Problem Chosen	2023 MCM/ICM Summary Sheet	Team Control Number
C		2322810

Analyzing and Predicting Wordle: A Multi-Model Approach to Difficulty and Number of Reports Prediction, with a Bonus Winning Strategy

To better understand the Wordle game and the player sharing behavior, we deisned three model to assess various aspects:

1. We devised a multiple linear regression model to quantify the difficulty of each word. Independent variables are solely from the solution word itself, so any word choice from current dictionary will produce a difficulty feedback.
2. We exploited the piecewise curve fitting method to understand the variation in the number of reported results and predict its future trend on March 1, 2023.
3. To predict the distribution of future reported results given a specific world, we built a Monte-Carlo Simulation to estimate the distribution of the reported results for any given five-letter word in the Wordle dictionary.

These models can work independently or interactively. Using three models together, we can predict the number of reports, whether they are in hard modes, and produce a prediction interval to the estimated value. For example, on March 1, 2023, there will be approximately (11024, 19548) reported results. There will be about 11024 reported results if the given puzzle is hard and 19548 reported results if the puzzle is relatively easy for players. Moreover, we anticipate that on March 1, 2023, if the word is “eerie,” the difficulty level will be categorized as medium. With this difficulty, we have evidence that the distribution of the reported score on March 1, 2023, will be (0.32, 8.57, 23.36, 39.95, 22.06, 9.87, 1.03) with respect to (1 try, 2 tries, 3 tries, 4 tries, 5 tries, 6 tries, and 7 or more tries).

Lastly, to explore beyond the posted questions, we figured out an interesting approach to winning the Wordle game based on what we found in our interaction with the dataset.

Table of Contents

1. Introduction *3*
1.1. Wordle 3
1.2. Game Rules 3
1.3. Data Cleaning Process 4
2. Assumptions *5*
3. Models *7*
3.1. Variables and Parameters 7
3.1.1. Model 3: OLS Multiple Linear Regression 7
3.1.2. Model 1: Model Fitting 8
3.1.3. Model 2: Monte Carlos Simulation 8
3.2. Constraints 8
3.3. The Model: 8
3.3.1. Model 1: Difficulty of a certain word 8
3.3.2. Model 2: Curve Fitting 11
3.3.3. Model 3: Monte-Carlo Simulation 13
4. Results *16*
4.1. Word attributes that affect score reports played in Hard Mode 16
4.2. Model predicting the distribution of reported results 17
4.3. Classification model based on difficulties 18
4.4. Wordle game solving strategy 18
5. Reflection *19*
5.1. Difficulty Regression Model 19
6. Conclusion *20*
7. Letter *21*
8. Reference *22*
9. Appendix *23*

1. Introduction

1.1. Wordle

The New York Times Wordle game is a popular online game where players have to guess a secret five-letter word by inputting different combinations of letters, which construct an actual word in the dictionary. This game has recently gained popularity, especially after the New York Times announced the purchase of the original creator, Josh Wardle. Since its acquisition, thousands of people have played each day. Nevertheless, the number of players that report their game stats has fluctuated from day to day. Various factors contribute to the variability of the reported numbers, such as the company's acquisition by the New York Times or the difficulty of words each day. In our analysis, we aim to find how different attributes affect the daily report numbers and how it relates to the number of reports played in hard mode. In addition, we aim to create a model to evaluate the difficulty of the words randomly generated by the company each day. Another model we developed is a simulation model to predict the future distribution of the percentages of the number of tries used to solve the game. Lastly, we will introduce some strategies to help the player win the game more efficiently.

1.2. Game Rules

To help readers and players understand the game better and comprehend the solving logic of this research paper, we reaffirm the rules of the Wordle game below:

1. Wordle puzzle involves guessing a five-letter word randomly generated by the computer that will be the same regardless of the physical position, device, and time you play during the specific day.
2. Each player has six attempts to guess the word correctly by entering different combinations of letters that must be an accurate, meaningful, and completed term from dictionaries.
3. After each guess, the game provides feedback by indicating the correctness of both spelling and position. Specifically, the green color represents a correct letter in the proper place. Yellow means the presence of a specific letter in the solution but is not in the right position. Finally, gray implies the absence of that letter in the puzzle.
4. Based on each feedback, the player will improve their guess and eventually figure out the keyword.
5. If the player did not guess the correct word within six trials, the player failed to solve the Wordle game. The result will be categorized as '7 or more tries (X)'.

1.3. Data Cleaning Process

To ensure the accuracy and completeness of data, we first start with a data cleaning process. We removed three data points associated with three words we believed were data errors. Data entries that we removed are:

- Entry 18 - "rprobe" - contest number: 545 - due to a typo in the word "probe"
- Entry 38 - "clen" - contest number: 525 - had four letters instead of five
- Entry 249 - "tash" - contest number: 314 - has four letters instead of five

We also removed a data point we believed to be an outlier from performing the standardized residual plot:

- Entry 34 - "study" - contest number: 529 - this data's number of reported results is an outlier - 2569 results

After removing these data errors and outliers. We proceed to perform data analysis and build models on this dataset.

2. Assumptions

We introduced the following assumptions for our three models:

- Players can guess all characters that appear in the solution within six trials: After submitting each trial, players will get information on whether a character appears in the solution. Since there are 26 letters in the alphabet and 30 entries for six trials, players should be able to figure out the word and win within six trials.
- The sum of the fifth and sixth trials for each word implies puzzle difficulty: We determine each word's problem based on the sum of the fifth and sixth trials used. Since players can guess all five letters within the solution after six trials, Wordle puzzles should be solvable within six attempts. If the keyword is too difficult to guess that day, players will need more attempts to get the solution, resulting in the significantly high number of fifth and sixth trials they need to use.
- The total number of second and third trials used for each word implies the simplicity of the puzzle: Opposite to complex terms, easier puzzles require fewer trials to solve, resulting in a significantly higher distribution of total shots toward the second and third trials.
- The total number of the fifth and sixth trials (T_{5+6}) and the total number of the second and third trials (T_{2+3}) are negatively correlated, which is tested with our correlation matrix between existing variables. Given the puzzle's difficulty, the negative correlation is reasonable as only T_{5+6} or T_{2+3} will dominate within the result distribution.
- There is a strong correlation between the total number of reported results and the number of games played in Hard Mode. We observed this correlation while building our correlation matrix between the existing variables. Evidence is below (*Figure1* & *Figure2*):

