The Effect of Multiple Unskilled Competitors on the Success of a Skilled Player in a Model Spelling Bee-Like Game

Robert Monahan, Department of Computer Science, Mathematics, and Engineering and Jeffrey R. Groff, Institute of Environmental and Physical Sciences, Shepherd University

In this paper we further investigate the role of randomness in competitions which was inspired by the results of other recent books and studies on the theory randomness. After a brief literature review, we construct a mathematical model of a game with spelling bee-like rules using an absorbing state Markov Chain. The model is then used to explore an interesting arrangement of competitors that explores the effect of multiple unskilled players on the success of a skilled player. We found that randomness plays quite a dominant role in competitions and the outcomes of events are not always what is expected or perceived.

Introduction

Biondo et al, conducted a study based on the economic premise that the stock market has predicable values and expected results. Because of this premise, professional economists sought to construct trading strategies that predicted the future asset prices of the market (Biondo 2013).

Biondo et al., however, challenged this premise by comparing the results of four common trading strategies to a purely random strategy. Interestingly, Biondo et al. found no significant difference between the outcomes of the professional trading strategies and the purely random strategy. The extreme nature of the wins and losses in the professional trading strategy, in fact, may, unnecessarily, be a more risky option than the random trading strategy.

The research conducted by Biondo et al. is suggesting an overarching theme in competitions that is vastly present but easy to forget about. This theme is the theory of randomness. Author Nassim Taleb discusses in his book *Fooled by Randomness*, how ubiquitous randomness is in everyday life and describes how easy it is to get caught in a human's attempt to rationalize and explain events that are purely random in nature (Taleb 2005).

Taleb illustrates this misconception by discussing the apparent animal-like shapes that clouds take when being closely watched. These clouds, however, are mere random generations in nature but are perceived by humans as having structure or reason (Taleb 2005).

The research conducted on randomness by Biondo and the insight of Taleb's book has laid the foundation for our research. We seek to answer questions such as: can we quantify how much ability matters in competitions? what role does randomness play in competitions? will the most qualified player succeed?

Within this research we will further investigate the theory of randomness by building a mathematical model of a game with spelling bee-like rules. First, we examine the construction of the game and its rules. Second, the model is checked for validity against another game modeling strategy. Third, we utilize our model to explore a particular organization of competitors that lead to interesting results. Fourth, we discuss the primary results of our research. Finally, we will draw conclusions from our research, discuss the implications, and make suggestions for future areas of study.

Formulation of the Model

In order to explore the interplay between chance and skill in a game with multiple players, a model game with spelling bee-like rules was constructed. The game is composed of N players that compete in a round-based fashion. Each player can either find success or failure during each round just like each competitor in a spelling bee can spell a word correctly or incorrectly each round. The probability of player n finding success during each round is a parameter of the model, h_n, and is a reflection of that player's skill. Therefore, the probability of failure for player n is $(1 - h_n)$.

If a player fails in any given round, that player cannot re-enter the game unless all other players also fail during that round. In the event that all players fail, no players are eliminated and the round is repeated. On the other hand, if at least one player succeeds during the round, all players who failed are eliminated from subsequent rounds. The game continues until only a single player remains.

A game with rules as outlined above can be formulated as a discrete-time absorbing-state Markov Chain (Norris 2006) as follows. First, the success probabilities of all N players are organized into a "skill" vector $h = (h_1\ h_2\ h_3 \cdots h_N)$, and the current state or competitive status of each player is structured is a state vector $S = (S_1\ S_2\ S_3 \cdots S_N)$ where S_n is a binary value with 1 or 0 indicating player n as still active in, or eliminated from, the game, respectively. For example, the state vector $S = (1, 1, 0, 1)$ indicates that players 1, 2, and 4 are still active while player 3 has been eliminated.

The probability of transitioning from one state vector to another depends only on the current state of each of the players and their respective success probabilities. Thus, the Markov condition is satisfied (Stewart 1994). Consider for example a two-player game. The state-space of such a game—that is, the enumeration of all the possible states—is $S \in \{(1, 1), (1, 0), (0, 1)\}$. Notice that the size of this state space is equivalent to $2^N - 1$, which is true regardless of the number of players.

