an example, I would submit `hgkshin-1a.zip`
2. Email `[sunet]-1a.zip` to `cs109l.submissions@gmail.com` (note that this is a separate submission email address) with the subject:
  - “Assignment 1a Turn In [Insert SUNET ID Here]”
  - “Assignment 1a Redo [Insert SUNET ID Here]”

The body may be blank as long as the `.zip` file is attached. You may turn the assignment in as many times you’d like before the deadline and I will only grade the most recent assignment.

As an example, if I were to submit assignment 1a for the redo deadline, I would send my file `hgkshin-1a.zip` with the subject:

**Assignment 1a Redo hgkshin.**

**Important honor code note:** As with all assignments, you must come up with your own original code. Copying/modifying an existing implementation would be a violation of the honor code. If you have any questions at all about this, feel free to email me at `hgkshin “at” stanford “dot” edu`.

## References

- [1] [http://en.wikipedia.org/wiki/Liar's\\_dice](http://en.wikipedia.org/wiki/Liar's_dice)
- [2] [http://en.wikipedia.org/wiki/Multinomial\\_distribution](http://en.wikipedia.org/wiki/Multinomial_distribution)