University of Exeter

College of Engineering, Mathematics and Physical Sciences

ECM3735 - Group 8

Finding the optimal strategy for the dice game Pig

Authors

059863
048874
057162
056854
054426

063602
071680
080004
650015588

Supervisor
Dr. B. Cooper

December 11, 2017

Abstract

Pig is a jeopardy dice game in which two players aim to be the first to a predetermined score. Pig combines strategy and chance in a fun and intuitive way. The more common name of ‘pass the pigs’ is a commercialised adaptation for which the core aims are identical. The ability to optimise one’s strategy is very subtle and is more challenging than it first appears.

This project considers both the theoretical and practical issues surrounding the finding of the optimal strategy. The biggest focus is on creating a method of finding optimums and statistically comparing these optimums so that we can see which strategy is best when played against another. Alongside this, we looked into how humans played Pig and the interactions that occur between players.

We managed to find an optimum strategy which matched T. Nellers’ Optimal, alongside using statistical analysis to consider how many real-world games are needed to compare to a computer. Additionally, when looking at how humans played Pig we found that an individual’s strategy is influenced by how well their opponent is doing and we also found that individual’s do not stick to their stated risk preference during game play.

Contents

List of Figures

1 Introduction

1.1 Aims and objectives

The ultimate aim of this project was to find an optimal strategy (or optimal strategies) for the dice game Pig. The optimal strategy is defined as a strategy that has the highest probability of winning when played against all other strategies. Another aim of the project was to compare our findings to the optimal strategy that T. Neller described in his 2004 paper[5]. Finally, we looked at the psychological aspect of playing Pig; including whether human players play according to their risk preferences (risk-neutral, risk-loving or risk-averse) and whether an individual's strategy is influenced by how well his or her opponent is playing.

We tried to achieve these aims by pursuing the following objectives:

1. Use MATLAB to write a function that calculates the probability of a given strategy winning against another, thus telling us which strategy is better.
2. Develop a script that finds the best strategy to play against any given strategy, which can be used iteratively to find the final optimal strategy.
3. Combine our probability calculator and strategy optimiser to run many tests on different strategies, producing a variety of results to study.
4. Have a sample of people play Pig and record their results so we could determine whether their strategies are influenced by how well (or badly) their opponent was doing, and whether they stick to their stated risk preferences (risk-averse, risk-neutral, risk-loving).

1.2 A brief history of Pig

1.2.1 Basics of Pig

The dice game Pig has many variations, so we chose to focus on the two-player single-die variant. When it is a player's turn, they can repeatedly roll the die as many times as they wish. The number on each roll signifies the points gained on that roll, which is added to a running total of points for the turn. These points can be banked to bring a player's turn to an end. However, all of the points in this running total are lost if the player rolls a 1 before banking, immediately bringing their turn to an end. The aim of the game is to be the first player to have a banked total of at least 100 points.

Many different strategies can be used in Pig. For example, a simple strategy would be to roll until passing a turn score of 20, then banking. A strategy can be thought of as the choice to either roll the die, or bank the turn score, at any game state.

A game state is defined by the current position of the game, player A's banked score, player B's banked score and whichever player's turn it is, turn score. These three scores can vary between zero and 100. The mistake would be to assume that this means there are $100 \times 100 \times 100 = 1$ Million game states. The reason this is not true is because many are not possible. Take a banked score of 90 and a turn score of 50 for example, this game would have ended a long time ago when the player reached a turn score of 10 winning them the game! This could be called an impossible game state, because it is not possible to reach this state, playing by the rules of the game. Another type of impossible game state is with a turn score of 1. The smallest turn score larger than 0 that a player can have is by rolling a 2, giving them a turn score of 2. ¿¿¿¿¿¿¿ 0bd14cdc236bd3c1de090b58ebe206b161988405

Let us now define a strategy as an array of choices, 1s for rolling, and 0s for banking, where the array is 3 dimensional, and containing every possible game state. Does this let us define every possible strategy or are there more variables to consider? If this does define every strategy, the next big question, and the main focus of this paper, is what is the best strategy to play? What strategy is most likely to win against any other, and be crowned the Optimal Strategy for the dice game Pig.

At any game state it is possible to produce linear equations to determine if it is better to roll or bank by looking at the probability of winning from that game state. Suppose there are two strategies A and B. Consider that it is A's turn and they are in position (i, j, k) where i is A's banked points, j is B's banked points, and k is B's sum total of points so far on this turn. Let P_{ijk} denote the probability of A winning from the current position and Q_{jik} denote probability of B winning from the equivalent position (Note that the position of j and i have swapped because it is from the perspective of each player). Then,

$$P_{ijk} = \frac{1}{6}(1 - Q_{ij0}) + \frac{1}{6}\sum_{r=2}^{6} P_{ijk+r} \tag{1}$$

if P rolls and

$$P_{ijk} = 1 - Q_{jik} \tag{2}$$

if P banks their points.

1.2.2 Neller's work

In *Dice Games Properly Explained*, R. Knizia takes the view of each roll of the die in Pig as a bet of not rolling a 1. He said of the best strategy, "Whenever your accumulated points are less than 20, you should continue throwing, because the odds are in your favour."[4]. The issue with this claim is, as T. Neller stated, "*risking points is not the same as risking the probability of winning.*"[5] From this, T. Neller attempted to find a more optimal solution for Pig by maximising equations 1 and 2. However this was not possible as he ended up with an equation of the form $x = \max(A_1x + b_1, A_2x + b_2)$ for which there is no known general method. As a result, T. Neller implemented value iteration to calculate an accurate estimate for the probabilities at each position (i, j, k) for both bank and roll. Figure 1 shows a visual interpretation of the resulting optimal strategy. Figure 1 shows that you should roll if the current game state is within the surface of the graph, and bank otherwise.

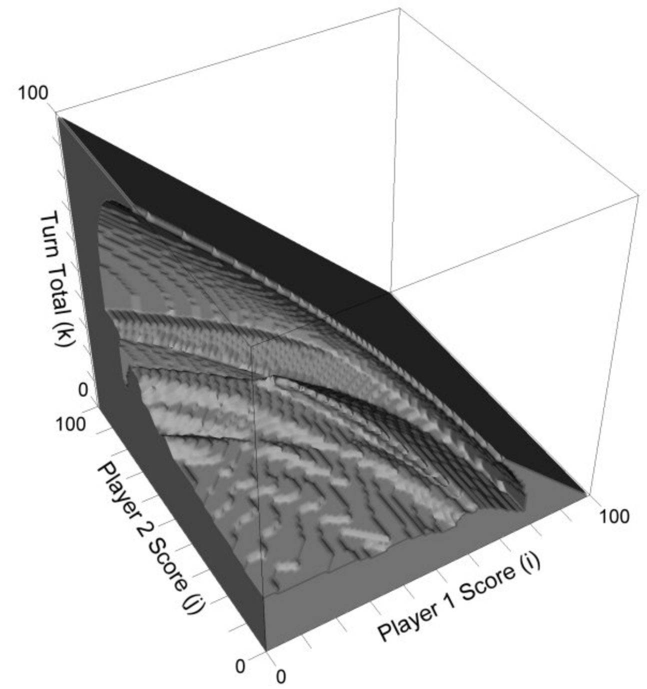


Figure 1: Roll/hold boundary for the optimal Pig play policy (Neller)

1.3 Optimal strategy

The optimal strategy is defined as a strategy that has the highest probability of winning when played against all other strategies. As an example, in a game of rock, paper, scissors, the optimum is to play each of the three choices about a third of the time. However it can occur that an optimal is to play to a draw as to minimise your ability to lose, for example a stalemate in chess. An optimal could arise whereby a win is all but impossible but going for a stalemate is possible and the best solution. The optimum is not set out to get the best win ratio but instead it wants to minimise the maximum loss an opponent can impose on it.

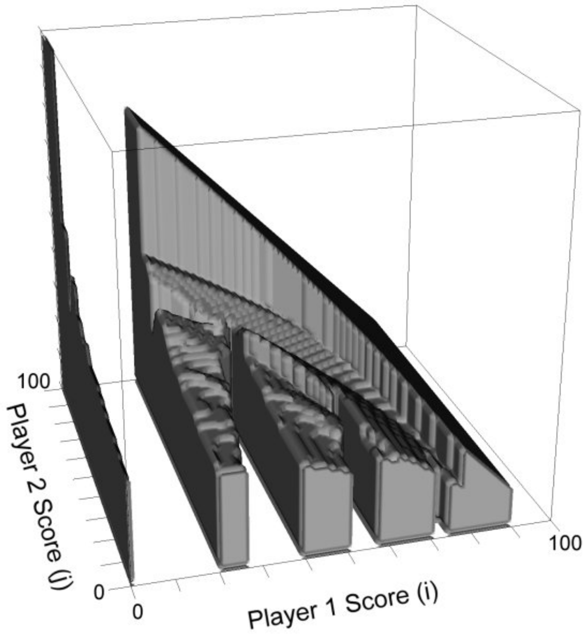
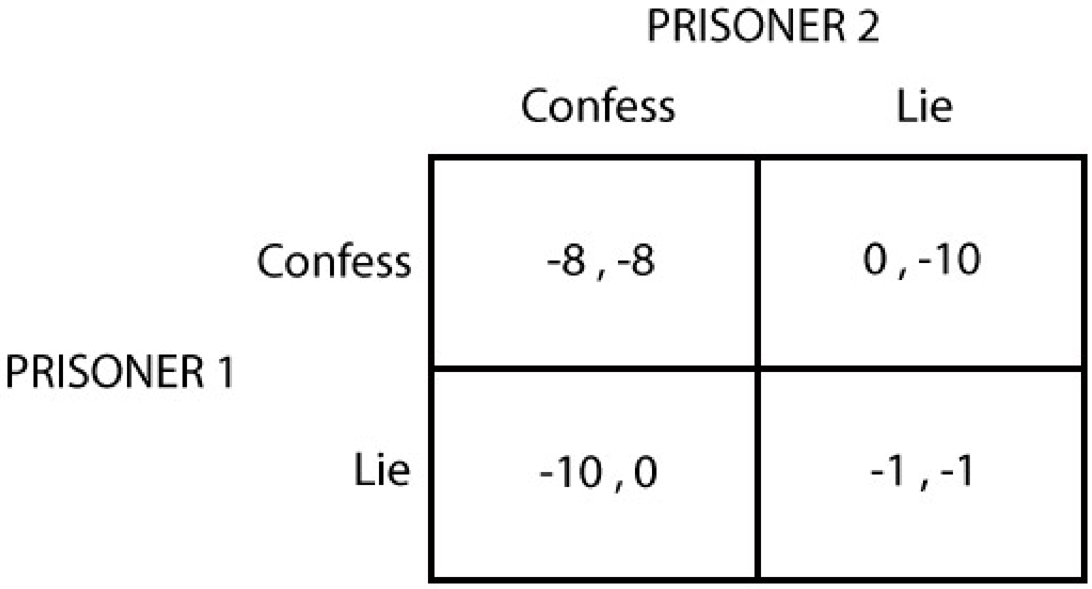


Figure 2: All of the states reachable by an optimal player (Neller)

One of the papers that T. Neller has published looks at finding the optimal strategy for Pig. As such the program looked at all possible game states within the Pig dimension and at each node picks the one with the highest win likelihood. It does this, simply by having the entire state space for the game as a matix of 1's and 0's to which it evaluates whether a 1 (roll) or a 0 (bank) is the best choice to maximise a win. With each of these game states are a series of equations that require solving. In exploring this, we sought to reach Neller's optimal strategy using our own approach or, ideally, find a strategy that surpasses it.

Another possible outcome when looking into the optimal strategy (or optimal strategies) is that we find a pair of strategies that, when played against one another, exist in a Nash equilibrium. A Nash equilibrium[2] is a point in a game space at which two strategies have to get worse against their opponent in order to adapt from their current position. Because of this, it would not be optimal to follow such a path so they remain in that state. This situation is demonstrated well in the prisoner's dilemma. It is defined as the following: *If player A confesses and player B fails to confess, then player A goes free. If player A fails to confess but player B confesses, then player A will be convicted and sentenced for the maximum term in jail and player B goes free. If both players confess, then both players will be convicted, but the maximum sentence will not be imposed. If neither player confesses, both players will be framed for another crime in which a conviction is certain.*

We can see from Figure 3 that the best outcome for each player is not to confess. However this is when one prisoner would look and say that confessing is the better option for them and then does so. Because this scenario is played such that both prisoners cannot discuss it then they would both make this discussion and you are left with the top left box after just a quick step. As you can see, this is not the best outcome for each of them, but they would have to take a greater loss in order to reach the best outcome and so are stuck in this Nash equilibrium.



		PRISONER 2	
		Confess	Lie
PRISONER 1	Confess	-8 , -8	0 , -10
	Lie	-10 , 0	-1 , -1

Figure 3: Prisoners' dilemma diagram

1.4 Preliminary findings

The first step we took to familiarise ourselves with Pig was to do some tests of the game . We began by testing a few basic strategies, such as "Bank at 20". We then tried to formulate our own strategies in an attempt to beat the aforementioned strategies. We were very successful in beating the previously defined strategies, which led us to believe that there are many strategies that in theory would be optimal but in practice are easily beaten.

Next, we started testing strategies on MATLAB using a script to simulate thousands of games between two strategies. R. Kniza theorised that banking at 20 would be one of the most optimal ways to play in "Dice Games Properly Explained". However, we were quickly able to show that this was false, given that it was a worse strategy to play than "Bank at 25". Even though it was a marginal victory it was enough for us to be sure it was a better strategy.

These findings highlighted that there are many ways to perceive what an optimal strategy may be. However, without extensive testing, we did not know if these hypothetical optimal strategies could be beaten easily.

1.5 Human Interactions in Pig

Maximising the probability of winning can be determined through the use of MATLAB, as discovered by T. Neller[5] to create an optimum solution. However, through the analysis of playing Pig ourselves, we discovered the human behavioural aspects that occurred during the game.

1.5.1 Human Interactions in Pig

Developing a strategy that maximises the probability of winning can be determined through the use of MATLAB. However, through the analysis of playing Pig ourselves we also wanted to look into the human interactions that occur when playing Pig.

