Roll the Bones: "A Study of Random Events Using Interactive Simulation"

Jason Saint

Follow this and additional works at: https://louis.uah.edu/honors-capstones

Recommended Citation
https://louis.uah.edu/honors-capstones/553

This Thesis is brought to you for free and open access by the Honors College at LOUIS. It has been accepted for inclusion in Honors Capstone Projects and Theses by an authorized administrator of LOUIS.
Roll the Bones
“A Study of Random Events Using Interactive Simulation”

Jason Saint
University of Alabama in Huntsville
Honor’s Senior Project
November, 1996
Table of Contents

I. Abstract

II. Philosophical Ideas and Placement in the Tradition

III. Methodology

IV. The Game of Craps

V. Formulation of the Model
   1. Overview
   2. Assumptions and Observations
   3. Derivation of the Model
   4. Evaluation

VI. Notes on the Simulation
   1. Instructions
   2. Implementation Notes

VII. Future Directions

VIII. Bibliography and Acknowledgements
   1. Works consulted
   2. Acknowledgements

Appendix A: Simulation Output

Appendix B: Source Code Listings
Abstract

"Roll the Bones : A Study of Random Events Using Interactive Simulation"

As the title suggests, this is an examination of a dice game from a mathematical perspective which utilizes a computer simulation of the game under consideration. The game studied is a wagering game called "craps" which is played with a pair of six-sided dice. An explanation of the rules is included with references to works on the history of the game. A brief sketch, with references, of the idea of randomness and its emergence in western thought is included also. The study of the game in the formulation of a mathematical model is used to attempt to motivate a method for the consideration of processes involving random events. The examination of the method focuses on the role of simulation in the development of the model. An interactive simulation of craps has been designed and implemented for UNIX based computers. The simulation interfaces with the user through the World Wide Web, and is available to anyone with a forms capable web browser at:

http://www.cs.uah.edu/cs/students/jsaint/craps.html
Philosophical ideas and placement in the tradition

In the latter half of this century, there has much debate involving questions about some of the most basic assumptions of Western scientific thought. These questions became amplified by the results of experimentation at the quantum level which implied an inherent uncertainty in scientific measurement. This uncertainty was highly inconsistent with the specific goal of the experiments, namely explaining the nature of light, as well as with the general goal of explaining all natural phenomena. Indeed, these results seemed to suggest that no explanation existed. Throughout much of the Modern period the idea that such a definite explanation did, indeed, exist for light or for any other aspect of reality would not have been questioned. Only lately in comparison to the length of the period have philosophers such as Camus, Nietzsche and Sartre offered opposition to the idea of an order fundamental in the nature of reality. Even the ideas that would lead to the concept of randomness, in a mathematical sense, were only beginning to emerge as late as the mid sixteenth century. Even after the idea of chance came into mathematical formulation, it was thought that a sufficiently comprehensive explanation could eliminate randomness (David, p. 172).

Mathematics is a tool used to build these explanations of reality, which take the form of mathematical models. The idea that there is an order inherent in reality leads mathematicians to try to predict the precise outcome or behavior of a situation with equations. Models which do this are called deterministic models. There is a process used in creating and analyzing these models which has been well established and has proven its worth in that much of what we call modern scientific knowledge is based on these models. The first step in this process is the identification and observation of the measurable quantities which affect the behavior of the system. Once identified, the relationship between these quantities is formed using the applicable existing knowledge and laws. Since many times these relations become complicated and difficult to deal with, simplifying assumptions are made and noted so that they may be later evaluated. These assumptions and laws produce a mathematical approximation of the situation being described. Mathematical techniques, such as those used to solve differential equations, are then employed to analyze this approximation to produce a mathematical model. The final step in this process is
to compare the results of the model to data collected in the observation of the system. When the results are far different from what is observed, it usually implies that there is a problem in one or more of the assumptions made in constructing the approximation of the situation. The assumptions can then be changed, and the model re-analyzed until the results are "close enough" to the data observed.

Since the inconsistencies between the model and reality arise with the use of simplifying assumptions, one might be led to ask why not discard all assumptions, and proceed with nothing assumed out of the model. The assumption that order is fundamental to reality leads to the conclusion that such a model is possible to obtain. In practice however, situations are often too complex to include everything which is involved, so apparent randomness or inconsistency with the notion of fundamental order has been thought to arise when a situation becomes complex. The prediction of weather is an example of this. Particular weather situations seem to defy the abilities of mathematical models to predict them. Weathermen often find out that the predictions they made for the current day are to some degree wrong. At the best, even predictions of the temperature cannot give the particular temperature at a particular time and place with absolute confidence. The traditional view is that the factors contributing to the particular weather of a place are too numerous and vary too much from place to place, even over short distances, to be taken account of in the model.

The above view has been universally accepted by western science until only this century. There exists now an increasing body of evidence which seems to imply that randomness is just as fundamental to the nature of reality as order. In light of this, there may be a reason to re-examine previously accepted models with attention given to the role of chance. There already exists a branch of mathematics which is concerned with situations which seem to exhibit random behavior, namely Probability and Statistics. In probability theory, the particular event is seen as fundamentally random and patterns (order) arise as the situation is repeated or becomes more complex. The variety of mathematical model which is used in analysis of such situation is called a stochastic model.