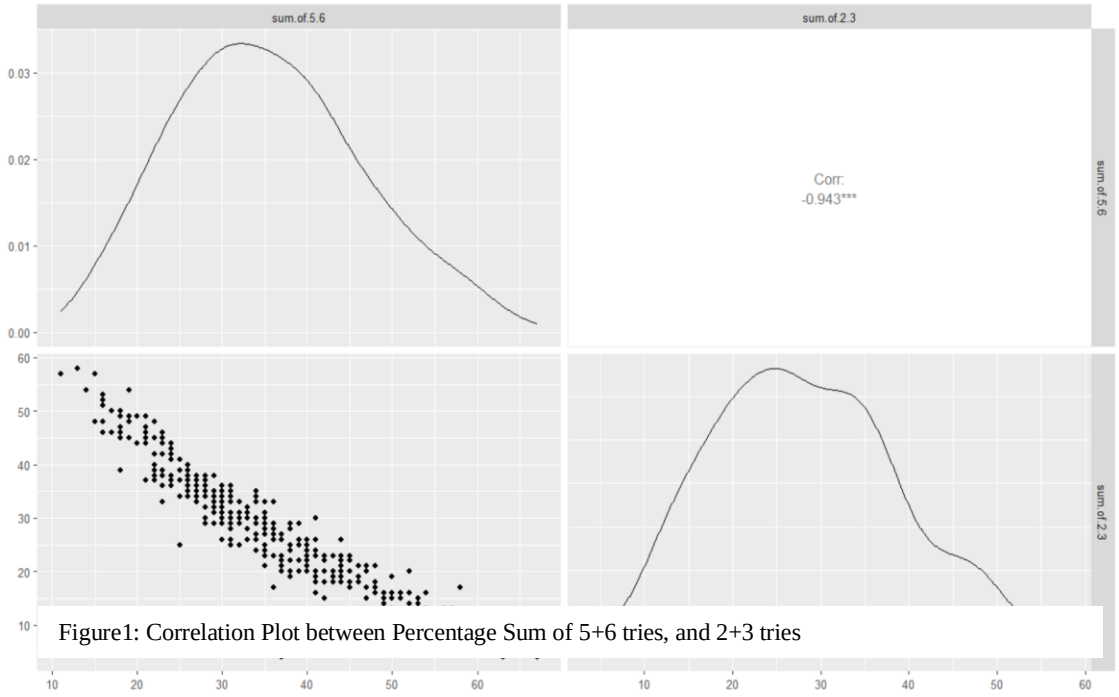

Figure1: Correlation Plot between Percentage Sum of 5+6 tries, and 2+3 tries

Team Control Number
2322810

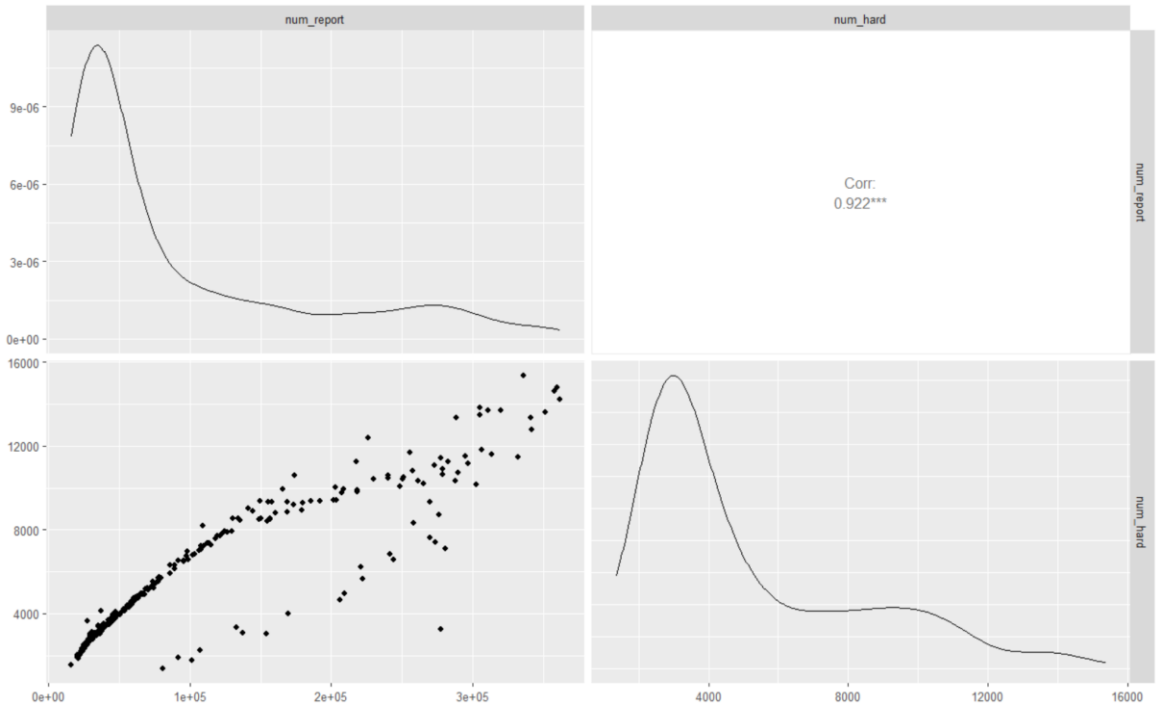

Figure2: Correlation Plot between number of reports and number of hard mode

- There are three levels of difficulty: easy, medium, and hard.
- There are no sudden changes within the predicted future of the game Worlde, such as no new features, acquisitions, or news associated with the game.

3. Models

We built three different models based on three distinguished techniques to estimate the number of reported results, the distribution of reported results, and the difficulty levels of each Wordle solution.

3.1. Variables and Parameters

3.1.1. Model 3: OLS Multiple Linear Regression

In this model, we introduce one dependent variable, and five independent variables as follows:

- Y: sum of the five and sixth trials used for each daily puzzle
- $x_1 = \frac{1}{5}\sum_{i=1}^{5} rank(letter_i)$

  To explain, x_1 is the average frequency score of each alphabetical letter appears in all possible words in the Wordle dictionary. The higher the score, the higher the frequency the letter appears in the Wordle dictionary.
- $X_2 \begin{cases} = 1 \; if \text{ there are 2 sets of duplicate letters} \\ = 0 \; otherwise \end{cases}$

  X_2 is the dummy variable for two sets of repeated letters. $X_2 = 1$ if two sets of repeated letters appear in the same word despite their consecutiveness. For example, $X_2 = 1$ if the word is “belle” as there are two sets of repeating letters: ‘e’ and ‘l.’
- $X_3 \begin{cases} = 1 \; if \text{ there are three same letters} \\ = 0 \; otherwise \end{cases}$

  If X_3 = 1, that means three same letters occurred in this word. For example, the term “fluff” contains three ‘f’ letters in which X_3 = 1 here.
- $X_4 \begin{cases} = 1 \; if \text{ one set of consecutive duplicate letter} \\ = 0 \; otherwise \end{cases}$

  X_4 = 1 if one letter repeats twice and they are next to each other. For example, X_4 for the word ‘skill’ equals one as it has ‘l’ as the repeat and consecutive.
- $X_5 \begin{cases} = 1 \; if \text{ one set of duplicate letter (must not be consecutive)} \\ = 0 \; otherwise \end{cases}$

  To explain, X_5 is also the reference variable in our model. $X_5 = 1$ only if a set of duplicated letters occurred in non-adjacent positions, e.g., “adapt.” $X_5 = 0$ if there are no repeated alphabet appeared in the word.