Therefore, we decided to investigate the following two questions:

We decided to investigate two ideas:

1. How are players' strategies influenced by their opponents'?
2. Do players stick to their stated risk preferences during game play?

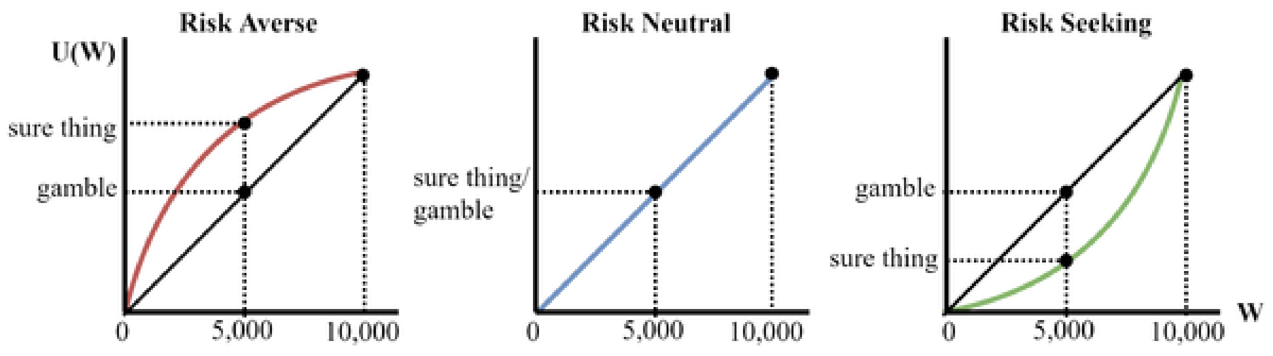


Figure 4: Risk preferences graph

The first investigation looks into a branch of economics called behavioural economics and the second investigation is to include the risk types: risk averse, risk seeking and risk neutral characteristics and to see whether players' stick to their stated risk preference when playing against each other.

Modelling the strategic interaction between the two players allowed us to analyse behavioural patterns (since the game involves known payouts and quantifiable consequences). This allowed us to apply our understanding of game theory to help determine the most likely outcomes.

1.5.2 Behavioural economics

Behavioural economics explains that not all actions taken are purely rationale and a good part of what drives peoples thinking is purely social in nature. For example, when playing Pig, humans are influenced by other aspects such as how well they are doing at a particular game state compared to their opponent.

1.5.3 Risk preferences

For our second investigation, we focussed on the chosen risk preference stated by each player and compared it to how they actually play Pig.

Risk can be defined as the intentional interaction with uncertainty, and individuals differ greatly in their attitudes towards risk.[1]

People differ greatly in their attitudes towards risk. Most individuals would, in general, prefer to take a less risky option when given a choice, they are risk-averse. However, some individuals prefer risk, risk-seekers, and some are indifferent towards risk and are considered risk-neutral. A risk-averse person will usually prefer a sure thing to a gamble where both have the same expected value. Conversely, a risk-seeking individual will try to choose a fair gamble over a sure thing; where a fair gamble is defined as a gamble with an expected payoff equal to that of a sure thing. Sometimes it happens that a risk-seeking person will end up choosing an unfair gamble (where the expected pay-off is less than zero). Finally, a risk-neutral individual prioritises the option with the highest expected monetary value.

We can apply these risk preferences to Pig as follows:
A risk-seeker will roll a greater amount of times on each turn and therefore bank on higher scores. A risk-averse individual will roll a fewer number of times at each turn and bank at lower scores and a risk-neutral player will be somewhere in between these two.

These three risk preferences are represented graphically in figure 4. On the x-axis is the payoff and on the y-axis is the expected utility. We wanted to see how an individuals stated risk preferences affected their strategy when playing Pig. This information could then be used to calculate the optimal strategy for each of the three risk preferences. Over time, a computer could then adapt its strategy to find an optimal strategy for a players risk preference.

1.6 Statistical analysis

In terms of statistical analysis, we wanted to test whether or not a strategy is better than the one it is playing against based on the games they play against one another, given as a probability of winning. Firstly, we needed to establish what we believed to be a fair test between two strategies and the importance of this to our project. We approximated playing two strategies against one another to a statistical distribution and then constructed a confidence interval, at a particular level of significance, and then worked backwards by assuming that a strategy should beat itself with probability 0.5. From this, we intended to find the optimal number of games required to run one single test between any two strategies, so that we could accurately say whether or not a particular strategy is consistently better than another.

After this had been determined, the next step was to decide the number of tests required to run in order to gain a desired degree of accuracy. Through this we were able to produce an good approximation of the probability that one of each strategy winning with the player going first alternated on each trial. If this probability lies above the upper bound of the confidence interval, then the particular strategy could be declared better than its opponent. Similarly, a probability falling below the lower bound indicated that it was worse. However, if the probability fell inside the confidence interval, the two strategies were considered inseparable at our given level of significance, meaning we could not strictly say that one is better than the other.

We have then applied this in the same way to outcomes from our optimal strategy against other strategies. We hoped to have shown that this strategy is superior to other simple strategies before comparing to those that are more complicated such as Nellers optimal strategy. We then, checked to see if our optimal strategy has converged to that of Nellers by the two being inseparable, after running the tests, at the given level of significance.

2 Methodology

2.1 Group organisation

This section discusses the methods used to organise all members of the group and to ensure that the project progressed smoothly.

2.1.1 Meetings

To ensure all 9 members of the group were able to meet up we created an Excel Google Sheets page and then input our timetable schedules into it to highlight all of our availabilities. Creating a Google Sheet meant all members could access and edit the timetable to ensure we were always aware of availability. We found that Wednesday morning was always a free space, so from week one we scheduled a weekly meeting on Wednesday morning for two hours where we could discuss current matters, updates on the project and delegate future tasks. This ensured that we kept in strong communication with each other to ensure all group members were up to date with our progress and most importantly make sure all information remained consistent across all group members.

In our first meeting, we were able to assign aspects of the project to each individual member. We firstly went round and every member expressed their personal strengths and weaknesses, therefore ensuring everyone's strongest attributes were utilised throughout the project. We decided on splitting the project into three sections: coding, human interactions and statistical testing, and assigned these accordingly to sub groups.

Communication was also key, and through the use of WhatsApp - a freeware and cross-platform instant messaging service - we were able to stay in constant communication and send documents safely. This also provided a platform to keep all documents in one place so that they were accessible to all members. An advantage of being able to share documents through WhatsApp is that it minimised the possibility of data loss and also made all information accessible at all times.

2.1.2 Creation of project plan

Creating our project plan highlighted the key risks that could occur within our project and how many concurrent activities were to be undertaken in our critical path. We endeavoured, as a group, to attempt to stick to the path and the times assigned to each section. This was completed successfully. In addition to this, we made sure to mitigate the risks identified in our project plan to ensure that our project progressed smoothly.

2.2 Understanding the equations

As discussed briefly in the introduction, linear equations can describe the probability of winning from any given game state. We let $P_{i,j,k}$ denote the probability of player A winning from the game state i, j, k and $Q_{j,i,k}$ denote the probability of player B winning from the game state j, i, k where i is A's banked score, j is B's banked score, and k is the turn score of the player whos turn it it.

Firstly, we looked at the probability A had of winning when it banks, denoted $P^{bank}_{i,j,k}$. A passes to B's turn by banking. So $P^{bank}_{i,j,k}$ will be equal to the probability A has on B's turn. The probability $Q_{j,i,k}$ is the probability that player B has of winning on their turn at their game state j, i, k. As long as the game is finite, Pig will always have a winner. Therefore, the sum of the probabilities from both players at any point in the game will be 1. Note, that this is from any point in the game, not at any game state, as a game state is specific to a player. From this we concluded that when it is player B's turn at game state j, i, k, player A will have a probability of winning that is $1 - Q_{j,i,k}$. By banking, A adds their turn score to their banked score, and B's turn starts with a turn score of zero. Thus, we define the probability A has at any game state of banking as

$$P^{Bank}_{i,j,k} = 1 - Q_{j,i+k,0}$$

Secondly, we looked at the probability A has of winning when it rolls, denoted $P^{roll}_{i,j,k}$. Similar to banking, the probability relies on a probability from B, $Q_{j,i,0}$, because when a 1 is rolled the game passes to B's turn. The main difference is that the turn score is not added to the banked score, as can be seen in the game state, and there is only a $\frac{1}{6}^{\text{th}}$ chance of it happening. $P^{roll}_{i,j,k}$ will also rely on the other successful outcomes of rolling each with a $\frac{1}{6}^{\text{th}}$ chance of happening. These other probabilities all have a higher turn score. r represents the scores that can be added, ranging from 2 to 6 depending on the number rolled. These probabilities are all summed together.

$$P^{Roll}_{i,j,k} = \frac{1}{6}(1 - Q_{j,i,0}) + \frac{1}{6}\sum_{r=2}^{6} P_{i,j,k+r}$$

How do we find out the useful probabilities that we have produced notation for? To find the useful probabilities that we have produced notation for , we looked at trivial probabilities and very simple calculations. The following equation gives the probability of winning from the game state where the banked score of player A is 100. By the rules of the game, A has won, meaning this probability is equal to 1.

$$P_{100,0,0} = 1$$

This is one of many trivial game state probabilities that are important to understand. Another would be

$$P_{82,0,20} = 1$$

In this case, strategy A has accumulated a big enough turn score to immediately bank and win. Generally, if $i + k \geq 100$ and $j < 100$ then the probability of that game state is 1, as it means the strategy has won. On the other hand, take the following game state probability from A's perspective:

$$P_{99,100,0} = 0$$

This game state is where the opponent has won the game, so strategy A has 0 probability of winning. Generally, if $i < 100$ and $j \geq 100$, then the probability at that game state is 0, as it means the other player has won.

These trivial game state probabilities allowed us to calculate the non-trivial probabilities of a game. We looked at this next.

2.2.1 Simultaneous equations

Firstly, we started with a simple pair of equations which describe the probabilities of game states near the very end of the game. Both of the strategies are on 99 points, and it is the beginning of strategy A's turn. We assume that both A and B roll on game state $99, 99, 0$. It may seem an obvious choice, as they have no turn score to lose, but the idea of banking on a turn score of zero is something we investigated further later on in this paper.

$$P_{99,99,0} = \frac{1}{6}\Big((1 - Q_{99,99,0}) + P_{99,100,2} + P_{99,100,3} + P_{99,100,4} + P_{99,100,5} + P_{99,100,6}\Big)$$

From what we know about trivial probabilities, we see that if A successfully rolls, i.e. doesn't roll a 1, they win the game. Therefore, the probabilities can be written as follows filling in the trivial probabilities:

$$\begin{aligned}
P_{99,99,0} &= \frac{1}{6}\Big((1 - Q_{99,99,0}) + (1) + (1) + (1) + (1) + (1)\Big) \\
P_{99,99,0} &= \frac{1}{6}(1 - Q_{99,99,0}) + \frac{5}{6} \\
P_{99,99,0} &= 1 - \frac{1}{6}Q_{99,99,0}
\end{aligned}$$

Similarly, for B's turn:

$$\begin{aligned}
Q_{99,99,0} &= \frac{1}{6}\Big((1 - P_{99,99,0}) + Q_{99,100,2} + Q_{99,100,3} + Q_{99,100,4} + Q_{99,100,5} + Q_{99,100,6}\Big) \\
Q_{99,99,0} &= \frac{1}{6}\Big((1 - P_{99,99,0}) + (1) + (1) + (1) + (1) + (1)\Big) \\
Q_{99,99,0} &= \frac{1}{6}(1 - P_{99,99,0}) + \frac{5}{6} \\
Q_{99,99,0} &= 1 - \frac{1}{6}P_{99,99,0}
\end{aligned}$$

Now the two equations can be solved simultaneously, giving the results:

$$\begin{aligned}
P_{99,99,0} &= \frac{6}{7} \\
Q_{99,99,0} &= \frac{6}{7}
\end{aligned}$$

This example is one complete set of simultaneous equations. It turns out that every probability can be collected into a set of equations to be calculated. While this one only contained two unknowns to be solved together, others may be much larger. A more complex simultaneous set can be seen if we start from A's turn at game state $97, 99, 0$. Let A's strategy be to roll at this game state.

$$P_{97,99,0} = \frac{1}{6}(1 - Q_{99,97,0}) + \frac{1}{6}P_{97,99,2} + \frac{4}{6}$$

After writing this out, we saw the other probabilities it relies on. Next, we wanted to see what the strategies do at these game states and then we wrote out the probability equations for them. In this example, let A roll at $97, 99, 2$ and B roll at $99, 97, 0$

$$\begin{aligned} P_{97,99,2} &= 1 - \frac{1}{6}Q_{99,97,0} \\ Q_{99,97,0} &= 1 - \frac{1}{6}P_{97,99,0} \end{aligned}$$

Now we have all three simultaneous equations that can be solved:

$$\begin{aligned} P_{97,99,0} &= 0.8325 \\ P_{97,99,2} &= 0.8565 \\ Q_{99,97,0} &= 0.8612 \end{aligned}$$

This method of calculation worked well. However, before we ran through every set of equations, we had to look at the interesting case of banking. For example, instead of rolling at $97, 99, 2$, A banks. The equations are now:

$$\begin{aligned} P_{97,99,0} &= \frac{1}{6}(1 - Q_{99,97,0}) + \frac{1}{6}P_{97,99,2} + \frac{4}{6} \\ P_{97,99,2} &= 1 - Q_{99,99,0} \\ Q_{99,97,0} &= 1 - \frac{1}{6}P_{97,99,0} \end{aligned}$$

We did not need to write out the equation for $Q_{99,99,0}$ because we already calculated it in our first example. Its value was $\frac{6}{7}$. So with this we managed to solve the equations, getting a much different result:

$$\begin{aligned} P_{97,99,0} &= 0.7102 \\ P_{97,99,2} &= 0.1429 \\ Q_{99,97,0} &= 0.8816 \end{aligned}$$

From this we concluded that when a strategy banks, it relies on knowledge of the probabilities of game states with higher banked scores than itself, which we shall be called "higher game states" from this point onwards. Note that it will not rely on game states "lower" than it as the game will always be progressing forward and a player cannot lose banked points. Once we had solved the sets containing higher game states first, we could run through the rest of the sets and solve them all successfully.

