Cribbage Simulation: Update

MATH 381 HW 8

Tim Ferrin, 1764309

December 1, 2020

I Introduction

The focus of this paper is to simulate different strategies that can be implement when playing the classic card game cribbage. It is an updated version of a previously written paper which didn't investigate how the different strategies for discarding and pegging affected each other. This paper extends upon the original simulation analysis and includes tables which do show how the strategies interact. The simulations are performed in a MATLAB script and the results are reported and analyzed in the sections below.

II Cribbage Rules

For those unfamiliar with the game's rules, a group of players (traditionally and in this paper two, but can include more) proceed through rounds of the game racking up points on a track on the game board, as shown in Figure 1. As points are earned, they are kept track of by "leapfrogging" the player's lagging peg ahead of the leading peg by the number of points earned. This is to keep track of what your previous point total was in case there is any doubt as to whether you advanced by the right number of points. This first player to the end of the board achieving 121 points or more is the winner of that game. Note that throughout the game, the numeric value of all face cards is 10, but that a Queen and King do not form a pair.

Each round of a game proceeds in three stages as follows;

1. Dealing: The players cut for the first deal and the player with the lower card deals. The dealer shuffles and deals six cards to the other player and themselves. Each player then chooses 2 of their cards to discard to the "crib". The crib is an extra set of cards that the dealer gets to utilize later in the game. The remaining four cards that each player has forms their hand that they can use for the next two stages. After both players have discarded to the crib the non-dealer cuts what is left of the deck and the dealer reveals the top card, often referred to as the "cut" or "flip" card. If the flip card is a jack, the dealer automatically scores two points for "his heels".

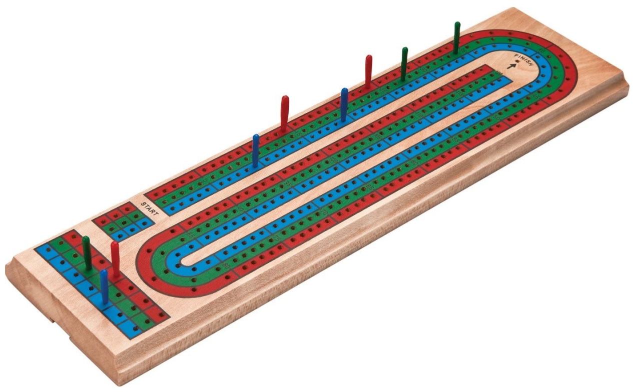

Figure 1: A standard cribbage board. Each color track and peg belongs to a different player. The section of track on the lower left end of the board is to keep track of the number of games won in a series for each player. This picture was taken from the Amazon ad for the "Mainstreet Classics Traditional Wooden Cribbage Board Game Set".

2. Pegging (or Playing): Starting with the non-dealer, the players take turns laying one card face up in front of them, stating the running total of the count (for example, the first player lays a 6 and says "six", the next player lays an eight and says "fourteen", and so on). This continues until the count is about to go above 31. If a player cannot play from the remaining cards in their hand without making the count go above 31, the say "go" and the counting continues with the other player. If this other player can continue to lay cards down without making the count exceed 31, they must do so. Once the count reaches 31 or neither player can lay cards down, the player who laid the most recent card down scores one point if the count is under 31 or two points if it is exactly 31. The count is then reset to zero and players with cards left in their hand continue this process. Players can score points during the pegging stage in the following ways:

   (a) For causing the count to reach exactly 15 a player scores 2 points.

   (b) Completing a pair (for example laying a king down immediately after the last card played was also a king) scores 2 points. Completing a three of a kind is the same as completing three different pairs, so this scores 6 points. Completing a four of a kind is the same as completing six different pairs, so this scores 12 points.

   (c) A run of three or more cards scores the number of cards in the run (For example if the following sequence occurs: P1:2, P2:4, P1:3, P2:5, P1:2, etc., P1 would get three points for the 2,4,3 run, P2 would get four points for the 2,4,3,5 run, and P1 would get four points for the 4,3,5,2 run.)

   When all cards have been laid down, play continues the next stage of the game.

3. Showing: Starting with the non-dealer the players display their hands and total up the number of points that is obtained through them *with the inclusion of the flip card*. Points are scored for:

(a) Combinations of any number of cards summing to 15 (2 points for each combination)

(b) Runs (1 point for each card in the run)

(c) Pairs (2 points each, so once again a three of a kind would be 6 points and a four of a kind would be 12 points)

(d) Flushes (A four-card flush scores 4 points and cannot include the starter card, a five-card flush scores 5 points)

(e) Nobs, aka having the Jack of the same suit as the flip card (1 point)

The dealer scores their hand with the flip card, places those points on the board, and then scores and places the points for the crib with the flip card as well. As a side note, a crib hand cannot score a four-card flush but can score a five-card flush with the flip card.

The game can be played with what is called ”Muggins”, which is where if a player at any point fails or forgets to claim any amount of points they have earned, the other player can call those points out and peg them as their own.

Since points are pegged on the board as soon as they are earned, games can often come down to the wire. Whoever gets to show their hand first or takes the point for the go in the final stretches of the game ending up the winner. This, along with the combination of luck and strategy is what makes cribbage such a fun game to play.

III Strategy

The particular strategy a player might employ is carried out by two means:

1. Which cards the player decides to discard

2. In what order they lay their cards down during the pegging stage

These are the two decisions that a player can make during the game and they act in a pivotal role in who wins any particular game and who wins the majority of a run of 1000 games. Each of the three different strategies for each of the two decisions employed by each player are coded for and have the option to use by simply commenting or uncommenting the relevant strategy specification lines.

