Hearts Card Game: Given Each Player's Score, Compute Each Player's Probability of Winning

Barry Zeeberg
barryzbooks.com
barryz2013@gmail.com

Abstract

In a hearts card game, we know that the player with the lowest score qualitatively has the highest chance of ultimately winning. However, the quantitative numerical probability of winning depends on the relative values of each player's scores, and also how close those scores are to foretelling the end of the game.

Quantitative estimates of the probabilities can be computed as the result of repeated simulations applied to a model of the scoring process. This is inherently stochastic in nature, so that the greater the number of simulations, the greater the precision and accuracy of the quantitative probability estimates. The simulation studies can be customized for groups of players at specified levels of experience. The results can be tabulated for easy reference. They can be used by either a human player, or by an artificial intelligence application. In fact, some of the result files are integrated into my upcoming R repository (CRAN) package *ProbTab*.

For the expert player, it is important to know the quantitative values of the probabilities in order to make rational choices among several alternative lines of play. For the non-expert player, it is an important learning tool to help move the player along to a higher level.

An important illustrative example is provided, namely the decision of whether to allow an opponent to shoot the moon, or to take the QS to stop the moon.

Introduction

Figure 1. Book on Hearts and Books on Bridge

In contrast to the game of bridge, there are very few books on hearts. I have come across only 2 books that specifically provide instruction in hearts (other than just a beginner-level introduction or as a brief chapter in a general book about many card games). Needless to say, for bridge there are probably dozens of books for advanced to expert players. Just about all of the advanced tactical concepts of bridge are directly applicable to hearts, so to learn to play hearts one needs to first study the bridge books. A partial list of such tactics includes:

- Counting (as stressed by Kelsey for bridge)
- Lead Placement
- Avoidance

- Psychological/Deceptive
- Squeeze/Pseudo-squeeze
- Elimination
- Endplay
- Card Reading/Inference
- Hypothesis/Imagination
- Entry/Exit Management
- Protecting Exit Cards
- Shifting the Burden/Responsibility
- Nervous Habit, Rhythm, Body Language Recognition/Deception/ Avoidance/Management (however, unlike poker, purposeful deception by mannerism is not permissible in bridge or hearts). As an aside, even in an online non-expert game, a break in a player's pace or rhythm can indicate that the player has a difficult choice to consider, and that can provide an inference of the holding. But a player may not stall on purpose to mislead.

However, overshadowing all of these is understanding the probability of winning based upon the score at the end of the current hand. This probability should determine the idealized distribution of dumping points upon the 3 opponents.

To this end, I generate and provide tables of probabilities of winning based upon scores. In addition to the tables, I also provide freely downloadable files containing the program code, so that the user can compute versions of the tables that might be based upon different underlying parameters than those that I settled on. Finally, I provide a specific illustrative example of how to use the probability tables to chart out the optimal tactics at a critical juncture.

Methods

The Rules of Hearts

I follow a fairly standard set of rules

- 4 players
- 1 point for each heart
- 13 points for the QS
- 26 points for the other 3 players for shooting the moon (*i.e.,* all 13 hearts plus the QS)
- Game over when at least 1 player has >= 100 points

Mapping Pseudo-Random Numbers to Players (winner.awk)

Since the files containing the program code are available, and the programs are pretty straightforward, I am only going to discuss the most interesting and challenging section of the code, namely the use of a pseudo-random number to denote which player gets hearts or the QS.

The basic idea is to construct 4 contiguous intervals, *e.g.*, 0 to 1, 1 to 2, 2 to 3, and 3 to 4. A random number is drawn from a uniform distribution between (for the present example) 0 to 4. For instance, say that the random number happens to be 2.345678. Then player 3 is credited with taking in the corresponding number of points. For each hand, this exercise is repeated 13 times for each heart and once for the QS.

This approach ignores the possibility of correlations arising from a player taking multiple hearts on 1 trick, it uses the simplification of independent probabilities. Implementing the more complicated and more realistic approach would require making additional assumptions about the frequency with which 1, 2, 3, or 4 hearts are taken on a single trick.

The equal-length intervals case described above might apply for a game of novice players. More experienced players would target the other players who have relatively lower scores. In this case, players with lower scores would have a higher probability of taking a higher number of points on a trick. So the intervals would need to be modified such that interval for the players with low scores would provide a larger “target” for the random number. For instance, if player 2 had a score that was significantly lower than the other 3 players, the intervals we use might be modified to, *e.g.*, 0 to 1, 1 to 2.5, 2.5 to 3.5, and 3.5 to 4.5. In this particular example, player 2

would have a 33.33 % higher chance of getting points. I refer to this altered probability as a "bias."

Depending on the general level of experience of the 4 players, the degree of bias will vary. The 4 defined levels are given in Table 1.

Table 1. Skill Levels

Skill Level	Degree of Bias
Novice	0.00
Intermediate	0.25
Advanced	0.50
Expert	1.00

No bias versus "novice" Modes (winner.awk)

The initial version of my program did not include bias, that was added later as a separate block of code. So for novice players, the program can be run using the original code that does not include bias, or, equivalently, using the bias code, and setting the degree of bias to 0 by specifying "novice" level. Comparing the results of these 2 functionally equivalent modes can provide a partial check on the correctness of the bias code. In practical terms, the bias code is a lot slower, as substantially more computation is performed in the "inner loop."