The probability of transitioning from state vector $S = (1, 1)$ with both players active to $S = (1, 0)$ with player two eliminated and player one victorious is $q = h_1(1 - h_2)$. Similarly, the probability of transitioning from state vector $S = (1, 1)$ with both players active to $S = (0, 1)$ with player one eliminated and player two victorious is $q = (1 - h_1)h_2$. On the other hand, transitioning from state $S = (1, 0)$ to state $S = (0, 1)$ is forbidden and has probability zero. In fact, both of these states represent different final outcomes of the game (player one victorious versus player two victorious) and are thus absorbing states out of which no transitions are allowed. Figure 1 illustrates the state space and transition probabilities for a two player game.

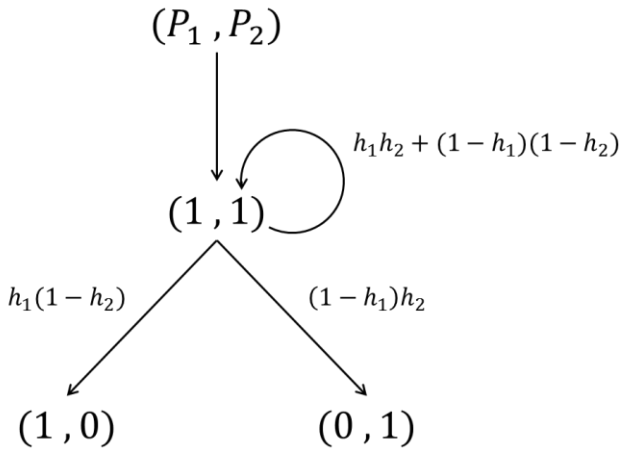


Figure 1: State Transition Diagram of a two player game. 1 is a binary representation of active players and 0 is a binary representation of inactive players.

The probabilities for all allowed transitions between states can be summarized in matrix form. For example, the transition probability matrix for the two-player game is

$$P = \begin{pmatrix} \diamond & h_1(1 - h_2) & (1 - h_1)h_2 \\ 0 & \diamond & 0 \\ 0 & 0 & \diamond \end{pmatrix}$$

where q_{ij} is the transition probability from state i to state j and the $\diamond$ are selected to ensure that the row sums are equal to one,

making P a stochastic matrix. This condition reflects the necessity that the total probability of all possible transitions out of the current state is equal to one.

Monte Carlo Simulations

After specifying the number of players, N, and each player's skill level, h, a game can be simulated using Monte Carlo methods as follows. First, a random number between zero and one is drawn from a uniform distribution for each of the active players each round of the game. If the random number drawn for each player is greater than that player's skill level, the player is unsuccessful in that round. If no active player is successful then the round is repeated. However, if at least one player is successful, $S$is updated to indicate any players eliminated during that round and the game continues. Each game is initialized with all players active, $S = (1\ 1\ 1\ \cdots 1)$, and ends when only a single player remains active, $\Sigma S = 1$. Many games can be simulated to determine the probability of each player winning any individual game.

Direct Matrix Calculations

The probability of each player winning any individual game can also be calculated directly using matrix-analytic methods. First, the transition probability matrix is reorganized so all the absorbing states are grouped together,

$$P = \begin{pmatrix} P_{TT} & P_{TA} \\ Z & U \end{pmatrix},$$

where P_{TT} is a sub-matrix containing the probabilities for transitions between transient states and P_{TA} is a sub-matrix containing the probabilities for transitions from transient states into one of the absorbing states. The Z sub-matrix is all zeros reflecting the fact that transitions from absorbing states to transient states are forbidden. Meanwhile, each row of the U sub-matrix has a single non-zero entry equal to one, reflecting the fact that once an absorbing state is entered it is never left.

The probability of ending up in each of the absorbing states can then be found using

$$A = (I - P_{TT})^{-1} P_{TA}$$

where I is an identity matrix commensurate in size with P_{TT} (Stewart 1994). This matrix has a row for each transient state and a column for each absorbing state where A_{mn} is the probability of ending in absorbing state n assuming the game began in transient state m. Assuming the states are ordered such that the first transient state is the state with all players active then

$$A_{1j} = (\pi_1, \pi_2, \pi_3, \cdots, \pi_N)$$

gives the probability of each player winning the game.