Strategies for discarding:

1. Random: Randomly discard two cards

2. Smart: Discard the cards such that you keep the max number of points in your hand

3. Total: Disard the cards such that you maximize the quantity

   $$\text{(Points you are keeping)} - \text{(Points you are giving to the other player)}$$

   This factors into the decision of discard whether or not you will get to claim potential points that you are discarding to the crib.

Strategies for pegging/playing:

1. Random: Randomly laying down cards that keep the count at or below 31
2. Smart: At any particular time, play a card that gives you the maximum number of additional points. For example if you are P1 with a hand of 4,4,7,Queen and the sequence P1:4, P2:4 has been played, you should play the other 4 next because the 4 will get you 6 additional points, whereas the 7 will give you 2 points and the Queen will give you 0 points.
3. Total: Same as the smart strategy, except make plays that will likely either give you points or deny easy points to the other player. This includes leading with a pair card so you could potentially get a three of a kind, not leading with a 10 or 5 as the other player are likely to also have 5s and 10s so they could get 2 points for 15, not making the count 21 as the other player likely has a 10 so they could get 2 points for 31.

IV Simulation

To demonstrate how our runs of simulation will work, we will first run simulations of 5000 games having player 1 use the random discard and smart pegging strategies with player 2 using both the random discard and random pegging strategies. Taking the number of times player 1 wins and dividing it by 5000 will give the approximate probability that a player can win with this combination of strategies against a player using the random strategies. Figure 2 shows a graph of 10 rounds of 5000 games and how their incremental proportion of wins converge to (roughly) the same number. Note that the proportions don't change much after about 1500 games.

The Central Limit Theorem, stated in a simplified form, that if we sample many values from an unknown distribution and average them, and do this multiple times, the averages will be approximately normally distributed around the mean of the distribution. Figure 3 shows a histogram of the simulated proportions of 100 runs of 100 games, demonstrating an approximate normal shape of the simulated proportions. Note that due to the relatively long computation time, not as many estimates were obtained to fill out the histogram. Computing the mean and standard deviations of the data from the histogram we get

$$\mu = 0.666$$

$$\sigma = 0.0140$$

This means that we can be 95% confident that the true proportion of winning from this strategy is within the interval (0.638,0.694).

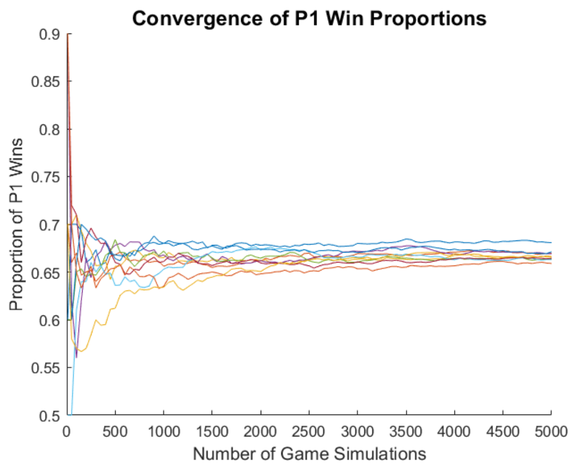

Figure 2: Graphs of the progression of the proportion of P1 wins using the random discard and smart pegging strategies against an opponent using the random discard and random pegging strategies.

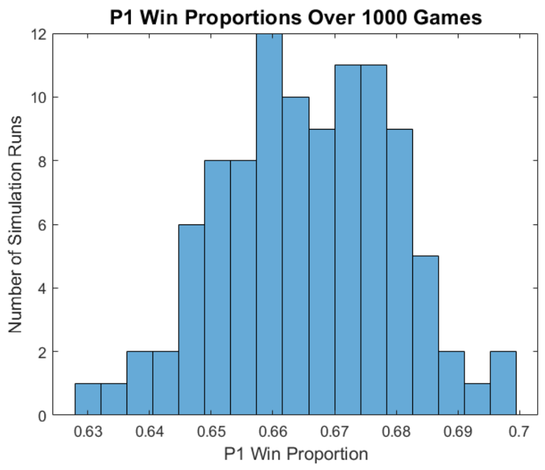

Figure 3: Histogram of the P1 win proportions using the random discard and smart pegging strategies against an opponent using the random discard and random pegging strategies.

V Strategy Combinations

As noted in the introduction, the previous version of this paper failed to include—due to coding and computation time—analysis of how the different discarding and pegging strategies affected the win rates of a player. The following two tables provide this information. The first table shows the average win proportions of player 1 when he and player 2 use the listed pair of strategies. The averages are taken over 5 runs of the win proportions of 300 games, where the starting dealer for each game is chosen at random. The second table is of the same format except it shows the average win proportions of the player that starts each the game as dealer. This is interesting to examine, as it shows how having the first crib of a game affects a player's ability to win any particular game.

Average Player 1 Win Proportion					
Discard Strategy \ Pegging Strategy	**P1 Random P2 Random**	**P1 Smart P2 Random**	**P1 Smart P2 Smart**	**P1 Total P2 Random**	**P1 Total P2 Total**
P1 Random P2 Random	0.51	0.67	0.51	0.69	0.53
P1 Smart P2 Random	0.95	0.98	0.94	0.97	0.94
P1 Smart P2 Smart	0.51	0.60	0.50	0.58	0.52
P1 Total P2 Random	0.96	0.98	0.93	0.98	0.95
P1 Total P2 Total	0.49	0.61	0.50	0.63	0.54

Average Starting Dealer Win Proportion					
Discard Strategy \ Pegging Strategy	**P1 Random P2 Random**	**P1 Smart P2 Random**	**P1 Smart P2 Smart**	**P1 Total P2 Random**	**P1 Total P2 Total**
P1 Random P2 Random	0.55	0.54	0.57	0.57	0.53
P1 Smart P2 Random	0.52	0.49	0.52	0.50	0.53
P1 Smart P2 Smart	0.55	0.55	0.56	0.54	0.54
P1 Total P2 Random	0.50	0.50	0.51	0.51	0.51
P1 Total P2 Total	0.57	0.54	0.54	0.55	0.55