Some situations seem to lend themselves to analysis by such models, particularly situations in which the indeterminate outcome has a finite number of possibilities. Games are good examples of such "model-friendly" situations. Game theory, especially as expressed by John von Neumann and Oscar
Morgenstern, is a formalization of the ideas involved in analyzing situations in which there is some non-deterministic element (Packel, 87). This element may be due to "human" activity or some randomizing device or some combination of the two. Human activity here is taken to mean that there is or may be something inherent in human (and perhaps also animal) behavior which defies explanation involving efficient causality. Game theory has many practical applications especially in the areas of economics and social science. Perhaps, then there is a method for studying such random events similar to the one for a deterministic model.
Methodology

With this in mind, I have chosen to model a game which employs a randomizing device. The game chosen is a dice game commonly called "craps". For a detailed discussion of the rules of craps please refer to "The Game of Craps" document included here. The intention was to model the game, paying particular attention to the method used. Some things that became apparent while considering this particular situation included the recognition that since patterns only arise with complexity, and that complexity here involves the repetition of a specific throw or set of throws of the dice. Thus, in order to be able to recognize a pattern, there would be a need to repeat a single craps turn many times. Since it is impractical to attempt to play craps for millions of turns and keep track of the results, I decided to simulate the craps game with a computer. For a detailed discussion of the simulation, please refer to the "Notes on the Simulation" document included here.

In formulation of the model, I first thought about what things I might want to have some idea about in advance if I were intending to play craps. Since craps is a common gambling game, one of the first things of this sort that occurred to me was the amount of money that I might have after a certain period of play. With this in mind, I set about discovering the things that determined the amount I would have. These things turned out to be the initial amount of money, and the amount won or lost on each turn. Again, it was important to identify what these values depended on. The initial amount of money depends upon the player, and upon the particular rules being used, but in any case will be known at the beginning of the game. Thus, it becomes a parameter to the model. The amount won or lost during each turn was dependent upon the amount bet during that turn, and whether the event the player bet on occurs or does not occur. Similarly to the initial amount, the amount bet on each turn is a parameter to the model. Whether or not the event occurs is dependent upon what event the player bet on, which is in turn dependent on the rules and the number of players (both parameters) and whether the score on the dice indicates "pass" or "no pass" (for the shooting player, this reads "win" or "loose"). The score, in craps, is the sum of the scores on each individual die. What are these individual scores dependent on? It is not clear that there are any causes for this, so we say that it is random.
Since the roll of a die is a random event, its outcome cannot be exactly predicted. We can, however, determine some things about the event. First, there are only six possible faces upon which the die can rest, so only six possibilities for the score on the die. The regularity of the shape of the die seem to suggest that no one side is any more likely to be up than any other side, so we say such dice are fair. It is possible to have dice which are more likely to exhibit some scores than other, known as "loaded dice", so whether or not the dice are fair or loaded will be a parameter effecting the likelihood of any particular value occurring. In the case of fair dice, the likelihood or probability of occurring for each score is the same. Mathematically, we say that these probabilities are uniformly distributed. Specifically, this probability is the number of ways in which a particular score can occur divided by the total number scores, or one-sixth. With loaded dice, these probabilities will change in a presumably known way, and may be accounted for in the model. Once this random event is identified and analyzed, the model can be formulated by backtracking through the chain of dependencies formed earlier. For a detailed description of this, please refer to the "Formulation of the Model" document included here.

Using the simulation constructed earlier, some idea of the level of complexity, in this case the number of repetitions, necessary to observe the predicted pattern can be obtained. The simulation, available at: http://www.cs.uah.edu/cs/students/jsaint/dice.html, provides a means of accepting the parameters to the model, simulates the game based on these parameters, and returns the results to be compared to the predictions. Varying these parameters and observing how they effect the outcome of the simulation can reveal patterns and relations between the parameters and the outcomes. This process may be repeated for any number of questions that might be raised about the game of craps.

This method can be used in the study of many such games. Summarized, the method used here to study a game which includes a randomizing device can be expressed as follows:

**Step 1:** Observe and investigate the process in order to determine what kind of questions can be asked.
Step 2: Once the questions are known, investigate the nature of the things which affect the 
answer to that question, noting those which are parameters, and further investigating 
those which are, in turn, dependent on other things.

Step 3: Upon reaching a fundamentally random element, see what can be known about that 
element (i.e. probability distributions).

Step 4: Retrace the hierarchy of dependencies so that the relation of randomness in the outcome 
to the initial question can be known. This will be the formulation of the mathematical 
model for the game studied.

Step 5: Show that the patterns predicted will occur in the playing of the game, and observe how 
the parameters effect the process. This may be achieved by simulation.

The above steps may be repeated, with possible iterations between steps.