All simulations and calculations were carried out using MATLAB running on either a modest laptop or desktop computer.

Results

Validating the Model: A Comparison Between Direct Matrix Calculations and Monte Carlo Simulations

If the direct matrix calculation was formulated correctly, it would yield comparable results to repeated Monte Carlo Simulations.

Consider a 4 player game with "skill" vector $h = (0.4\ 0.5\ 0.6\ 0.7)$. The direct matrix calculation returns the probability matrix

$$p = (0.0944\ 0.1661\ 0.2791\ 0.4605)$$

which shows the probability of player 1, 2, 3 or 4 winning the entire competition. Using the same "skill" vector h from before, the result from 10 Monte Carlo Simulations is (0 0.4 0.2 0.4). The aforementioned outcome matrix shows that player 1 won 0% of the games (0 games), player 2 won 40% (4 games), player 3 won 20% (2 games), and player 4 won 40% (4 games). These results are not unique as running 10 additional Monte

Carlo Simulations could yield different outcomes.

These results are quite different than the Direct Matrix Calculation's results and are summarized in Figure 2. Figure 3 compares 100 Monte Carlo Simulations to the Direct Matrix Calculation.

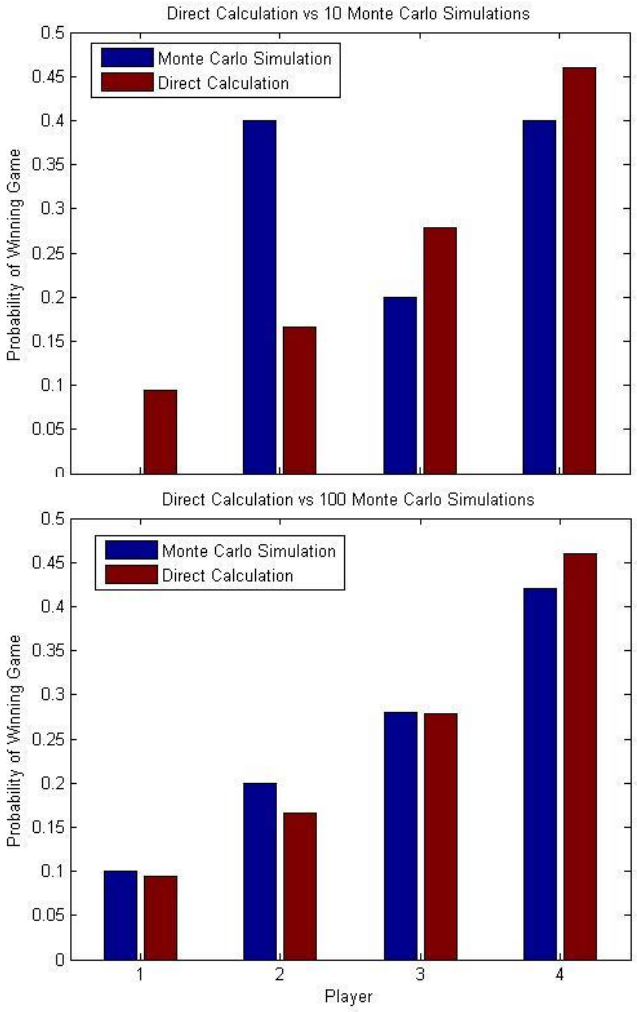


Figure 2: Compares the Direct Calculation results to 10 Monte Carlo Simulations. Figure 3: Compares the Direct Calculation results to 100 Monte Carlo Simulations. Skill vector used: $\boldsymbol{h} = [\mathbf{0.4\ 0.5\ 0.6\ 0.7}]$.

Notice how the overall differential between the Monte Carlo Simulations and Direct Matrix Calculation in Figure 3 is significantly smaller than that of Figure 2. The results began to agree after repeated Monte Carlo Simulations. Figure 4 summarizes the differentials between the outputs of repeated simulations and the direct calculation.