It seems that these two tables confirm some pretty common sense intuitions one might have about the game. Most fundamentally, when player 1 has a more advanced discard or pegging strategy than player 2, they will win a majority of the games. This indicates that the strategies described are indeed beneficial. Secondly, it seems that The discarding strategy has much more of an effect on the outcome of the game than the pegging strategy, seen from the rows of .9 ranged proportions on the first table. This makes sense, as the potential for points is much greater in the showing stage than the pegging stage. Additionally, whenever the two players are using identical strategies for discarding and for pegging, each player wins approximately 50% of the games. This too intuitively makes sense. Finally, note that when player 1 has a better discard strategy, the starting dealer advantage is essentially nullified—something that is not necessarily true for a better pegging strategy.

VI Code

The following is the MATLAB code used for the simulations and plotting. It is a whopping 754 lines!

```
P1AvgWinStratMat = zeros(5)
DealerAvgWinStratMat = zeros(5)
for stratDis = 1:5
for stratPeg = 1:5
numRuns = 5;
P1WinVec = zeros(1,numRuns);
DealerWinVec = zeros(1,numRuns);
P1WinMat = zeros(numRuns,100);
P1Win10s = zeros(numRuns,1);
for run=1:numRuns
P1Wins = 0;
DealerWins = 0;
n=300;
for game = 1:n
   dealer = randi(2);
   firstDealer = 3-dealer;
   Scores = zeros(1,2);
   for keepPlaying = 1:100
      dealornodeal = [3-dealer,dealer]; % Flips the dealer from the last round
      dealer = 3-dealer;                % Updates who the dealer is
      deck(1,:) = [repmat(1,1,13),repmat(2,1,13),repmat(3,1,13),repmat(4,1,13)];
      deck(2,:) = repmat(1:13,1,4);
      deck(3,:) = randperm(52);

      if dealer==2
         P1dealt = deck(1:2,ismember(deck(3,:),[1 3 5 7 9 11]));
         P2dealt = deck(1:2,ismember(deck(3,:),[2 4 6 8 10 12]));
      else
         P2dealt = deck(1:2,ismember(deck(3,:),[1 3 5 7 9 11]));
         P1dealt = deck(1:2,ismember(deck(3,:),[2 4 6 8 10 12]));
      end
      Flipcard = deck(1:2,ismember(deck(3,:),randi([13 52])));
      if Flipcard(2)==11 % For his heels points
         Scores(dealer) = Scores(dealer) + 2;
         if sum(Scores>=121)==1
            break
         end
      end

      if stratDis == 1
        [P1hand, P2hand, crib] = bothRandDiscard(P1dealt,P2dealt);
      elseif stratDis == 2
        [P1hand, P2hand, crib] = P1SmartDiscard(P1dealt,P2dealt);
      elseif stratDis == 3
        [P1hand, P2hand, crib] = bothSmartDiscard(P1dealt,P2dealt);
      elseif stratDis == 4
        [P1hand, P2hand, crib] = P1TotalDiscard(P1dealt,P2dealt,dealer);
      else
        [P1hand, P2hand, crib] = bothTotalDiscard(P1dealt,P2dealt,dealer);
      end

      if stratPeg == 1
        pegPoints = bothRandPegging(P1hand,P2hand,dealer,Scores);
      elseif stratPeg == 2
        pegPoints = P1SmartPegging(P1hand,P2hand,dealer,Scores);
      elseif stratPeg == 3
        pegPoints = bothSmartPegging(P1hand,P2hand,dealer,Scores);
      elseif stratPeg == 4
        pegPoints = P1TotalPegging(P1hand,P2hand,dealer,Scores);
      else
        pegPoints = bothTotalPegging(P1hand,P2hand,dealer,Scores);
      end

      Scores = pegPoints;
      if sum(Scores>=121)==1
         break
      end
      showPoints = showing(P1hand,P2hand,crib,Flipcard,dealer,Scores);
      Scores = Scores + showPoints;
      if sum(Scores>=121)==1
         break
      end
   end
```

```
   Scores = min(121,Scores);
   if sum((Scores==121).*(1:2)) == firstDealer
       DealerWins = DealerWins+1;
   end
   if sum((Scores==121).*(1:2))==1
       P1Wins = P1Wins+1;
   end
   %if mod(game,n/100)==0
   %   P1WinMat(run,game/(n/100)) = P1Wins/game;
   %end
   %if game==10
   %    P1Win10s(run) = P1Wins/10;
   %end
end
P1WinVec(run) = P1Wins/n;
DealerWinVec(run) = DealerWins/n;
end
P1AvgWinStratMat(stratDis,stratPeg) = mean(P1WinVec);
DealerAvgWinStratMat(stratDis,stratPeg) = mean(DealerWinVec);
end
end
P1AvgWinStratMat
DealerAvgWinStratMat
%%
histogram(P1WinVec,20)
%%
hold on
for j=1:numRuns
   plot([10,(n/100):(n/100):n],[P1Win10s(j),P1WinMat(j,:)])
end
xlabel('Number of Game Simulations','Fontsize',12)
ylabel('Proportion of P1 Wins','Fontsize',12)
title('Convergence of P1 Win Proportions','Fontsize',14,'FontWeight','bold')

function points = countFlush(hand,Flipcard,isCrib)
   points = 0;
   if (~isCrib) && sum(groupcounts(hand(1,:)')==4) == 1
      points = 4;
   end
   if sum(groupcounts([hand(1,:)';Flipcard(1)])==5) == 1
      points = 5;
   end
end
function points = count15s(hand,Flipcard)
   sums = [];
   values = min(10,[hand(2,:),Flipcard(2)]);
   for i=2:5
      sums = [sums;sum(nchoosek(values,i),2)];
   end
   points = 2*sum(sums == 15);
end