In keeping with the ideas expressed here, this method is offered as applicable only to a relatively 
small class of processes, namely those including some randomizing element. This method may not be 
appropriate for all random processes or even all games. The method for modeling a particular situation 
may not be able to be known in advance, however, with the repetition of many models of many different 
types, perhaps a more general pattern can be noticed. As only a single model of a single type has been 
completed at the present time, it might be a bit early to offer predictions about the modeling process in 
general, but I feel confident in stating that simulation will play a role in such a method. The website which 
contains the model and the simulation may be periodically updated with new models and simulations, and 
any patterns that emerge will be noted there as well.
The Game of Craps

It has been speculated that the dice game called craps had its origin in an ancient game which involved the throwing of animal heel bones, called astragali (see David, p. 1). The throws were given a value depending upon what face(s) of the bone(s) landed face-up. There is debate about whether these throws were seen by the ancients as "random" (i.e. having no definite cause), or as the result of some unseen cause. Either way, this game developed into a number of modern games, one of which is craps.

Craps, as it is commonly played today, involves the throwing of two six-sided dice. A player's score is calculated by summing the scores face-up on each die. Based on this value, each turn is assigned the status of either pass, or no-pass. A roll totaling 7 or 11 is called a "natural win" and is scored as a pass. A throw resulting in 2, 3, or 12 is called a "crap-out", and is scored as a no-pass. A throw of 4, 5, 6, 8, 9, or 10 is called the shooting player's "point". When a point has been rolled, the shooting player must continue rolling the dice until either the point comes up again, or a 7 is rolled. If the second occurrence of the point happens before a 7 is rolled, the turn is scored as a pass. If the 7 comes up first, the turn is scored as a no-pass. The rules on when the shooting player must give up the dice vary, so for the purpose of this model, the convention of giving up the dice when no pass on a point throw occurs.

Craps is a game which people often place wagers on. The rules on how this may be done vary from game to game, so this model will begin by laying out some simple betting rules.

(R1) Only the conditions of pass and no-pass will be bet upon.

(R2) The shooting player will bet pass, all others will bet no-pass.

(R3) The bet amount will be constant over all turns.

These rules are very restrictive, and are probably not used in any "real" situation. One reason for this is that the rules assume a "house" which will pay-off the bets. However, we will begin with this very general set of rules with the idea that the model may be adapted, later, to any particular set of rules chosen.
Overview

What follows here is the formulation of the mathematical model of a craps game which can be used to make predictions about the amount of money a given player will have at the end of a craps game given an initial amount of money and an initial bet. The model will proceed from the random event of the roll of a die, and build a way to describe the amount won or lost. The amount the player ends up with, is simply the initial amount plus the amount "won" in bets, where this amount may be positive or negative. Results will be evaluated using a simulation. The random events will be called here "random variables" (see Ross, p. 108 for a definition of this), and some knowledge of the basic principles of probability is assumed (references for the ideas used will be provided).

Assumptions and Observations

(1) The betting and the form of play follow the rules outlined in "The Game of Craps" document.

(2) The dice are assumed to be either "fair" or "ace-six flats" where fair dice means that no side is more likely to occur than any other, and ace-six flat dice means that the scores of 1 and 6 are twice as likely to occur. The random variable associated with the score on die $i$, call it $X_i$, will depend on the type of dice.

   For fair dice, we say the probabilities have a uniform distribution (see Ross, 159), so we say that the probability that $X_i$ is equal to $n$, $P(X_i = n)$ is as follows:

   $P(X_i = 1) = P(X_i = 2) = P(X_i = 3) = P(X_i = 4) = P(X_i = 5) = P(X_i = 6) = 1/6$.

   For ace-six flats, the distribution is as follows:

   $P(X_i = 1) = P(X_i = 6) = 1/4$ and,

   $P(X_i = 2) = P(X_i = 3) = P(X_i = 4) = P(X_i = 5) = 1/8$.

(3) We observe that the score on any dice does not effect the score on any other dice, so we say that the dice are independent (see Ross, 72-75).

(4) We observe that the score on any roll does not effect the score on any other roll, so we say that the rolls are independent.
(5) We observe that the amount won has the dependencies as follows in the diagram.

![Diagram]

Derivation of the Model

I. Score on the die.

We begin with the idea that the score on a particular die on a particular throw is essentially random. In craps, only 2 dice are used. The score which is important to the play of craps is the sum of the scores on the individual dice. Since each die can take values from 1 to 6, the sum will take values from $(1+1) = 2$, to $(6+6) = 12$. We will call the random variable for the sum $Y_2 = (X_1 + X_2)$. Since $X_1$ and $X_2$ are identically distributed, independent random variables, the probability distribution for $Y_2$ will be given by the convolution of the distributions of $X_1$ and $X_2$ (see Siegrist, p. 175). This is given by:

$$f_Y(j) = P(Y_2 = j) = f_X*f_X = \sum_{i=1}^{6} f_X(i) f_X(j-i)$$
For fair dice, this gives:

\[ f_y(2) = f_y(12) = 1/36 \]
\[ f_y(3) = f_y(11) = 1/18 \]
\[ f_y(4) = f_y(10) = 1/12 \]
\[ f_y(5) = f_y(9) = 1/9 \]
\[ f_y(6) = f_y(8) = 5/36 \]
\[ f_y(7) = 1/6 \]

For ace-six flats, this gives:

\[ f_y(2) = f_y(12) = 1/16 \]
\[ f_y(3) = f_y(11) = 1/16 \]
\[ f_y(4) = f_y(10) = 5/64 \]
\[ f_y(5) = f_y(9) = 3/32 \]
\[ f_y(6) = f_y(8) = 7/64 \]
\[ f_y(7) = 3/16 \]

These values are important because they are used in determining the probability of the conditions of "pass" or "no-pass" occurring ("pass" and "no-pass" are defined in the "Rules and Instructions" document).

