**Applying Statistics and Information Theory in "Battleship"**

**Mathematics Analysis and Approaches: Standard Level**

*1.1 Introduction*

As a child, board games were one of my favorite pastimes. They provided hours of entertainment and could be enjoyed by friends and family alike. While some games relied purely on luck, I found that the most engaging games were those that combined strategy with elements of uncertainty. The sense of risk, formulating predictions, and the surprise created by these games always kept me on the edge of my seat. Personally, one game that epitomized these qualities was "Battleship."

In Battleship, the objective is to sink all five of your opponent's ships, which are hidden somewhere on their 10x10 board. The gameplay is turn-based, with each player taking turns guessing the location of their opponent's ships. A hit results in the opponent declaring "hit," a sunk ship results in "sink," and a miss results in "miss."

While a random guessing strategy may work in some cases, the high level of uncertainty in Battleship renders this approach inefficient. I believe that there exist more effective strategies for winning Battleship, and I aim to explore this throughout my paper.

*1.2 Aim of Exploration*

Due to the nature of this game, I hypothesize that statistics and probability can be applied to develop a winning strategy. Although you are typically searching for 5 different ships, to simplify my exploration I will explore methods to efficiently target one ship in the entirety of the 10x10 board. Therefore, the final goal of this exploration should be to develop a methodology or algorithm that maximizes hits and minimizes misses to locate the final position of ships. By analyzing previous guesses and responses, it should determine the most likely locations of opponent ships and adjust my strategy accordingly. Additionally, the algorithm should evaluate the effectiveness of different guesses and optimize my gameplay for maximum success.

*1.3 Additional Definitions and Context*

Before moving forward, there are a couple specific words and definitions that should be defined. The Battleship board is shown below:

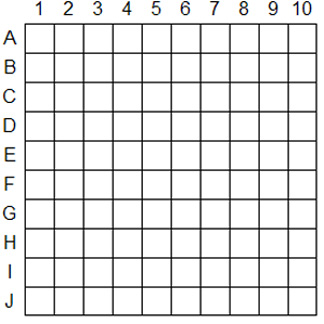


Fig 1.1 – Battleship board that will be referred to in this exploration

For formatting purposes, the table of important definitions is mentioned on the following page.

Table 1.1 – List of important definitions

| Board | It is where ships are arranged. There exists 100 spaces on the board |
| --- | --- |
| Space | Any quadrant or coordinate on the grid. Example of spaces are:  A1, J10, E5, G7 |
| Ship | It is any of the 5 boats that can be placed on the board. Each boat possess different lengths:   * A Patrol Boat occupies 2 spaces * A Submarine or Cruiser occupies 3 spaces * A Battleship occupies 4 spaces * An Aircraft Carrier occupies 5 spaces     Source: <https://hobbysprout.com/how-to-play-battleship/> |
| Shot  Taking a shot  Playing a move  Taking a guess | This refers to asking the opponent whether a space of interest either hit a ship or missed entirely |
| Position of a ship  Arrangement of a ship | Refers to the set of spaces that a particular ship occupies. Take an Aircraft Carrier as an example: it can possess some of the following arrangements:  A1, A2, A3, A4, A5  C4, D4, E4, F4, G4 |
| Hit | Refers to accurately pinpointing one of the spaces of an opponent ship |
| Miss | Refers to inaccurately pinpointing one of the coordinates of an opponent ship |

*2.1 Methodology: What are existing ways or tips to win?*

Before delving into mathematics, it is important to review what others have done to increase their chances of winning. This review provides a basis of where to begin. In reading 4 articles and blogs, there were 3 recurring concepts:

* Do not group shots in one area
* Use the “checkerboard method” or parity. This is where you consider the entire board has checkered spaces where you only target dark spaces
* Shoot initial shots at the centre of the board

For an efficient method or algorithm it should include one or more of these approaches. Aside from simple tips, I also explored 2 articles that applied logic and statistics to solve this problem. In both articles, they notably used computers to calculate the likelihood of a ship residing in each position on the board. By doing this, they were able to rank each square appropriately.

As an example, consider the space E1. For a Cruiser, there are 4 possible arrangements that contain the space. As a result, it will have a weight of 4. But when E5 is considered, there are 6 possible arrangements, thus it will have a higher weight of 6. However, as more shots are taken, the possible arrangements in one space might decrease, reducing the weighting and favouring other spaces.

Although this system intrigued me, their weighting of each space seemed oversimplified; the likelihood or *information content* should be better described than an occurrence count. Therefore, “how can this be measured?”. Fortunately, one mathematical way to describe this information can be found in information theory.

*2.2 Methodology: Utilizing information theory to describe information.*

Consider the entire sample space of a theoretical ship of length 1. Since it is of length 1, its sample space would be the 100 different arrangements that can be positioned on the 10x10 board. Now let us say a theoretical observation reduces the sample space by half -- therefore it is now 50. In information theory, one such observation equates to 1 **bit** of information.

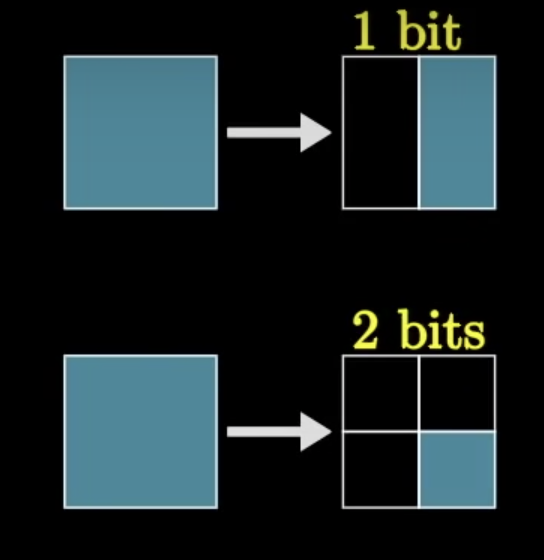
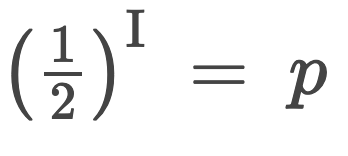


Fig 2.1 – A visual representation of “bits of information”.