3.1.2. Model 1: Model Fitting

To measure the variability of the number of posted results over time, we assign our y value as the number of reported results and x as the contest number, increasing by one daily.

3.1.3. Model 2: Monte Carlos Simulation

Since Monte-Carlo Simulation is more a computation loop that will emulate how real players play the game, there are no variables included in this model.

3.2. Constraints

Based on our assumptions and the characteristics of our models, we introduce the following constraints:

- For multiple linear regression models, inputs should be an existing word in the dictionary. As the variables in our regression come from evaluating actual words from the Wordle dictionary, using words that have no meaning will skew the distribution of reported results.
- Due to the nature of the curve fitting method, it will work best with predicting parameters with minimal variance. Since we observed that the variability of the reported results wore out over time, our fitted curve will do a great job predicting the future value of the total number of reports, given the puzzle's difficulty. However, suppose there is any sudden fluctuation in the reported results due to external factors, such as in-game change features or news related to the game. In that case, we need more adjustments to the fitted function.

3.3. The Model:

3.3.1. Model 1: Difficulty of a certain word

To figure out the difficulty of a Wordle word, either already appeared or a possible future solution, we need to evaluate multiple attributes to determine the relative word hardness toward Wordle players. To most users, the answer as "input" will be more straightforward for them to solve than the word "fluff." For some words, the difficulty may be easy to distinguish, but for terms such as "slump," "crank," and "aloft," that is neither ubiquitous nor unfamiliar to players, a model to determine the hardness is necessary.

For these reasons, we analyze the words in the current data set from various perspectives to construct variables and form an ordinary least square multiple linear regression to estimate the difficulty of any possible word choice.

We noticed that the words "fluff" and "mummy" appeared in our data as the words that need the most percentage of 5 or 6 tries to solve. This rank gave us insights

into the specific letters inside the word. For example, "fluff" contains three 'f' and "mummy" contains three "m." Then, a question formed: Will this attribute affect players' accuracy in solving the Wordle game? Furthermore, other top 10 ranking words that require 5 or 6 attempts to solve are "coyly," "gawky," "booze," "forgo," "cacao," "woken," and "judge." It is easy to find that about half of them share a common feature of multiple occurrences of the same letter. For example, 'y' in "coyly," 'o' in "booze," and the word "cacao" even contains two pairs of repeated letters.

Thus, we designed a model with multiple attributes as our independent variables: average frequency of letters relative to the Wordle word dictionary (Figure 3), if a word contains two sets of repetitive letters, if a single letter occurred three times in the word, if there are two same consecutive letters in the word, and if there are two same letters in the word (not consecutive). The first variable is numerical, and all the latter four are dummy variables. Moreover, we set the dependent variable as the total percentage of using 5 and 6 tries. Taking more trials could indicate that the word is difficult and needs more attempts to solve.

We divide the repeated letter into four cases because when a player already has one letter in either green or yellow color, they are less likely to try words with the same letter at another position. For instance, if the target word is "woody" and the player tries "lodge," Wordle will give feedback with 'o' in the second position as green. Then, it will be less likely for that player to try to think of a word with multiple 'o'.

	Letter	Frequency	Rank		Letter	Frequency	Rank
1	a	975	25	14	n	573	18
2	b	280	9	15	o	753	23
3	c	475	17	16	p	365	12
4	d	393	14	17	q	29	2
5	e	1230	26	18	r	897	24
6	f	229	8	19	s	668	19
7	g	310	10	20	t	729	22
8	h	387	13	21	u	466	16
9	i	670	20	22	v	152	5
10	j	27	1	23	w	194	6
11	k	210	7	24	x	37	3
12	l	716	21	25	y	424	15
13	m	316	11	26	z	40	4

Figure3: Alphabet frequency table ranking from Wordle dictionary

Furthermore, we set the first independent variable (x1) as the average frequency of the letters relative to the frequency in the overall frequency table of letters of the Wordle dictionary (*Figure 3*) since the higher appearance of a letter in our daily usage, or the Wordle dictionary, indicates the higher probability of it occurring in the actual Wordle game.

Hence, we achieved our regression model through ordinary least square (OLS) multiple linear regression calculation:

$$y = 67.95 - 1.92x_1 + 16.86x_2 + 12.53x_3 + 8.97x_4 + 10.51x_5$$

All the $\hat{\beta}'s$ in our model are reasonable. For example, the coefficient of -1.92 for x_1 means with every one-unit increase in the average frequency of the letters in a particular word, the difficulty level will decrease by 1.92. Similarly, for the dummy variables, if the word contains any attributes from the list, then the difficulty will increase between 8.97 to 16.86.

We construct individual T Hypothesis Tests and an overall F Test to test the predicting power and if there are linear correlations between the dependent variable and independent variables. The hypothesis of the T tests is:

$$H_0: \beta_k = 0,\ H_a: \beta_k \neq 0, where\ k = 1, 2, 3, 4, 5$$

The results of the Hypothesis Testing are shown in the table below:

	Estimate	**Standard Error**	**t value**	**P-value**
x_1	-1.9161	0.2291	-8.365	1.45e-15
x_2	16.8567	4.6744	3.606	3.56e-04
x_3	12.5294	6.8645	1.825	6.88e-02
x_4	8.9739	1.6316	5.500	7.33e-08
x_5	10.5106	1.3875	7.575	3.23e-13

Table 1: Testing statistics for multiple linear regression

According to the statistics above, all x's, despite x_3, gain substantial evidence that they are helpful, and there is evidence to reject the null hypothesis even under an alpha level of 0.001. For x_3, even though the p-value is not as small as the other four explanatory variables, we can still reject the null hypothesis under the alpha level of 0.1. Of course, some people will claim that we cannot reject the null hypothesis, so we use an added variable plot (AV Plot) to analyze (*Figure 4*).