Squash Function (winner.awk)

The implementation of bias is inherently a nonlinear process, since a score differential of 20 would motivate the players to target the low player just as much as a differential of 50. I was not able to find an analytic function (*e.g.*, trig, inverse trig, hyperbolic trig, rectangular hyperbola, etc.) that provided the desired mapping between score differential and bias, so I decided to tabulate a function ("Squash Function"). The values in the Squash Function are calibrated for an "expert" player. For the other levels, those values are scaled down by a degree of bias associated with each skill level, as mentioned above. Note that the degree of bias for "expert" is 1.00, so the

full value of the Squash Function is used. For “novice,” the degree of bias is 0.00, so the Squash Function in effect is ignored.

The parameters for the tabulation were developed by trial and error, but they can optionally be provided as a set of command line arguments for a user who wants to explore other scenarios. The default parameters that I recommend are based on a visual examination of the changes in selected probability scenarios between “novice” and “expert.”

I want to stress that, whereas the (unbiased) probabilities for “novice” quantitatively reflect what they are in physical reality, the biased probabilities for the other 3 levels are based on my manual selection of what I deem as an appropriate Squash Function. If sufficient data from real world games of various levels were collected, then the Squash Function could be set in a less arbitrary manner by using the empirical data.

Noise as a Function of Number of Simulations (computeNoise.awk)

The awk program computeNoise.awk was used to estimate the noise within the results of a winner.awk run. This program compares probabilities that arise from identical scores within a hand and should therefore theoretically be identical. The coefficient of variation is computed for each instance, and a summary statistic is computed as the mean of the coefficients of variation.

Encoding/Decoding the Names of Output Files to Retrieve the Command Line Parameters

The meta data are encoded in the names of output files by the awk program winner.awk. In addition to that encoding, winner.awk also generates a meta data file that contains a decoded version of much of the same encoded information. Here is the line of code in winner.awk that encodes:

```
name0=sprintf("%s_%d_%.4f_%d_%d_%s_%d_%d_%d_%.4f_%.4f_%.4f_%.4f",
systime_start_format,DEBUG,Pmoon,N,BIAS,level,initial,final,del,squash5,squash10,squash20,squash100)
```

The names of output files for subsequent processing (*i.e.*, computeNoise.awk) carry along the original identifying meta data.

Available Program and Result Files Archive

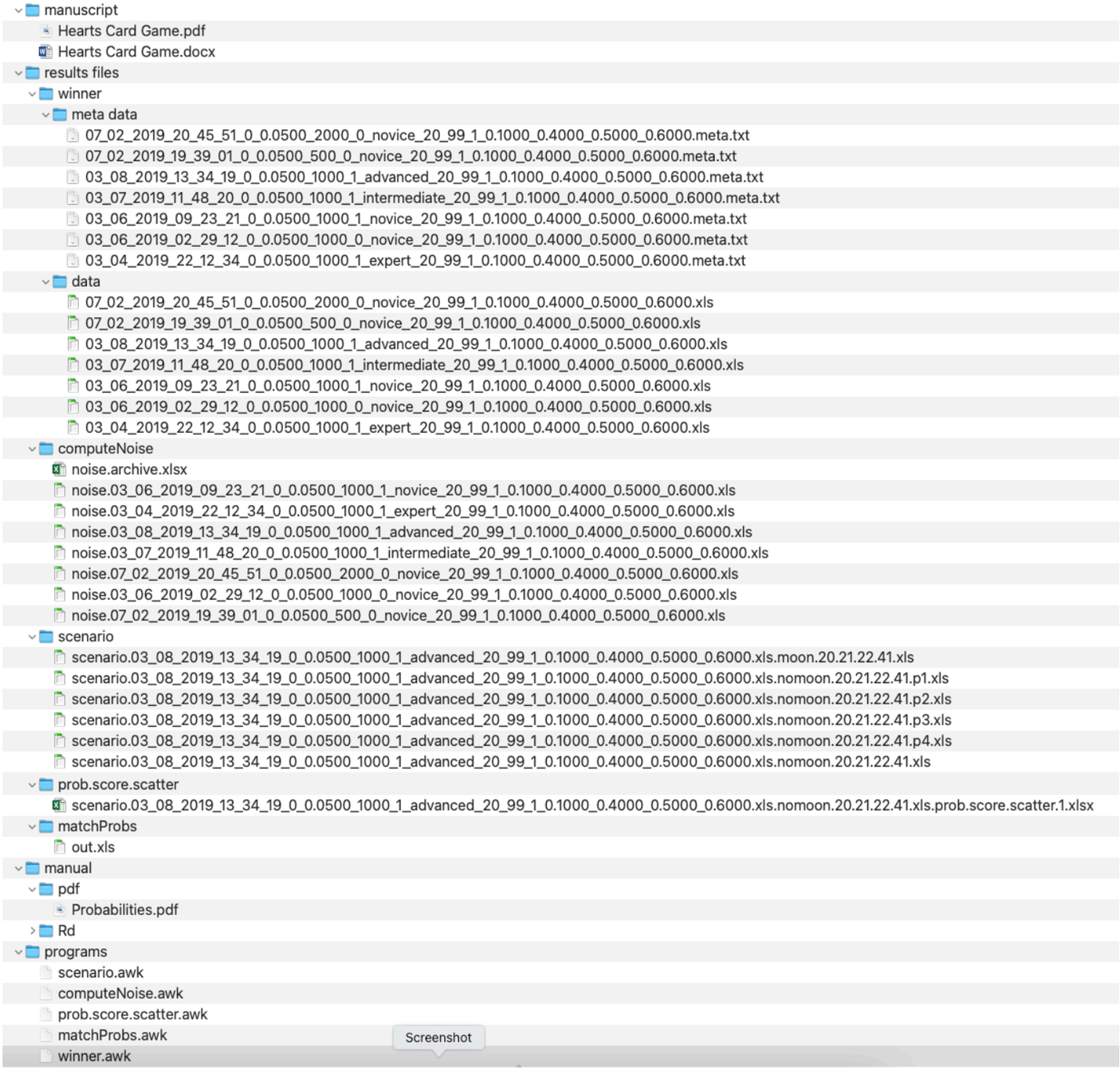
Table 2 lists the programs and their functions