Solving these equations meant we could calculate every probability of winning from every game state for any given strategy. From this we could check $P_{0,0,0}$, which is the probability that any strategy has of winning the game. Computational power was required to give us the probability values that we wanted to calculate. This is where we turned to MATLAB for assistance.

2.2.2 Matrices

To make use of MATLAB, we wrote the equations in terms of matrices. Every set of equations can be written in the form $CX = D$, where C, X and D are matrices containing the coefficients, the probabilities, and the constant values respectively. We looked at the examples above to build these matrices.

The first part was to simplify the equations. We wanted them in terms of separated probabilities with the coefficient values on the left-hand side of the equation, and a combined constant value on the right. This involved expanding brackets and adding constants together.

We have:

$$\begin{aligned} P_{99,99,0} &= \frac{1}{6}\Big((1 - Q_{99,99,0}) + 1 + 1 + 1 + 1 + 1\Big) \\ Q_{99,99,0} &= \frac{1}{6}\Big((1 - P_{99,99,0}) + 1 + 1 + 1 + 1 + 1\Big) \end{aligned}$$

This becomes:

$$\begin{aligned} P_{99,99,0} + \frac{1}{6}Q_{99,99,0} &= 1 \\ Q_{99,99,0} + \frac{1}{6}P_{99,99,0} &= 1 \end{aligned}$$

Using basic ideas of matrix multiplication, we can construct a matrix:

$$\begin{bmatrix} 1 & \frac{1}{6} \\ \frac{1}{6} & 1 \end{bmatrix} \begin{bmatrix} P_{99,99,0} \\ Q_{99,99,0} \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

This was useful as we could then take the inverse of the coefficients matrix, using it in left matrix multiplication to find the probabilities.

$$\begin{aligned} C^{-1}CX &= C^{-1}D \\ X &= C^{-1}D \end{aligned}$$

MATLAB has a function that did this for us, giving the probability values in the matrix X. Looking at the other two examples:
Example where A rolled on $97, 99, 2$.

$$\begin{aligned} P_{97,99,0} + \frac{1}{6}Q_{99,97,0} - \frac{1}{6}P_{97,99,2} &= \frac{5}{6} \\ P_{97,99,2} + \frac{1}{6}Q_{99,97,0} &= 1 \\ Q_{99,97,0} + \frac{1}{6}P_{97,99,0} &= 1 \\ \begin{bmatrix} 1 & \frac{1}{6} & -\frac{1}{6} \\ \frac{1}{6} & 1 & 0 \\ \frac{1}{6} & 0 & 1 \end{bmatrix} \begin{bmatrix} P_{97,99,0} \\ P_{97,99,2} \\ Q_{99,99,0} \end{bmatrix} &= \begin{bmatrix} \frac{5}{6} \\ 1 \\ 1 \end{bmatrix} \end{aligned}$$

Example where A banked on $97, 99, 2$.

$$P_{97,99,0} - \frac{1}{6}P_{97,99,2} + \frac{1}{6}Q_{99,97,0}) = \frac{5}{6}$$
$$P_{97,99,2} = 1 - Q_{99,99,0}$$
$$Q_{99,97,0} + \frac{1}{6}P_{97,99,0} = 1$$
$$\begin{bmatrix} 1 & \frac{1}{6} & -\frac{1}{6} \\ 0 & 1 & 0 \\ \frac{1}{6} & 0 & 1 \end{bmatrix} \begin{bmatrix} P_{97,99,0} \\ P_{97,99,2} \\ Q_{99,99,0} \end{bmatrix} = \begin{bmatrix} \frac{5}{6} \\ 1 - Q_{99,99,0} \\ 1 \end{bmatrix}$$

Notice how $Q_{99,99,0}$ is in the constant values matrix. As discussed before, this is because its value will have already been calculated from a previously solved simultaneous set.

2.2.3 Piglet

To simplify the task, we first attempted to construct the matrices for a variant of Pig known as Piglet. In this game, players take it in turns to flip a coin until either a tail is flipped, or the player banks their turn score.

This simpler game contained all the same elements as Pig but on a smaller scale. Plainly, it is played with a coin rather than a die, so that probabilities are generated by $\frac{1}{2}$, instead of the $\frac{1}{6}$ given by a die. The other difference is that you typically play to a much lower score than the 100 in Pig. This score could be chosen to be whatever we wanted, often 10 but sometimes as low as 2 or 3 for the most simple games.

A subtle yet vital difference is that 1 is added for a successful flip. Consider if the coin was a two-sided die, with 1 and 2 rather than tails and heads respectively. If it was the same as Pig, the face value of the roll would be added to the turn score, so 2, not 1. We accepted that Piglet adds 1 to the turn score on a flip of a head.

With a coin you can only score 1 per flip, meaning you must flip a minimum of 10 times for you to reach your end goal. In Pig this is not the case, as you can roll anywhere from 2 to 6 points on a successful roll, giving you a minimum of 17 rolls to achieve your goal. Having a change in the minimum number of flips allows luck to have a much greater presence in the outcome of the game, as we observed upon playing it ourselves.

Our goal was to find the probability of a strategy winning when going first. For strategy A, the probability of winning is $P_{0,0,0}$, as this is the starting point of the game on A's turn. Piglet's simplicity made it the perfect first stepping stone to solving this.

2.2.4 Handwritten notes

We first played Piglet to a total score of 3, taking strategy A as banking on a turn score of 1 and strategy B as banking on a turn score of 2. (Both strategies would bank by default as soon as they got to the winning score.) From this we wrote out all the simultaneous equations by hand. This starts at the set that considers the probability $P_{2,2,0}$ and $Q_{2,2,0}$ whereby each strategy wins on its next go, similar to our first example in Pig. Once we had these equations written in terms of matrices, we manually put them into MATLAB and got it to solve them. From this we worked down through all simultaneous sets to obtain the probabilities of each strategy winning from game state $0, 0, 0$.

We repeated this process for a game to total score 2. We used the same strategies again; strategy A was to bank on a turn score of 1; strategy B was to bank on a turn score of 2, which is equivalent to rolling at every game state, what we called "rolling to win". We achieved another answer which looked relatively correct. As a group we noticed a pattern at this point to which the number of sets of simultaneous equations is that of the total score squared. It was here we realised that working out Piglet to 10 would require 100 sets of equations.

2.3 Coding

Once we had familiarised ourselves with the rules of the game, the equations, the matrices, and a few simple Piglet examples, we began building MATLAB scripts. We had two functions we wanted to produce. One to calculate probabilities as we described so far and another to calculate optimised strategies.

2.3.1 Initial Calculations

Our first attempt at getting our ideas down as code was a very simplified calculation of Piglet. Although we coded in a dice probability variable that could be changed, the code only worked for a dice flip, and when the variable was set to $\frac{1}{2}$. It was also limited to taking in only Bank on strategies. We could however run to any winning score.

By using the hand calculated simple Piglet probabilities, we could check to see if our coded matrices were correct and fix any bugs. We even found that we could use the MATLAB-produced matrices to correct our handwritten mistakes. Once this was working we had our first runnable code. Next we had to fully generalise the code for Pig. There were two steps to doing this, firstly taking any strategy as an input, and secondly, having any dice probability working in the code. The former was easier; an independent problem that we made a function in MATLAB. The latter was a little trickier but was a fundamental part of building the matrices.

2.3.2 Strategies to matrices

In our introduction, we defined a strategy as a 3 dimensional array of choices. Another way, and the way that we were introduced to strategies, is by a MATLAB function. These functions take inputs of “my score”,“your score”, and “turn score”, then output a “choice” that is a logical value of 1 or 0. Below is an example of Bank on 20.

```matlab
function choice = Bank20(myscore,yourscore,turnscore)
    if turnscore < 20
        choice = 1; % Roll
    else
        choice = 0; % Bank
    end
end
```

This way of defining a strategy was used when running the code for the game Pig, which called the function strategies within it. However, it limited the way that more abstract strategies could be represented. Take a strategy that rolls in very specific locations for example, the function strategy would have to include many ‘if’ statements to handle this, leading to a very big and hard to read function. An array of choices on the other hand are always on the same size, and importantly it is easy to change if we wanted to. Additionally, it was easy to locate the choice at a given game state.

It made sense to work with matrix strategies, but we still wanted to use function strategies in our code. Therefore, we devised a function *Strategies to Matrices* that could convert functions to matrices but leave the strategy alone if it is already a matrix. First of all, we checked that the input was a function handle, then it ran through every game state, outputting the choice that the function produces, then stored that value in the corresponding matrix. We also used this function to check that the Matrix was the correct size for the game. Unlike function strategies, the matrix strategies only worked for a certain game length.

This allowed us to check what the strategy wanted to do at any game state, which was important for the type of equation to use when building the matrices.

2.3.3 Coding Matrices

This section covers how to build general matrices for any set of simultaneous equations.

Firstly, we needed to consider which probability values rely on each other to produce a single set. Take $P_{i,j,0}$ the probability of A winning at the beginning of their turn. Assuming A rolls (we looked at banking on 0 later on in this paper), $P_{i,j,0}$ will rely on other probabilities. On a successful roll, the new probabilities have different turn scores k, but same banked score i and j. On an unsuccessful roll, a new probability from B’s perspective is required; $Q_{j,i,0}$. This has not been previously calculated as it is not from a higher game state, therefore must be considered. Continuing on from the probabilities from A’s perspective that we must now also consider, they can either bank or roll again. If A banks, that probability will be equal to $1 - Q_{j,i+k,0}$, where k is the now non zero turn score. As $Q_{j,i+k,0}$ is for a higher game state, it would have already been calculated, so there were no new probabilities to consider. Banking was an easy choice to deal with for this reason. If A rolls again at the new game states with a higher turn score, more probabilities must also be considered. The important fact however is that only the turn score has changed again. They also rely on $Q_{j,i,0}$ if the roll is unsuccessful, which is one we are already considering. It seems then that we must consider all P probabilities containing a higher turn score but the same banked scores, as our original $P_{i,j,0}$, up until the turn score has won A the game.

So far we have only considered P probabilities from A's perspective, but we also rely on $Q_{j,i,0}$. Similarly to what we have discussed with repeats to A's probabilities P, we also had to consider all Q probabilities with higher turn scores up until the turn score where B had won the game. Again however, only the turn score is changing. It can be concluded then that each set of simultaneous equations is built around fixed banked scores i and j. This fact could also have been reached by realising that any higher game state (higher banked values) than the current one would have already been calculated so would not produce another equation, and no lower game state (lower banked values) would ever be called because the players cannot lose their banked score.

From this we can see that there must be a simultaneous set for every banked score i for A and every banked score j for the opponent B. In the Piglet section we explained how we noticed that the number of sets was the "to win" score squared. It is now clear to see why this is the case. i and j can take the same amount of values as the "to win" score so multiplying them together for every possible combination gives the "to win" score squared. This gives us some scope on how many sets will need to be solved in Pig: $100 \times 100 \times 100 = 1$ million.

Now we have a rough idea of every equation in any simultaneous set. For clarity, we will order them as follows in the probabilities matrix, X. There is an artificial line to seperate the Ps and Qs, for our visual understanding only.

$$\begin{bmatrix} P_{i,j,0} \\ P_{i,j,1} \\ \vdots \\ P_{i,j,n-1} \\ P_{i,j,n} \\ - \\ Q_{j,i,0} \\ Q_{j,i,1} \\ \vdots \\ Q_{j,i,m-1} \\ Q_{j,i,m} \end{bmatrix} \tag{3}$$

where $n =$ "to win score" $- i - 1$ and $m =$ "to win score" $- j - 1$.

The first problem to our assumptions was that we needed all of these equations. This only outlines all of the ones that could be needed. Take for example, the set built around $0, 0$, where A is banking on 20. $P_{0,0,50}$ for example would never be considered, as the strategy would have banked many rolls ago. For banking on 20 strategy, the game state $0, 0, 50$ and many others would be impossible. This idea of impossible game states was touched on in the introduction, the difference here is that this one is specific to a strategy. As discussed in the introduction, a turnscore of 1 is also not possible for Pig.

We found that this problem had an elegant solution. The definition of an impossible game state is one that cannot be reached from other game states. The coefficient's matrix is filled from top to bottom, meaning lower turnscore probabilities are completed first. Therefore, if we got to any probability that is an impossible game state, it would not have been called by any other game states before it. Therefore all the coefficient values for it would be zero. This is a simple to check in our program. We then decided to ignore this row of the matrix, which is the same as it not being there. With this problem solved, we could use the set probability matrix described above, without worrying if the probability is relevant or not.

Next we could fill in the coefficient matrix and the constant value matrix. This was done by checking what the strategy does at each game state, either rolling or banking. This wasnmuch easier with strategies in matrix form. By making use of MATLAB's variables, for loops and general programming techniques, we successfully put all the needed values into the matrices.

$$\left[\begin{array}{ccccccccccccc}
1 & 0 & -\frac{1}{6} & \cdots & -\frac{1}{6} & 0 & 0 & | & \frac{1}{6} & 0 & 0 & \cdots & 0 \\
\vdots & \ddots & & & & & \vdots & | & & & & & \\
\vdots & & \ddots & & & & 0 & | & & & & & \\
0 & \cdots & 0 & 1 & 0 & 0 & \cdots & | & 0 & 0 & \cdots & 0 & 0 \\
\vdots & & & & \ddots & & \vdots & | & & & & & \\
\vdots & & & & & \ddots & 0 & | & & & & & \\
0 & 0 & 0 & \cdots & 0 & 0 & 1 & | & 0 & 0 & \cdots & 0 & 0 \\
- & - & - & - & - & - & - & + & - & - & - & - & - \\
 & & & & & & \vdots & | & \ddots & & & & \\
 & & & & & & \vdots & | & & \ddots & & & \\
\frac{1}{6} & 0 & \cdots & 0 & 0 & 0 & 0 & | & 0 & \cdots & 1 & 0 & -\frac{1}{6} \\
 & & & & & & \vdots & | & & & & \ddots & \\
0 & 0 & 0 & \cdots & 0 & 0 & 0 & | & 0 & 0 & \cdots & 0 & 1
\end{array}\right]
\left[\begin{array}{c}
P_{i,j,0} \\ \vdots \\ \vdots \\ P_{i,j,k} \\ \vdots \\ \vdots \\ P_{i,j,n} \\ - \\ \vdots \\ \vdots \\ Q_{j,i,m-2} \\ \vdots \\ Q_{j,i,m}
\end{array}\right]
=
\left[\begin{array}{c}
\frac{1}{6} \\ \vdots \\ \vdots \\ 1-Q_{j,i+k,0} \\ \vdots \\ \vdots \\ 0 \\ - \\ \vdots \\ \vdots \\ \frac{5}{6} \\ \vdots \\ 1-P_{i,j+m,0}
\end{array}\right]$$

We then ran through every game state backwards, collecting the probabilities as we solved them. Remembering that matrices might need the probabilities calculated from higher game states, so whilst we built our collection of probabilities, we had to call upon them as we worked down. Once we had completed every matrix calculation, we had a generalised working code that could calculate the probabilities of winning at any game state, for any two strategies, for any given dice probability and to any winning score.