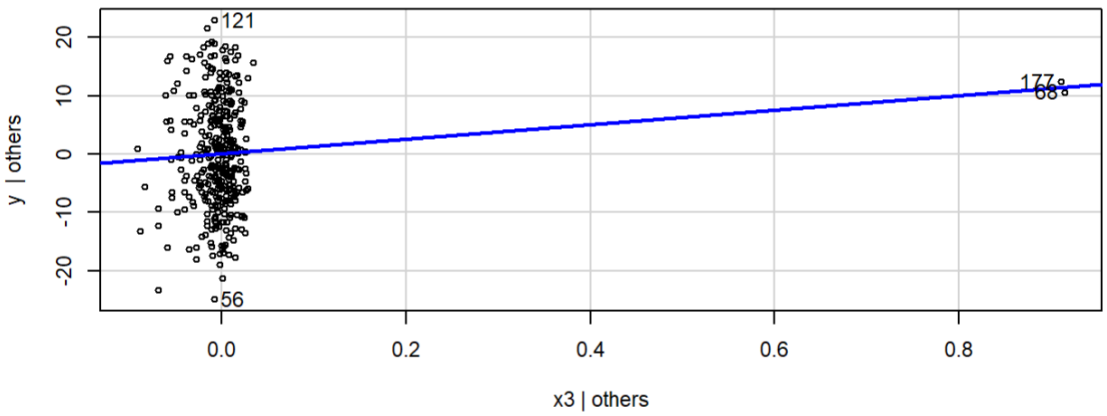

Figure 4: Added variable plot for x_3

The estimated linear regression is: $y = -3.75 \times 10^{-16} + 12.53$, and the t hypothesis testing about the β indicated there is linear correlation between x_3. If a single letter occurs three times in the word, it will affect the difficulty of the word.

While we can produce the difficulty index, y, for every input word, we still need to segment the difficulty levels. As we observed, the distribution of the dependent y variable is approximately in the normal distribution, so we set the 25% and 75% quantile line as the separation of different difficulty levels. In addition, Wordle needs to consist of neutrality and moderate difficulty to hold players, so it's reasonable that the distribution of difficulty index is normally distributed. Overall, the categorization of complexity is listed below:

$$\begin{cases} easy\ if\ 0 \le y \le 27 \\ medium\ if\ 27 < y \le 42.5 \\ hard\ if\ 42.5 < y \end{cases}$$

3.3.2. Model 2: Curve Fitting

To visualize the movement of the reported results, trend, and variation, we plotted the total number of reported results over days, which the contest number represents *(Figure 5).*

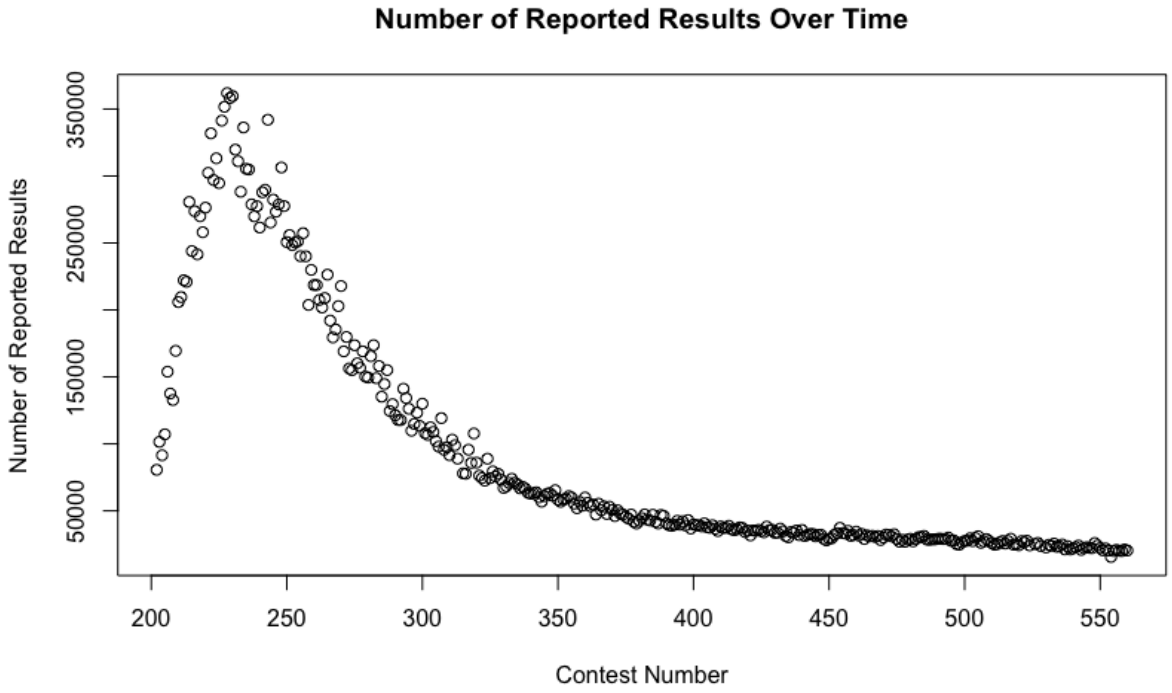

Figure 5: Plot for number of reported results against contest number

There are two strongly observed trends from this graph (*Figure 5*). The peak of the reported results was recorded on February 2, 2022, with 361,908 reported results, shortly after the New York Times announced its game purchase on January 31, 2022. Before this acquisition, the Wordle game gained popularity and millions of players per day, which explains its steep increasing slope. The game's popularity peaks at the acquisition announcement and steadily declines afterward. There was variation along the rise of its popularity. However, the decreasing trend power won over its variation, and the number of reported results daily also merged, becoming a smoothed line, especially toward the tail of this graph.

Due to the nature of this graph, where there is an abrupt change in the rising slope before February 2 and the declining slope after February 3 until April 23, which is the 30th percentile of the data, we divide this dataset into three different periods. For each period, we figured out the mathematical equations that fit the data with the help of Fitter and Scipy. Optimize libraries in Python. Using these libraries guarantees that the formulas that fit these points have the least squared residuals.

After minimizing the squared residuals, the result of this piecewise function is:

$$f(x) = \begin{cases} \frac{-2.0332e+07}{x-1.6910e+02} + 7.0981e+05 & \text{if } x \leq 228 \text{ (red)} \\ \frac{1.4217e+08}{x-5.1874e+01} - 4.6274e+05 & \text{if } 229 \leq x \leq 308 \text{ (orange)} \\ \frac{4.7515e+06}{x-2.5769e+02} + 7.0735e+03 & \text{if } x \geq 309 \text{ (green)} \end{cases}$$

There are three main functions representing three lines in this function (Figure 6):

1. Red line represents the fitted line for the reported results before February 2, when players are still interested in posting their Wordle results to Twitter.
2. Orange line represents the fitted line for the number of reported results from February 3, after its maximum of sharing, until April 23, 2022, which is the 30th percentile of this dataset.
3. The green line represents the fitted line for the reported results after April 23. To preserve the predicting power of this equation, we approximated this line so that it fits 70% of this dataset.