Table 2. Available Programs

Program	Function
winner.awk	• Given each player's score, compute each player's probability of winning
matchProbs.awk	• Given set of 4 scores, compare the 4 corresponding probabilities across multiple files that had been generated by winner.awk • This allows spot checking the reproducibility of repeated runs • Or the effect of number of iterations • Or the effect of changing parameters such as "novice" *versus* "expert" level • Or calibration of proper squash function parameters
computeNoise.awk	• Estimate the noise within the results of a winner.awk run • Compare probabilities that arise from identical scores within a hand and should theoretically be identical
scenario.awk	• Given a current score and 26 points to be added to the score, how are those points best distributed to optimize the chance of winning? • Retrieves the probabilities for all scenarios for all players • Outputs the data to generate a histogram of probabilities for each player
prob.score.scatter.awk	• For a given original position, retrieve the subsequent scores and probabilities from the scenario output file

Table 3. Available Data Results Files

File #	File Name
1	07_02_2019_19_39_01_0_0.0500_500_0_novice_20_99_1_0.1000_0.4000_0.5000_0.6000.xls
2	07_02_2019_20_45_51_0_0.0500_2000_0_novice_20_99_1_0.1000_0.4000_0.5000_0.6000.xls
3	03_06_2019_02_29_12_0_0.0500_1000_0_novice_20_99_1_0.1000_0.4000_0.5000_0.6000.xls
4	03_06_2019_09_23_21_0_0.0500_1000_1_novice_20_99_1_0.1000_0.4000_0.5000_0.6000.xls
5	03_07_2019_11_48_20_0_0.0500_1000_1_intermediate_20_99_1_0.1000_0.4000_0.5000_0.6000.xls
6	03_04_2019_22_12_34_0_0.0500_1000_1_expert_20_99_1_0.1000_0.4000_0.5000_0.6000.xls
7	03_08_2019_13_34_19_0_0.0500_1000_1_advanced_20_99_1_0.1000_0.4000_0.5000_0.6000.xls
8	out.xls
9	noise.07_02_2019_19_39_01_0_0.0500_500_0_novice_20_99_1_0.1000_0.4000_0.5000_0.6000.xls
10	noise.07_02_2019_20_45_51_0_0.0500_2000_0_novice_20_99_1_0.1000_0.4000_0.5000_0.6000.xls
11	noise.03_06_2019_02_29_12_0_0.0500_1000_0_novice_20_99_1_0.1000_0.4000_0.5000_0.6000.xls
12	noise.03_06_2019_09_23_21_0_0.0500_1000_1_novice_20_99_1_0.1000_0.4000_0.5000_0.6000.xls
13	noise.03_07_2019_11_48_20_0_0.0500_1000_1_intermediate_20_99_1_0.1000_0.4000_0.5000_0.6000.xls
14	noise.03_04_2019_22_12_34_0_0.0500_1000_1_expert_20_99_1_0.1000_0.4000_0.5000_0.6000.xls
15	noise.03_08_2019_13_34_19_0_0.0500_1000_1_advanced_20_99_1_0.1000_0.4000_0.5000_0.6000.xls
16	noise.archive.xlsx
17	scenario.03_08_2019_13_34_19_0_0.0500_1000_1_advanced_20_99_1_0.1000_0.4000_0.5000_0.6000.xls.moon.20.
18	scenario.03_08_2019_13_34_19_0_0.0500_1000_1_advanced_20_99_1_0.1000_0.4000_0.5000_0.6000.xls.nomoon.
19	scenario.03_08_2019_13_34_19_0_0.0500_1000_1_advanced_20_99_1_0.1000_0.4000_0.5000_0.6000.xls.nomoon.
20	scenario.03_08_2019_13_34_19_0_0.0500_1000_1_advanced_20_99_1_0.1000_0.4000_0.5000_0.6000.xls.nomoon.
21	scenario.03_08_2019_13_34_19_0_0.0500_1000_1_advanced_20_99_1_0.1000_0.4000_0.5000_0.6000.xls.nomoon.
22	scenario.03_08_2019_13_34_19_0_0.0500_1000_1_advanced_20_99_1_0.1000_0.4000_0.5000_0.6000.xls.nomoon.
23	
24	scenario.03_08_2019_13_34_19_0_0.0500_1000_1_advanced_20_99_1_0.1000_0.4000_0.5000_0.6000.xls.nomoon.20.21.22.41.xls.prob.score.scatter.1.xlsx



A comprehensive summary of all the components is given above. These are available as individual components or as a .zip file from https://github.com/barryzee/Hearts-Scores-Probs. The command lines that generated the output files are given at the top of each program file, but the directory path names will need to be edited for the user's configuration. All files are protected by U.S. Copyright, but they are freely available for download for personal, research, academic, or other non-commercial research and use.