II. Probability of a pass condition.

Refering to the rules mentioned above, we see that the probability of a pass condition is the probability of rolling a 7 or an 11 or a 4 followed by 0 or more rolls which are neither 4 nor 7, followed by another 4, or terms similar to the last for each member of \{5, 6, 8, 9, 10\}. Calling this random variabe \( W \), we see that:

\[ P(W=1) = P(Y_2=7) + P(Y_2=11) + P(Y_2=4)^2 \sum_{i=0}^{\infty} (1 - P(Y_2=4) - P(Y_2=7)) \]
\[ P(Y_2=5) - P(Y_2=7)) + P(Y_2=6)^2 \sum_{i=0}^{\infty} (1 - P(Y_2=6) - P(Y_2=7)) + P(Y_2=8)^2 \sum_{i=0}^{\infty} (1 - P(Y_2=8) - P(Y_2=7)) + \]
\[ P(Y_2=9)^2 \sum_{i=0}^{\infty} (1 - P(Y_2=9) - P(Y_2=7)) + P(Y_2=10)^2 \sum_{i=0}^{\infty} (1 - P(Y_2=10) - P(Y_2=7)) \]

The term \( \sum_{i=0}^{\infty} (1 - P(Y_2= j) + P(Y_2=7)) \) is an example of a geometric series, and will thus converge to:
For fair dice, this yields:

\[ P(W=1) = 0.4929 \]

For ace-six flats, this yields:

\[ P(W=1) = 0.4390 \]

Thus we note that the pass condition is more likely to occur with fair dice.

III. Number of pass and no-pass outcomes

To know the number of pass and no-pass outcomes, we must know the number of turns, and the likelihood of each outcome. We know from above the probability of a pass condition \( p \), so because no-pass is the only other possibility, we say that \( W=0 \) when a no-pass condition occurs. Since \( W=1 \) or \( W=0 \) are the only possible values of \( W \), we say that \( W \) is an indicator variable. Further, \( P(W=0) = (1 - p) \). The number of turns will be a parameter, call it \( n \). Let the number of pass conditions that occur in \( n \) trials be denoted by the random variable \( N_n = (W_1 + W_2 + \ldots + W_n) \). Since there are only two possible outcomes we see that this is an example of Bernoulli Trials (see Siegrist, p. 25), with \( p = P(W=1) \). Thus \( N_n \) has the binomial distribution with parameters \( n \) and \( p \). This leads us to conclude that the probability of the number of pass conditions being equal to some number \( k \) is:

\[ P(N_n = k) = \binom{n}{k} p^k (1-p)^{n-k} \]

This only gives us the probability that the number of pass conditions will be a given number. What we wish to have here is some expectation of what this value will be over time. This leads us to the concept of expected value (see Ross, p. 246). Because the outcomes of craps are discrete as opposed to continuous (see Siegrist, p. 151, 157), we say the random variables associated with them are discrete. For discrete random variables, the expected value is defined to be:

\[ E(X) = \sum_{x} x f(x) \quad (\text{Siegrist, p. 161}). \]
This is a sort of "average" value for the outcomes over a "large" number of repetitions. Since $N_n$ is the sum of indicator variables $W_1, \ldots, W_n$, we need to know something about the expected value for the indicator variables. Applying the definition above results in:

$$E(W) = \sum_{i=0}^{1} i f_w(i) = 0(1-p) + 1(p) = p,$$

Where $f_w$ is the distribution function for $W$. We may now attempt to determine the expected value for the sum of the indicator variables. Since the expectation for each $W_i$ is finite, the expected value of the sum is the sum of the expected values (see Ross, p. 260), we have:

$$E(N_n) = E(W_1 + \ldots + W_n) = E(W_1) + \ldots + E(W_n) = n(E(W)) = np.$$

For fair dice, this yields: $E(N_n) = .4929(n)$.

For ace-six flats, it yields: $E(N_n) = .4390(n)$.

IV. Bets won.

From our hierarchy of dependencies, we see that we need to know how we bet each turn in order to get the number of bets won. Since the number of bets won depends upon the number of players and the rules for betting pass or no-pass, we will make the following further assumption:

(6) Since the rules used here state that all players except the shooting player will be betting no-pass, we will say that, over time, the number of times a player is betting pass will behave as if the dice is given to a new player on each turn. Thus, $b_p = n/r$, where $n$ is the number of turns, and $r$ is the number of players. This can be evaluated in the simulation.

We now see that now we know the number of turns each player will be betting pass. We also know that the expected number of pass conditions in $n$ turns is $np$. Thus in $b_p$ turns, the expected number of successful pass bets for player $k$ is:
\[ S_k = (b_p)p = (n/r)p. \]

Using the same sort of analysis, we can determine the number of successful no-pass bets that a player will win, as follows:

\[ T_k = (n - b_p)P(W=0) = (n - (n/r)) \left( 1 - p \right). \]

So the total number of successful bets is, in terms of the parameters \( n \), and \( r \):

\[ S_k + T_k = (n/r) p + (n - (n/r)) \left( 1 - p \right). \]

V. Bets lost

This can be determined in a manner similar to the one for the number of bets won, because the two numbers depend upon the same things. The difference lies in the fact that you want to know the number of times a no-pass condition occurred when the player was betting pass. Analysis yields:

\[ U_k + V_k = (n/r) \left( 1 - p \right) + (n - (n/r)) (p), \]

where \( U_k \) is the number of pass bets lost, and \( V_k \) is the number of no-pass bets lost. In our case, since each player bets every turn, the sum of the bets lost and the bets won should total to \( n \), which they indeed do.