function points = countPairs(hand,Flipcard)
   values = [hand(2,:),Flipcard(2)];
   combos = nchoosek(values,2);
   points = 2*sum(combos(:,1)==combos(:,2));
end
function points = countRuns(hand,Flipcard)
   points = 0;
   values = [hand(2,:),Flipcard(2)];
   for i=3:5
      combos = nchoosek(values,i);
      sortCombos = sort(combos,2);
      shiftMat = sortCombos-repmat(sortCombos(:,1),1,i);
      compMat = shiftMat==repmat(0:(i-1),nchoosek(5,i),1);
      points = points + i*sum(sum(compMat,2)==i);
   end
end
function points = countNobs(hand,Flipcard)
   points = 0;
   if ismember(11,hand(2,:)) && ismember(Flipcard(1),hand(1,hand(2,:)==11))
      points = 1;
   end
end
function points = countHand(hand,Flipcard,isCrib)
   points = 0;
   points = points + countFlush(hand,Flipcard,isCrib);
   points = points + count15s(hand,Flipcard);
   points = points + countPairs(hand,Flipcard);
   points = points + countRuns(hand,Flipcard);
   points = points + countNobs(hand,Flipcard);
end
function points = showing(P1hand,P2hand,crib,Flipcard,dealer,Scores)
   points = [0 0];
   if dealer==1
      points(2) = points(2) + countHand(P2hand,Flipcard,0);
      if sum(Scores+points>=121)==1
         return
      end
      points(1) = points(1) + countHand(P1hand,Flipcard,0);
      points(1) = points(1) + countHand(crib,Flipcard,1);
```

```
        if sum(Scores+points>=121)==1
            return
        end
    else
        points(1) = points(1) + countHand(P1hand,Flipcard,0);
        if sum(Scores+points>=121)==1
            return
        end
        points(2) = points(2) + countHand(P2hand,Flipcard,0);
        points(2) = points(2) + countHand(crib,Flipcard,1);
        if sum(Scores+points>=121)==1
            return
        end
    end
end

function check = check15(count)
    check = 0;
    if count == 15
        check = 1;
    end
end
function check = checkPair(playStack)
    check = 0;
    if size(playStack,2) >= 2
        if playStack(2,end) == playStack(2,end-1)
            check = 1;
        end
    end
end
function check = checkTrio(playStack)
    check = 0;
    if size(playStack,2) >= 3
        if (playStack(2,end) == playStack(2,end-1)) && (playStack(2,end) == playStack(2,en
            check = 1;
        end
    end
end
function check = checkQuad(playStack)
    check = 0;
    if size(playStack,2) >= 4
        if (playStack(2,end) == playStack(2,end-1)) && (playStack(2,end) == playStack(2,en
            check = 1;
        end
    end
```

```
end
function check = checkRun(playStack,runLength)
    check = 0;
    stLength = size(playStack,2);
    if stLength >= runLength
        potRun = sort(playStack(2,stLength-(0:runLength-1)));
        if (~isempty(potRun)) && isequal(potRun-potRun(1),0:runLength-1)
            check = 1;
        end
    end
end
function points = checklist(playStack,count)
    points = 0;
    if check15(count)
       points = points+2;
    end
    checkQuads = checkQuad(playStack);
    checkTrios = checkTrio(playStack);
    checkPairs = checkPair(playStack);
    if checkQuads
        points = points+12;
    elseif checkTrios
        points = points+6;
    elseif checkPairs
        points = points+2;
    end
    checkRuns = zeros(1,5);
    for i=3:7
        checkRuns(i-2) = checkRun(playStack,i);
    end
    points = points+max(checkRuns.*(3:7));
end
function Scores = bothRandPegging(P1hand,P2hand,dealer,Scores)
   curCount = 0;
   curPlayer = dealer; % curPlayer indicates who is laying down the next card on the pil
   stack = [];
   curCard = [0;0];
   log = zeros(5,8);
   cardsPlayed = 0;
   while size(P1hand,2)+size(P2hand,2) > 0
      P1hand = P1hand(:,(ismember(P1hand(1,:),curCard(1))+ismember(P1hand(2,:),curCard(2
      P2hand = P2hand(:,(ismember(P2hand(1,:),curCard(1))+ismember(P2hand(2,:),curCard(2
      P1cardOptions = P1hand(:,min(10,P1hand(2,:))<=31-curCount);
      P2cardOptions = P2hand(:,min(10,P2hand(2,:))<=31-curCount);
      P1numOptions = size(P1cardOptions,2);
      P2numOptions = size(P2cardOptions,2);
      curPlayer = 3-curPlayer;
      cardsPlayed = cardsPlayed + 1;
      if (P1numOptions == 0) && (P2numOptions == 0) % 1pt for go or last card
         curPlayer = 3-curPlayer;
         Scores(curPlayer) = Scores(curPlayer)+1;
         if sum(Scores>=121)==1
            break
         end
         cardsPlayed = cardsPlayed-1;
         curCount = 0;
         stack = [];
         log(4:5,cardsPlayed) = Scores;
      else % If someone can play
         if (curPlayer == 1) && (P1numOptions == 0) % If player 1 is curPlayer and can't
            curPlayer = 2;
            curCard = P2cardOptions(:,randi(P2numOptions));
         elseif (curPlayer == 2) && (P2numOptions == 0) % If player 2 is curPlayer and c
            curPlayer = 1;
            curCard = P1cardOptions(:,randi(P1numOptions));