Results

Overview of Some Selected Scores and Probabilities

Table 3 shows the parameter values that were constant across all of the simulation studies. Table 4 shows the parameter values that were varied across the simulation studies.

Table 3. Values of the Parameters that were Held Constant across All Simulation Studies

Parameter	Value
squash5	0.10
squash10	0.40
squash20	0.50
squash100	0.60
Pmoon	0.05
initial	20
final	99
del	1

Table 4. Parameter Values that Varied across the Simulation Studies

N	LEVEL	MODE
1000	novice	bias
1000	intermediate	bias
1000	advanced	bias
1000	expert	bias

1000	novice	no bias
500	novice	no bias
2000	novice	no bias

Table 5 shows several selected lines from the output of winner.awk for a simulation at the advanced level. Each row depicts the scores and probabilities at the end of a completed hand. The first 4 columns show the scores, and the final 4 columns show the corresponding probabilities. Note that, for a given row, the 4 scores, from left to right, are sorted from low score to high score. Thus, each score is *not* identified with a particular player from row to row. That is, the score for a given player may shuttle back and forth between the 4 positions from row to row.

Let us take line 130 as an example. The first 3 scores (*i.e.,* 20, 21, and 22) are nearly identical, and, as we intuitively expect, the corresponding probabilities of ultimately winning the game (0.284047, 0.282101, and 0.315175) are also nearly identical. We note that the third probability is slightly higher than the second probability, although the scores would lead us to expect the opposite. This slight discrepancy is the result of the fact that the level of noise associated with 1000 simulations introduces a small degree of statistical uncertainty in the numerical values of the probabilities. We could compute probabilities to any desired degree of accuracy by increasing the number of simulations, but this would require a lot of additional computation time. In practice, the degree of accuracy for 1000 simulations suffices.

The fourth score (41) is considerably higher than the first 3, and so the probability is much lower (0.118677). The tabulated probabilities allow us to make a quantitative assessment of the chances of winning that is not possible to infer from the scores alone. These quantitative assessments can be used as an *absolute* for a single player, or *relative* comparing multiple players.

Table 5. Selected Lines from the Output of winner.awk for a Simulation at the Advanced Level (Parameters Given in Table 4, Simulation Number 3)

Line Number	Scores				Probabilities			
	1	2	3	4	1	2	3	4
1	20	20	20	44	0.302033	0.312682	0.289448	0.095837
130	20	21	22	41	0.284047	0.282101	0.315175	0.118677
131	20	21	22	67	0.361868	0.300584	0.320039	0.017510
132	20	21	22	93	0.389767	0.337674	0.272558	0.000000
133	20	21	23	40	0.309478	0.289168	0.271760	0.129594
134	20	21	23	66	0.323017	0.330754	0.323017	0.023211
135	20	21	23	92	0.398855	0.345420	0.255725	0.000000
136	20	21	24	39	0.296981	0.315482	0.258033	0.129503
137	20	21	24	65	0.340798	0.359299	0.284323	0.015579
138	20	21	24	91	0.455315	0.324553	0.220132	0.000000
139	20	21	25	38	0.312682	0.289448	0.258470	0.139400
140	20	21	25	64	0.367961	0.340777	0.268932	0.022330
141	20	21	25	90	0.392586	0.390684	0.216730	0.000000
142	20	21	26	37	0.307841	0.317522	0.240077	0.134560
143	20	21	26	63	0.387755	0.344023	0.243926	0.024295

The lines are taken from 03_08_2019_13_34_19_0_0.0500_1000_1_advanced_20_99_1_0.1000_0.4000_0.5000_0.6000.xls

The total number of lines in this file is 70,709.

Noise Characteristics as a Function of the Number of Simulations

The probabilities that arise from identical scores within a hand should theoretically be identical. A measure of the noise associated with various numbers of simulations per study can be determined by assessing the variance in the simulated probabilities for identical scores within a hand. The coefficient of variation is computed for each such hand, and a summary statistic is computed as the mean of the coefficients of variation.

For example, in row 1 of Table 5, the identical score 20 appears 3 times. Theoretically, the associated probabilities should be identical. However, there is a finite degree of variance between the 3 probabilities 0.302033, 0.312682, and 0.289448, resulting in a coefficient of variation of 0.03858962.

The overall degree of variation across all probability levels is exemplified by Figure 2. The coefficient of variation is lower for higher probabilities. This phenomenon results from the fact that higher probabilities, by definition, appear more often in the simulation results, and so are subject to less statistical noise.