VI. Amount won or lost

This amount, call it \( C \), will depend on the amount bet on each turn, a parameter we will call \( A \).

From the rules we are operating under, this amount will be the same from turn to turn. We say that the total amount won or lost will be the sum of the amount won minus the sum of the amount lost. The amount won is the amount bet per turn, multiplied by the number of successful bets. Applying similar analysis for the amount lost yields:

\[ C = A(S_k + T_k) - A(U_k + V_k). \]

Using the substitutions for \( S_k \), \( T_k \), \( U_k \), \( V_k \), and simplifying, we obtain:

\[ C = A \left( 4(n/r)p - 2(n/r) - 2np + n \right). \]
Clearly, as \( n \) goes to infinity, \( C \) will as well which seems to bode well for craps players. However, actual casinos have different rules which will alter this expected outcome. Techniques similar to the ones offered here can be used to evaluate the effect of these rules.

**Evaluation**

The above developed model is used to predict an average amount won in \( n \) turns of craps. To test this result, the number of turns would need to be fixed, and a number of simulations of \( n \) turns run. Taking note of the average of the amount won by each player, an average of this value could be obtained to compare to the predicted value of, using 100,000 turns as an example, $11,360. Simulating for 20 runs of 100,000 turns each resulted in an observed average of $11,197. This seems to offer empirical evidence that the above result can, in fact, be trusted.

In a mathematical sense, this result does not seem to have a firm grounding. The assumption that the number of times a player will be betting pass will converge to some average value over time is not firmly supported. A more mathematically rigorous accounting of the behavior, suggested by Dr. Siegrist, follows:

Let \( Z \) be the random variable which represents the net amount won during a game.

For the case of a pass, one player (the shooting player) will win, all others will lose:

\[
Z = A(1 - (r - 1)) = A(2 - r) .
\]

For the no-pass case, one player loses and all others win, so:

\[
Z = A((r - 1) - 1) = A(r - 2) .
\]

Thus, in each case, the average amount won per player will be:

\[
Z = A(2 - r)/r = A((2/r) - 1) , \text{ and }
\]

\[
Z = A(r - 2)/r = A(1 - (2/r)) .
\]

The expected value of this variable will then be given by:

\[
E(Z) = p(A((2/r) - 1)) + (1 - p)(A(1 - (2/r)) .
\]

Let \( Z_i \) be the average amount won per player on turn \( i \). The above, random variable \( C \), becomes:

\[
C = (Z_1 + Z_2 + ... + Z_n) .
\]

Since for each game \( Z \) is independent and has the same distribution, the expected value of \( C \) is:
\[ E(C) = (E(Z_1) + E(Z_2) + \ldots + E(Z_n)) = nE(Z). \]

So with the same inputs as above, (i.e. \( n = 100,000 \); \( r = 10 \); \( A = $10 \)) we get:

\[ E(C) = n[ p(A((2/r) - 1)) + (1 - p)(A(1 - (2/r)) ] = $11,360. \]

Since this analysis yields the same result as the analysis above, I feel more confident about the result obtained. Perhaps a future direction for this project would be to provide some sort of rigorous proof for the assumption made.

The scope of the above model is somewhat limited by the rules of craps that were assumed at the beginning. Future versions of the model can incorporate different rules for betting, and perhaps an entire class of models can be produced. Casino versions of craps are altered so that the expectation of winning is negative. Models based on such rules could be used to determine a method of minimizing the losses.
Notes on the Simulation

Instructions

These are the instructions for using the craps game simulation located at:

http://www.cs.uah.edu/cs/students/jsaint/dice.html.

1. Using a forms capable web browser, open the URL listed above. The screen will be divided into two sections, Required Parameters, and Display options.

2. In the Required Parameters section, first enter the number of turns to run the simulation. This can be a value between 1 and the maximum long integer in C. Next enter the number of players in the simulated game, between 1 and 20. The third required parameter is the amount to bet, in whole dollar amounts, on each turn. Play initially starts with each player having $100. The final required parameter is the type of dice to use, fair or ace-six flats. If both options are selected, ace-six flats are assumed. Each of these values will be reported by the simulation in the Results section of the report screen.

3. In the Display Options section, any of the following options may be selected.

   Display relative frequencies for the scores - Selecting this option will cause the simulation to display a chart of the relative frequency of the occurrence of each possible outcome of the roll of two dice.

   Number of turns to display - This can be a value between 0 and the number entered in the Number of Turns field. If the number to display field is left blank, it is assumed to be 0.

4. If mistakes have been made, the Clear Form button will clear all fields and selected options.

5. Once the required parameters have been entered and the desired display options have been selected, click the Submit Form button to send the entered data to the simulation.

Implementation Notes

The implementation notes are divided into the following sections.