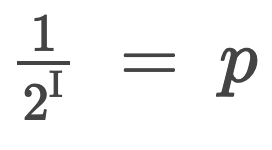
Source: <https://www.youtube.com/watch?v=v68zYyaEmEA&t=932s>

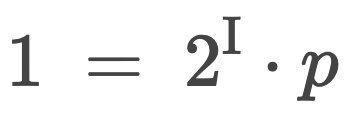
However, because this was halved, the bit of information also describes the new probability of the ship’s real location occurring in the sample space -- ½ of the original. Similarly, if the sample space is halved once again, this equates to 2 bits of information -- ¼ sample space of the original. This can be further summarized in the following equation

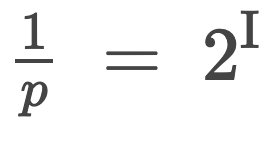


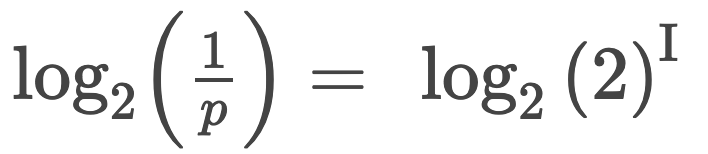
Where “I” is the number of bits and “p” represents the probability of the ship’s location being in sample space.

With this, we can then solve for “I”:











The formula for information was first introduced by Claude Shannon in his paper, “A Mathematical Theory of Communication”. This then begs the question: “what is the information content of every ship in the game?”. As such, I applied this formula to each one to see how much they vary. In each case, the probability of each ship is the likelihood of finding its actual arrangement in its sample space. Thus: p = 1/total arrangements.

| SHIP | NUMBER OF APPROXIMATE BITS |
| --- | --- |
| Aircraft Carrier  (Contains 120 possible arrangements) | **p = 1/120** |
| Battleship  (Contains 140 possible arrangements) | **p = 1/140** |
| Submarine/Cruiser  (Each contains 160 possible arrangements) | **p = 1/160** |
| Patrol Boat  (Contains 180 possible arrangements) | **p = 1/180** |

Table 2.1 – Information content of each ship

So to find the location of the Aircraft Carrier, one would need to halve its 120 sample space around 7 times consecutively. This is because information is additive; a series of observations can be used to find the **total information gain**. As the youtube channel, 3blue1brown describes it: “in the same way probability likes to multiply, information likes to add”. A visual representation is found below:

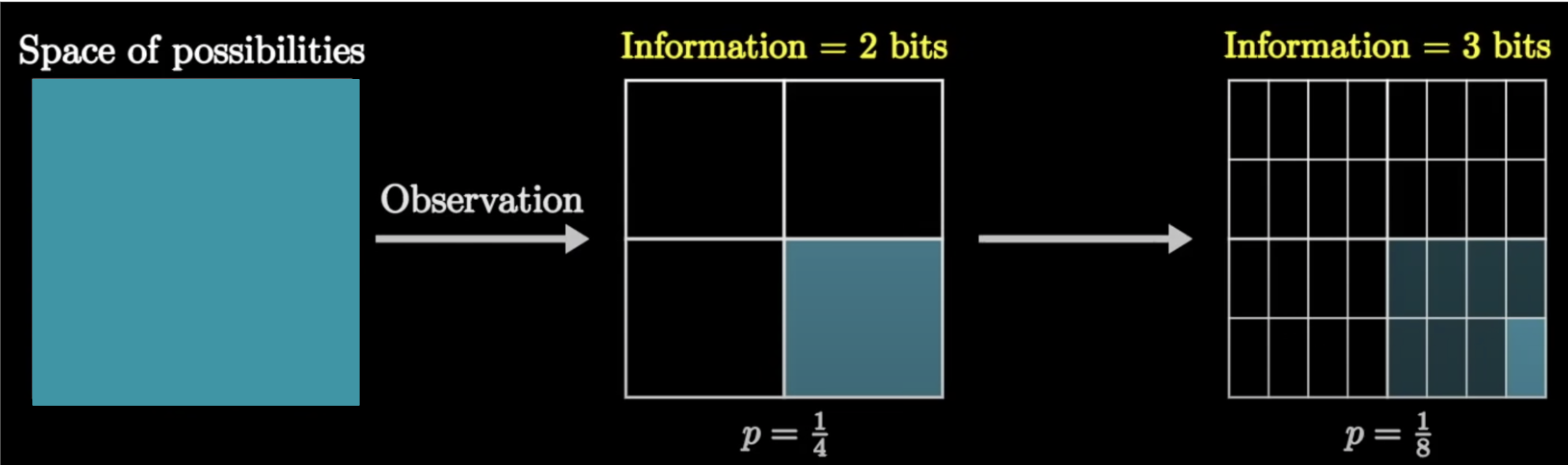


Fig 2.2 – In the above example one observation reduced the original sample space to ¼ -- 2 bits. Then, the subsequent observation took the ¼ of sample space and reduced it to ⅛ -- 3 bits. But in total, the set of observations resulted in **5 bits of information gain.** Source: <https://www.youtube.com/watch?v=v68zYyaEmEA&t=932s>

*2.3 Methodology: Utilizing entropy to evaluate spaces.*

With a new definition of, “information content”, we could use this to evaluate each space on the board. Take space A1 as an example:

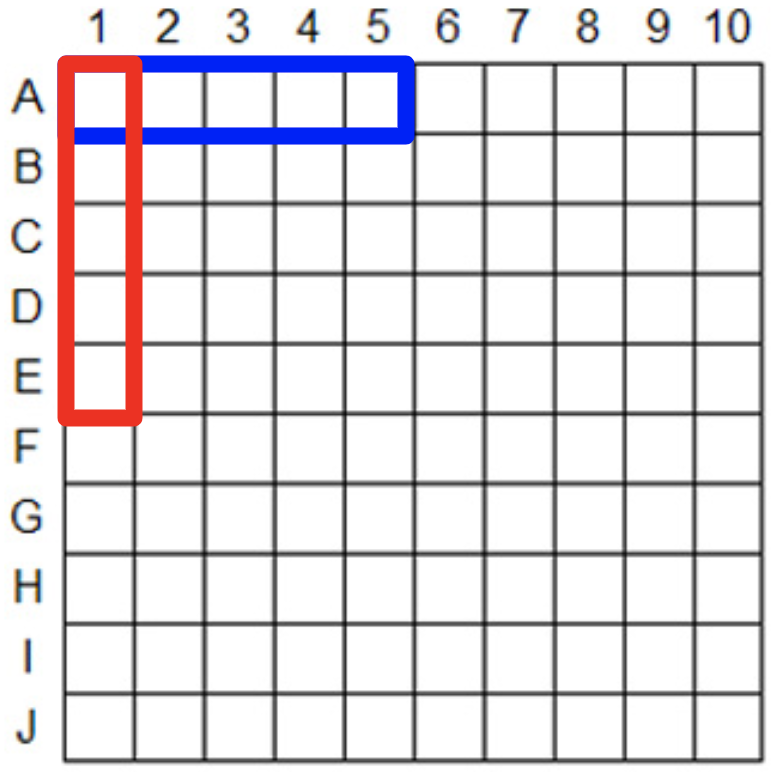
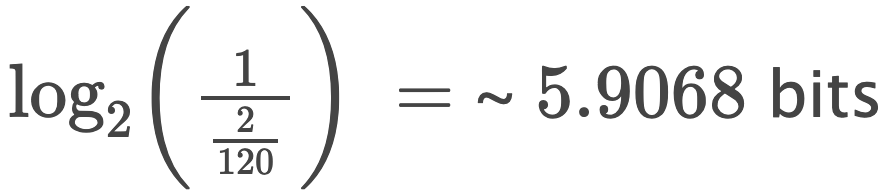
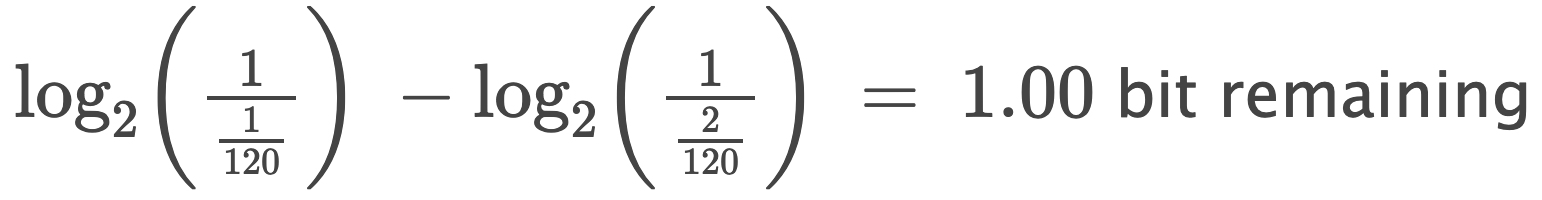