The matrices above are a very generalised version used for Pig. Although the number of line can vary, each written out row in this example shows a different situation that is dealt with from both strategies perspective. The dotted lines show the seperation between Ps and Qs.

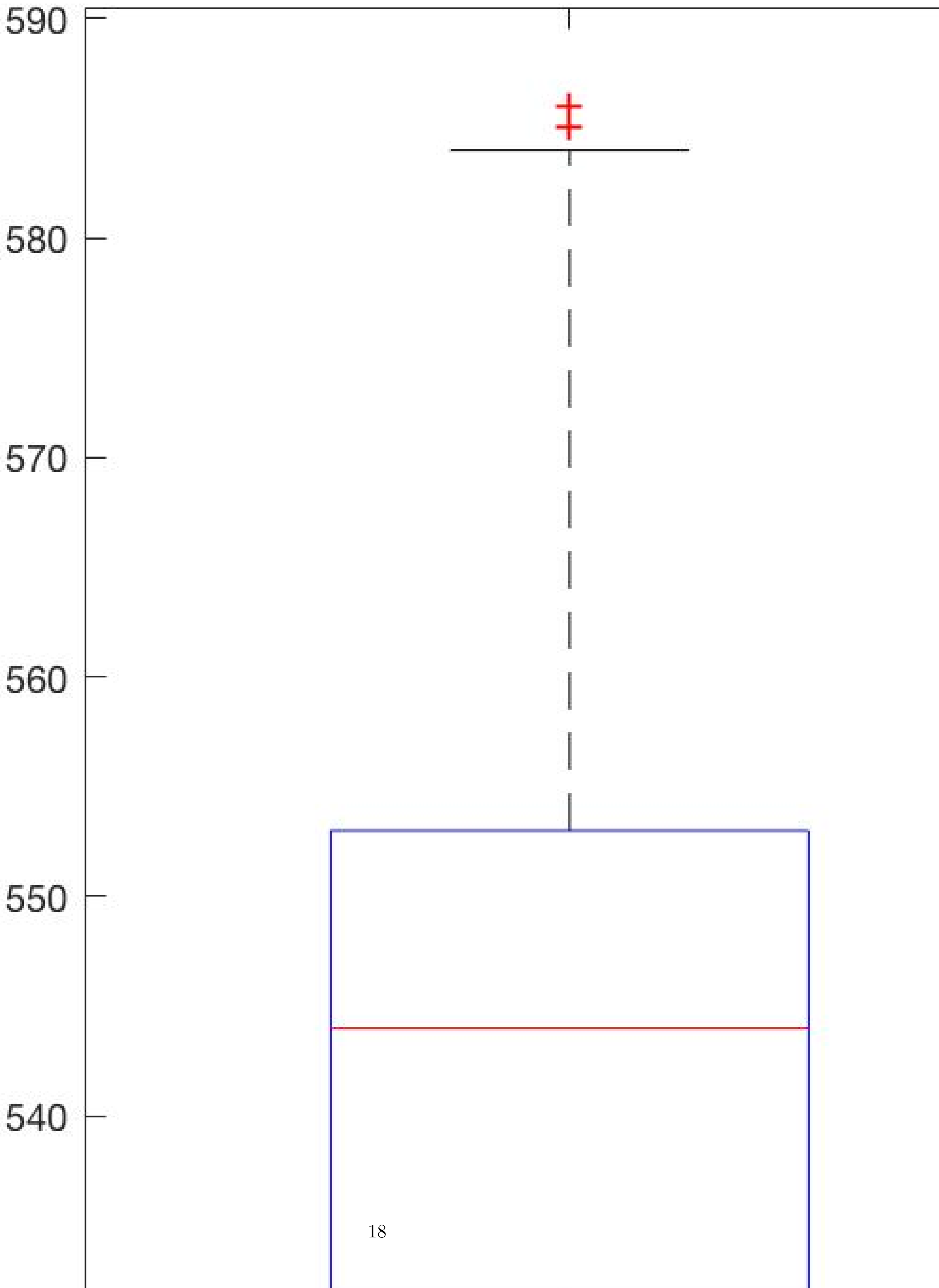
- The first line shows the values inserted when A rolls. There will be 5 $\frac{1}{6}$s in C
- The second line shows the values inserted when A banks, going to a higher game state probability in terms of Q
- The third line shows how an impossible game state is dealt with from A's perspective.
- The forth line shows the values inserted when B rolls near the end of the game. The main difference being that trivial game states exist, changing the value in D.
- The final line shows the values inserted when B banks, going to a higher game state probability in terms of P

For Pig, MATLAB completes all 1 million sets in 5 seconds. Although we had every probability on Strategy A's turn $P_{i,j,k}$ and every probability on Strategy B's turn $Q_{i,j,k}$, the main outputs that we were interested in, were $P_{0,0,0}$ and $Q_{0,0,0}$. These are the probabilities each strategy has of winning, when going first.

2.3.4 Testing the code

To test the output of the code, we compared the calculated probabilities to experimental results. It was sufficient to test only the value in the lowest position in the matrix, $P_{0,0,0}$, since this value relies on the higher game states (and therefore any errors in higher positions should be reflected in this output).

$P_{0,0,0}$ represents strategy A's probability of winning at the start of the game, assuming strategy A is the first to roll. To test this value, we simulated a series of games of Pig with strategy A always being the first to roll, and outputted the number of wins for strategy A. For example, the $P_{0,0,0}$ for the input strategies "bank on 20" and "bank on 21" was 0.542626, indicating that "bank on 20" has a 54.26% chance of beating "bank on 21" when going first. Running 1,000 series of 1,000 games gave us a distribution for the number of wins for "bank on 20" displayed in section 2.3.2 . The median on the box-plot is very close to the projected probability which indicated that our code had correctly calculated $P_{0,0,0}$.


590
580
570
560
550
540

We similarly tested some more simple strategies with successful results. Then we tried to create awkward strategies that would test the limitations of our program. Of special interest were strategies that involved banking at turn score 0, i.e. "skipping" a turn. In certain circumstances this decision could theoretically lead to a stalemate, where both strategies refuse to roll. While this would never happen in practical play, it was important to equip our MATLAB script to deal with these scenarios so that it wouldn't cause issues when we later come to computer-generated strategies.

A particularly interesting example is a strategy which skips a turn whenever the two players' banked scores are equal. (For convenience we excluded the starting game state where both players have score 0.) When this strategy plays against itself, around a quarter of games result in stalemates. But because the majority of games resulted in victories for one player, we still wanted to be able to analyse this strategy, and therefore had to update the code to accommodate stalemates.

The goal was that a stalemate would be counted as a loss for both players, since this was not a desirable outcome for our optimal strategy. Our modification therefore had the entire code run twice if a stalemate was detected. On the first run, every stalemate was declared a victory for player B (hence a loss for player A) to give an accurate recording of $P_{0,0,0}$. On the second run, we reversed the order of the strategies so that stalemates were counted as a victory for player A (hence a loss for player B), giving us an accurate value for $Q_{0,0,0}$.

Adjusting the script for simulating series of games to output the number of stalemates, we could then accurately test the $P_{0,0,0}$ and $Q_{0,0,0}$ values for strategies involving skips. Now satisfied that our code gave reliable outputs for any input strategies, we were ready to use this script to compare optimal strategies.

2.3.5 Coding for an optimal

The piece of code that we produced was one that could produce an optimal strategy against any given input strategy. Note that this is not the overall optimal for Pig, but an important step in finding it. With the foundations that we have explained so far, regarding equations, matrices and strategies, we can explain how we produced this code.

As a brief explanation of how it works, the code takes two important inputs: the main strategy that we wish to optimise against (let us call this strategy B) and another strategy which we will call strategy A. Strategy A will be iteratively modified until it becomes the optimal opponent for strategy B. The reason we wanted to be able to start from any point was to see if there were multiple optimals against B, i.e. local maxima. Strategies A and B can be inputted in any format since we make use of the "strategies to matrices" function to get them in our preferred matrix form. The code then runs through the simultaneous sets again, modifying A to be the best it possibly can against B.

As in the previous coding task, we built matrices, but this time a little differently. In the probability matrices we were looking up what the strategy does and building the matrices around what the choice was. However, this time we wanted to consider both rolling and banking at each game state. We could then see which option proudced the highest probability of winning. If the strategy's choice is not already the one with the highest chance of it winning, the script changes it and records its new probability of winning.

The method behind this is a little more complicated as all the probabilities are interlinked. By changing one probability, all other probabilities are affected, potentially making rolling or banking at another game state a better choice. This meant that we may not have had to change strategy A multiple times before it stabablises at each set.

The probability matrix at each set now looks like this.

$$\begin{bmatrix} P^{Roll}_{i,j,0} \\ P^{Roll}_{i,j,1} \\ \vdots \\ P^{Roll}_{i,j,n-1} \\ P^{Roll}_{i,j,n} \\ - \\ P^{Bank}_{i,j,0} \\ P^{Bank}_{i,j,1} \\ \vdots \\ P^{Bank}_{i,j,n-1} \\ P^{Bank}_{i,j,n} \\ - \\ Q_{j,i,0} \\ Q_{j,i,1} \\ \vdots \\ Q_{j,i,m-1} \\ Q_{j,i,m} \end{bmatrix} \tag{4}$$

where $n =$ "to win score" $- i - 1$ and $m =$ "to win score" $- j - 1$.

When strategy A looks at a $P^{Roll}_{i,j,k}$, it relies on future $P_{i,j,k+r}$s. The question now is how do we choose which future $P_{i,j,k+r}$ to rely on, as we have a rolling version and a banking version? This is where we use the current version of A to choose. We then build the matrices accordingly, solving them again to get our new probabilities. At this point we compare rolling and banking, and see if A needs to change. If A does change, then we must run through the same process again, recalculating probabilities and changing A. This will converge for each set. This process will modify A as it works down, recalculating probabilities when it makes changes to the strategy. Once it has finished every set, A will have been completely modified to choose the best choice at every game state against B.

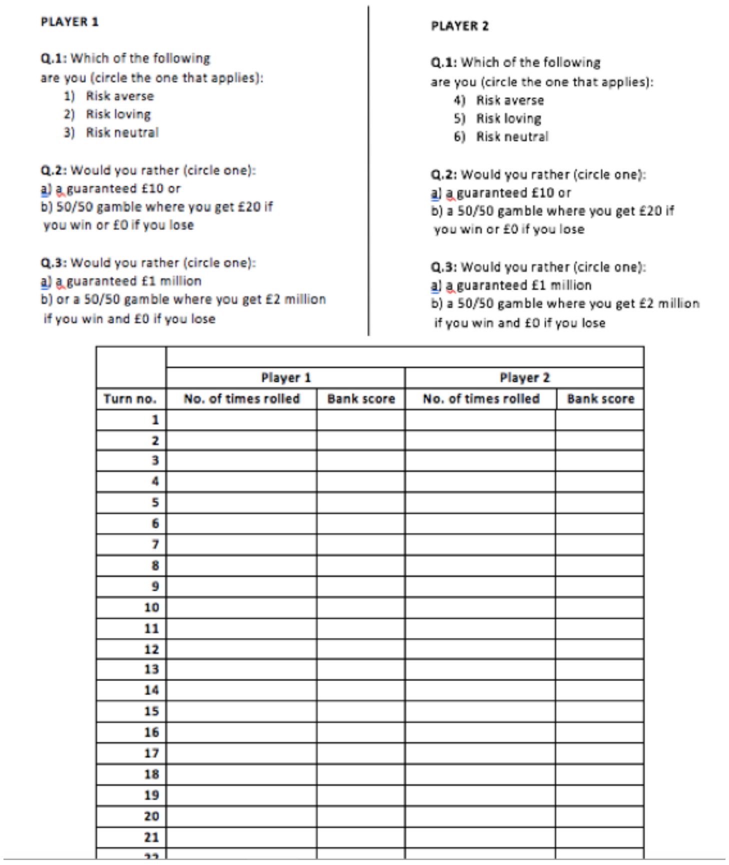
This script took a lot more computational power, and generally took MATLAB a few minutes to run. The results were promising, however, as it always produced a new strategy that beat its opponent by a large margin.

2.4 Human interactions

To investigate if players aligned game play with their stated risk preference and see if players were influenced by how well their opponent is doing, we decided to get a sample of people from the ECM3735 (Mathematical Group Project) module to play Pig. We gathered 15 samples, which was 15 games played with 30 peoples risk preference and game strategies gathered.

2.4.1 Method

1. Contacted Barrie to ask if we would be able to use time in our drop-in sessions to get students to participate.
2. Created a tracker form, Figure 6.
3. Purchased dice and printed off tracker form.
4. Explained to the participants how to play Pig to 100.
5. Explained to the participants to circle each of the three questions, without letting their opponent see (limits any chance of dependent preference choosing).
6. After each players go they had to write down the amount scored (bank score) and the amount of times they had rolled (No. of times rolled) allocated to each turn number (turn no.).
7. Each player did this until one player won.