I. Algorithm used.

II. Implementation of the algorithm.

III. Reasons for the interface.
I. Algorithm used.

The heart of the craps simulation is a random number generator, which produces values between 1 and 6. This is done two separate times for each roll, and the sum of the two is taken as the score. Dependent upon this score and the current bet-type of each player, money is either added or subtracted from each player's total. The algorithm works as follows:

Program Play_Craps
Var
i : index;
diel, die2, score : integer;
Begin
    die1 := get_random;
die2 := get_random;
score := die1 + die2;
until (n turns simulated)
do
    if (pass condition) then
        foreach player betting pass
            player_total := player_total + bet amount;
    foreach player betting no-pass
        player_total := player_total - bet amount;

    if (no-pass condition) then
        foreach player betting pass
            player_total := player_total - bet amount;
        foreach player betting no-pass
            player_total := player_total + bet amount;
        if (no-pass on a point situation) then
            give dice to next player;
        else
            current player shoots again;
End

II. Implementation of the algorithm.

This algorithm was implemented in C on the UNIX operating system. The operating system choice was driven by concerns of speed, and was influenced by availability of suitable machines. C was chosen as the language due to my own familiarity with the language and its ease of use in a UNIX environment. Once chosen, the simulation program was designed to use command line arguments to accept the parameters to
the simulation as well as the display options. Thus, while the simulation is running, there is no interaction with the user. This was driven by the desire to be able to run the simulation for large numbers of turns in a short amount of time. Command line arguments were also chosen so that the simulation could be called from within a Perl script using the elements of an associative array of parameters as the command line arguments. The main body of the Perl script is used to translate the parameters, which come from the web server in Common Gateway Interface (CGI) format (see Tittel, 203). These are the values which the user has entered on the web-page. Listings for the code are included in Appendix B.

III. Reasons for the interface.

The World Wide Web was chosen as the user interface, because I believe that the ideas contained here can be useful to anyone interested in random processes, from gambler to mathematician. The Web makes it possible for anyone who is interested to test conjectures about the process simulated without the need of building this simulation from scratch. This is in addition to the already well-known benefits of Web publishing to the speed, availability and cost of publication.
Future Directions

There are three areas in which this specific project may be expanded. These are the model, the simulation, and the interface. Other expansions could include the addition of similar analysis and simulation for other games of this nature.

I. The Model

Future directions for the model include versions of the model under different sets of rules. Essentially a new model will be generated for each new set of rules. Also, allowing changeable bets based on a specific strategy could be added to the model.

II. The Simulation

As new versions of the model are produced, the simulation will have to be changed to adapt the play of the game to the different sets of rules used. Since the rules most likely to change involve the rotation of the dice among the players and the allowed strategies of betting, this will involve the addition of subroutines for each set of rules. Which betting strategy used, for example, would depend on the user's input.

III. Interface

Future directions for the interface will involve some sort of graphical display for the results of the simulation. This will perhaps be accomplished by using such relatively new tools such as Java.
Bibliography and Acknowledgements

Works Consulted


Reference Works


Acknowledgments

Images for the web pages were created by Rob Thomas, rthomas@hiwaay.net

My advisor on this project was Dr. Kyle T. Siegrist, siegrist@math.uah.edu

This project was completed under the guidelines of the UAH Honor's Program.
Simulation of a Craps Game

For instructions on how to use the simulation, refer to the Notes on the Simulation document. For an explanation of the rules used, please refer to the Game of Craps document.

Required Parameters

Number of Turns
Number of players (max 20)
Amount to bet each turn
Choose one of the following
Fair Dice
Ace-Six Flats (loaded dice)

Display Options

Number of Turns to Display
Display Relative Frequencies for scores

Back to the Table of Contents.

Although this server is maintained by the University of Alabama in Huntsville, Computer Science Dept, the content of student home pages are works of their respective authors. If you have administrative questions regarding student pages, please contact us at support@cs.uah.edu.
Simulation of a Craps Game

Required Parameters

Number of Turns
Number of players (max 20)
Amount to bet each turn
Choose one of the following
Fair Dice
Ace-Six Flats (loaded dice)

Display Options

Number of Turns to Display
Display Relative Frequencies for scores.

Back to the Table of Contents.
Number of Wins: 4885
Winners(net):12840
Player: 8
Number of Shots: 9990
Number of Wins: 4961
Winners(net):16300
Player: 9
Number of Shots: 9793
Number of Wins: 4741
Winners(net):11440
Player: 10
Number of Shots: 10150
Number of Wins: 5084
Winners(net):18020

Simulation of a Craps Game

Required Parameters
Number of Turns
Number of players (max 20)
Amount to bet each turn
Choose one of the following
Fair Dice
Ace-Six Flats (loaded dice)

Display Options
Number of Turns to Display
Display Relative Frequencies for scores.