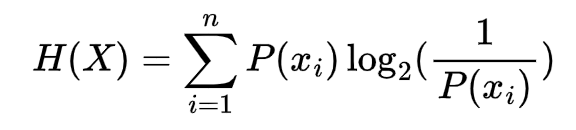
Fig 2.3 – Different arrangements an Aircraft Carrier can be positioned containing space A1

Since there exists two possible ways to arrange an Aircraft Carrier on A1, if there is a hit on that space, we immediately reduce the sample space from 120 to 2. Mathematically, we gain 5.9068 bits. Subtracting this value from the total bits required to find the single arrangement, we are left with 1 bit of information -- simply, a remaining ½ chance of finding its final arrangement.





If we rank spaces based on this information gain, spaces like A1 or A10 would be favoured over others. This may work occasionally, but it is flawed because the probability of hitting an Aircraft Carrier in this space would still be very low since it is in a corner. To address this, the ranking algorithm should also take into consideration the probability and information gained if it misses entirely on a space. This can be done by taking a type of expected value called **entropy** that was also introduced by Claude Shannon:



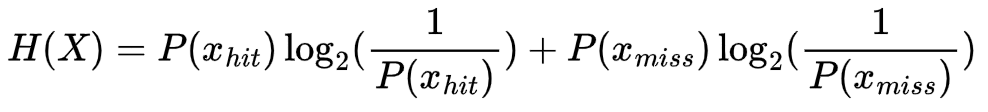
Where:

* “H(X)” is the expected value of information or **entropy**
* “X” is a random variable that can take different values, “x”.
* “P(x-i)” is the probability of the “i-th” to the “n-th” event occurring.
* The logarithm base 2 of the reciprocal of P(x-i) is the information content of x-i

To apply this in the current context, consider X as any space on the board that takes two events; being a hit or a miss:



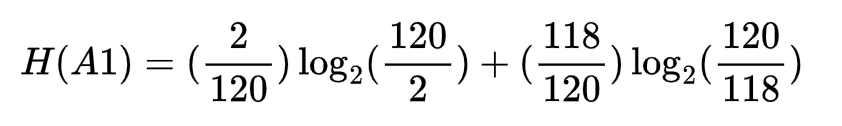
Therefore, I only need to calculate the entropy considering only x-hit and x-miss:

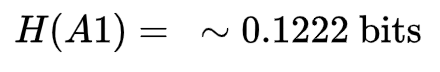


Where:

* “H(X)” is the expected information of a specific space
* “P(x-hit)” is the probability of obtaining a hit in that space
* “P(x-miss)” is the probability of obtaining a miss in that space

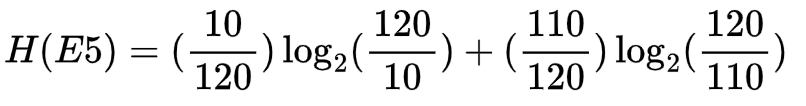
With this, we can then calculate the expected information of a particular space. Returning to the example of A1, its expected information or entropy is 0.1222 bits.





Here, 2/120 is the probability of obtaining a hit since there are 2 arrangements in A1 out of the sample space of 120. Likewise, 118 is the remaining probability of missing.

This process can then be iterated over the entire board to find the space with the highest entropy. Since this would take a substantial amount of time, I decided to create a Python computer program that does this computation-heavy task. According to the program, the best opening shot to find an Aircraft Carrier would either be E5, E6, F5, F6 since they all possess the same entropy:

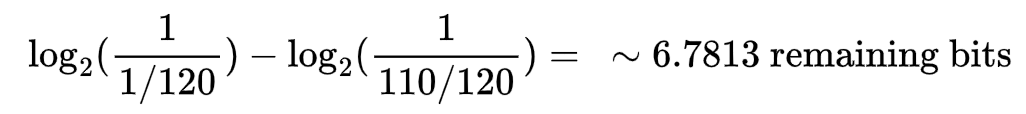




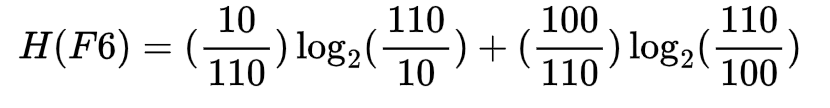
Here, 10/120 represents the probability of a hit. It is 10/120 instead of 2/120 in the previous example since there are 10 different ways an Aircraft Carrier can be arranged containing E5. 110/120 is therefore the probability of missing.

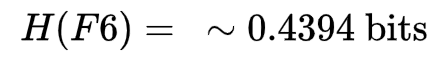
*2.4 Methodology: Reducing Sample Space*

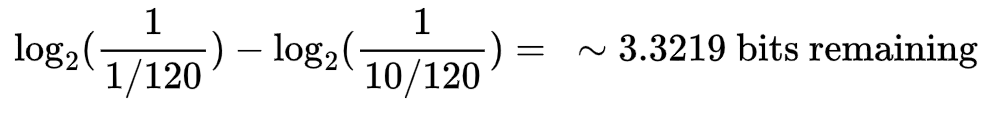
Since the algorithm suggests E5 as a move, if I choose to play it, it could either be a hit or a miss. In case of a miss, the arrangement of the Aircraft Carrier would still contain 6.7813 bits.