PLAYER 1

Q.1: Which of the following are you (circle the one that applies):

1) Risk averse
2) Risk loving
3) Risk neutral

Q.2: Would you rather (circle one):
a) a guaranteed £10 or
b) 50/50 gamble where you get £20 if you win or £0 if you lose

Q.3: Would you rather (circle one):
a) a guaranteed £1 million
b) or a 50/50 gamble where you get £2 million if you win and £0 if you lose

PLAYER 2

Q.1: Which of the following are you (circle the one that applies):

4) Risk averse
5) Risk loving
6) Risk neutral

Q.2: Would you rather (circle one):
a) a guaranteed £10 or
b) a 50/50 gamble where you get £20 if you win or £0 if you lose

Q.3: Would you rather (circle one):
a) a guaranteed £1 million
b) a 50/50 gamble where you get £2 million if you win and £0 if you lose

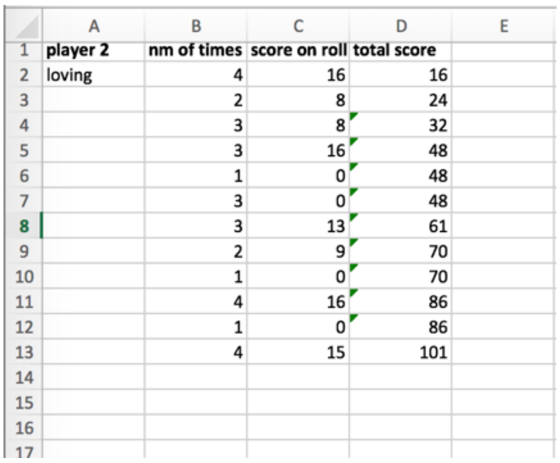
	Player 1		**Player 2**	
Turn no.	**No. of times rolled**	**Bank score**	**No. of times rolled**	**Bank score**
1				
2				
3				
4				
5				
6				
7				
8				
9				
10				
11				
12				
13				
14				
15				
16				
17				
18				
19				
20				
21				

Figure 6: Pig tracker form

8. Collected in the forms.
9. Inputted all the data into excel documents, categorising each player into one of the three risk preferences. Example shown in Figure 7.
10. We calculated the accumulated scores for each player, and began our analysis shown in findings (put section reference).

The reasoning behind the questions on the tracker form:
Question 1
This question was used to find out participants risk preference (risk neutral, risk averse, risk seeking) so we could see if individuals actually stuck to their stated risk preference whilst playing.
Questions 2 and 3
We created these questions to test whether this would correlate with the stated preference in question 1. The risk neutral person (in theory) should have no preference to option a or b as they both have the same expected pay off. A risk averse person (in theory) should pick option a in both cases and a risk seeking person will (in theory) pick option b in both cases. Both question 2 and 3 represent gambles, however the expected pay off in question 3 is much greater than that of question 2. Therefore question 3 allows us to see how risk seeking a participant may be, as a greater payoff will be a lot more attractive to a risk seeker.

These questions created were motivated by a branch of economics called Prospect Theory. As in Kahneman and Tverskys 1979 paper, prospect theory is the Analysis of decision under risk[3]. The theory explains that humans dislike losing far more than we like winning and that depending on whether a scenario is framed as win or a loss, we act differently. Humans dislike losing so much, that they are willing to take huge risks to avoid a loss scenario. As an example, in the case of Pig, if player A is close to winning and player B is far behind, then player B will usually try to keep rolling on their turn, to build up a big score and then bank it, when actually their optimal strategy might be to bank earlier on. (this is later discussed in REFERENCE)



	A	B	C	D	E
1	player 2	nm of times	score on roll	total score	
2	loving	4	16	16	
3		2	8	24	
4		3	8	32	
5		3	16	48	
6		1	0	48	
7		3	0	48	
8		3	13	61	
9		2	9	70	
10		1	0	70	
11		4	16	86	
12		1	0	86	
13		4	15	101	
14					
15					
16					
17					

Figure 7: Information inputted into Excel

2.5 Statistical testing

2.5.1 Fair test

Throughout our project, we tested numerous different strategies against one another to determine whether a certain strategy was better than another. To do this accurately, we had to ensure that the test we conducted was fair and would give a reliable result. This was particularly important for our final findings when we needed to test our found strategy. We wanted to know whether it was indeed an optimal strategy, or whether there were still some strategies that were able to beat it.

To determine whether a strategy is better than another is not as simple as playing them against each other once and deciding that the winner is the better strategy. A strategy will have a theoretical probability of winning against another given strategy, but playing only one game may not give a reliable result. Say you were to play a strategy against itself. The theoretical probability of either player winning would be 0.5 if you were to alternate which goes first and to play a large number of games. By playing only one game you would deduce that one of the strategies is better than the other even though they are the same strategy. You could also play two strategies where one has a significantly larger theoretical probability of winning than the other, but the test could conclude that the worse strategy wins, meaning you would accept that the lesser strategy is better even though theoretically it is not. This false result would be due to statistical inference.

Theoretical probabilities don't always agree with the practical results, though there are ways where you can improve the accuracy of a practical test to make it more accurate - that is, closer to its theoretical probability. This is the justification for identifying what quantifies a fair test: to try to find a result that is accurate within a certain confidence limit of the theoretical results. In our case, we can increase the number of games we play so that inference is reduced, thereby giving a reliable score when testing strategies. Instead of playing only one game, we can play numerous games and then take scores. If we played n games, each individual game of Pig would result in a winner which awards that strategy a score of 1. This continues until n games are completed. The probability for each strategy is given by its score divided by n.

The true value of n needed to be calculated and not chosen arbitrarily. To do so we used hypothesis testing. From theoretical probability values, we could define an interval around the value in which we want the simulated result to fall. The practical test involved playing n games of Pig, where a strategy received a score of 1 for each game they win. After n games were completed, we could calculate the probability of the strategy winning against the other by dividing the number of games it has won by the total number of games. In this way, we could test these results for many different values of n and observe the distribution of p as n increases. From there, we chose an appropriate value for the number of games which gave us constant results within our confidence interval. Once this value was obtained, the testing of strategies could result in reliable results and therefore be as fair of a test as possible.

2.5.2 Deciding that a strategy is better than another

Once we had decided on a value for n at which we decide that the test is fair, we could easily do tests to conclude that one strategy is better than another. By finding n, we had chosen an upper and lower limit of the p-value at which we wanted our results to lie within. To illustrate this, suppose we concluded that 100 games of Pig is a fair test, and we wanted our practical probability to lie within 0.1 of the true (theoretical) probability. If we were to test the same strategy against itself, theoretically either strategy should win with a probability of 0.5. Therefore after simulating many games, we would want the representative probability to lie within 0.4 and 0.6.

As the probability was calculated by dividing a strategy's number of wins by the total number of games, we could then either perform this calculation to find the representative probability or we could calculate directly from the confidence interval how many games a certain strategy wwould need to win in order to conclude that it is a better strategy. In the above case, as it needs to have a probability higher than 0.6 to conclude that it is a better strategy, it would therefore need to win a minimum of $100 \times 0.6 = 60$ games in order to be considered a better strategy.

2.5.3 Testing our optimal

To finalise our findings we needed to test our version of the optimal to determine whether it actually was an optimal, and at least as good as Nellers[5]. Therefore we ran some tests on it and evaluated the findings.

We began our testing by playing our version of the optimal against simply "hold at" strategies, which had actually been found to be quite effective (even against Nellers strategy). By doing this we could decide whether it is actually better than simple "hold at" strategies or whether it needed adjusting before playing it against Nellers version.

Throughout these tests we made sure to keep to our definition of whether a strategy is better than another. This was very important in analysing the results of our optimal strategy.

3 Findings

Need general brief on overall findings here

3.1 Pig

From the two functions we built in MATLAB, we were able to run many tests and find lots of results. The main objective from these finding was to produce an absolute optimal for Pig.

3.1.1 Did we solve Pig

From any strategy A, we could produce an optimal against it. We will call this a first order optimal against A. From this optimal, we could run the same process again, thus producing an optimal against our first order optimal. We will call this the second order optimal against A. By repeating this proccess we can get higher and higher ordered optimals against any strategy. We found that this converges to a single strategy, from most starting strategies. Could this be our final optimal?

We used our probability calculator to test it against other strategies. It beats all strategies that we had produced. We then tested it against Neller's optimal, to which both it and Neller's optimal has the same probability of winning when each going first. From previous testing, we had learnt that if these probabilities were the same, it was highly likely that the strategies were the same. We could test to see if the strategies were the same by simply comparing them in MATLAB with the command **MATLAB code here**. This confirmed this idea, as every element in our strategy was the same as Neller's.

When optimising against Neller's strategy, the strategy produced is an exact copy of Neller's. This is the same as how repeated implications of the optimal finder eventually converges and doesn't change, meaning that it

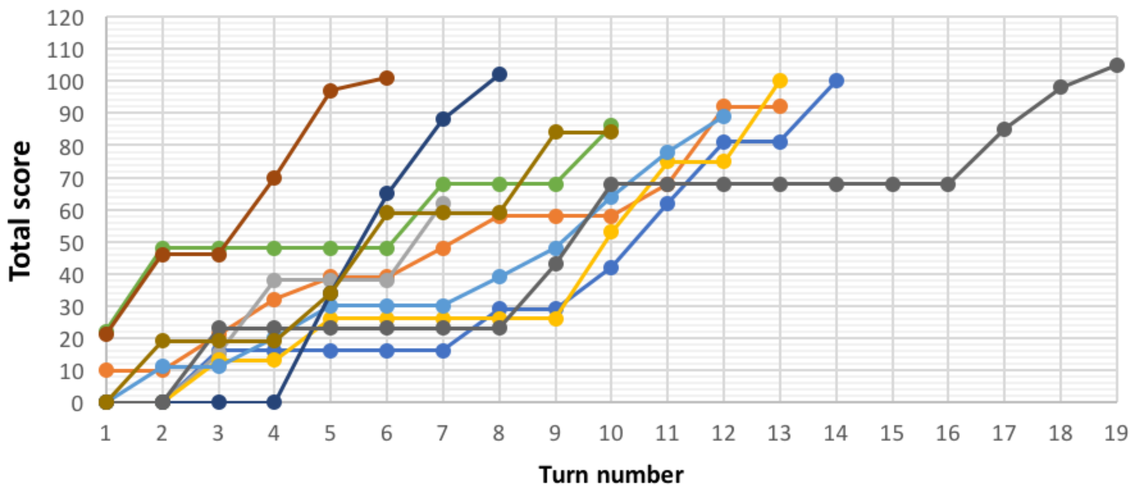


Figure 8: Graph to show total score at each turn for all risk neutral players

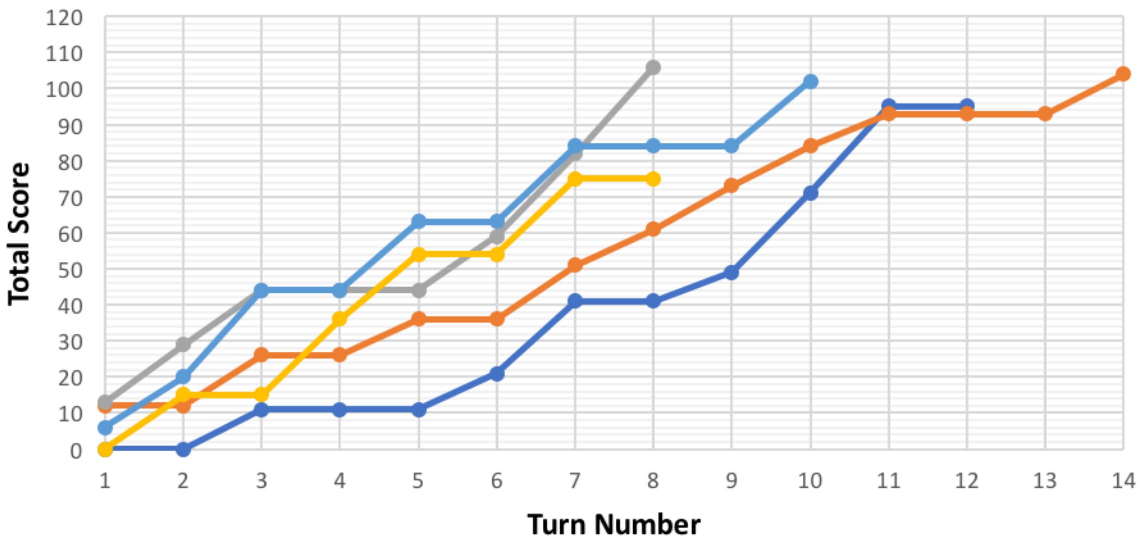


Figure 9: Graph to show total score at each turn for all risk averse players

reached Neller's optimal.

To check that it cannot be beaten by any strategies by brute force, we would have to test it against every strategy. This is not even possible by MATLAB as there are roughly $2^{100\times100\times100}$ possible strategies which would not be possible to test all against Neller's. However we could take the fact that the optimal against Neller's is itself, therefore meaning no strategy could do any better than its optimal, and so Neller's will beat every other strategy. We have found the optimal!