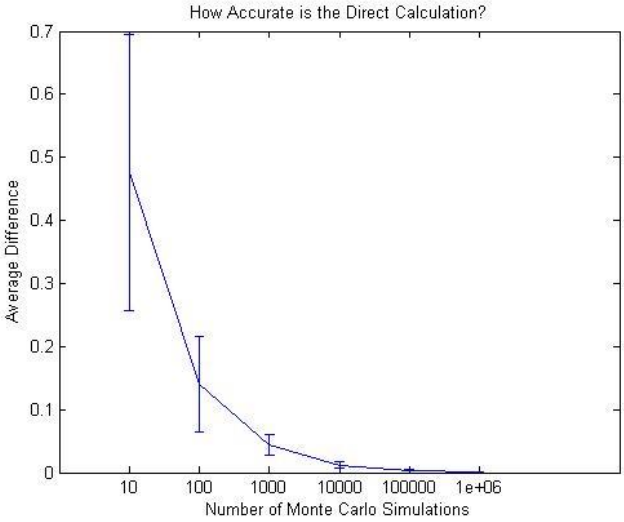


Figure 4: Compares the average difference between the Direct Calculation and Monte Carlo Simulation as a function of the number of Monte Carlo Simulations.

Figure 4 illustrates how as the number of simulations increases the differential between the simulation and direct calculation decreases. In other words, the Monte Carlo Simulation agreed with the Direct Matrix Calculation.

On a side note, the direct calculation requires a heavy computational load for competitions with larger amounts of players because, again, the Markov Chain takes the form of a matrix with $2^N - 1$ rows and columns. Because the Monte Carlo Simulations validated the results of the direct calculation, the Monte Carlo Simulation can be used to approximate the probability of any given player winning the competition when the number of players is too large to use the direct calculation.

A Diamond In The Rough: A Study of a Skilled Player's Probability of Success When Competing Against Multiple Unskilled Players

Consider a 2 player game in which player 2 has "skill" $h_2 = 01$. How "skilled" must player 1 be in order to maintain a win percentage of at least 50% in the competition? In this trivial example, h_1 must be 0.1 as well. The outcomes of this 2 player game mirror a simple coin toss. What happens, however, as

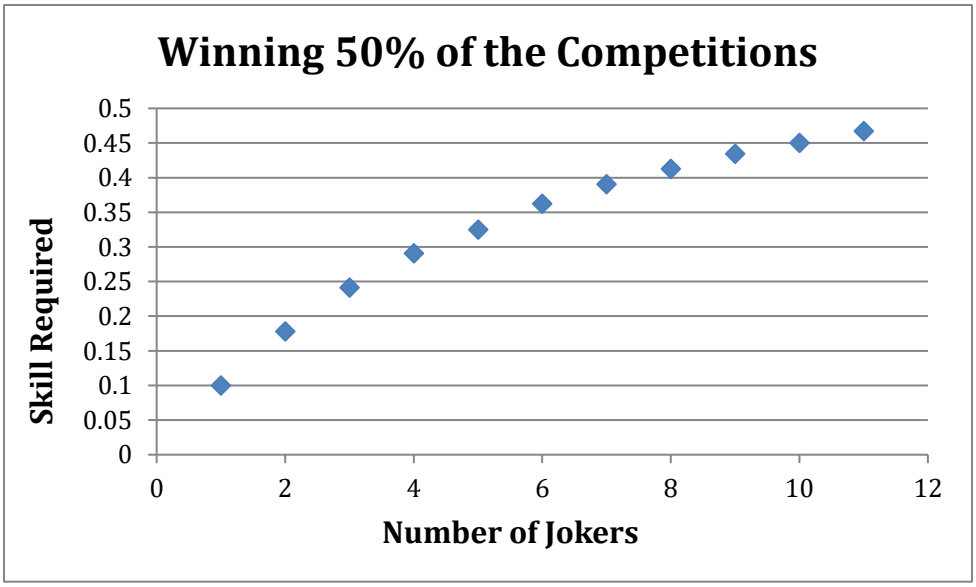
the number of unskilled players increase? The Diamond in the Rough Model seeks to answer this question.