         elseif curPlayer == 1
            curCard = P1cardOptions(:,randi(P1numOptions));
         else
            curCard = P2cardOptions(:,randi(P2numOptions));
         end
         curCount = curCount + min(10,curCard(2));
         stack = [stack,curCard];

         Scores(curPlayer) = Scores(curPlayer)+checklist(stack,curCount);
         log(3,cardsPlayed) = curCount;
         if curCount == 31
            Scores(curPlayer) = Scores(curPlayer)+2;
            curCount = 0;
            stack = [];
         end

         log(curPlayer,cardsPlayed) = curCard(2);
         log(4:5,cardsPlayed) = Scores;

         if sum(Scores>=121)==1
            break
         end
      end
   end
end
```

```
function Scores = P1SmartPegging(P1hand,P2hand,dealer,Scores)
   curCount = 0;
   curPlayer = dealer; % curPlayer indicates who is laying down the next card on the pil
   stack = [];
   curCard = [0;0];
   log = zeros(5,8);
   cardsPlayed = 0;
   while size(P1hand,2)+size(P2hand,2) > 0
      P1hand = P1hand(:,(ismember(P1hand(1,:),curCard(1))+ismember(P1hand(2,:),curCard(2
      P2hand = P2hand(:,(ismember(P2hand(1,:),curCard(1))+ismember(P2hand(2,:),curCard(2
      P1cardOptions = P1hand(:,min(10,P1hand(2,:))<=31-curCount);
      P2cardOptions = P2hand(:,min(10,P2hand(2,:))<=31-curCount);
      P1numOptions = size(P1cardOptions,2);
      P2numOptions = size(P2cardOptions,2);
      curPlayer = 3-curPlayer;
      cardsPlayed = cardsPlayed + 1;

      if (P1numOptions == 0) && (P2numOptions == 0) % 1pt for go or last card
         curPlayer = 3-curPlayer;
         Scores(curPlayer) = Scores(curPlayer)+1;
         if sum(Scores>=121)==1
            break
         end
         cardsPlayed = cardsPlayed-1;
         curCount = 0;
         stack = [];
         log(4:5,cardsPlayed) = Scores;
      else % If someone can play
         if (curPlayer == 1) && (P1numOptions == 0) % If player 1 is curPlayer and can't
            curPlayer = 2;
            curCard = P2cardOptions(:,randi(P2numOptions));
         elseif (curPlayer == 2) && (P2numOptions == 0) % If player 2 is curPlayer and c
            curPlayer = 1;
            potPoints = zeros(1,P1numOptions);
            for i = 1:P1numOptions
               potCard = P1cardOptions(:,i);
               potPoints(i) = checklist([stack,potCard],curCount+min(10,potCard(2)));
               if (curCount+min(10,potCard(2))) == 31
                  potPoints(i) = potPoints(i)+2;
               end
            end
            maxPotCards = P1cardOptions(:,max(potPoints)==potPoints);
            numMaxPotCards = size(maxPotCards,2);
            curCard = maxPotCards(:,randi(numMaxPotCards));
         elseif curPlayer == 1
```

```
            potPoints = zeros(1,P1numOptions);
            for i = 1:P1numOptions
               potCard = P1cardOptions(:,i);
               potPoints(i) = checklist([stack,potCard],curCount+min(10,potCard(2)));
               if (curCount+min(10,potCard(2))) == 31
                  potPoints(i) = potPoints(i)+2;
               end
            end
            maxPotCards = P1cardOptions(:,max(potPoints)==potPoints);
            numMaxPotCards = size(maxPotCards,2);
            curCard = maxPotCards(:,randi(numMaxPotCards));
         else
            curCard = P2cardOptions(:,randi(P2numOptions));
         end
         curCount = curCount + min(10,curCard(2));
         stack = [stack,curCard];

         Scores(curPlayer) = Scores(curPlayer)+checklist(stack,curCount);
         log(3,cardsPlayed) = curCount;
         if curCount == 31
            Scores(curPlayer) = Scores(curPlayer)+2;
            curCount = 0;
            stack = [];
         end

         log(curPlayer,cardsPlayed) = curCard(2);
         log(4:5,cardsPlayed) = Scores;

         if sum(Scores>=121)==1
            break
         end
      end
   end
end
function Scores = bothSmartPegging(P1hand,P2hand,dealer,Scores)
   curCount = 0;
   curPlayer = dealer; % curPlayer indicates who is laying down the next card on the pil
   stack = [];
   curCard = [0;0];
   log = zeros(5,8);
   cardsPlayed = 0;
   while size(P1hand,2)+size(P2hand,2) > 0
      P1hand = P1hand(:,(ismember(P1hand(1,:),curCard(1))+ismember(P1hand(2,:),curCard(2
      P2hand = P2hand(:,(ismember(P2hand(1,:),curCard(1))+ismember(P2hand(2,:),curCard(2
      P1cardOptions = P1hand(:,min(10,P1hand(2,:))<=31-curCount);
```

```
P2cardOptions = P2hand(:,min(10,P2hand(2,:))<=31-curCount);
P1numOptions = size(P1cardOptions,2);
P2numOptions = size(P2cardOptions,2);
curPlayer = 3-curPlayer;
cardsPlayed = cardsPlayed + 1;

if (P1numOptions == 0) && (P2numOptions == 0) % 1pt for go or last card
   curPlayer = 3-curPlayer;
   Scores(curPlayer) = Scores(curPlayer)+1;
   if sum(Scores>=121)==1
      break
   end
   cardsPlayed = cardsPlayed-1;