Back to the Table of Contents.
Appendix B: Source Code Listings
**For a general description of how this simulation was implemented, */
* please refer to http://www.cs.uah.edu/cs/students/jsaint/simimp.html. */

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>

define MAX_PLAYERS 20
#define START_AMOUNT 100

typedef struct player_M{
  int num_shoots,
      num_shoots_win,
      is_shooting,
      bet_pass,
      bet_amount,
      total_money,
      start_money;
}player;

* This function generates two random numbers between 1 and 6 with */
* either equally likely outcomes or weighted outcomes, based on */
* the variable pflag. The sum of these two scores is returned. */

int get_rand(char pflag)
{
  int dice, temp, temp2;
  if (pflag == 'f')
  {
    temp = rand();
    temp2 = rand();
    dice = ((temp % 6) + 1) + ((temp2 % 6) + 1);
  }
  else
  {
    dice = 0;
    temp = (rand() % 8) + 1;
    temp2 = (rand() % 8) + 1;
    if (temp < 3)
      dice += 1;
    else if (temp > 6)
      dice += 6;
    else
      dice += (temp - 1);
    if (temp2 < 3)
      dice += 1;
    else if (temp2 > 6)
      dice += 6;
    else
      dice += (temp2 - 1);
  }
  return (dice);
}

* This function assigns the theorital probabilities of each score to */
* an element of an array. */
void assign_probs(float prob1[11], float prob2[11])

prob1[0]=1.0/36.0;
prob1[1]=1.0/18.0;
prob1[2]=1.0/12.0;
prob1[3]=1.0/9.0;
prob1[4]=5.0/36.0;
prob1[5]=1.0/6.0;
prob1[6]=5.0/36.0;
prob1[7]=1.0/9.0;
prob1[8]=1.0/12.0;
prob1[9]=1.0/18.0;
prob1[10]=1.0/36.0;
prob2[0]=1.0/16.0;
prob2[1]=1.0/16.0;
prob2[2]=5.0/64.0;
prob2[3]=3.0/32.0;
prob2[4]=7.0/64.0;
prob2[5]=12.0/64.0;
prob2[6]=7.0/64.0;
prob2[7]=3.0/32.0;
prob2[8]=5.0/64.0;
prob2[9]=1.0/16.0;
prob2[10]=1.0/16.0;

return;

* This is the function which actually emulates the play of craps. An */
* integer value is returned based on the outcome of the turn. */

int play_craps(int print_flag, long *roll_num, char pflag, int freq[11])

int point=0;
long score_val;
long roll;

roll=1;
score_val=get_rand(pflag);
freq[score_val-2]++;

if (((score_val == 7) || (score_val == 11)) /* natural */
{
  if (print_flag)
    printf("Shooter rolled a %d NATURAL WIN", score_val);
  *roll_num+=roll;
  return(1);
}
else if (((score_val == 2) || (score_val == 3) || (score_val == 1
{
  if (print_flag)
    printf("Shooter rolled a %d CRAP OUT", score_val);
  *roll_num+=roll;
  return(0);
}
else /* a 4, 5, 6, 8, 9, or 10 was rolled */
{
  if (print_flag)
printf("Point is %d -- Following rolls: ", score_val);
score_val = 0;

/*This is the play after the initial roll. It continues until the */
/* player rolls the point again, or a seven is rolled. */
while ((score_val != point) && (score_val != 7))
{
    score_val = get_rand(pflag);
    if (print_flag)
        printf("%d, ", score_val);
    freq[score_val-2]++;
    roll++;
    if (score_val == 7)
    {
        if (print_flag)
            printf(" -- LOSE with point %d", *roll_num+=roll);
    }
    if (score_val == point)
    {
        if (print_flag)
            printf(" -- WIN with point %d.", point);
        *roll_num+=roll;
        return(1);
    } else
        return(-1);
}

* This function finds the next shooter. */
int get_next_player(player list[], int total_players)

int index=0;
int shooter;
while (! list[index].is_shooting ) /* find the shooting player */
    index++;
list[index].is_shooting=0;
shooter=(index + 1)%total_players; /* give dice to next player */
list[shooter].is_shooting=1;
return(shooter);

int main(int argc, char *argv[])

char pflag;  /*flag for setting type of dice */
long turn,  /*current turn */
    num_turns,  /*total number of turns*/
    num_rolls,  /*number of rolls*/
    ind;  /*general purpose index variable*/
int error=0;
int print_flag=0;
int rel_freq[11]; /*relative frequencies for the scores */
float prob1[11];
float prob2[11];
float prob_ar[11];
float freq;
float avg_winnings=0.0;
nint num_wins=0;
nint num_disp=0;
nint turn_result;
nint num_players;
nint pass_dice;
player play_list[MAX_PLAYERS];
player *current_player;
nint curr_player_index=0;

if (argc > 0)
{

/* initializations */

printf("<HTML>");
srand(time(0));
assign_probs(prob1, prob2);

num_rolls=0;
nun_turns=0;

for(ind=0;ind<11;ind++)
    rel_freq[ind]=0;

num_turns=atoi(argv[1]);

/*get the probability to use*/
if (*argv[2] == 'f')
    pflag='f';
else if (*argv[2] == 'l')
    pflag='l';
else
{
    printf("ERROR: YOU MUST SELECT THE DICE!!\n");
    error = 1;
}

num_disp=atoi(argv[3]);

num_players=atoi(argv[5]);
for(ind=0;ind<num_players;ind++)
{
    play_list[ind].num_shoots=0;
    play_list[ind].num_shoots_win=0;
    play_list[ind].is_shooting=0;
    play_list[ind].bet_pass=0;
    play_list[ind].bet_amount=atoi(argv[6]);
    play_list[ind].total_money=START_AMOUNT;
}