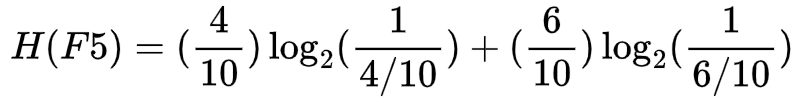
Additionally, it would need to remove the 10 ship arrangements containing E5 from the sample space. Therefore the next space to guess would be calculated differently using 110 as the size of the sample space. As a result, the program suggests F6 as the space contains 10 possible arrangements, thus having a hit probability of 10/110:

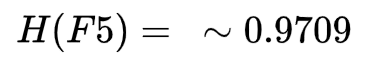




But if guessing at E5 registers a hit, then the location of the Aircraft Carrier is drastically reduced to 3.3219 bits. 

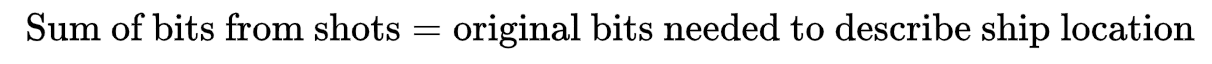
This also means the sample space needs to be reduced to 10 possible arrangements. Following this hit, the algorithm recommends guessing either on spaces E4, E6, D5 or F5 that each have a 4/10 probability of hitting and 6/10 probability of missing. Thus, the expected information is nearly 1 bit:

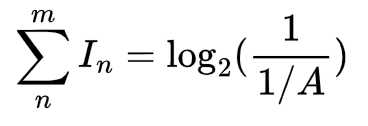




Overall, each time a guess is made, the calculations for the next spaces should take into account the reduced sample space.

It is also important to note that the sum of each guess in the end will always equal the original bits of information needed to describe the location of the ship at the start. For any ship, this can be described in the following:





Where:

* “I” is bits of information gained at the “n-th” shot that is summed over the total number of shots, “m”.
* “A” is the total number of arrangements for a specific ship at the beginning of a game.

*3.1 Analysis: Simulations*

To better understand the efficiency of the algorithm, I performed 1000 simulated games where it tried to pinpoint the location of the ship. The number of games that took a specific amount of shots is graphed below using matplotlib. Note that each game’s target location was randomized.

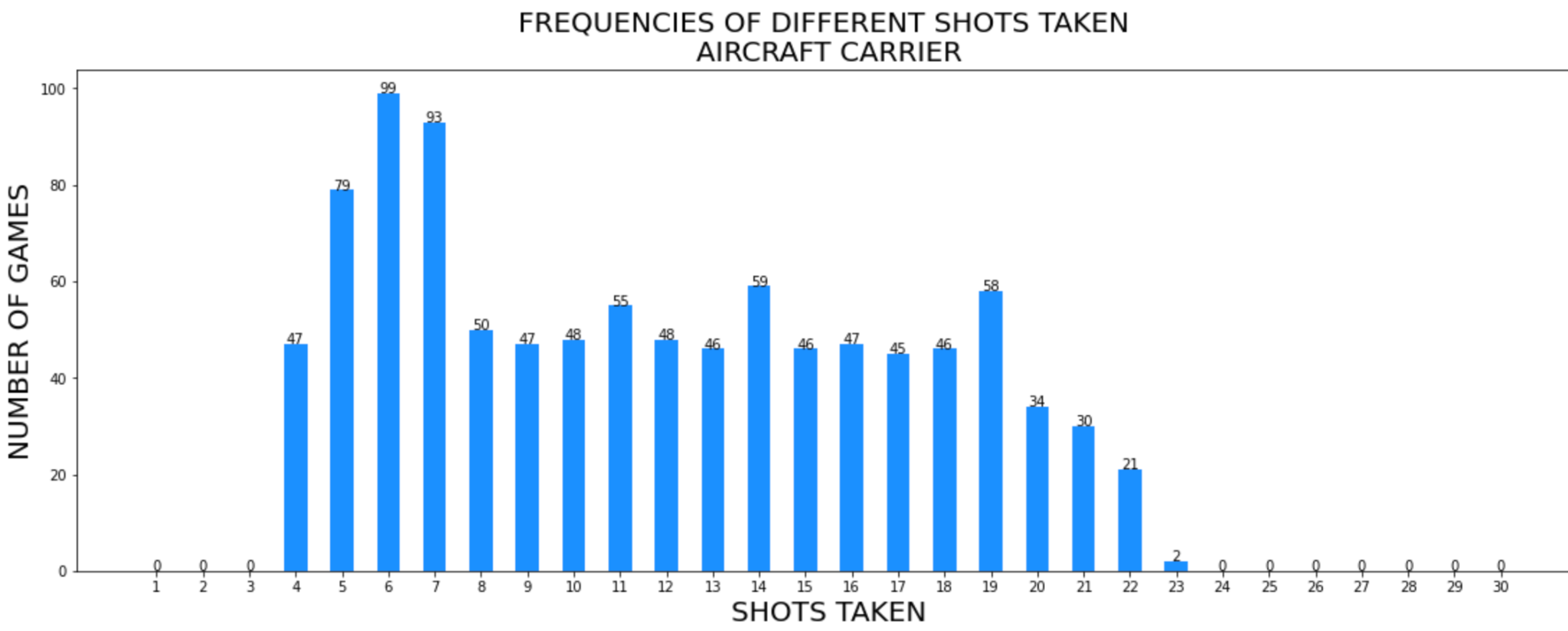


Fig 3.1 – Graph illustrating the shot distribution to find the Aircraft Carrier. Overall, its values are not very spread apart