3.2 Human Interactions of playing Pig

3.2.1 Do players stick to their stated risk preference

To determine whether players stuck to their stated risk preferences, we divided all of the samples into the three risk preference categories (risk averse, risk seeking, risk neutral). After doing this, we plotted the total score of each player at each turn For all three of the graphs (Figure 8, Figure 9, Figure 10) the plotted results are all fairly similar in shape and gradient, which we didnt expect initially. So, to investigate further we decided to find the average total score at each turn number for each of the three risk preferences (Figure 11, Figure 12,

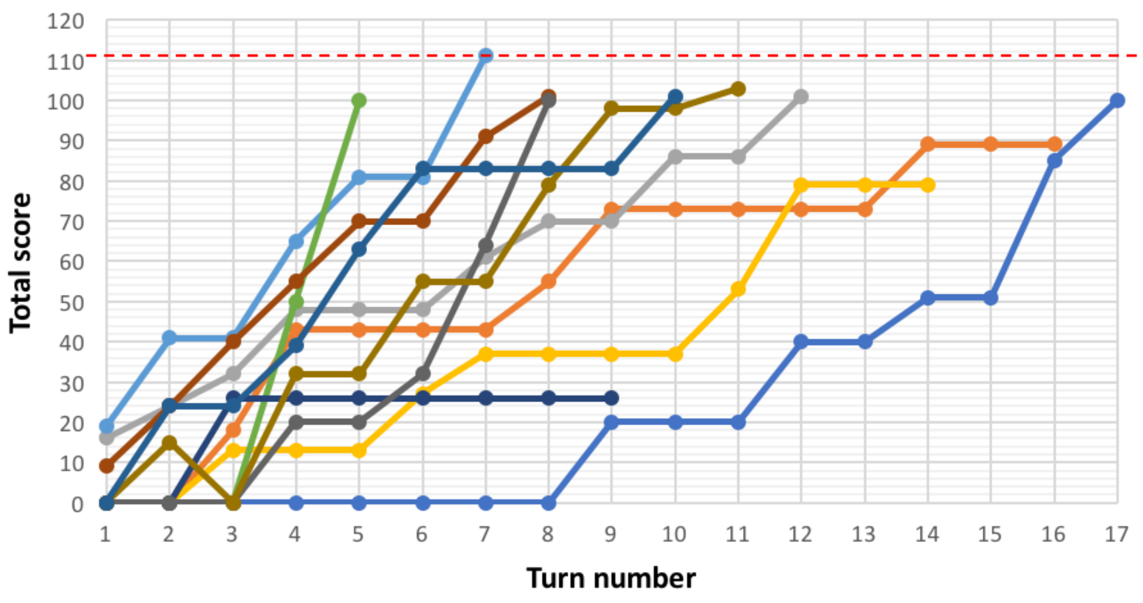


Figure 10: Graph to show total score at each turn for all risk seeking players

Figure 13).

Looking at Figure 11, 12, 13, we can see that they are actually all very similar. On further inspection of the gradients of the trend lines, we can see that the risk neutral average graph has a gradient of 4.75, the risk seeking average graph has a gradient of 4.85 and the risk averse average graph has a gradient of 4.76.

The gradients are therefore all very similar. The results have shown that the gradient is greater for risk seekers than risk averse participants, however, we expected there to be a greater change in the gradients. This is because risk seekers would roll more each turn to increase the chances of a getting a high bank score. Whereas a risk averse participant would bank on a smaller score at each turn as this is a safer option.

A risk seeker would potentially believe the outcome of rolling and scoring a greater bank score would be of greater interest then banking at a safe score, therefore they would be at a greater risk of rolling a 1 and losing all the points from that round. This could therefore explain why the gradient is only a smaller percentage greater, as looking through the game strategies performed by risk seekers a lot of them banked on a score 0, due to rolling a greater amount of times thus a greater chance of scoring 1. Therefore their overall accumulated score was still achieved by the same amount of turn numbers as risk adverse participants.

In addition, we expected the risk neutral gradient to be equally between the risk seekers gradient and risk averse gradient. Conversely, its the lowest gradient of the three. This could be due to experimental error as the difference between the greatest and lowest gradient is only 2%.

Moreover, looking at all three graphs the points plotted at turn number 15-16 are all below the average gradient by a significant amount. This could be because at this point in the game the players opponent is close to winning and therefore the player begins to panic and roll too each turn number, causing them to roll a 1 and lose. This links back to the ideas exposed in Prospect theory[3] as humans compensate to win by taking extra risks which they usually wouldnt. In this case the player is rolling excessively in order to rebuild up a greater score to bank on. This therefore shows, that for our results gathered, that at around the turn number 14, on average, players strategies change dependent on the opponents play. ¿¿¿¿¿¿¿ c95b138e788e0fc585b41586451b32f6d1ea47f9

Overall, a possible explanation for the similarity of the gradients is that in the long run of playing Pig, players preferences are evened out by the game, much like the stock market evens out returns in the long run.

3.2.2 Players Interactions

For our second experiment we looked into how players A's strategy was influenced by player B's strategy and vice versa. To do this we took all 15 sample games that were played and plotted the total scores of player A

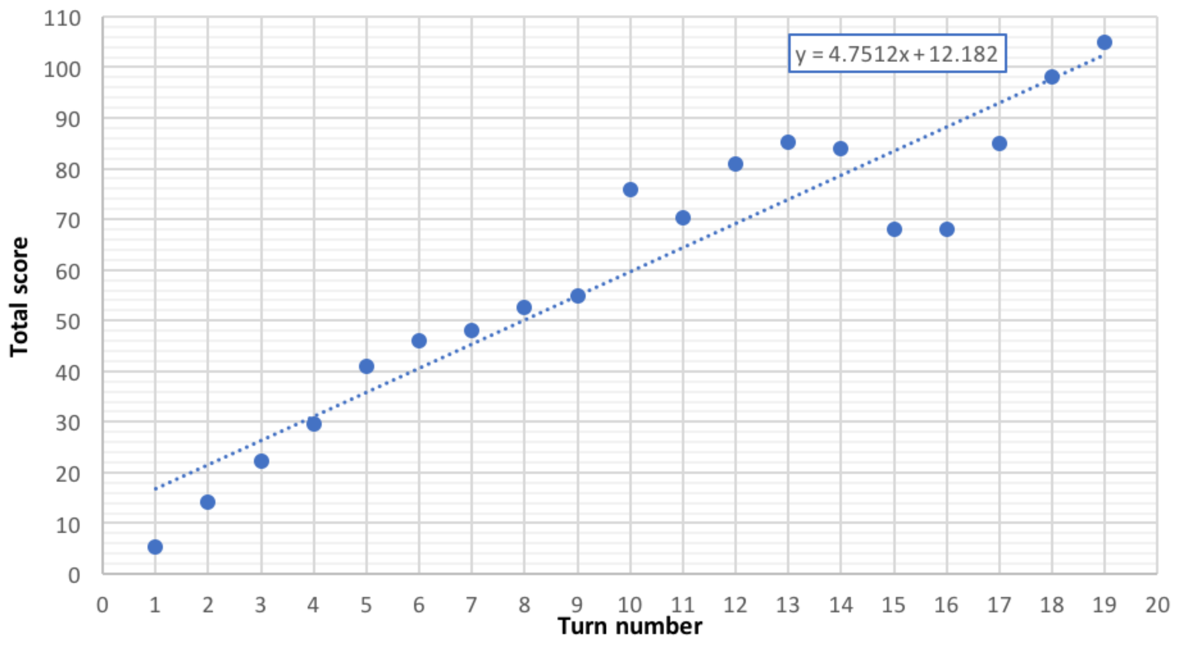


Figure 11: A graph to show the average total score at each turn for all risk neutral players

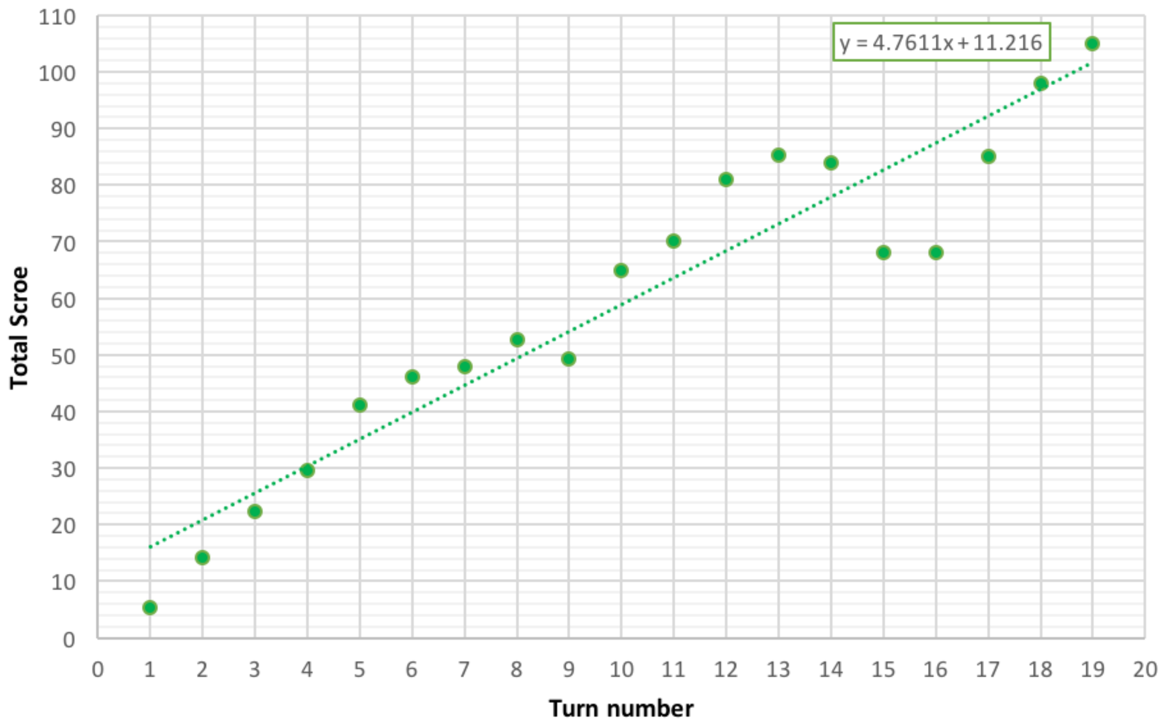


Figure 12: A graph to show the average total score at each turn for all risk averse players

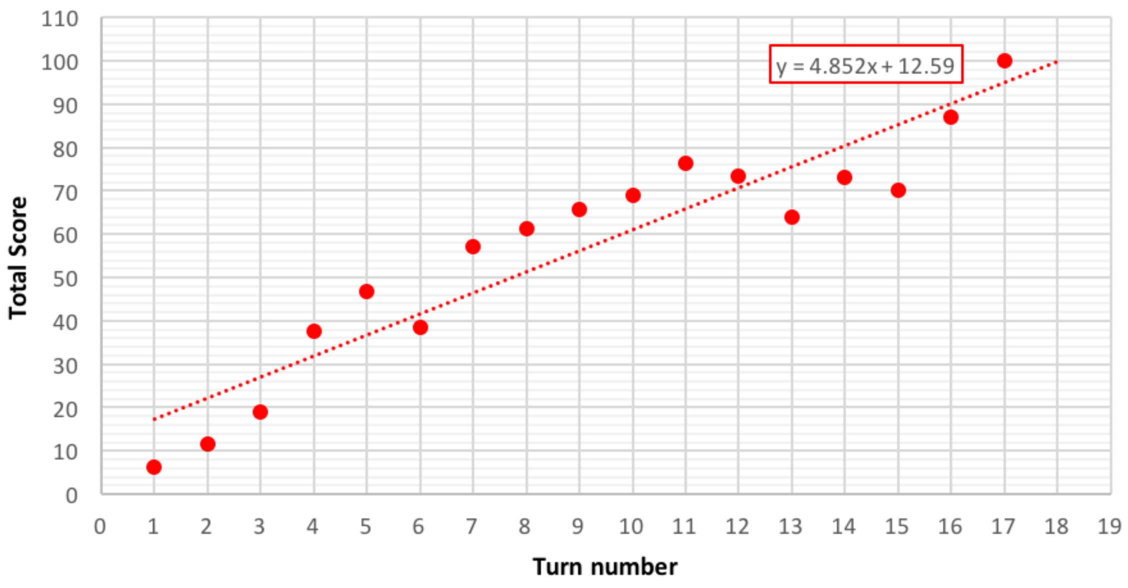


Figure 13: A graph to show the average total score at each turn for all risk seeking players

and player B at each turn.

Firstly, an interesting game that we wanted to look at was Game 14 (Figure 14).

Looking at Game 14, we found that from turn 4 onwards, as player 1s score increases and gets closer to the target score of 100, player 2s total score stays the same until turn 8. A possible explanation for this is that as player 1 continues to increase their total score, player 2 wants to try and build up a big bank score in order to catch up with player 1. As a result, they've roll for too long, eventually rolling 1, thus scoring a turn total of 0.

This shows us that player 1's strategy is influencing the strategy used by Player 2. At first, both players represent similar game techniques, but after an increased score from Player 1, Player 2 demonstrates traits of a risk seeker (although they stated to be risk neutral) by rolling a greater amount of times to attempt to gain a better outcome. However, in this case they have been unsuccessful. This highlights two concepts. Firstly, player 2 has not stuck to their stated risk preference during the game, and secondly, the strategy implemented by player 1 has resulted in a change of behaviour in player 2. As this game lead to a lose for player 2 it may show that a strategy shouldnt be changed during game play, and that individuals should play a strategy that is independent of their opponents.

Whilst looking through all of our results we found a common theme that is shown well through Game 11 (Figure 15) in particular. We can see that players start to mirror one anothers strategies.

The stated risk preferences of each player are different and therefore we would expect slightly different graphs from each player. However, they both follow the same pattern from the beginning of the game. The shape of the graphs follow that of one expected by a risk neutral participant and thus may show that player 1 (stated risk loving) has actually adopted the strategy of player 2.

3.3 Statistical Testing

3.3.1 The test

As mentioned in the methodology, to calculate the number of games required to play to perform a fair test we will use hypothesis testing.

Let S denote a certain strategy within the game of Pig.
Let n denote the number of games played.
Let X denote the number of games won by that strategy after n games.