To enhance the presentation and allow quantitative comparison between runs, the data were binned and the mean value of the coefficient of variation was computed within each probability bin. For example, to assess the effect of number of simulations upon the noise, Figure 3 compares mean coefficient of variation *per* probability bin. We conclude that using 1000 simulations represents a good compromise between the accuracy and the precision of the estimated probability values, and the computational time.

Analogous graphs for 1000 simulations show that there is no discernable difference between the various skill levels and bias modes (data not shown).

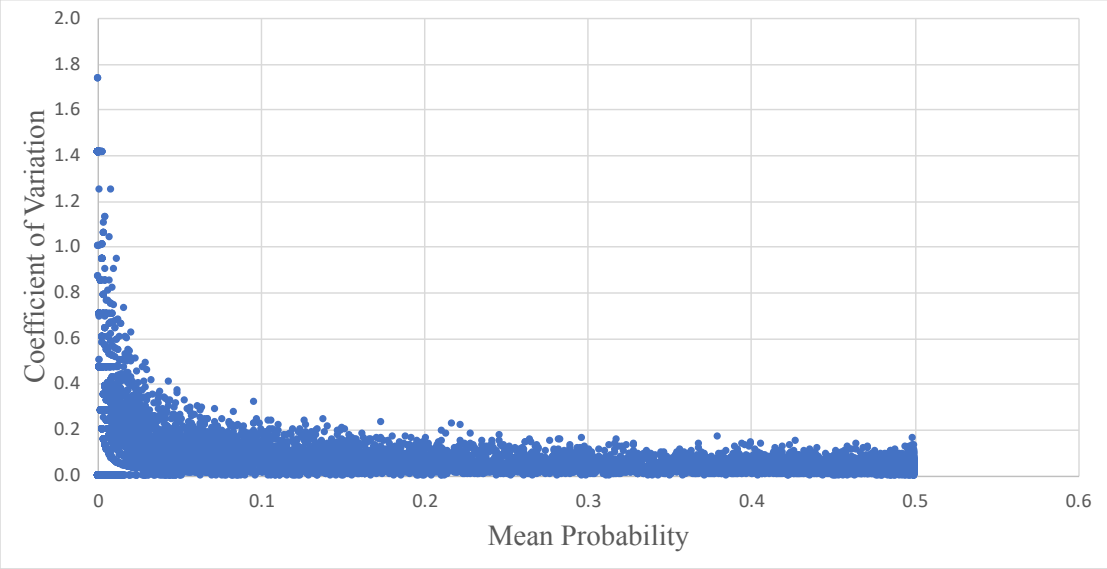


Figure 2. Coefficient of variation *versus* mean probability for scores that appear multiple times in 1 hand. The results are for novice level, no bias mode, 1000 simulations. The awk program computeNoise.awk was used to estimate the noise within the results of a winner.awk run.