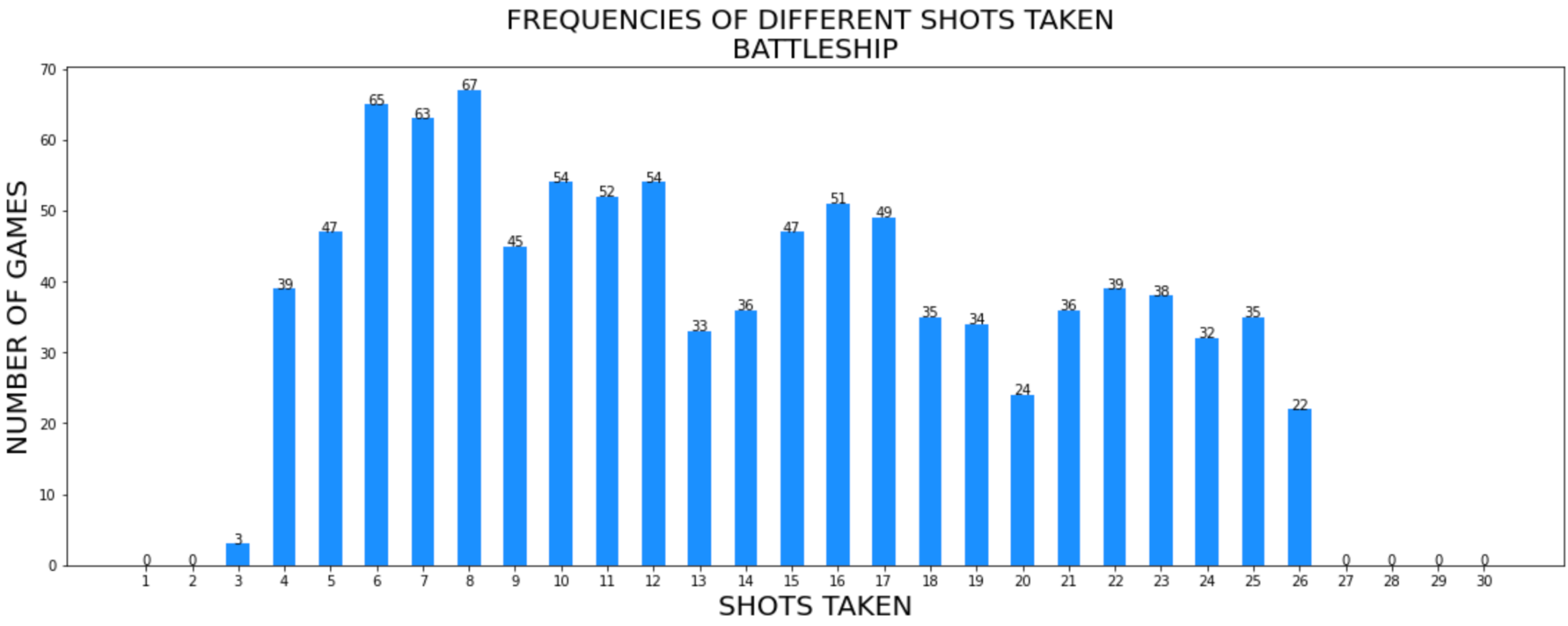


Fig 3.2 – Graph illustrating shot distribution to find the Battleship. Visually, there is a slightly greater variety of shots, but the mode still lies between 6 to 8 shots taken.

From initial observations, the amount of shots taken to sink an Aircraft Carrier or Battleship are quite similar. This could be due to their lengths being 5 and 4 – the largest in the game.

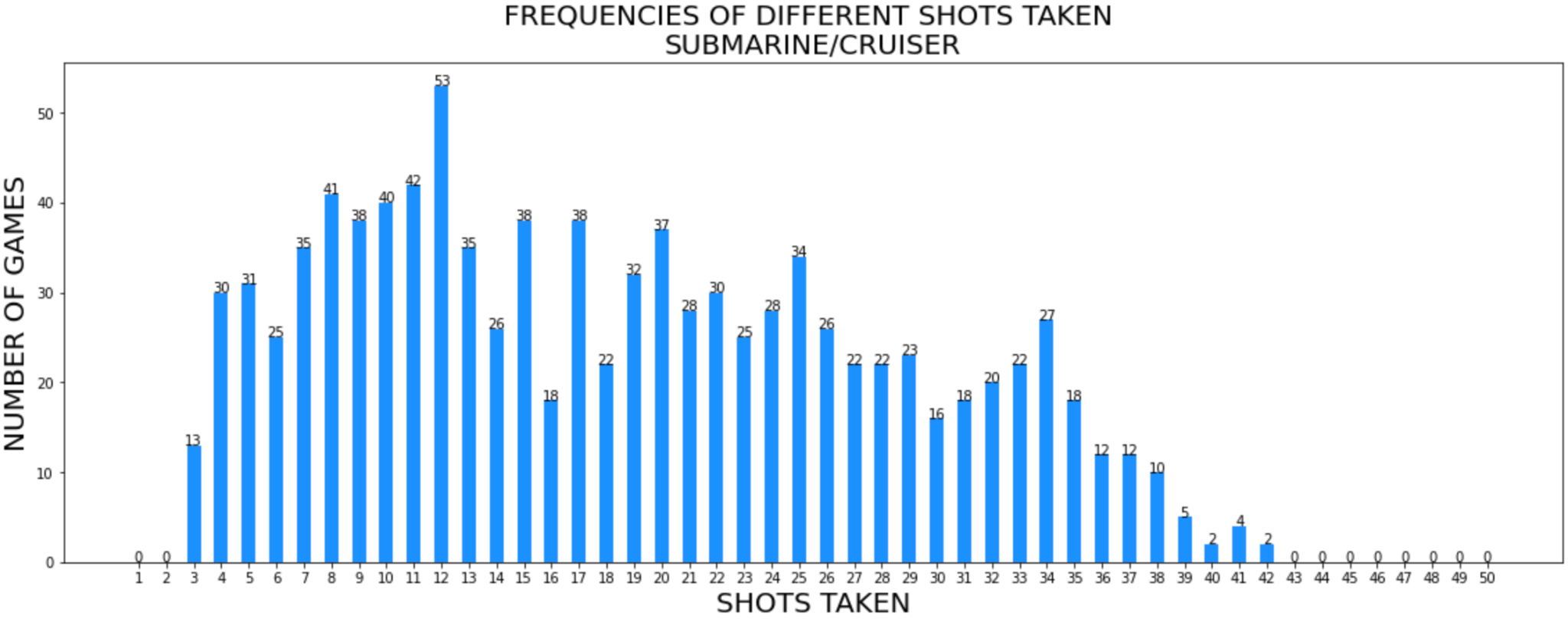


Fig 3.3 – Graph illustrating shot distribution to find the Submarine or Cruiser. I decided to group these two ships together as they both possess the same length of 3. The range of this distribution extends to 39, which is significantly greater than the previous 2 that were in the 20s.