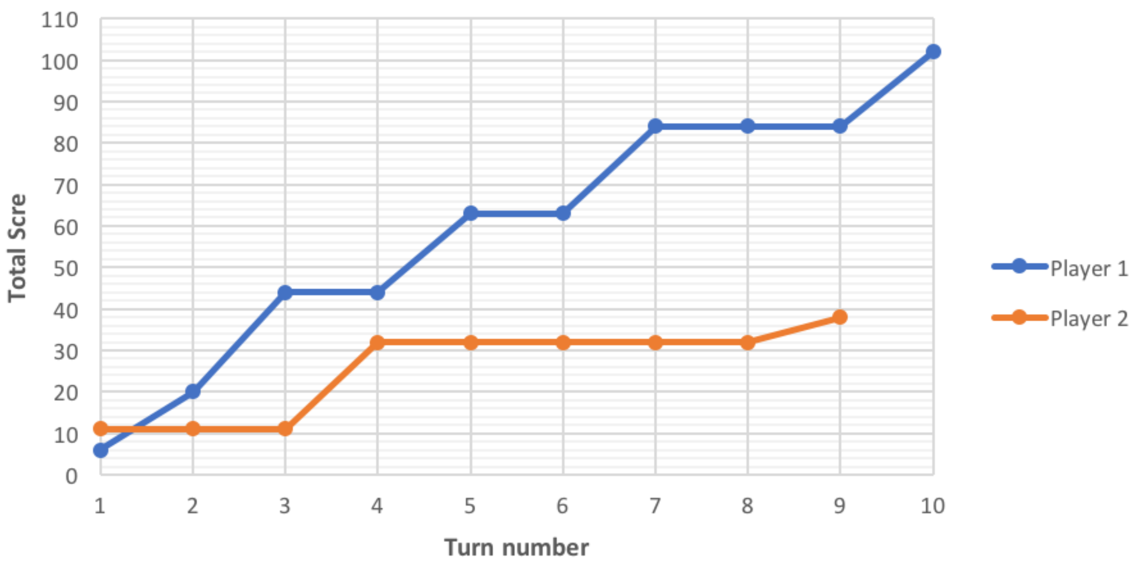


Figure 14: A graph to show the total score of each player at each game state for Game 14 (Player 1: Risk Averse, Player 2: Risk Neutral)

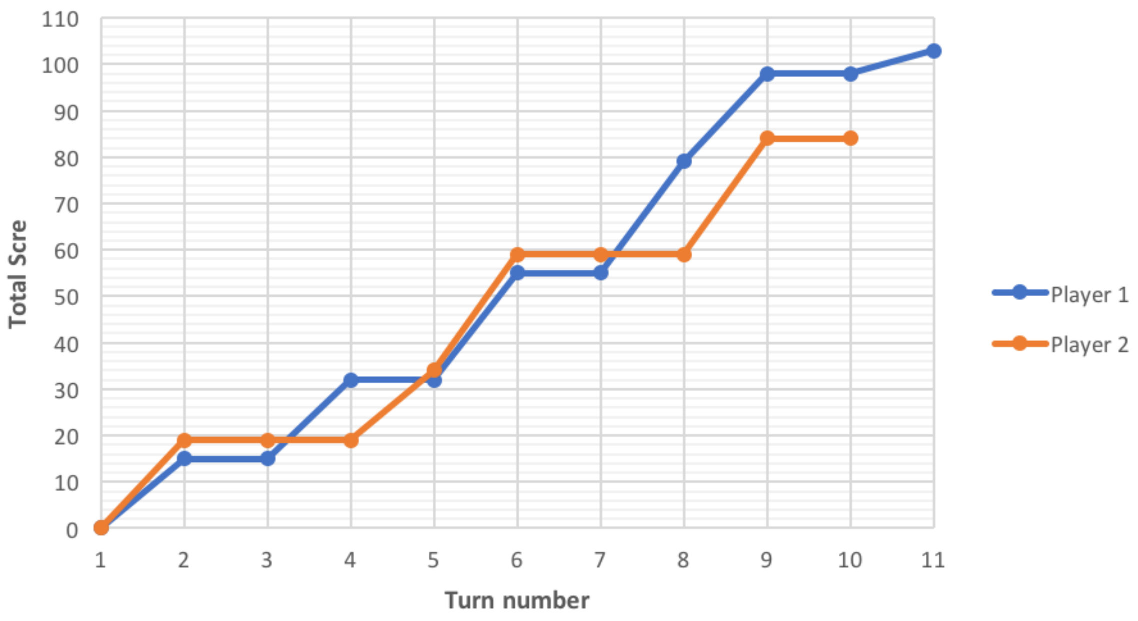


Figure 15: A graph to show the total score of each player at each game state for Game 11 (Player 1: Risk loving, Player 2: Risk Neutral)

As we will be playing S against itself, we can say $E(X) = \frac{n}{2}$, the expected amount of games S should win against itself.
We can approximate X to the binomial distribution to conduct the hypothesis test. $X \sim Bin(n, p = 0.5)$ by the binomial distribution

$$E(X) = np \qquad SD(X) = \sqrt{var(X)}$$
$$= \frac{n}{2} \qquad = \sqrt{np(1-p)}$$
$$= 0.5\sqrt{n}$$

By using the Z distribution at a 95% confidence interval ($z = 1.96$), we compute the following interval.

$$\begin{aligned} 95\%Cl &= (\frac{n}{2} - 1.96(0.5\sqrt{n}), \frac{n}{2} + 1.96(0.5\sqrt{n})) \\ &= (\frac{n}{2} - 0.98\sqrt{n}, \frac{n}{2} + 0.98\sqrt{n}) \end{aligned}$$

We would like the p value that we obtain to be within 0.025 of the true value of the probability. As probabilities range from 0 to 1, this would mean that the practical result is within 95% of the theoretical value. For this system this implies that $0.475 < p < 0.525$. From the above interval for X, we can say that as $X = np$, using the upper and lower bounds for our p, we can then calculate a value for n, i.e. $0.475n$ would be equal to the lower bound for X of the confidence interval with $0.525n$ being the upper bound.
i.e.

$$0.475n = \frac{n}{2} - 0.98\sqrt{n}$$

Solving this for n, we get that $n = 1537$, and similarly for the upper bound.

This would imply that playing a game of Pig 1537 times would end up giving a probability of the strategies winning against each other that is within 95% of the true value. As theoretical stats is not 100% accurate, this theory was simulated in MATLAB. We tested this theory by running n games of Pig 100 times for a given strategy against itself, for n ranging from 1 to 4000, and calculating the p-value from each of the games.

Figure 16 shows the distribution of the p-values over the number of games played. Each point on the x-axis has 100 values for the y-axis. The two lines on the y axis at $y = 0.475$ and $y = 0.525$ show the confidence interval we want for p. The orange colour that runs through the graph shows a higher concentration of values. This graph shows that the distribution for p is scattered greatly for small number of games played and converges as the number of games increases. We can see that around the 1500 number of games played, the orange centre is located within the interval for p, but there are still a lot of stray values. These stray values still occur, but less often, even when the number of games is increased. By looking at the graph we can chose a value at which the majority of points lie within the interval.

After the number of games passes 2500, there doesnt appear to be much decrease in the width of the distribution, so we will chose 3000 as the appropriate number of games to play for a fair test. It would be a mistake to now only use $n = 3000$ to test strategies. As the Figure 16 shows, even for high values for n, you can still get stray results. Therefore, like what was done in the test to find n, for strategies we want to compare we will take 100 results of 3000 games. Then we can take the average of these results to further improve the reliability of the result.

3.3.2 Corollary

In MATLAB, we tested the fair test numerous times by creating a code which ran the test 100 times, that is, 100 results 3000 games played 100 times with the average computed. Figure 17 illustrates 1 of the results of running this code. You can see that for 100 results of the test, the distribution of p is still very small. For this particular result, p ranged from 0.4974 and 0.5020 which is well within our confidence interval for p. The mean of the results was (coincidently) 0.5 exactly.

The purpose of the fair test is that our practical results are accurate to within a certain degree of confidence to the theoretical results, in our case, within 0.025. As we have coded a programme which can calculate the theoretical probabilities, we can run multiple simulations with the fair test to see how accurate our results are to the theoretical values.

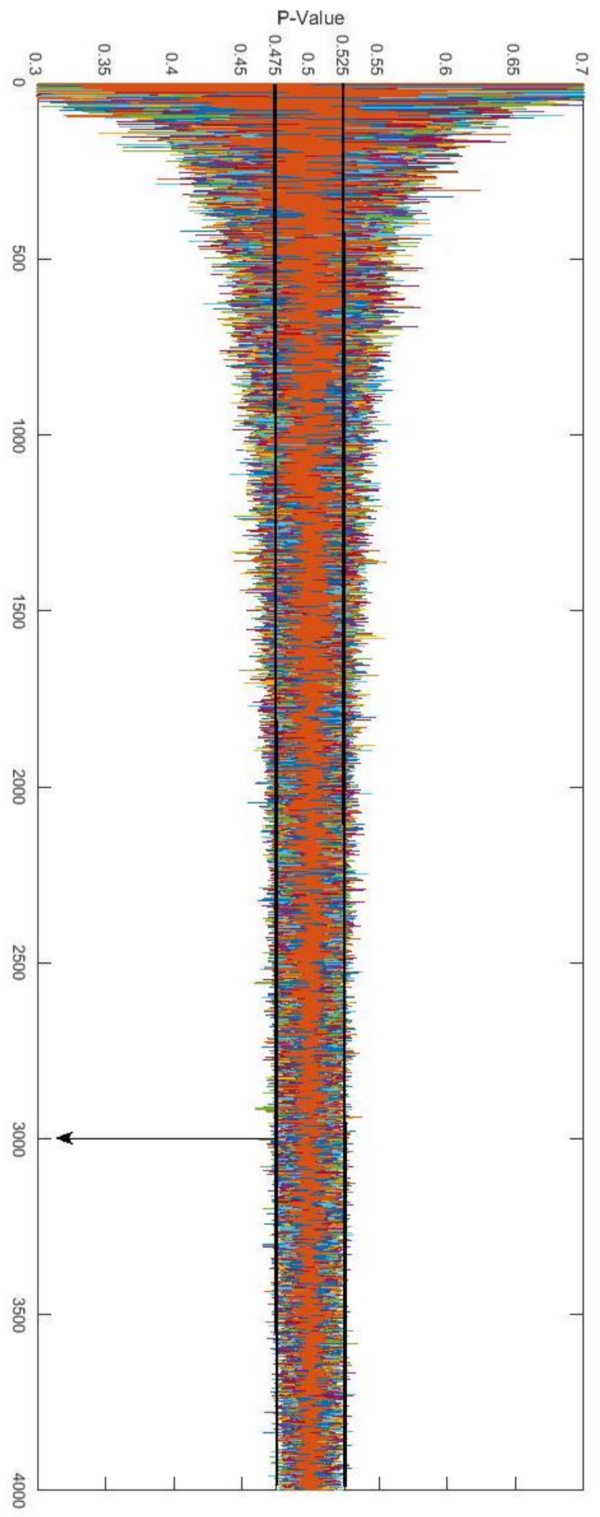


Figure 16: The distribution of p-values against the number of games played between a strategy and itself

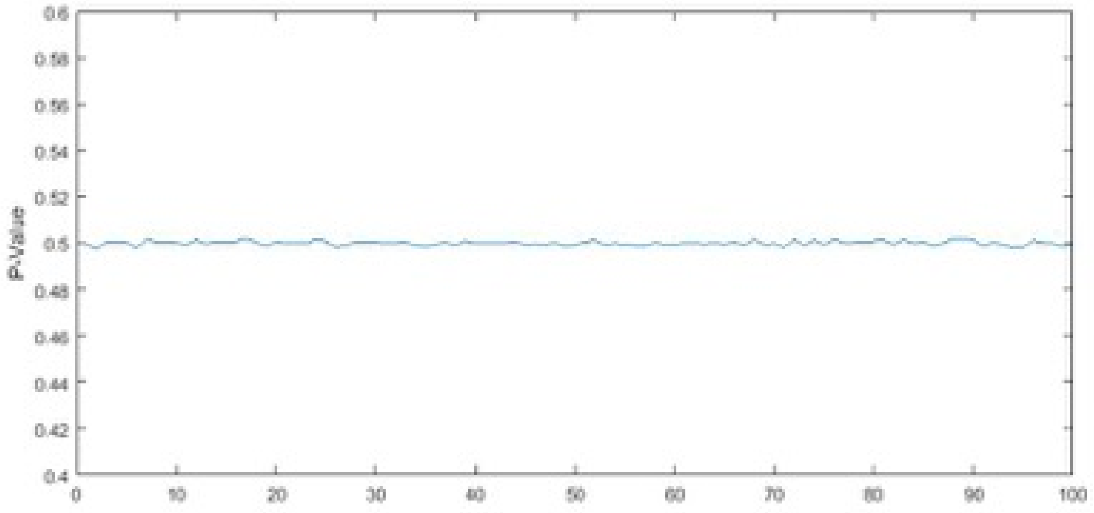


Figure 17: p-values calculated when running a strategy against itself in 100 fair tests

3.3.3 Testing theoretical probability values to practical values with the fair test

We have showed that for a strategy playing against itself, playing 3000 games 100 times gives a reliable result for the probability of winning, but we now should ensure that this test also works for different strategies playing against itself. From using the theoretical probabilities calculated in MATLAB, we will run numerous tests for different strategies playing against themselves and comparing the theoretical results to the practical results obtained from the test. We will consider simple hold at strategies with this. Due to time constraints we will not compare every hold at strategy, though we will test a wide variety of hold at strategies. Figure 18 shows the result from 10 different games of 2 different strategies under the fair test. The last column shows the deviation from the theoretical values of the probability and the practical values obtained from the test. By our conditions, if the deviation is greater than 0.025 then this would imply that the practical results arent consistent with the theoretical results implying that the test wasnt reliable. As you can see the results are all less, and comfortably so, than 0.025, therefore we can say that the test conducted gives reliable results and is therefore fair.

3.3.4 Determining whether a strategy is better than another

There are two ways which we could analyse the results. As we are playing 3000 games of Pig and giving each strategy a score of 1 for each game they win, we could analyse the score for both strategies and use that to determine whether it is better or worse than the other strategy. Though we are playing 100 independent 3000 games and then taking the average score of both strategies, the following method will still apply.

From looking at our maximal value for the probability in the confidence interval, as $X = np$ where $n = 3000$ and $p = 0.525$, we can say that if one strategy wins on average $X = 3000 \times 0.525 = 1575$ then it is better than the other strategy.

We could also look at the respective probability of the strategies from the theoretical results. As $p = \dfrac{X}{n}$, if the value for p is outside our determined confidence interval, we can then deduce that one of the strategies is better than the other.