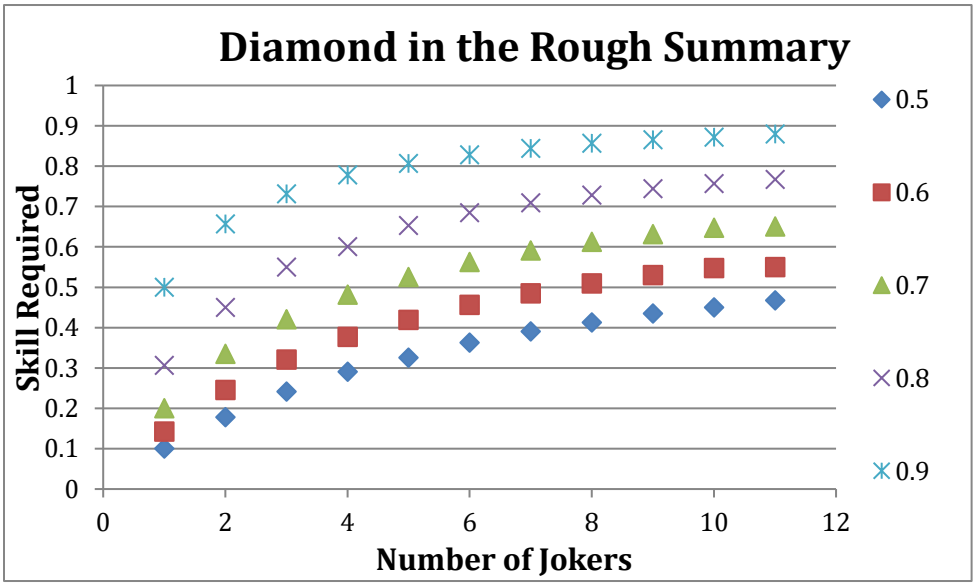
In this model with N players, there are $N-1$ unskilled players, called Jokers, and 1 skilled player, called the Ace. The initial skill vector is, therefore, given by $h = (h_a\ h_{j1}\ h_{j2}\ \cdots h_{j(n-1)})$ where h_a is the "skill" of the Ace that must be determined and $h_{ji} = 0.1$ is the standard Joker "skill" where $i \in 1, 2, \cdots (n-1)$.

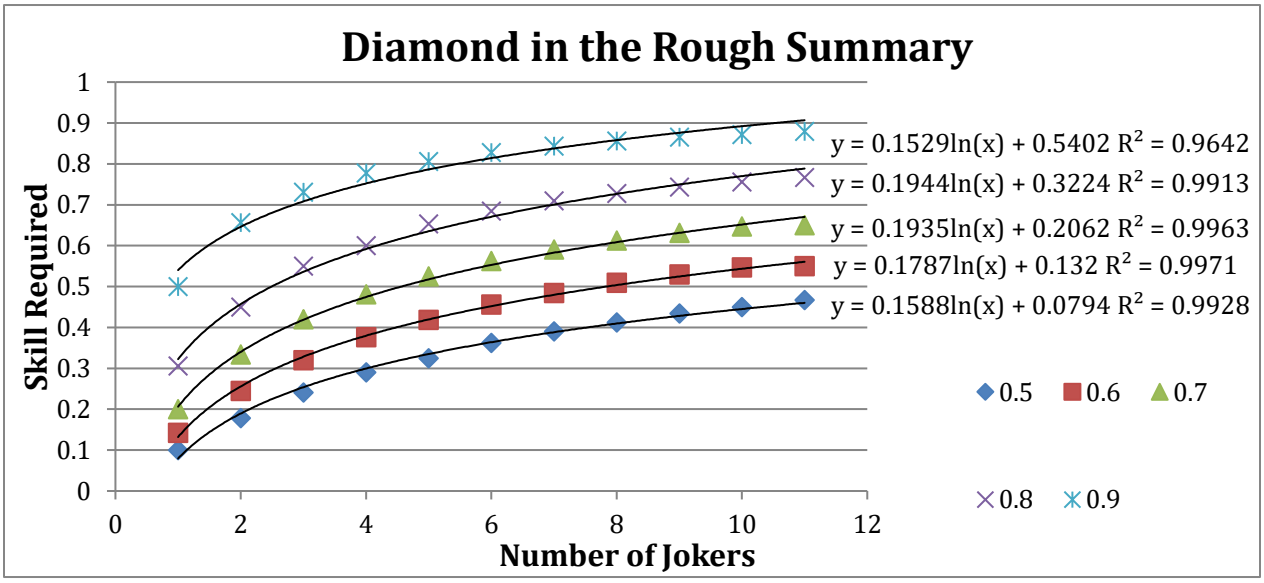
Using the direct calculation, the "skill" required of the Ace to beat multiple unskilled competitors 50% of the competitions is summarized in Figure 5.