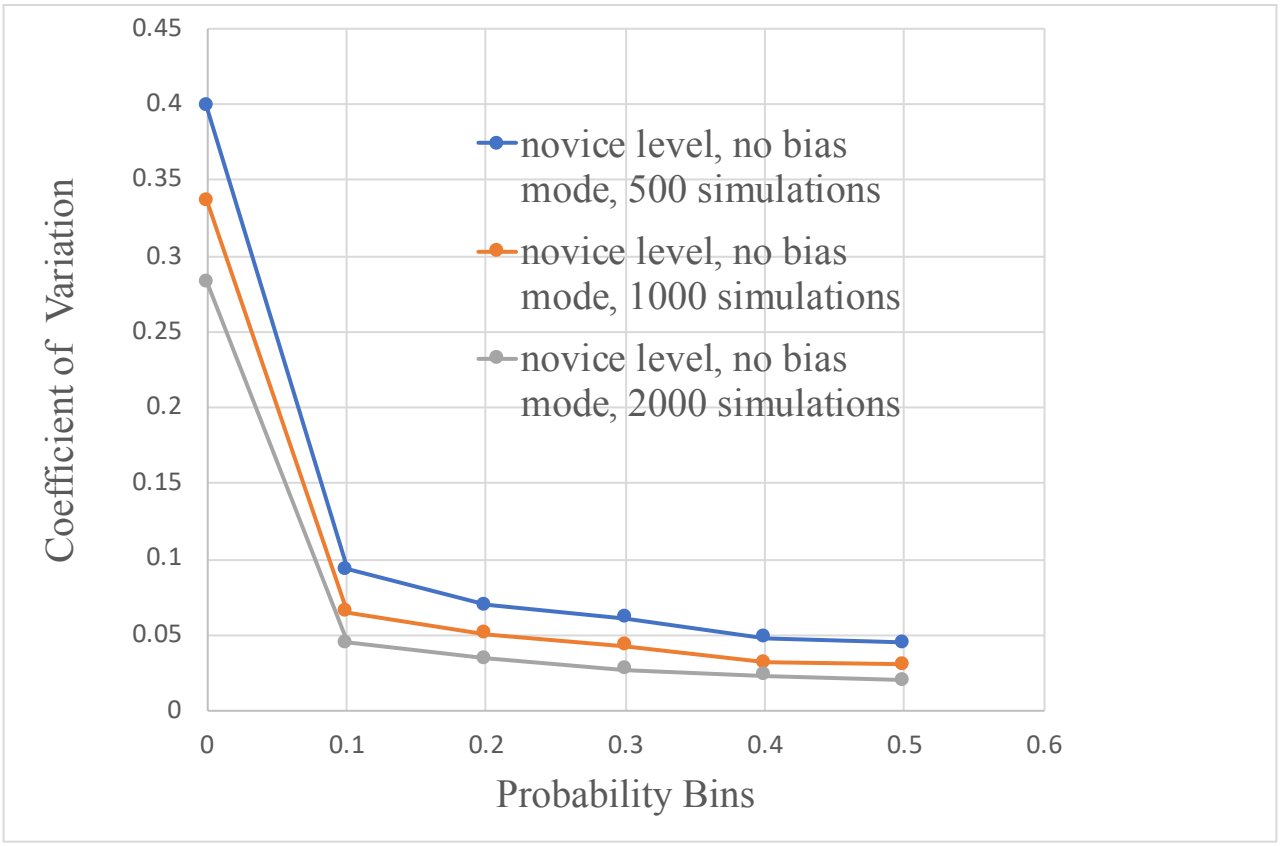


Figure 3. Binned coefficient of variation *versus* probability bins. The axis values of Probability Bins represent the lower end of each bin range. The awk program computeNoise.awk was used to estimate the noise within the results of winner.awk runs.

Comparison of Probabilities for Each Skill Level

Table 6 presents a comparison of probabilities for each skill level. It incidentally shows the agreement between the no bias mode *versus* novice level (see Methods). The main observation is that increasing the level of expertise diminishes the probability differential between the 4 scores. This is a direct result of expert players more effectively targeting the lower score. For example, the probabilities for the position with score 20 systematically decrease from 0.76 for novice to 0.56 for experts. The opposite is true for the position with score 77.

Table 6. Comparison of Probabilities for Each Skill Level

Level	**Probabilities**			
	1	2	3	4
No Bias	0.7640	0.2006	0.0344	0.0010
Novice	0.7647	0.1931	0.0412	0.0010
Intermediate	0.7027	0.2424	0.0461	0.0088
Advanced	0.6634	0.2541	0.0815	0.0010
Expert	0.5614	0.2908	0.1238	0.0240

The awk program matchProbs.awk retrieved probabilities for score 20 35 50 77 (from output files generated by winner.awk)

Moon versus no-moon scenarios

The same file that had been used as the example in Tabel 5 was now used as the input file to the awk program scenario.awk, in 2 successive runs. For both runs, the same starting condition was used, namely the 4 scores given as line 130. In the first run, the parameter *moonflag* was set to 1 in order to retrieve the 4 lines that corresponded to each of 4 players shooting the moon. In the second run, *moonflag* was set to 0 in order to retrieve all of the (many) lines corresponding to no player shooting the moon.

The results for *moonflag* set to 1 are given in Table 7. For reference, the original line 130 is included again in this table. Now consider in some detail

line 2401. In this case, the score for position 1 remained the same (20), and the other 3 score in line 2401 correspond to 3 scores in line 130 plus 26 points each. For instance, 47 = 21 + 26. Thus line 2401 was retrieved as 1 of the 4 lines that represent shooting the moon.

Line 2401 represents a fortuitous circumstance in which the player in original position 1 remained in position 1 in line 2401, and this is also true for the other 3 players. However, the circumstance is changed in line 5628. In this case, the original player in position 2 shoots the moon, but this player is now in position 1 of line 5628. We can see that original player 1 received 26 points, and has moved to position 2 in line 5628 (46 = 20 + 26). It is left as an exercise for the reader to confirm the ordering of positions in lines 8786 and 49554.

As mentioned above, the scores for the first 3 positions of line 130 are nearly identical, and so the corresponding probabilities are nearly identical. The score for position 4 is significantly higher, and so the corresponding probability is significantly lower.

Table 7. Shooting the Moon Scenario

		Line Number in awk Output File	Scores				Probabilities			
			1	2	3	4	1	2	3	4
	Original	130	20	21	22	41	0.284047	0.282101	0.315175	0.118677
Original Position of Player Mooning	**1**	2401	20	47	48	67	0.70374	0.134843	0.121063	0.040354
	2	5628	21	46	48	67	0.680934	0.15856	0.117704	0.042802
	3	8786	22	46	47	67	0.647405	0.165524	0.156709	0.030362
	4	49554	41	46	47	48	0.324402	0.245933	0.22488	0.204785

The output file from which these lines are taken is
scenario.03_08_2019_13_34_19_0_0.0500_1000_1_advanced_20_99_1_0.1000_0.4000_0.5000_0.6000.xls.moon.20.21.22.41.xlsx

After the player in original position 1 shoots the moon, the corresponding probability increases dramatically from 0.284047 to 0.70374. The probabilities for the other 3 positions decrease dramatically. Similar behavior is observed for lines 5628 and 8786. However, in line 49554 it was the player with the original high score (41) who shoots the moon. In this case, the 4 resultant scores are not all that far from one another, and so the 4 probabilities are also somewhat similar.