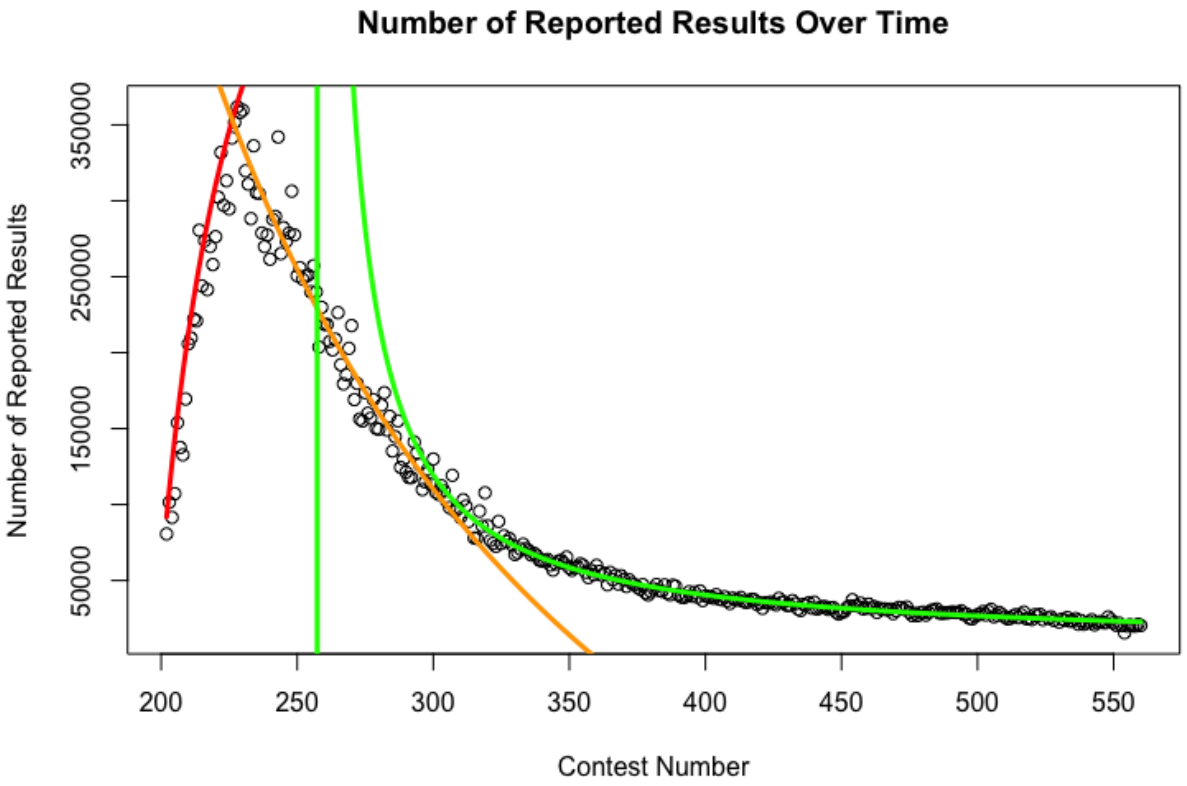

Figure 6: Plot for number of reported results against contest number with piecewise equation

From the result of this piecewise function, assuming there will be no change to the Wordle game shortly, we can use the third piece of this function to make predictions. Specifically, this model will help determine the forecast for the reported results on March 1, 2023, which we will show in the following analysis.

3.3.3. Model 3: Monte-Carlo Simulation

Another interest is to find the distribution of the number of tries in future Wordle games. Given this objective, we tried to construct several simulation models in R,

python, and java to predict the distribution. After all, we implemented the Monte-Carlo Simulation method to simulate the Wordle game.

We designed the working algorithm for the Monte-Carlo simulation. We determined the line of the process to be in the cycles of steps below:

1. Randomly select a word from the word list that contains appropriate features (e.g., a letter in the correct position if green occurred)
2. Fill out the result table after each try (use to calculate the correctness of each letter, 0 indicates gray, 1 indicates yellow, and 2 indicates green)
3. Delete all the words with letters labeled '0', gray, to avoid the random function from selecting these words.
4. Delete all the words with letters labeled '1', yellow, that are in the same position to prevent the random function from selecting these words.

Add all words with correct letters in the corresponding position to a word list, the list from which the latter loop will select guess words.

According to this algorithm, we run the loop 719,358 times, two times the maximum number of reported results in the given data set, to estimate the average future distribution of the number of tries.

Our algorithm is solely based on computational calculation, and the word list of the Wordle game is infinite to real players to some extent. Therefore, the calculated distribution of the number of tries may be some percentages skewed to the right overall. Differences between the average distribution of the number of trials and the distribution produced from simulation can be calculated, and the confidence interval can be computed to add up to the current data.

Monte-Carlo Simulation in total 719,358 run times that set random target word from 2,309 words produced the percentage distribution as follows:

	1 try	2 tries	3 tries	4 tries	5 tries	6 tries	X (fails)
%	0.32	8.59	24.64	37.38	19.19	8.98	0.92

Table 2: Percentages of number of tries from Monte-Carlo

The average of the percentage distribution of the given user reported data set is:

	1 try	2 tries	3 tries	4 tries	5 tries	6 tries	X (fails)
%	0.47	5.78	22.64	32.95	23.71	11.61	2.82

Table 3: Average of percentages of number of tries from actual dataset

To view and directly compare the difference between these datasets, the bar plot of the two sets of percentage distributions is shown below:

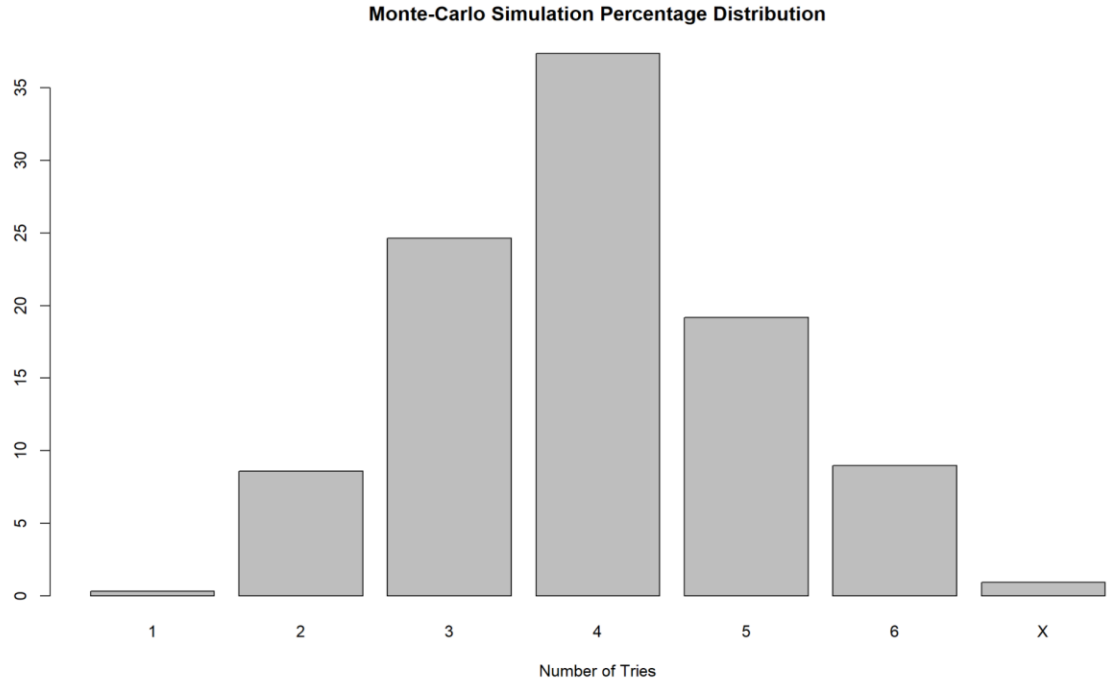

Figure 7: Bar plot of the distribution of percentages of number of tries from Monte-Carlo