   curCount = 0;
   stack = [];
   log(4:5,cardsPlayed) = Scores;
else % If someone can play
   if (curPlayer == 1) && (P1numOptions == 0) % If player 1 is curPlayer and can't
      curPlayer = 2;
      potPoints = zeros(1,P2numOptions);
      for i = 1:P2numOptions
         potCard = P2cardOptions(:,i);
         potPoints(i) = checklist([stack,potCard],curCount+min(10,potCard(2)));
         if (curCount+min(10,potCard(2))) == 31
            potPoints(i) = potPoints(i)+2;
         end
      end
      maxPotCards = P2cardOptions(:,max(potPoints)==potPoints);
      numMaxPotCards = size(maxPotCards,2);
      curCard = maxPotCards(:,randi(numMaxPotCards));
   elseif (curPlayer == 2) && (P2numOptions == 0) % If player 2 is curPlayer and c
      curPlayer = 1;
      potPoints = zeros(1,P1numOptions);
      for i = 1:P1numOptions
         potCard = P1cardOptions(:,i);
         potPoints(i) = checklist([stack,potCard],curCount+min(10,potCard(2)));
         if (curCount+min(10,potCard(2))) == 31
            potPoints(i) = potPoints(i)+2;
         end
      end
      maxPotCards = P1cardOptions(:,max(potPoints)==potPoints);
      numMaxPotCards = size(maxPotCards,2);
      curCard = maxPotCards(:,randi(numMaxPotCards));
   elseif curPlayer == 1
      potPoints = zeros(1,P1numOptions);
```

```
            for i = 1:P1numOptions
               potCard = P1cardOptions(:,i);
               potPoints(i) = checklist([stack,potCard],curCount+min(10,potCard(2)));
               if (curCount+min(10,potCard(2))) == 31
                  potPoints(i) = potPoints(i)+2;
               end
            end
            maxPotCards = P1cardOptions(:,max(potPoints)==potPoints);
            numMaxPotCards = size(maxPotCards,2);
            curCard = maxPotCards(:,randi(numMaxPotCards));
         else
            potPoints = zeros(1,P2numOptions);
            for i = 1:P2numOptions
               potCard = P2cardOptions(:,i);
               potPoints(i) = checklist([stack,potCard],curCount+min(10,potCard(2)));
               if (curCount+min(10,potCard(2))) == 31
                  potPoints(i) = potPoints(i)+2;
               end
            end
            maxPotCards = P2cardOptions(:,max(potPoints)==potPoints);
            numMaxPotCards = size(maxPotCards,2);
            curCard = maxPotCards(:,randi(numMaxPotCards));
         end
         curCount = curCount + min(10,curCard(2));
         stack = [stack,curCard];

         Scores(curPlayer) = Scores(curPlayer)+checklist(stack,curCount);
         log(3,cardsPlayed) = curCount;
         if curCount == 31
            Scores(curPlayer) = Scores(curPlayer)+2;
            curCount = 0;
            stack = [];
         end

         log(curPlayer,cardsPlayed) = curCard(2);
         log(4:5,cardsPlayed) = Scores;

         if sum(Scores>=121)==1
            break
         end
      end
   end
end
function Scores = P1TotalPegging(P1hand,P2hand,dealer,Scores)
   curCount = 0;
```

```
curPlayer = dealer; % curPlayer indicates who is laying down the next card on the pil
stack = [];
curCard = [0;0];
log = zeros(5,8);
cardsPlayed = 0;
while size(P1hand,2)+size(P2hand,2) > 0
   P1hand = P1hand(:,(ismember(P1hand(1,:),curCard(1))+ismember(P1hand(2,:),curCard(2
   P2hand = P2hand(:,(ismember(P2hand(1,:),curCard(1))+ismember(P2hand(2,:),curCard(2
   P1cardOptions = P1hand(:,min(10,P1hand(2,:))<=31-curCount);
   P2cardOptions = P2hand(:,min(10,P2hand(2,:))<=31-curCount);
   P1numOptions = size(P1cardOptions,2);
   P2numOptions = size(P2cardOptions,2);
   curPlayer = 3-curPlayer;
   cardsPlayed = cardsPlayed + 1;

   fives = P1cardOptions(2,:)==5;
   tens = P1cardOptions(2,:)>=10;
   numPot5s10s = sum(fives+tens);
   if (curPlayer==1) && (curCount==0) && (numPot5s10s<P1numOptions)
       P1cardOptions = P1cardOptions(:,~(fives+tens));
       P1numOptions = P1numOptions-numPot5s10s;
   end
   pairVec = sum(P1cardOptions(2,:)==P1cardOptions(2,:)')-1;
   numPairs = max(pairVec);
   if (curPlayer==1) && (curCount==0) && (numPairs>0)
      P1cardOptions = P1cardOptions(:,pairVec>0);
      P1numOptions = numPairs;
   end

   if (P1numOptions == 0) && (P2numOptions == 0) % 1pt for go or last card
      curPlayer = 3-curPlayer;
      Scores(curPlayer) = Scores(curPlayer)+1;
      if sum(Scores>=121)==1
         break
      end
      cardsPlayed = cardsPlayed-1;
      curCount = 0;
      stack = [];
      log(4:5,cardsPlayed) = Scores;
   else % If someone can play
      if (curPlayer == 1) && (P1numOptions == 0) % If player 1 is curPlayer and can't
         curPlayer = 2;
         curCard = P2cardOptions(:,randi(P2numOptions));
      elseif (curPlayer == 2) && (P2numOptions == 0) % If player 2 is curPlayer and c
         curPlayer = 1;
```

```
            potPoints = zeros(1,P1numOptions);
            for i = 1:P1numOptions