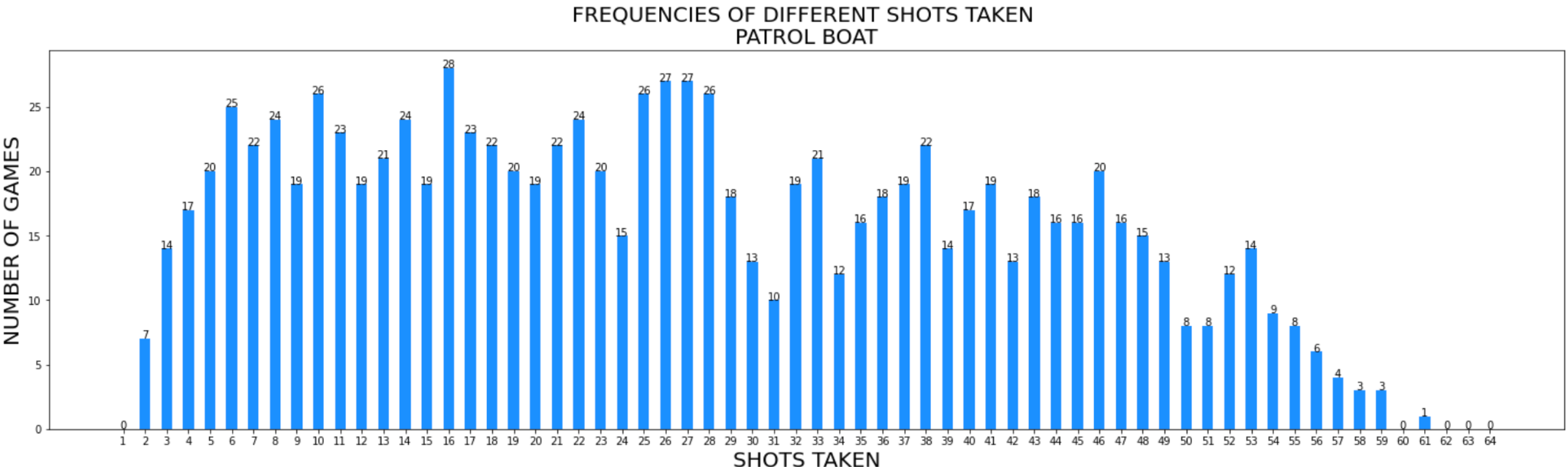
**

Fig 3.4 – Graph illustrating shot distribution to find the Patrol Boat. The range and distribution is significantly greater than all ships. This means it is the hardest ship type of the 4 to find. Despite this, it is not surprising as a smaller ship would naturally be more difficult to find.

This data can also be described the following measurements:

| SHIP TYPE | MODE  (Shots taken) | RANGE | MEAN NUMBER OF SHOTS | STANDARD DEVIATION  OF SHOTS |
| --- | --- | --- | --- | --- |
| Aircraft Carrier | 6 | 19 | 11.691 | 5.309 |
| Battleship | 8 | 23 | 13.678 | 6.424 |
| Submarine/  Cruiser | 12 | 39 | 18.573 | 9.732 |
| Patrol Boat | 16 | 59 | 26.672 | 14.907 |

Table 3.1 – Summary of different statistical measures applied to each ship data. Each measurement was calculated using the “statistics” module found in the Python Standard Library.

Before proceeding with the simulations I did attempt to play a couple games by myself. Unfortunately I did not record every game, but for an Aircraft Carrier I averaged 15 shots to find its location. Seeing that my algorithm bested this score by around 3 shots, I was slightly satisfied. However, I still was eager to improve this by any means possible.

*3.2 Analysis: Using Parity*

Fortunately, the review of the different articles in section 1 re-introduced the topic of parity or the checker-board method. As the name states, this approach consists of viewing the entire board as a checkered, where one should only aim on the darkened squares. Because each ship has a minimum length of 2, in theory this should reduce the amount of available spaces to guess from 100 to 50. Therefore, I re-modified my algorithm and simulated another 1000 games. However, because of bugs in my program and computation power of my computer, I was only able to record data for the Aircraft Carrier and Battleship:

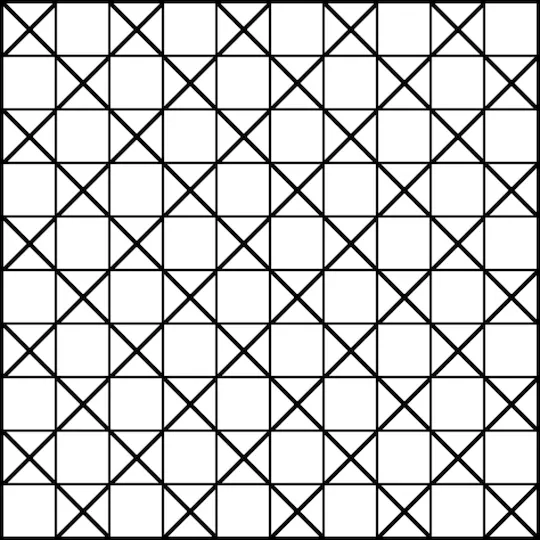


Fig 3.6 – An example of a board utilizing parity to facilitate shot-taking.