3.3.5 Statistical findings on the optimal

It happens to be that simply hold at strategies, i.e. If your turn score is less than 20 (say, then you always roll, are quite effective. This made us think that testing our version of the optimal strategy against simply hold at strategies would be a good place to start. We could find the optimal and then play it against numerous hold at strategies and see the results, though we thought it would be good to find whether there was a best hold at

	Theoretical			Practical (under the fair test)			
Hold at strategies	Probability of the former winning when it goes first	Probability of the latter winning when it goes first	Probability of former winning in general	Probability of the former winning when it goes first	Probability of the latter winning when it goes first	Probability of former winning in general	Deviation of results (general)
20v21	0.5426	0.5258	0.5084	0.5429	0.5267	0.5081	0.0003
15v20	0.4784	0.5933	0.4426	0.4773	0.5939	0.4417	0.0009
1v80	0.5049	0.5078	0.4986	0.5057	0.5093	0.4982	0.0003
1v5	0.0236	0.9860	0.0188	0.0233	0.9862	0.0186	0.0002
80v85	0.5568	0.4541	0.5514	0.5557	0.4518	0.5520	0.0006
45v55	0.5544	0.4740	0.5402	0.5555	0.4725	0.5415	0.0013
1v100	0.7393	0.2682	0.7356	0.7400	0.2683	0.7359	0.0003
36v52	0.6354	0.3969	0.6193	0.6329	0.3949	0.6190	0.0002
5v21	0.0788	0.9401	0.0694	0.0785	0.9393	0.0696	0.0003
13v31	0.5006	0.5559	0.4724	0.5024	0.5563	0.4731	0.0007

Figure 18: Examples of probabilities between various hold at strategies

strategy so that we only need to use that to analyse our version of the optimal.

As we saw earlier, Figure 18 shows some of the probabilities we obtained from running these tests along with their theoretical probabilities. We decided to run a test, in compliance with our rules for a fair test, in MATLAB of every hold at strategy from hold at 1 to hold at 100 in order to determine which particular hold at strategy was better than all of the others. As we saw earlier, the strategy hold at 0 cannot win any games it plays and so, to reduce computing time, we will not include this in our simulations. We wanted to know whether there was one hold at strategy which beat all other strategies, as opposed to finding which one won the most against the others. By this we mean whether the strategies are transitive. By saying that the strategies are transitive, we mean that if we have strategies A, B and C, where A is better than B and B is better than C, then A is better than C also. This would mean that if we find one strategy that is the best, it will beat all other strategies. We ran a MATLAB simulation, again alongside our rules for a fair test, which returned a result that confirmed that these simple hold at strategies are in fact transitive.

After we had looked at the transitivity of strategies, we looked at playing a tournament between all of the hold at strategies from 1 to 100 in order to find which such strategy was the most superior and beat all of the rest. We again ran this in compliance with our rules for a fair test and the outcome of this tournament was that hold at 25 came out to be victorious from 10 tournaments.

3.3.6 Interesting note

After we evaluated all of the simple hold at strategies, we decided to look at more complicated strategies that have other rules other than just at what turn score to bank at. When evaluating strategies of this description, we were able to find three strategies that are non-transitive. The strategies we have found that have such properties are named turtle, roll to win and roll and wait. Turtle is a slow strategy, as suggested by its given name, banking on a low score each turn and thus slowly making its way towards 100. Roll to win is a very simple strategy, where it only banks once the turn score reaches 100. Roll and wait is a strategy that behaves such that it gets up to a banked score of 80, before not rolling and waiting for the opponent to catch up, before then rolling to try to win. We ran fair tests between these strategies and have found both practical and theoretical probabilities of each strategy winning against another. When turtle played against roll to win, we got a practical probability that turtle will win with probability of 0.8127 when going first, compared to 0.1949 for that of roll to win. Comparatively, we get theoretical probabilities of 0.812840 for turtle to win going first and 0.195448 for roll to win. Using these number to calculate who wins in general from multiple games is fairly simple and will be explained further on, but the calculate to be 0.8089 for turtle winning and 0.1911 for roll to win to win. This is again showing that our tests are accurate, with very little deviation between the practical and theoretical values. This shows that turtle is a better strategy as both the practical and theoretical probabilities of it beating roll to win lie above 0.525, the upper bound of our previously stated confidence interval. Similarly,

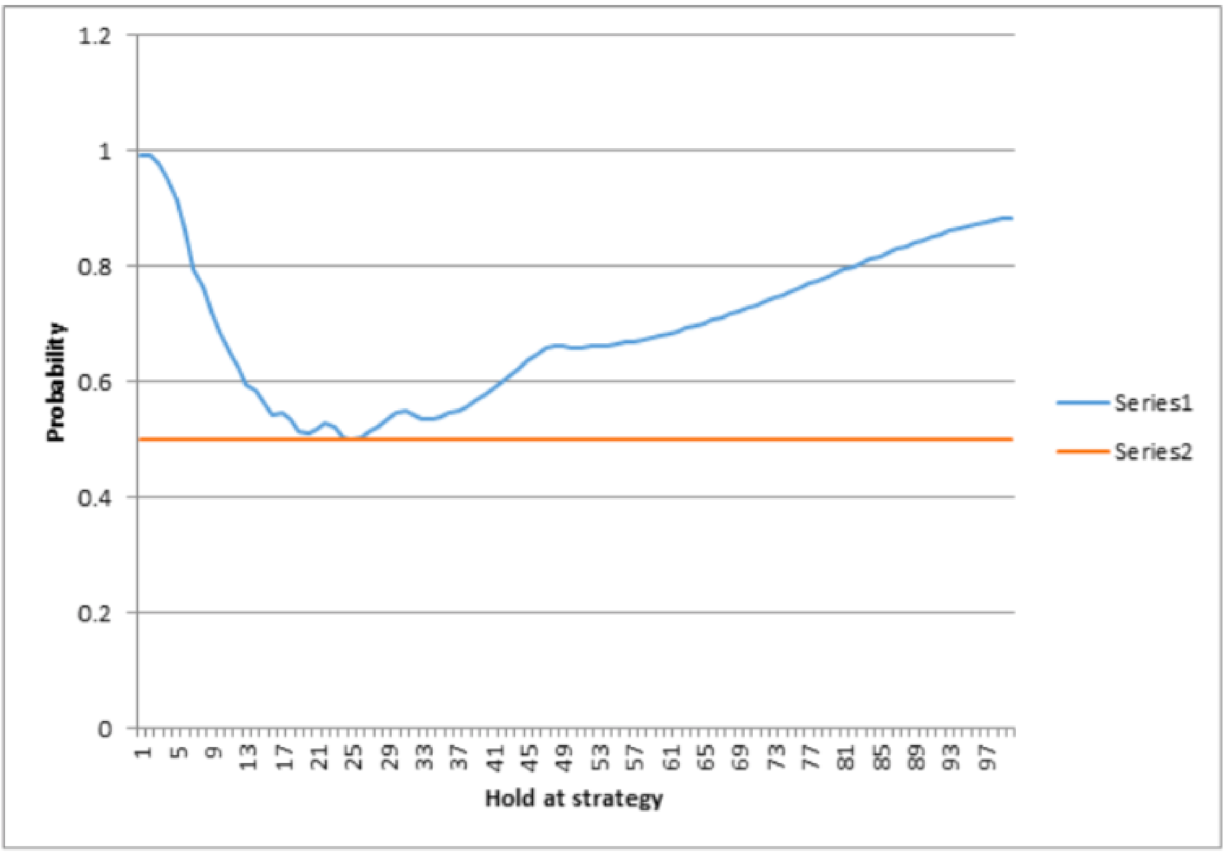


Figure 19: General probabilities of hold at 25 beating every hold at strategy

roll to wins probability lies below the lower bound of 0.475, further confirming our conclusion that turtle is superior. We then played roll to win against roll and wait and got both practical and theoretical probabilities of 1 for roll to win winning, regardless of who went first. Roll and wait produces a practical probability of beating turtle 0.7101 when going first and 0.7062, whereas the latter wins with probability 0.2983 when it goes first. This compares to the practical probabilities of 0.710606 for roll and wait and 0.297148 for turtle. The probabilities for roll and wait to win clearly lie above the upper bound of our confidence interval, and turtle below the lower bound, hence we conclude that it is the better strategy. As this experiment shows, for both practical and theoretical probabilities, turtle is better than roll to win, roll to win is better than roll and wait, however, roll and wait is better than turtle, giving us a 3-way loop, and hence proving that these such strategies are non-transitive.

Our function in MATLAB calculates the probability of a strategy winning given that it starts the game, therefore, theoretically we calculate the general probability of it beating every other hold at strategy as follows:
Let x denote the probability hold at 25 winning when going first.
Let y denote that of its opposing strategy when it goes first.
Let X be the general probability of hold at 25 winning.
For 3000 games, hold at 25 goes first 1500 times and second the same amount of times, so we have

$$X = \frac{1500(x) + 1500(1-y)}{3000}$$
$$X = \frac{x + (1-y)}{2}$$

As an example, lets compare it against hold at 30. We calculated the probability of hold at 25 winning when going first as 0.573741 and that of hold at 30 as 0.48091. If we put this in our formula, we get $X = \frac{0.573741 + (1 - 0.48091)}{2} = 0.546416$. Applying this against all other hold at strategies, we can produce the graph in Figure 18. As we can see in Figure 19, all of the general probabilities for hold at 25 winning lie strictly above the line of $y = 0.5$, other than when it plays against itself, when it gives a value of 0.5 as expected. From this, we can deduce that, theoretically, this is also the best of these simple hold at strategies.

Upon finding our “optimal strategy”, we then had to test it against other strategies in order to show that it is in fact the superior of the two in question and, in turn, the best to play against any given strategy. As we saw from the tournament of all of the hold at strategies, and the subsequent table of probabilities, hold at 25 was the best of the simple old at strategies. As the hold at strategies are also transitive, it made sense to test our optimal against this one and then conclude that our strategy beats all of the rest also. We ran the two against each other in line with our fair test and the simulation returned a practical probability of 0.55076 that our optimal would win when going first and 0.52136 in general, compared to hold at 25 winning with probability 0.50804 when going first and just 0.47864 in general. As we can see our optimal strategy beats hold at 25 when going first with a probability that lies above our confidence interval, showing that our strategy is superior. This is also in agreement of theoretical calculations, with our optimal theoretically winning 0.55127 in general and bank at 25 winning with probability 0.50914. As we know that the hold at strategies are transitive, this then means that our optimal strategy beats all such simple strategies also.

We then tested our optimal strategy against that of Nellers. These simulations returned results of a probability of 0.53182 when our strategy went first and 0.53176 when Nellers went first, which are values within each-others confidence interval which suggest that practically the two strategies could have the same probability. This was also backed up, with theoretical results showing that both nellers and our optimal won when going first with a probability of 0.53059. This then gave us a general probability of each strategy winning of 0.5, showing that our optimal strategy has in fact converged to that found by Neller previously.

4 Conclusions

4.1 Overall findings

4.2 Humans playing Pig and their interactions

Do players stick to their stated risk preferences?
No, we found out that players dont actually play in align with their stated risk preferences. It seems that on average, once participants play Pig they align with a different risk preference. We found that a lot more people are risk seeking than they originally stated. As we can see with the risk preference graphs, we expected them to look like the Figure 4, as shown in the introduction. However we found that this wasnt the case. We explained this as the risk preferences are evened out during the process of a game.

Player interactions
We have concluded that an individuals strategy is definitely influenced by their opponents. Game 11 (Figure 14) and game 14 (Figure 15) best represent our overall results found. In summary, we found that players change their strategy and do not stick to their stated risk preference and players copy their opponents strategy. If people stick to mirroring each other they actually have a more equal chance of winning, whereas if they change their strategies later on in the game it causes a worse outcome.

Limitations
Our sample size was fairly small. Therefore, if this experiment was to repeated, we would want to ensure that we collected data from a greater number of participants, so that we could be more confident in our conclusions.

4.3 Statsical analysis

To conclude statistical analysis, we first had to acknowledge that probabilities for our strategies winning could be measured both practically and theoretically and these can often differ greatly. In order to ensure that these were as close as they could be, we needed to create some rules for a fair test by which we would have to follow when playing two strategies against each other, in order to accurately compare them.

We first noted that a particular strategy will win against itself with a general probability of 0.5. We then constructed a confidence interval at the 95% significance level with the idea that we didnt want p to differ more than 0.025 away from 0.5, where p denotes the probability of a strategy winning against itself. From this, we could work backwards from our confidence interval and find the optimal number of games, n, required for one single fair test. This method produced $n = 1537$. However, we saw from Figure 16 that this value of n gave too many outliers in the data, so we instead chose $n = 3000$ as it appeared accurate for our particular confidence interval. As this number still produced some outlying data, we decided to run 100 fair tests in order to minimise the error as much as we could in order to obtain accurate practical probabilities against theoretical ones.

We decided first to look at simple hold at strategies and found them to be transitive. By this, we mean, for three strategies A, B and C; if A is better than B and B is better than C, then A is better than C. This means we could have one strategy superior than the rest and we found this to be hold at 25, when we ran a tournament between the strategies.

We were then required to test if the optimal strategy that we thought we had found was in fact superior. As we knew the hold at strategies to be transitive, we only needed to test it against the best of these in order to know that it was better than the rest, assuming that it won. Our optimal strategy won with general probability of 0.52136, thus allowing us to conclude that it was better than all of the hold at strategies.

We then tested our optimal against that found by T. Neller[5] to see if our strategy had in fact converged to this. Both strategies, when going first, won with the same probability of 0.53182, giving a general probability of 0.5 for each strategy. We could then conclude that our strategy was in fact the same as the optimal that T. Neller had found and written about himself.

References

[1] IGI Global. `https://www.igi-global.com/dictionary/risk/25414`, 2017.

[2] John Forbes Nash jr.

[3] Daniel Kahneman and Amos Tversky. Prospect theory: An analysis of decision under risk. In *HANDBOOK OF THE FUNDAMENTALS OF FINANCIAL DECISION MAKING: Part I*, pages 99–127. World Scientific, 2013.

[4] Reiner Knizia. *Dice Games Properly Explained.* Elliot Right-Way Books, 1999.

[5] Todd W Neller and Clifton GM Presser. Optimal play of the dice game pig. *The UMAP Journal*, 25(1), 2004.

[6] Todd W Neller and Clifton GM Presser. Practical play of the dice game pig. *The UMAP Journal*, 31(1), 2010.

[7] B Myerson Roger. Game theory: analysis of conflict, 1991.

[8] Theodore L Turocy. Texas a&m university. *Bernhard von Stengel, London School of Economics Game Theory CDAM Research Report (October 2001)*, 2001.