               potCard = P1cardOptions(:,i);
               potPoints(i) = checklist([stack,potCard],curCount+min(10,potCard(2)));
               if (curCount+min(10,potCard(2))) == 31
                  potPoints(i) = potPoints(i)+2;
               end
            end
            maxPotCards = P1cardOptions(:,max(potPoints)==potPoints);
            numMaxPotCards = size(maxPotCards,2);
            curCard = maxPotCards(:,randi(numMaxPotCards));
         elseif curPlayer == 1
            potPoints = zeros(1,P1numOptions);
            for i = 1:P1numOptions
               potCard = P1cardOptions(:,i);
               potPoints(i) = checklist([stack,potCard],curCount+min(10,potCard(2)));
               if (curCount+min(10,potCard(2))) == 31
                  potPoints(i) = potPoints(i)+2;
               end
            end
            maxPotCards = P1cardOptions(:,max(potPoints)==potPoints);
            numMaxPotCards = size(maxPotCards,2);
            curCard = maxPotCards(:,randi(numMaxPotCards));
         else
            curCard = P2cardOptions(:,randi(P2numOptions));
         end
         curCount = curCount + min(10,curCard(2));
         stack = [stack,curCard];

         Scores(curPlayer) = Scores(curPlayer)+checklist(stack,curCount);
         log(3,cardsPlayed) = curCount;
         if curCount == 31
            Scores(curPlayer) = Scores(curPlayer)+2;
            curCount = 0;
            stack = [];
         end

         log(curPlayer,cardsPlayed) = curCard(2);
         log(4:5,cardsPlayed) = Scores;

         if sum(Scores>=121)==1
            break
         end
      end
   end
```

```
end
function Scores = bothTotalPegging(P1hand,P2hand,dealer,Scores)
   curCount = 0;
   curPlayer = dealer; % curPlayer indicates who is laying down the next card on the pil
   stack = [];
   curCard = [0;0];
   log = zeros(5,8);
   cardsPlayed = 0;
   while size(P1hand,2)+size(P2hand,2) > 0
      P1hand = P1hand(:,(ismember(P1hand(1,:),curCard(1))+ismember(P1hand(2,:),curCard(2
      P2hand = P2hand(:,(ismember(P2hand(1,:),curCard(1))+ismember(P2hand(2,:),curCard(2
      P1cardOptions = P1hand(:,min(10,P1hand(2,:))<=31-curCount);
      P2cardOptions = P2hand(:,min(10,P2hand(2,:))<=31-curCount);
      P1numOptions = size(P1cardOptions,2);
      P2numOptions = size(P2cardOptions,2);
      curPlayer = 3-curPlayer;
      cardsPlayed = cardsPlayed + 1;

      fives = P1cardOptions(2,:)==5;
      tens = P1cardOptions(2,:)>=10;
      numPot5s10s = sum(fives+tens);
      if (curPlayer==1) && (curCount==0) && (numPot5s10s<P1numOptions)
          P1cardOptions = P1cardOptions(:,~(fives+tens));
          P1numOptions = P1numOptions-numPot5s10s;
      end
      pairVec = sum(P1cardOptions(2,:)==P1cardOptions(2,:)')-1;
      numPairs = max(pairVec);
      if (curPlayer==1) && (curCount==0) && (numPairs>0)
         P1cardOptions = P1cardOptions(:,pairVec>0);
         P1numOptions = numPairs;
      end

      if (P1numOptions == 0) && (P2numOptions == 0) % 1pt for go or last card
         curPlayer = 3-curPlayer;
         Scores(curPlayer) = Scores(curPlayer)+1;
         if sum(Scores>=121)==1
            break
         end
         cardsPlayed = cardsPlayed-1;
         curCount = 0;
         stack = [];
         log(4:5,cardsPlayed) = Scores;
      else % If someone can play
         if (curPlayer == 1) && (P1numOptions == 0) % If player 1 is curPlayer and can't
            curPlayer = 2;
      potPoints = zeros(1,P2numOptions);
      for i = 1:P2numOptions
         potCard = P2cardOptions(:,i);
         potPoints(i) = checklist([stack,potCard],curCount+min(10,potCard(2)));
         if (curCount+min(10,potCard(2))) == 31
            potPoints(i) = potPoints(i)+2;
         end
      end
      maxPotCards = P2cardOptions(:,max(potPoints)==potPoints);
      numMaxPotCards = size(maxPotCards,2);
      curCard = maxPotCards(:,randi(numMaxPotCards));
   elseif (curPlayer == 2) && (P2numOptions == 0) % If player 2 is curPlayer and c
      curPlayer = 1;
      potPoints = zeros(1,P1numOptions);
      for i = 1:P1numOptions
         potCard = P1cardOptions(:,i);
         potPoints(i) = checklist([stack,potCard],curCount+min(10,potCard(2)));
         if (curCount+min(10,potCard(2))) == 31
            potPoints(i) = potPoints(i)+2;
         end
      end
      maxPotCards = P1cardOptions(:,max(potPoints)==potPoints);
      numMaxPotCards = size(maxPotCards,2);
      curCard = maxPotCards(:,randi(numMaxPotCards));
   elseif curPlayer == 1
      potPoints = zeros(1,P1numOptions);
      for i = 1:P1numOptions
         potCard = P1cardOptions(:,i);
         potPoints(i) = checklist([stack,potCard],curCount+min(10,potCard(2)));
         if (curCount+min(10,potCard(2))) == 31
            potPoints(i) = potPoints(i)+2;
         end
      end