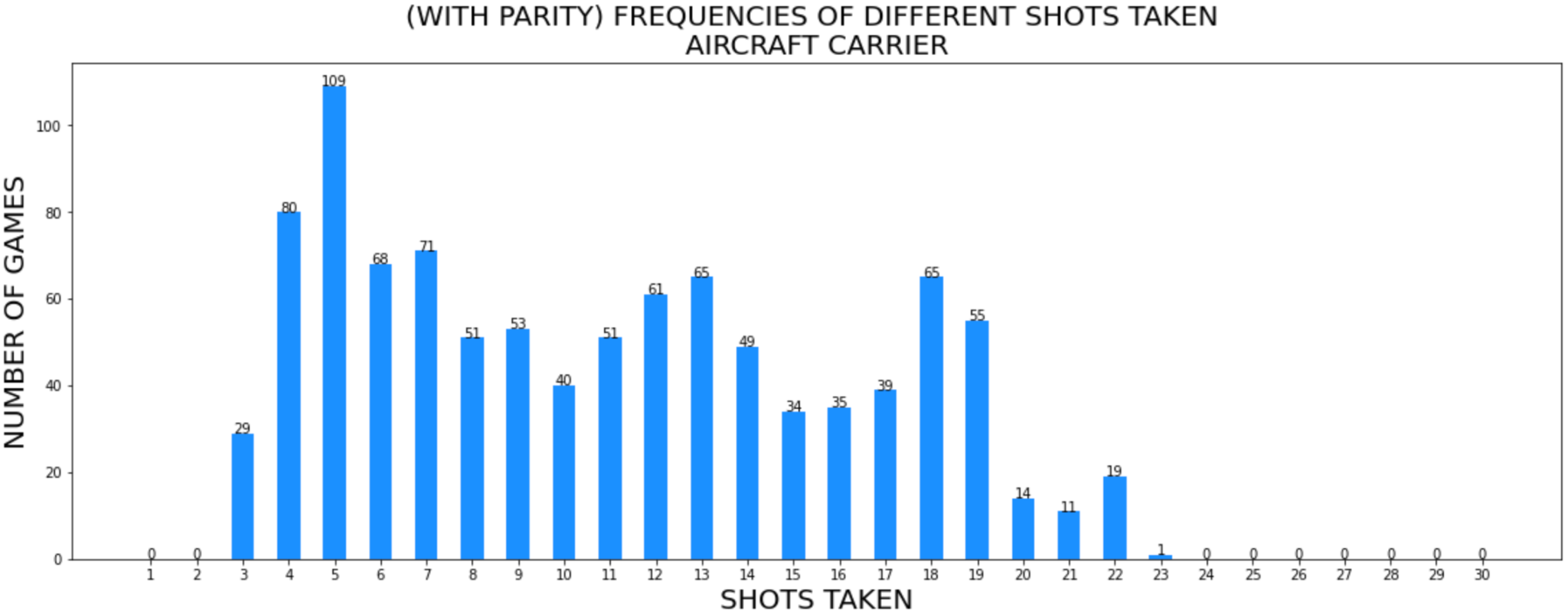


Fig 3.7 – Graph illustrating the shot distribution to find the Aircraft Carrier using parity. Overall, the range grew smaller and the distribution is focused towards shots 4 and 5. The new mode is 5 as opposed to 6.

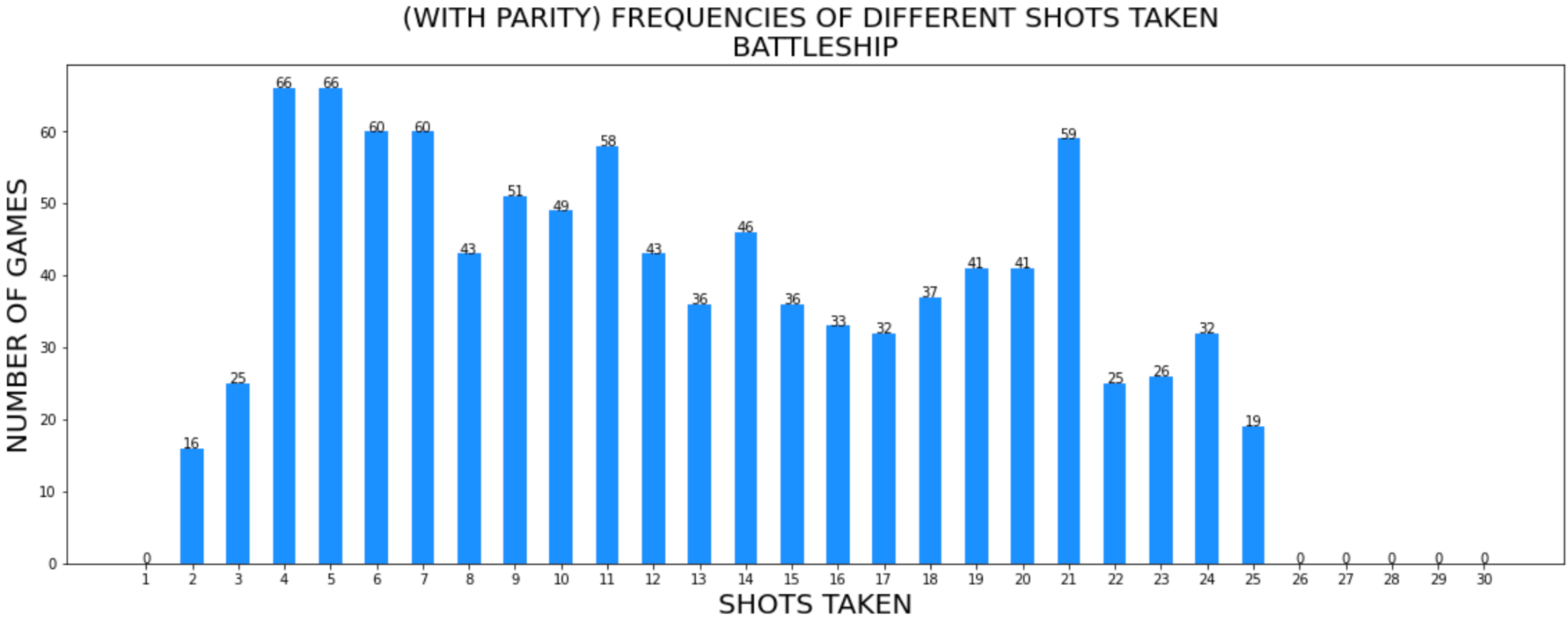


Fig 3.8 – Graph illustrating the shot distribution to find the Battleship using parity.

Like previously, the new data can be interpreted using statistical measures:

| SHIP TYPE | MODE  (Shots taken) | RANGE | MEAN NUMBER OF SHOTS | STANDARD DEVIATION  OF SHOTS |
| --- | --- | --- | --- | --- |
| Aircraft Carrier | 5 : 6 | 20 : 19 | 10.866 : 11.691 | 5.312 : 5.309 |
| Battleship | 4 and 5 : 8 | 23 : 23 | 12.547: 13.678 | 6.478 : 6.424 |

Table 3.2 – Summary of statistical measures applied to each ship using parity. Values in red denote measurements using parity while values in black denote measurements not using parity. Each measurement was calculated using the “statistics” module found in the Python Standard Library.

By utilizing parity the mean number of shots for each ship were reduced by around 1. Additionally, the mode for each ship was reduced, especially for the Battleship type. However, the distribution of shots was more distributed and varied shown by the slight increases in standard deviation and range. This was even more highly evident in the Battleship graph when there was a resurgence of 59 games that took 21 shots. Although I could say implementing parity did improve the algorithm, it was not by a significant factor. For such a method to greatly increase the effectiveness of my algorithm, it needs to decrease the value of the mode and mean. Additionally, a specific method should make all data points on a graph converge to 0 shots taken. This suggests that a particular approach should also reduce the standard deviation.