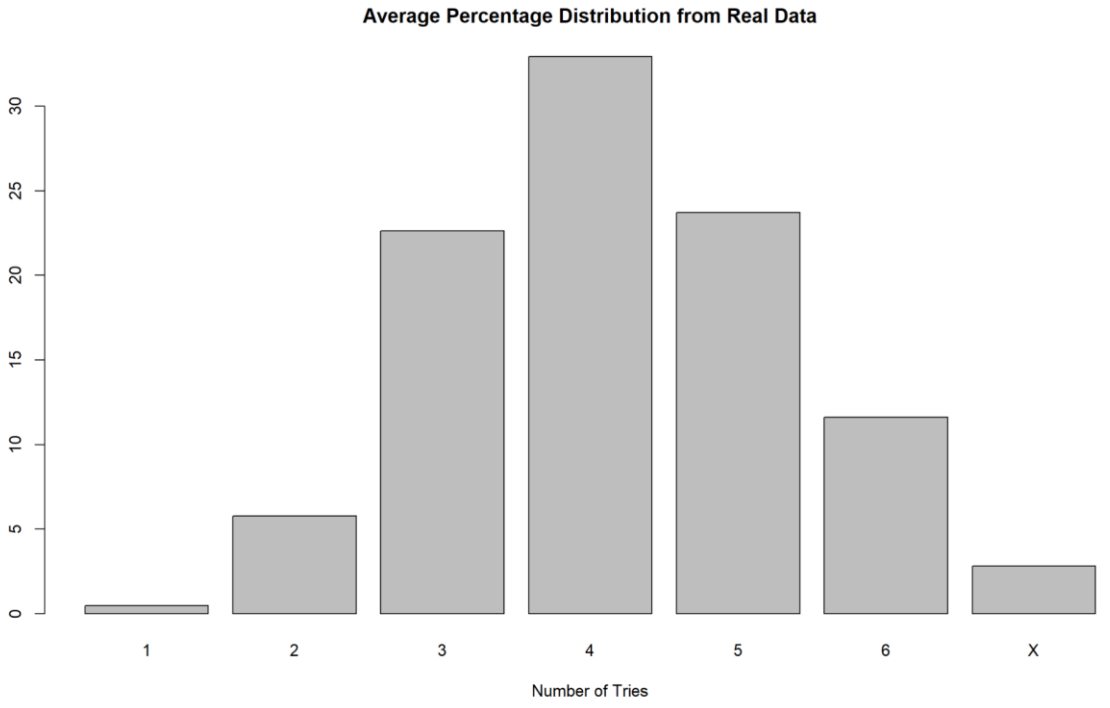

Figure 8: Bar plot of the distribution of percentages of number of tries from actual data

The percentage of category 2 tries, 3 tries, and 4 tries from the Monte-Carlo Simulation is, on average, 3.08% higher than those from the given data set. As

explained above, actual human players cannot just eliminate all the words containing 'gray' or 'yellow' letters on the same spot. Yet, scientific computation can accomplish this complex action.

As different words, or the difficulty level, will affect the number of trials real players use to solve the problem, we always need to add or minus a confidence interval based on the variability of the hardness. Therefore, we divided the difficulty model into three levels: easy, medium, and hard. We will apply this model to calculate the weighted confidence interval that needs to be added to the average distribution. The formula for the weighted average for easy and hard levels is shown below (the medium level holds constant with the original):

$$\begin{cases} 1\ tries + 0,2\ tries + 0.04, 3\ tries + 2.33, 4\ tries + 1.83, 5\ tries - 1.47, 6\ tries - 2.71, X - 0.02; easy\ level \\ 1\ tries - 0,2\ tries - 0.02, 3\ tries - 1.28, 4\ tries + 2.57, 5\ tries + 2.87, 6\ tries + 0.89, X + 0.11; hard\ level \end{cases}$$

The data is secured from controlling the random function in the Monte-Carlo Simulation only to pick the words assigned with easy and hard level, and run for 359,679 times, respectively.

4. Results

4.1. Word attributes that affect score reports played in Hard Mode

The number of reports played in hard mode can be affected by many aspects. Possible factors include: difficulty level, the number of tries the player needs to solve the puzzle, the unique interest of the player in the specific word, other personal factors.

In this case, we use available data to explore what word attributes affect the number of reports played in hard mode. With the limited given dataset, the three attributes that affect the score reports played in hard mode are the number of total reports, the number of the first trials, and the number of the fourth trials.

To first observe every possible correlation, we create a correlation matrix between variables and run a single linear regression for each variable against the number of reports posted in hard mode. The correlation matrix and our single linear regression result confirm a strong linear relationship between the number of reports in hard mode versus the number of total reports. Specifically, there is a strong positive linear relationship between these two variables. Since we can predict the number of total reports on a specific day, and the total number of reports is strongly linearly correlated with the number of reports in hard mode, predicting the number of reports in hard mode is possible.

Furthermore, our correlation matrix confirms the strong positive linear relationship between the number of first trials and the number of results posted in hard mode, indicating that players tend to share their results more frequently if they get the answer

for the hard mode on the first try. On the other hand, if players need more trials to solve a puzzle (four trials), they will tend not to share their results on Twitter.