      maxPotCards = P1cardOptions(:,max(potPoints)==potPoints);
      numMaxPotCards = size(maxPotCards,2);
      curCard = maxPotCards(:,randi(numMaxPotCards));
   else
      potPoints = zeros(1,P2numOptions);
      for i = 1:P2numOptions
         potCard = P2cardOptions(:,i);
         potPoints(i) = checklist([stack,potCard],curCount+min(10,potCard(2)));
         if (curCount+min(10,potCard(2))) == 31
            potPoints(i) = potPoints(i)+2;
         end
      end
```

```
            maxPotCards = P2cardOptions(:,max(potPoints)==potPoints);
            numMaxPotCards = size(maxPotCards,2);
            curCard = maxPotCards(:,randi(numMaxPotCards));
         end
         curCount = curCount + min(10,curCard(2));
         stack = [stack,curCard];

         Scores(curPlayer) = Scores(curPlayer)+checklist(stack,curCount);
         log(3,cardsPlayed) = curCount;
         if curCount == 31
            Scores(curPlayer) = Scores(curPlayer)+2;
            curCount = 0;
            stack = [];
         end

         log(curPlayer,cardsPlayed) = curCard(2);
         log(4:5,cardsPlayed) = Scores;

         if sum(Scores>=121)==1
            break
         end
      end
   end
end

function [hand1, hand2, crib] = bothRandDiscard(dealt1,dealt2)
   discardOrd1 = randperm(6);
   discard1 = discardOrd1(1:2);
   discardOrd2 = randperm(6);
   discard2 = discardOrd2(1:2);

   crib = [dealt1(:,discard1), dealt2(:,discard2)];
   hand1 = dealt1(:,~ismember(1:6,discard1));
   hand2 = dealt2(:,~ismember(1:6,discard2));
end
function [hand1, hand2, crib] = P1SmartDiscard(dealt1,dealt2)
   handIndOptions = nchoosek(1:6,4);
   pointOptions = zeros(1,nchoosek(6,4));
   for i = 1:nchoosek(6,4)
      handInds = handIndOptions(i,:);
      potHand = dealt1(:,handInds);
      pointOptions(i) = countHand(potHand,[-50;-50],0);
   end
   [~,maxPointInd] = max(pointOptions);
   maxHandInds = handIndOptions(maxPointInd,:);
```

```
    hand1 = dealt1(:,maxHandInds);

    discardOrd2 = randperm(6);
    discard2 = discardOrd2(1:2);
    hand2 = dealt2(:,~ismember(1:6,discard2));

    crib = [dealt1(:,~ismember(1:6,maxHandInds)), dealt2(:,discard2)];
end
function [hand1, hand2, crib] = bothSmartDiscard(dealt1,dealt2)
    handIndOptions = nchoosek(1:6,4);
    pointOptions = zeros(2,nchoosek(6,4));
    for i = 1:nchoosek(6,4)
        handInds = handIndOptions(i,:);
        potHand1 = dealt1(:,handInds);
        pointOptions(1,i) = countHand(potHand1,[-50;-50],0);
        potHand2 = dealt2(:,handInds);
        pointOptions(2,i) = countHand(potHand2,[-50;-50],0);
    end
    [~,maxPointInd1] = max(pointOptions(1,:));
    maxHandInds1 = handIndOptions(maxPointInd1,:);
    hand1 = dealt1(:,maxHandInds1);
    [~,maxPointInd2] = max(pointOptions(2,:));
    maxHandInds2 = handIndOptions(maxPointInd2,:);
    hand2 = dealt2(:,maxHandInds2);

    crib = [dealt1(:,~ismember(1:6,maxHandInds1)), dealt2(:,~ismember(1:6,maxHandInds2))]
end
function [hand1, hand2, crib] = P1TotalDiscard(dealt1,dealt2,dealer)
    handIndOptions = nchoosek(1:6,4);
    pointOptions = zeros(1,nchoosek(6,4));
    placeCards = [-100 -150;-100 -150];
    for i = 1:nchoosek(6,4)
        handInds = handIndOptions(i,:);
        potHand = dealt1(:,handInds);
        potDiscard = dealt1(:,~ismember(1:6,handInds));
        is1dealer = 2*(dealer == 1)-1;
        pointOptions(i) = countHand(potHand,[-50;-50],0)+is1dealer*countHand([potDiscard,p
    end
    [~,maxPointInd] = max(pointOptions);
    maxHandInds = handIndOptions(maxPointInd,:);
    hand1 = dealt1(:,maxHandInds);

    discardOrd2 = randperm(6);
    discard2 = discardOrd2(1:2);
    hand2 = dealt2(:,~ismember(1:6,discard2));
```

```
   crib = [dealt1(:,~ismember(1:6,maxHandInds)), dealt2(:,discard2)];
end
function [hand1, hand2, crib] = bothTotalDiscard(dealt1,dealt2,dealer)
   handIndOptions = nchoosek(1:6,4);
   pointOptions = zeros(2,nchoosek(6,4));
   placeCards = [-100 -150;-100 -150];
   for i = 1:nchoosek(6,4)
      handInds = handIndOptions(i,:);
      potHand1 = dealt1(:,handInds);
      potDiscard1 = dealt1(:,~ismember(1:6,handInds));
      is1dealer = 2*(dealer == 1)-1;
      pointOptions(1,i) = countHand(potHand1,[-50;-50],0)+is1dealer*countHand([potDiscar
      potHand2 = dealt2(:,handInds);
      is2dealer = 2*(dealer == 2)-1;
      potDiscard2 = dealt2(:,~ismember(1:6,handInds));
      pointOptions(2,i) = countHand(potHand2,[-50;-50],0)+is2dealer*countHand([potDiscar
   end
   [~,maxPointInd1] = max(pointOptions(1,:));
   maxHandInds1 = handIndOptions(maxPointInd1,:);
   hand1 = dealt1(:,maxHandInds1);
   [~,maxPointInd2] = max(pointOptions(2,:));
   maxHandInds2 = handIndOptions(maxPointInd2,:);
   hand2 = dealt2(:,maxHandInds2);

   crib = [dealt1(:,~ismember(1:6,maxHandInds1)), dealt2(:,~ismember(1:6,maxHandInds2))]
end
```