*4.1 Other Extensions/Improvements*

Aside from this simple checker-board parity, I believe there are other approaches that should be explored not covered in this paper. To increase the efficiency of the algorithm, one or more of these proposals should be visited. One strategy that could yield improvements is diagonal parity as shown below:

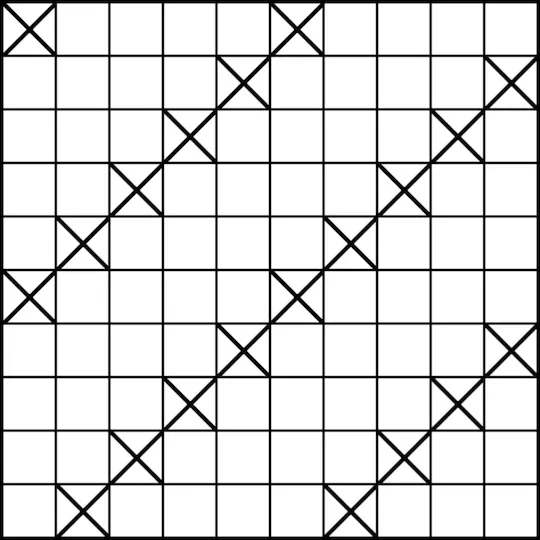


Fig 4.1 – Diagonal parity

This type of parity could potentially reduce the sample space of Battleships and Aircraft Carriers. Because of their longer lengths, doing shots in this pattern could be beneficial

Secondly, another possible improvement could be dividing the board into segments and analyzing its overall entropy. The initial algorithm has a drawback in that it considers the entire board in terms of calculations. This “chunking” could be explored to test whether grouping or spreading one’s shots translate into better results. Many of the previous articles highlight this strategy, therefore could this could be implemented in the algorithm?

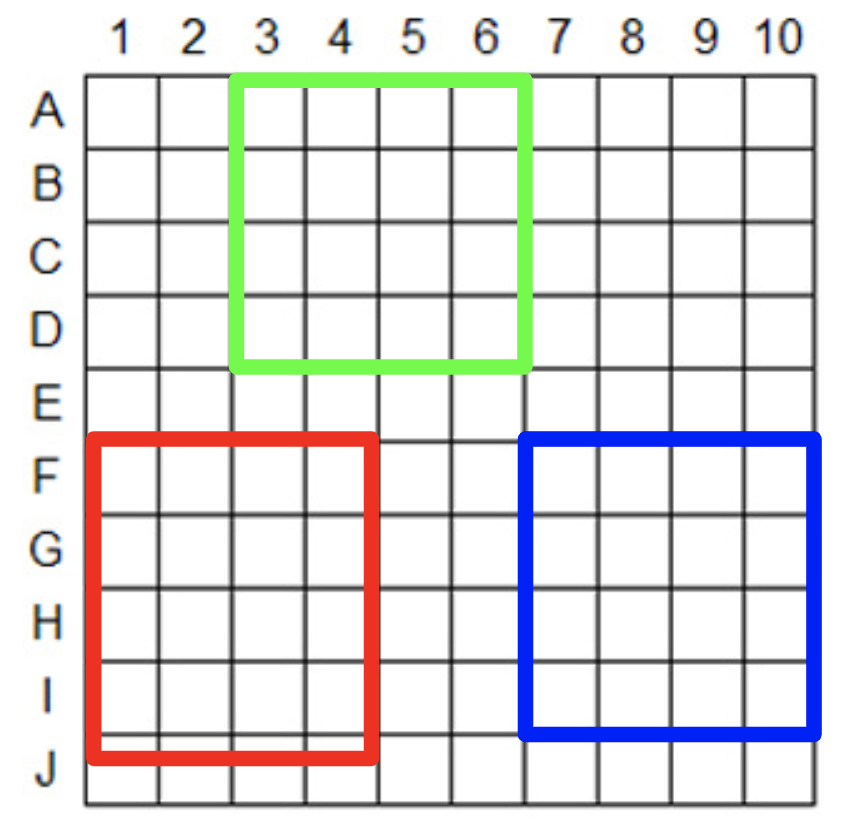


Fig 4.2 – In this case, perhaps information theory can be used to calculate the total information gain of each highlighted segment. This divides the entire board into smaller problems that can be analyzed. Then, the algorithm can aim in more probable areas instead of looking at the board as a whole.

Lastly, this exploration was mainly concerned with the offensive, shot-taking aspect of Battleship. Since this algorithm can play thousands of simulated games, perhaps I can use it for defensive purposes. In particular, can I use this algorithm to find the best arrangement of ships that avoids shots? By looking from the other side of the problem, perhaps I can find an answer to this question.

*4.2 Conclusion*

To address the initial aim of this exploration, an algorithm and methodology was developed to locate ships. However, its efficiency and accuracy can still be improved if the previously mentioned topics of interest are investigated.

In the end, I found this exploration paper to be quite interesting and quite nostalgic. Battleship was a game that brought me so much fun as a child and I still remember days when I scratched my head, desperately trying to figure out the best space to guess in front of haughty parents. To choose this topic paid a great homage to one of my favourite games. In terms of mathematics, I have always found statistics, although seen in contempt by many classmates, as very practical in the real world. It is amazing how it can be used in many scenarios to make sense of data or a problem that can often seem undecipherable at first glance. Ultimately, this interest led me to information theory that I find just as amazing. The work of Claude Shannon helped quantify an ambiguous concept like information that sees applications in technology and communications.

Overall, this exploration has exposed me to a different field of mathematics, a different way of thinking and a different way of problem solving. Statistics is a field that encompasses many aspects of the real world – even a simple game of, “Battleship”.

*Bibliography*

wikiHow. (2021, August 4). *3 ways to win at battleship*. wikiHow. Retrieved March 3, 2023, from https://www.wikihow.com/Win-at-Battleship

*Tips to win Battleship*. UltraBoardGames. (n.d.). Retrieved March 3, 2023, from https://www.ultraboardgames.com/battleship/tips.php?utm\_content=vc-true

Brown, S. (2020, August 11). *Win battleship with these basic strategies*. The Spruce Crafts. Retrieved March 3, 2023, from https://www.thesprucecrafts.com/how-to-win-at-battleship-411068

Team, G. (2022, July 12). *How to win at battleship: 15 tips, tricks & strategies (boost your win rate)*. Gamesver. Retrieved March 3, 2023, from https://www.gamesver.com/how-to-win-at-battleship-tips-tricks-strategies-boost-your-win-rate/

O'Neill, P. (2022, March 12). *Battleship*. nulliq. Retrieved March 3, 2023, from https://www.nulliq.dev/posts/battleship

Schwartz, A. (2022, April 20). *Coding an intelligent battleship agent*. Medium. Retrieved March 3, 2023, from https://towardsdatascience.com/coding-an-intelligent-battleship-agent-bf0064a4b319

3Blue1Brown. (2022, February 6). *Solving wordle using information theory*. YouTube. Retrieved March 3, 2023, from https://www.youtube.com/watch?v=v68zYyaEmEA&t=932s

[J. D. Hunter, "Matplotlib: A 2D Graphics Environment", Computing in Science & Engineering, vol. 9, no. 3, pp. 90-95, 2007](https://doi.org/10.1109/MCSE.2007.55).

Shannon, C. E., & Weaver, W. (1949). *The mathematical theory of communication.* University of Illinois Press.