A few of the results for *moonflag* set to 0 are given in Table 8. In this case 1 of the 4 players has taken the QS, and possibly some hearts. All of those scenarios (numbering 2,213) are retrieved by the awk program scenario.awk. The output format is more complex than for the shooting the moon scenario (Table 7), since the connection between the ordering of positions in the original line 130 and the resultant lines needs to be made explicit.

Let us take line 186 as our example, as this line illustrates an interchange of positions 3 and 4, relative to the original line 130. This can be inferred by the order of positions given in the columns "Original Positions." Comparing Subsequent Scores against Original Scores, we see that the 2 original positions 1 and 2 were point-less, but original position 4 took in 44 – 41 = 3 points, and original position 3 took in 45 – 22 = 23 points. We note that 3 + 23 = 26, so a total of 26 points were taken in. We also note that original position 3 took in the QS plus 10 hearts. As a consequence, we note that the probability for original position 3 dropped significantly from 0.315175 to 0.128180.

For any given hand, 1 of the 4 positions will take the QS (plus possibly some hearts). The consequence will be that the subsequent

Table 8. Taking the QS scenario

	Parameters used Primarily by the Programmer																													
Line Number	Permutation Parameter x	Permutation Parameter y	Permuted Position of the Player who got >= 13	Original Position of the Player who got >= 13	Running Total of Number of Lines Retrieved		Original Positions					Subsequent Scores					Original Scores					Subsequent Probabilities					Original Probabilities			
130																	20	21	22	41							0.284047	0.282101	0.315175	0.118677
134	1	1	4	4	4		1	2	3	4		20	21	23	66		20	21	22	41		0.323017	0.330754	0.323017	0.023211		0.284047	0.282101	0.315175	0.118677
137	1	1	4	4	8		1	2	3	4		20	21	24	65		20	21	22	41		0.340798	0.359299	0.284323	0.015579		0.284047	0.282101	0.315175	0.118677
140	1	1	4	4	12		1	2	3	4		20	21	25	64		20	21	22	41		0.367961	0.340777	0.268932	0.02233		0.284047	0.282101	0.315175	0.118677
143	1	1	4	4	16		1	2	3	4		20	21	26	63		20	21	22	41		0.387755	0.344023	0.243926	0.024295		0.284047	0.282101	0.315175	0.118677
186	1	2	4	3	76		1	2	4	3		20	21	44	45		20	21	41	22		0.407045	0.333659	0.131115	0.128180		0.284047	0.282101	0.118677	0.315175

The output file from which these lines are taken is scenario.03_08_2019_13_34_19_0_0.0500_1000_1_advanced_20_99_1_0.1000_0.4000_0.5000_0.6000.xls.nomoon.20.21.22.41.xls. The total number of lines in this file is 2,213.

probability for that position will drop (as just detailed above). The awk program scenario.awk generates an additional output file for each original position. Each of these additional files contains a list of the subsequent probabilities for the corresponding original position taking the QS. Of course, these exclude the 4 cases of shooting the moon, and there may be a small number of "ambiguous" cases in which 2 positions each take exactly 13 points, so a unique determination of who took the QS is precluded.

A scatter plot of the points taken *versus* subsequent probabilities allows us to determine when it is better to let an opponent shoot the moon, and when it is better to take the QS plus some hearts to prevent an opponent from shooting the moon. The data for constructing the scatter plot can be retrieved from the scenario output file by the awk program prob.score.scatter.awk.