*get first player*/
current_player=&play_list[0];
current_player->is_shooting=1;

pass_dice=0;
*end of initializations */

*play craps!!*/
if (error != 1)
{
    printf("<H2>Craps Game Results</H2>

    for(turn=1; turn<=num_turns; turn++)
    {
        if (pass_dice)
        {
            curr_player_index=get_next_player(play_list, num_players
            current_player=&play_list[curr_player_index];
        }
        current_player->num_shoots++;
        if (turn<=num_disp)   /* if still in display boundaries */
            print_flag=1;
        else
            print_flag=0;

        if (print_flag)
            printf("<H3>Turn -- %d :: Player -- %d is shooting.</H3>

        /* set up betting for the turn */
        for(ind=0; ind<num_players; ind++)
            play_list[ind].bet_pass=0;

        current_player->bet_pass=1;
        turn_result=play_craps(print_flag, &num_rolls, pflag, rel_freq);
        if (turn_result == -1)
            pass_dice=1;
        turn_result=0;
    } else
        pass_dice=0;
    num Wins+=turn_result;
    current_player->num_shoots_win+=turn_result;

    /* Bookkeeping for money exchanged during the turn */
    for(ind=0;ind<num_players;ind++)
    {
        if (turn_result)
            if (play_list[ind].bet_pass)
                play_list[ind].total_money+=play_list[ind].total_money;
    }
if (!play_list[ind].bet_pass)
    play_list[ind].total_money+=play_list[ind].total_money
else
    play_list[ind].total_money-=play_list[ind].total_money


else
    play_list[ind].total_money-=play_list[ind].total_money

/* Calculate the average winnings */
for(ind=0;ind<num_players;ind++)
    avg_winnings += play_list[ind].total_money - START_AMOUNT;
avg_winnings /= num_players;

* Print the summary */
printf("<H2>Summary:\n\n

Number of turns: %d\n\n\nNumber of wins: %d\n\n\nAverage number of rolls per turn: %.0f\n\n\n\n* Comparison to the predicted outcomes */
if (pflag == 'f')
    printf("Type of dice: fair\n\n\n\nelse
    printf("Type of dice: ace-six flats\n\n\n\n/* for each score */
if (*argv[4] == 'y')
    {
        printf("<H3>Probabilities and Relative Frequency\n\n\n\nif (pflag == 'f')
        {
            for(ind=0;ind<ll;ind++)
                prob_ar[ind]=probl[ind];
        }
else
            for(ind=0;ind<ll;ind++)
                prob_ar[ind]=prob2[ind];
for(ind=0;ind<ll;ind++)
    {
        freq=((float) rel_freq[ind])/(float) num_rolls;
        printf("%d %s %f %f \n", ind+2, prob_ar[i], prob_ar[i], freq);
    }
}
/* for the probability of passing and of the amount won */
if (pflag == 'f')
    {
        printf("Theoretical Probability of Passing: .4929<P>\n\n\n\nelse
        printf("Average amount won (predicted): %.2f<P", atoi(argv[6])*(4*(num_tur
    }
else
    {
        printf("Theoretical Probability of Passing: .4390<P>\n\n\n\n
printf("Average amount won (predicted): \%.2f", atoi(argv[6]) * (4 * (num_turns)));

freq = ((float) num_wins) / ((float) num_turns);
printf("Observed Probability: \%.1f", freq);
printf("Observed Average: \%.2f", avg_winnings);

* Print player data */
printf("<HR><H3>Player Data</H3>");
for (ind = 0; ind < num_players; ind++)
{
    printf("<H3>Player: %d</H3>
    Number of Shots: %d
    Number of Wins: %d

* Display parameter form */
printf("<HR>");
printf("<HR><H2>Simulation of a Craps Game</H2>

<FORM METHOD="POST" ACTION="http://www.cs.uah.edu/cgi-bin/jsaint/mytry.cgi"

<H2>Required Parameters</H2>
<input size=25 name="n" type="text" value="" placeholder="Number of Turns">
<br><p>
<input size=10 name="players" type="text" value="" placeholder="Number of players (max 20)">
<br><p>
<input size=25 name="bet" type="text" value="" placeholder="Amount to bet each turn">
<br><p>
<H3>Choose one of the following</H3>
<input type="radio" value="fair" name="die" value="fair" checked="checked">Fair Dice
<br><p>
<input type="radio" value="loaded" name="die" value="loaded">Ace-Six Flats (loaded dice)
<br><p>
<H2>Display Options</H2>
<input size=25 name="turns" type="text" value="" placeholder="Number of Turns to Display">
<br><p>
<input type="radio" value="y" name="display" value="y">Display Relative Frequencies
<br><p>
<input type="reset", value="Clear Form"/>
<br><p>
<input type="submit", value="Submit Form"></FORM>");

printf("<P><HR>"");
printf("Back to the <A HREF="http://www.cs.uah.edu/cs/students/jsaint/craps.html"/>

else
    printf("ERROR PROCESSING REQUEST TRY AGAIN: <H1> <H2> IF PROBLEM
    PERSISTS SEND MAIL TO jsaint@cs.uah.edu</H2>"");
printf("</body></HTML>");