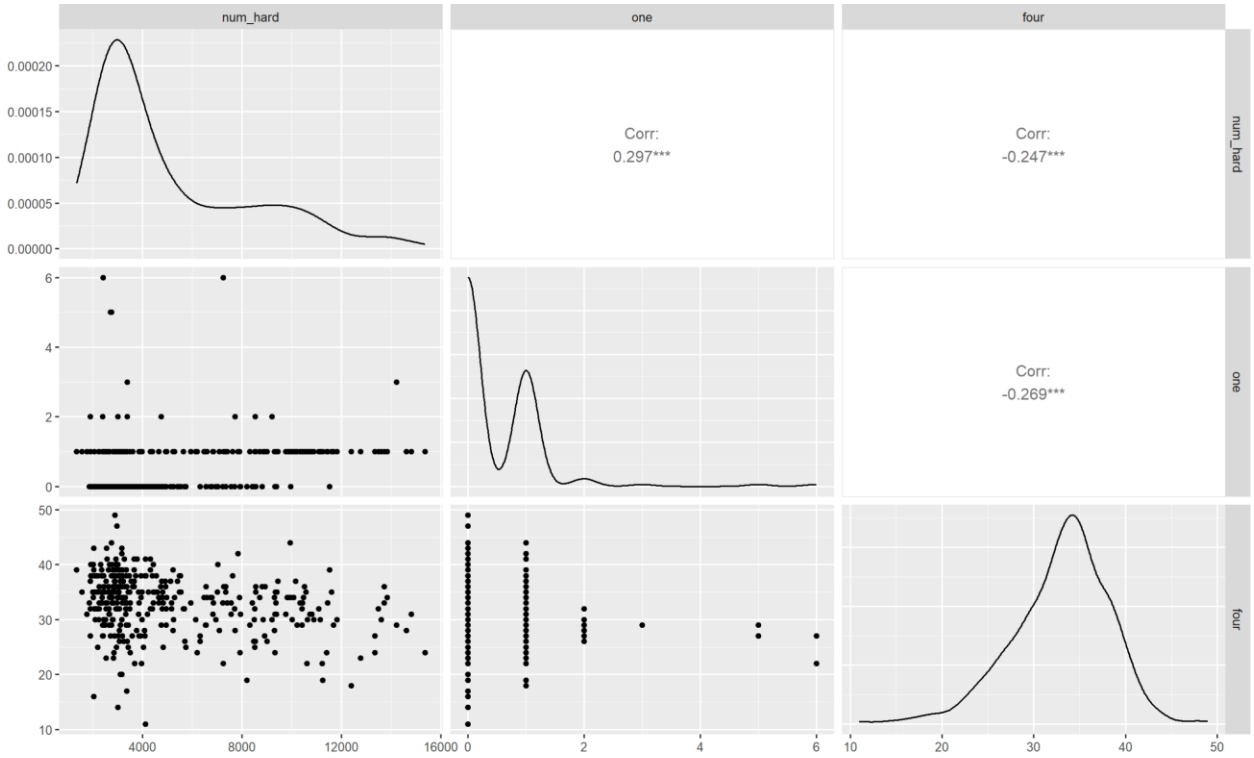

Figure 9: Correlation plot with percentage in one try, four tries, and number of reported hard model

4.2. Model predicting the distribution of reported results

On March 1, 2023, 60 days from the last day of the given dataset, December 31, 2022, the contest number will be 560 + 60 = 620. According to this equation, on March 1, 2023, the approximated number of reported results will be 20188 reports. However, this prediction will fluctuate based on the given puzzle of the day. Based on the distribution of the percentage of trials given the user reports dataset, we have the following percentage distribution on March 1, 2023:

	1 try	2 tries	3 tries	4 tries	5 tries	6 tries	X (fails)
Trials	94.88	1166.87	4570.56	6651.95	4786.57	2344.83	569.30

Table 4: Number of trials for each number of tries

Applying the formula for the weighted average for easy and hard levels, we have the confidence interval of the number of reported results will be (11024, 19548). In this essence, there will be approximately 11024 reported results if the puzzle is

challenging and 19548 reported results on March 1, 2023, if the puzzle is solvable to most players.

4.3. Classification model based on difficulties

To test and also explore our designed model for difficulty, we can plug in some future possible Wordle words as input to produce the y value, which is the difficulty index. Let's make an example with the word "eerie."

Before computing the difficulty index, we can assume that "eerie" is a challenging puzzle. This is because "eerie" has a similar structure to "fluff" and "mummy," which are calculated and categorized into the hard level based on the available data set. The calculation formula is given below:

$$y = 67.95 - 1.92x_1 + 16.86x_2 + 12.53x_3 + 8.97x_4 + 10.51x_5$$

$$y = 67.95 - 1.92 \times 24.4 + 16.86 \times 0 + 12.53 \times 1 + 8.97 \times 0 + 10.51 \times 0$$

$$y = 33.63$$

Despite our preassumption, the numeric result produced is 33.63, between 27 and 42.5. So that indicates that the word “eerie” belongs to the medium level. This result seems incompatible with our conjecture, but the letter ‘e’ is the most frequent letter in the Wordle word list that we use. Thus, the difficulty index is reasonable for our example here.

Another word example we can use to test the regression model is “elate.” Before experimenting with our numerical model, we can first analyze the letters. The letter ‘e’ is frequently used, and “late” is a word that often appears daily. Then, we have some evidence to predict it will fall into the easy level. This calculation process is below:

$$y = 67.95 - 1.92x_1 + 16.86x_2 + 12.53x_3 + 8.97x_4 + 10.51x_5$$

$$y = 67.95 - 1.92 \times 24 + 16.86 \times 0 + 12.53 \times 0 + 8.97 \times 0 + 10.51 \times 0$$

$$y = 21.96$$

The difficulty index we retrieved is certainly a small one, which is smaller than the measure line 27. This result also corresponds to our presumption. Therefore, we can conclude that “elate” is an easy puzzle.

4.4. Wordle game solving strategy

As the Monte-Carlo Simulation requires the creation of a computing algorithm by us, we also concluded a set of logic that players can use to solve the Wordle game. Some essential guidelines or suggestions that players should consider as listed below:

- Try to find a word with five different letters at the first trial

- Try to use more consonants in the first trial than vowels: there are only five vowels, so they are easier to guess; For example, “v_w_l” is more straightforward to guess than “_o_e_.”
- If there are green letters, try to think of words structure beyond these letters
- If there are yellow letters, but you are unsure of their correct position, try to think of words with different consonants first. Elimination of letters sometimes works better than consistently trying to fit one single letter.
- If letters such as ‘e’ and ‘c’ appear in the fourth position, think of words ending with “er” and “ch.” Letter combinations frequently appear in the word list.

5. Reflection

5.1. Difficulty Regression Model

As we refer to various research papers about the difficulty of a text, we found Natural Language Processing, a text analysis system. According to this system, we can also develop a similar program, or assessment model, to process the familiarity of a specific word. Potential aspects of being considered include the frequency of appearances on social media, articles, or research papers. Suppose a word appears more in research papers rather than social media. In that case, the word might be less familiar to the public since more people spend more time on social media than reading professional papers.

Word familiarity systems could include metrics such as overall letter frequency and whether the word appears in a formal or informal context. Besides, more respect is needed to explore. If we construct the familiarity system, we can also include this as a variable in our difficulty regression model and thus enhance the overall prediction accuracy.

Moreover, the difficulty level may change depending on the number of trials the player is currently at. For example, if a player guesses "spear" on the first try and the target words are "eerie" and "elder" (we have two target words here because we want to compare the variation in the difficulty level with respect to the number of trials), which are both in hard difficulty level. The feedback will be 'xx!x!' and 'xx!xo'. For the "eerie" case, we know one 'e' is inside, so we are less likely to think of words with multiple 'e.' In contrast, the feedback for "elder" is more straightforward. We have one 'e' and 'r' and do not need to think about duplicate letters. Thus, we should include a new variable contributing to the variation of difficulty with respect to the number of trials.