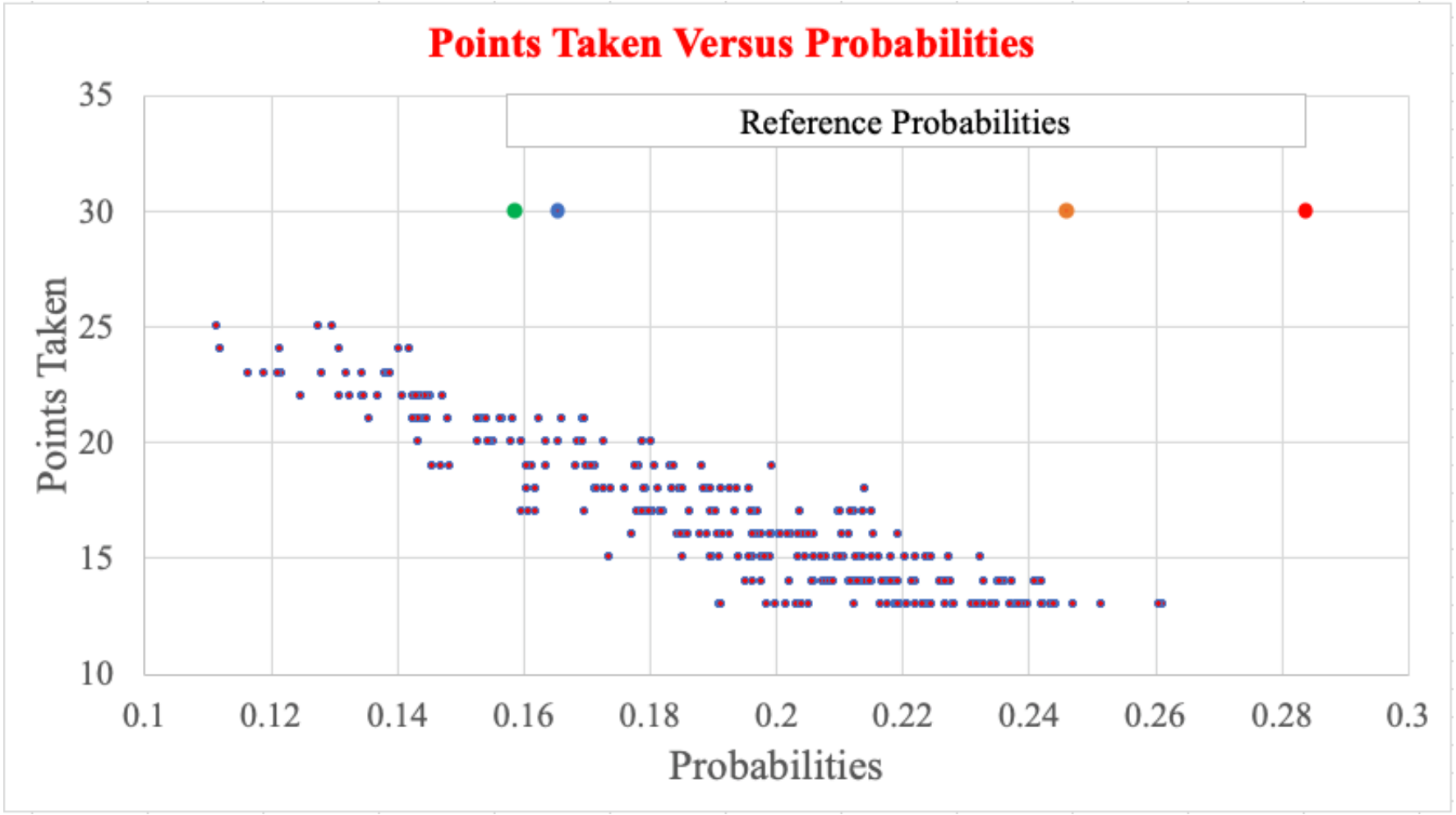


Figure 4. Points taken *versus* subsequent probabilities for original position 1 after hands in which position 1 took at least 13 points. For reference, the probabilities are shown for the same position after hands in which the original table positions 2 (green, p = 0.15856), 3 (blue, p = 0.165524) and 4 (orange, p = 0.245933) shot the moon. For reference, the original probability (red, p = 0.284047) is shown for the same position prior to the hand. These reference probabilities were arbitrarily positioned at 30 points taken, in order to disambiguate them from the real data points that never exceed 25 points taken. The awk program prob.score.scatter.awk retrieved the data from the scenario.awk output file scenario.03_08_2019_13_34_19_0_0.0500_1000_1_advanced_20_99_1_0.1000_0.4000_0.5000_0.6000.xls.nomoon.20.21.22.41.xls.

The scatter plot for original position 1 taking the QS (and possibly some hearts) is shown in Figure 4. These are taken from the full range of results that are exemplified in Tables 7 and 8. For reference, the original probability for position 1, and the probabilities for position 1 subsequent to the other positions shooting the moon, are graphically represented.

The first observation is that all of the subsequent probabilities lie far to the left of the original reference probability (red), as we expect, since taking the QS plus possibly hearts will lower the subsequent probability.

We also see that it is almost always better to allow reference position 4 (orange) to shoot the moon, rather than taking the QS plus hearts to stop the moon. For reference positions 2 (green) and 3 (blue), the determination of the advisability of stopping the moon by taking the QS cannot be generalized, we need to dig down into the data to see the specifics, in particular how many hearts can be taken in addition to the QS before it is better to just allow the moon. For instance, if original position 2 (green) were to shoot the moon, then the subsequent reference probability for original position 1 would be 0.15856. Original position 1 can afford to take no more than 18 points (*i.e.,* the QS plus 5 hearts) to ensure that the subsequent probability would not fall below 0.15856.

Discussion

Playing a hand of hearts requires constant evaluation of off-setting risks and benefits, that goes into the ultimate choice of which card to play. The most dramatic example of this type of dilemma is the choice of taking in the QS in order to stop another player from shooting the moon. Of course, if you take the QS, it is a "personal" liability in that only that your score will be adversely affected. On the other hand, if an opponent shoots the moon, then you share the liability with 2 of your fellow players. Perhaps it is better to accept a shared 26-point liability rather than a personal 13-point liability.

This musing is no longer relegated to the realm of philosophy, as we can compute the probabilities for the 2 competing scenarios, and therefore have a basis for making a rational choice.

The main shortcoming of the current implementation is discussed in the Methods section *"Mapping Pseudo-Random Numbers to Players (winner.awk)."* That is, we still need to implement a model that realistically takes into account the statistical nature of multiple hearts being taken in 1 trick. This could be implemented in the same manner as the current implementation of taking the QS. That is, we could assign a probability to taking 2 or 3 or 4 hearts in 1 trick, and use a random number to indicate when one of those instances occur. That probability would depend on how many hearts are remaining in the current trick (*i.e.,* if there are only 2 hearts remaining then the probability for taking 3 hearts would need to be set to 0). My expectation is that this would introduce a fairly negligible second order correction to the current implementation.

References

See Figure 1.

Acknowledgements

I appreciate the review and feedback by Robert Dalessandro