What if the Ace desires to overcome the Jokers more than 50% of the competitions? Figure 6 shows the results with desired win percentages of 60%, 70%, 80%, and 90%.

Figure 6 shows that the skill required of the Ace player to overcome the Jokers increases as the number of Jokers increase. It can also be seen that the skill required of the Ace increases as the desired win percentage increases.







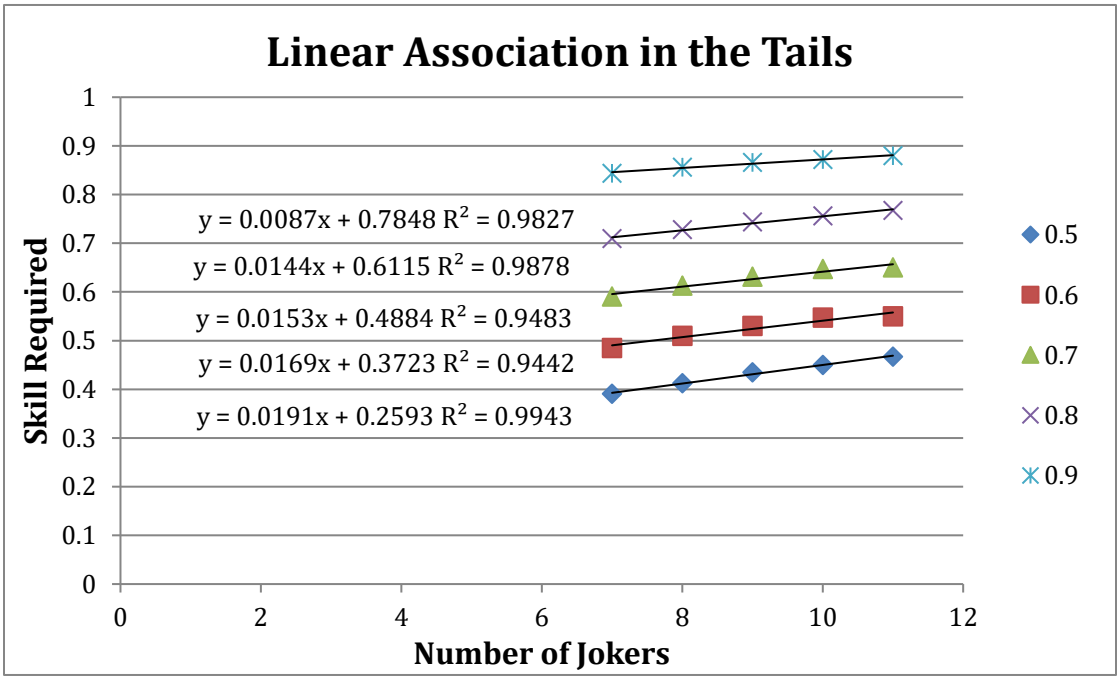


Figure 5: Shows the skill required of the Ace player to beat a certain number of Jokers 50% of the time. Figure 6: shows the skill required of the Ace player to beat a certain number of Jokers 50%, 60%, 70%, 80% and 90% of time, respectively. Figure 7: A reproduction of Figure 5 but with logarithmic trend lines and corresponding correlation coefficients. Figure 8: Focuses on the last 5 points of each win percentage (50%, 60%, 70%, 80%, and 90%) illustrates their linear tendency with corresponding correlation coefficients.

Conclusions

Interestingly, the data in Figure 6 suggests a pattern or perhaps a plateau point. In other words, is there an Ace player skill other than the trivial skill required, 1, (premature limit) that is expected to beat any number of Jokers? Figure 6 is reproduced with corresponding logarithmic trend lines and correlation coefficients.

Overall, the correlation between these logarithmic functions and our data is pretty strong but it is important to keep in mind that these trend lines, although strongly correlated, are still approximations.

Looking back at the original data points in Figure 6, the "tail" end is visually

linear. Figure 8 examines the linear relationship in the tail of each desired win percentage.

Using the five logarithmic equations from Figure 7 and the five linear equations from Figure 8, we can approximate the number of Jokers (x) it takes to overcome the Ace player. In other words, we can set the Ace players skill required (y) to 1 and solve for the number of Jokers that require the Ace to be perfectly skilled. Figure 9 summarizes these findings.

Linear Limit	Win percentage	Logarithmic Limit
24.7	0.9 (90%)	20.2
27.0	0.8 (80%)	32.6
33.4	0.7 (70%)	60.5
37.1	0.6 (60%)	128.7
38.8	0.5 (50%)	329.4

Figure 9: Shows the expected number of Jokers to overwhelm an Ace player unless he had perfect skill using the five logarithmic and five linear equations.

Unfortunately, the logarithmic and linear trend lines do not suggest the existence of a premature skill required of the Ace player. As the number of Jokers increase and/or the desired win percentage increases, the skilled required of the Ace played indefinitely approaches 1, or perfect skill.

Although we couldn't find a premature limit point, it is pretty clear that we are able to answer our three introductory questions at the conclusion of this research. Can we quantify how much ability matters in competitions? What role does randomness play in competitions? Will the most qualified player succeed?

Discussion

Results from a Diamond in the Rough suggest that ability matters, to a degree, within competitions. Ability alone, however, is not the only influential factor because players do not compete in isolation. The outcome of a competition is affected by the number of players in the game as well as the skill of each of those players. In fact, an Ace player with a skill of 0.46 is expected to win 50% of the competitions against only 11 Jokers. Although the Ace was more than four times as "skilled" as any one of the Jokers, it didn't take many Jokers to overcome the Ace.

The underlying principle in this study is the existence of randomness in competitions. The most qualified, or in this context, skilled, is not necessarily the winner at the end of the competition. Events in competitions are not repeated until the expected winner, or most skilled player, has succeeded so the role of chance cannot be discounted. The existence of randomness in competitions really illustrates the limitations of a meritocratic view.

Applications of this research can be seen within elections. Whether the election demographics are federal, state, or local, the "skilled" or most informed voters can be drowned out by the sea of unskilled competitors. Does the most qualified candidate in the election win? Not necessarily, the role of chance and randomness may indirectly elect a candidate who is not the most qualified.

Looking back at the research conducted by Biondo et al., the stock market is a prime illustration of the dominance of randomness. Biondo et al. found that on average, the skilled trading strategies did not yield any significant difference in outcome when compared against a random trading strategy (Biondo 2013). In fact, the skilled trading strategies lead to more extreme and sporadic, or varied, results where the losses were more detrimental but the wins were greater than that of a random strategy.

The research and results in this study really suggest some sort of "mysterious" analytical solution. Is there an analytical generalization that can be made which connects the desired win percentage of the Ace, the number of jokers, and the skill required of the Ace player? Is there a more compact representation than the matrix built in this study? In fact, the matrix may, simply, be that compact representation.

Unfortunately, we were not able to formulate an analytical solution to the Diamond in the Rough and, therefore, further research is necessary.

Perhaps the greatest limitation to this research was the lack of computational power. All calculations and figures were generated in Matlab on a modest laptop computer. In a Diamond in the Rough, generating results for games with 12 Jokers actually depleted the laptops memory before completion. A more powerful computer should be able to collect data points for games that have more Jokers and, as a result, can make more accurate predictions regarding the skill required of an Ace player to beat a sea of Jokers a certain percentage of games.

In this study, particularly within the Diamond in the Rough, the Ace player competed against a sea of Jokers with homogeneous skill. All the Jokers were set to have a skill of $h_j = 0.1$. The matrix can certainly handle a competition where the Jokers' skill is heterogeneous but it was not explored in this study. Sample code for further exploration can be found in the Appendix.

References

Biondo A. 2013. Are Random Trading Strategies More Successful than Technical Ones? PLoS ONE 8(7): e68344. doi:10.1371 /journal.pone.0068344

Norris J.R. 2006. Markov Chains. New York (NY): Cambridge University Press.

Stewart W. 1994. Introduction to the Numerical Solution of Markov Chains. Princeton (NJ): Princeton University Press.

Taleb N.N. 2005. Fooled by Randomness: The Hidden Role of Chance in the Markets and in Life. New York: Random House.

Appendix

Monte Carlo Simulation Example MATLAB Program

```
% Define a skill vector for the game
h = [0.5 0.1 0.1]

% then, calculate the number of players
N = length(h);

% and specify the total number of games to be simulated.
nGames = 10000;

% Create a vector to count how many times each player wins the game
nSuccess = zeros(1,N);

% then play the game many times
for ii = 1:nGames

      % starting each time with all players active.
      S = ones(1,N);
      flag = 1;

      % Play each game so long as at least one player is still active
      while (sum(S) > 1)

            % and keep repeating each round of the game if no players
            % are successful.
            while (flag == 1)
            test = rand(1,N);
            SNew = (h.*S)>test;
            flag = sum(SNew)==0;
         end
         S = SNew;
         flag = 1;
    end

      % After determining who won each game, increment the win tally
      % for the player that won the game.
      nSuccess = nSuccess + S;
end

% After all games are played, calculate the probability of each player
% winning the game.
pSuccess = nSuccess./nGames
```

Direct Matrix Calculations Example MATLAB Program

```
% Define a skill vector for the game
h = [0.5 0.1 0.1]

% then, calculate the number of players
N = length(h);
```

```
% and the size of the state space, that is, the number of possible
% states in the Markov Chain.
nStates = 2^N-1;

% Find the probability of each player failing a round of the game.
f = 1-h;

% Find the binary representation of each state and convert this
% representation from a character array into a matrix.
stateSpace = flipud(dec2bin(1:nStates));
stateSpace = arrayfun(@str2num,stateSpace);

% Initialize the transition probability matrix
P = zeros(nStates,nStates);

% then build it by finding each element.
for ii = 1:nStates

     inow = stateSpace(ii,:);
     for jj = ii+1:nStates % j has to be greater then or equal to i

     jnow = stateSpace(jj,:);

          % if inow-jnow has negative elements then a zero turned
          % back to a one, which is forbidden
          if min(inow-jnow)>=0

               fail = bitxor(inow,jnow).*f;
               pass = bitand(inow,jnow).*h;

               P(ii,jj) = ...
               prod(fail(find(fail>0)))*prod(pass(find(pass>0)));
          end
     end
end

% Insert appropriate values into the diagonal elements of P so the row
% sum to one.
P = P + diag(1-sum(P,2));

% Rearrange P so the matrix is in this form P = (PTT PTA; Z U) where
% PTT holds the transition probabilities for transitions between the
% transient states, PTA holds the transition probabilities for
% transitions from transient states into the absorbing states, and Z
% and U are a commensurate matrix of zeros and a commensurate identity
% matrix, respectively.

iabs = find(sum(stateSpace,2)==1); % indices of absorbing states
nabs = length(iabs); % the number of absorbing states
itrans = find(sum(stateSpace,2)~=1); % indices of transient states
ntrans = length(itrans); % the number of transient states
inew = [itrans; iabs];
stateSpaceShuffle = [stateSpace(itrans,:) ; stateSpace(iabs,:)];
P = P(:,inew); % reshuffle the P matrix
P = P(iNew,:);
```

```
PTT = P(1:ntrans,1:ntrans);
PTA = P(1:ntrans,ntrans+1:ntrans+nabs);

% Calculate the probability of each player winning the game by first
calculating matrix A where A(i,j) is the probability of absorption
into state j when starting in state i.
A = inv(diag(ones(1,length(PTT)))-PTT)*PTA;

% The first row in A represents the probability of each player winning
the game because this row corresponds to the starting state with all
players active (the beginning of the game).
pSuccess = A(1,:);
```