5.2. Monte-Carlo Simulation Model

The Monte-Carlo Simulation consists of a complete and reliable loop of finding appropriate word guesses. However, some optimizations are still applicable to our

model. For example, most times, real players can only eliminate some but not all words with gray or yellow letters in the wrong position. A possible improvement would be not to delete all the words that did not meet the criteria and retain some to make up noise (error) manually.

A potential improvement is to include an algorithm that can automatically test letter combinations such as ‘er,’ ‘ch,’ and ‘ve.’ In this case, the probability of success will increase in some cases when the program detects there may be evidence of words with letter combinations.

5.3. Future Prediction Model

As mentioned earlier, we used a curve-fitting method with the help of computational algorithms such as Scipy. Optimize and the Fitter library in Python to minimize the squared residuals. Therefore, we are confident that our curve has the least squared residuals compared to the given dataset. Furthermore, as seen in Figure 5, the variance of the number of reported results wears out over time, leading to minimal fluctuations in our prediction. Our prediction from a curve fitting method with a prediction interval based on the difficulties of a given word is reasonable. However, a risk may occur if there may exist a function that can fit the model better than our existing model. Furthermore, since curve fitting will make a great prediction when there is low variability, if the variance of the number of predicted results suddenly increases due to any circumstance, we will need to adjust our calculation for this prediction.

6. Conclusion

In this study, we constructed an OLS multiple linear regression, a Monte Carlo Simulation model, and a system of piecewise equations to describe and predict the current and future trend of the number of reports. We found that these models all interact, or at least correlate, with each other. That means if the New York Times company sets a word with low difficulty, in which the simulated distribution of the number of trials will be skewed to the right, more players will share results on social media. As we observed a clear pattern of decreasing number of reports, the New York Times company may intend to increase the Wordle players' social media sharing by various efforts to promulgate the game. Thus, setting the word in easy mode will likely boost the number of reports.

Moreover, we discovered that on December 25th, Christmas, the number of reports experienced a sudden decrease. As our guess, an important holiday will force everyone to focus on their personal lives, reducing interaction with Wordle. A possible strategy to mitigate the decrement is incorporating holiday themes like "merry" or "Santa" into the game. Through the word-of-mouth communication, this action will help alleviate the substantial decrease that may happen in the future.

In addition, we designed some solving gambits for Wordle players to win the game with higher efficiency. The strategy is slightly different from our Monte-Carlo Simulation's algorithm but is a more accessible way for regular players.

Overall, our models are significantly useful based on hypothesis testing and can be used to predict future behaviors of the game Wordle and Wordle's players. Nevertheless, there are still various aspects that we can improve on, just as written in the reflection section, and they are all potential topics and issues that we aim to address in future investigations.

7. Letter

Dear New York Times,

Under comprehensive data analysis and modeling, we strongly acknowledge that Wordle is a popular and tactful word game. As evidence, the maximum daily number of Twitter players who report their game stats is 359,679. And that's the only part of players who share the results. However, we also noted a gradual decrease in reports as the day passed. In our analysis, we built models to detect the future potential distribution of percentages of tries, what attributes of a word will affect difficulty level, and the future data trend of the number of reports if everything holds constant. These models not only help to analyze given data but also help to improve the Wordle game.

Firstly, games that always maintain the difficulty level of either hard or easy will lose their market or existing users. Players consistently find themselves solving the problem at a fast speed or can not reach the solution daily, will quit the game and find alternatives. Of course, a game cannot only include moderate/medium difficulty. Adding easy or hard sometimes will boost the player's interest. Thus, the multiple linear regression model to estimate the difficulty level is essential while randomly generating the solution for that day. Setting up the probability of manually picking easy, medium, and challenging levels can reserve the current players and even attract new users.

Secondly, the mathematical equation used to estimate the general future data about the number of reports shared indicate a continuously decreasing trend. As the declining rate is decelerating, it is an excellent opportunity for the business to plan some events to attract new players. Possible ideas such as April Fool's Theme and Easter Event might sound enchanting to the public, especially those who use social media often. This idea appeared as we noticed a significant decrease in the reported results on December 25th, 2022, which is Christmas, which is reasonable because people want to engage in the festival with family and friends rather than solve a word puzzle. However, things will change if there is a unique Christmas UI or related event.

In addition, the Wordle game now only has one mode to play with. It will be necessary for the developer team to construct new modes for players. Otherwise, maintaining the same mode to guess a 5-letter word will be more varied for players. New modes could include timed mode, in which there is a 60-second countdown for every trial for the player to add some tense feeling. Another mode that developers can consider is self-design words. Players can customize their words and send them to their friends to guess the answer. In this way, new users will flow into the pool and thus expand the spread of the game.

In conclusion, despite Wordle's game popularity, it needs some improvements to maintain or regain its popularity. Our suggestions are purely based on the aspect of statistical analysis. However, when incorporated with real business consultants and developer teams, we believe it will help the New York Times to improve the Wordle game and bring it back to its peak popularity.

8. Reference

[1] *5 Letter Words - Word Unscrambler*. (n.d.). https://www.wordunscrambler.net/words/5-letter

[2] B. (2022a, February 5). *Three Million Wordle Tweets Later - Towards Data Science*. Medium. https://towardsdatascience.com/three-million-wordle-tweets-later-3d3af23bd5c7

[3] B. (2022b, February 22). *Peak Wordle & Word Difficulty - Towards Data Science*. Medium. https://towardsdatascience.com/peak-wordle-word-difficulty-64907be4c177

[4] B. (2022c, November 3). *What I Learned from Playing More than a Million Games of Wordle*. Medium. https://towardsdatascience.com/what-i-learned-from-playing-more-than-a-million-games-of-wordle-7b69a40dbfdb

[5] Broz, M. (2023, January 22). *Wordle Statistics, Facts, & Strategies (2023)*. Photutorial. https://photutorial.com/wordle-statistics/

[6] C. (n.d.). *Time-Series Analysis on Cinema's Ticket Used and Total Sales*. https://rstudio-pubs-static.s3.amazonaws.com/858323_f81ac21852fc4859a151a981fcd6e18a.html

[7] Chalabi, M. (2022, February 21). *What the numbers tell us about the best – and worst – Wordle start words*. The Guardian. https://www.theguardian.com/games/2022/feb/21/wordle-starting-words-numbers-probability

[8]*Facebook - Meld je aan of registreer je*. (n.d.-a). Facebook. https://analyticsindiamag.com/top-10-r-packages-for-natural-language-processing-nlp/

[9]*Facebook - Meld je aan of registreer je*. (n.d.-b). Facebook. https://towardsdatascience.com/the-cost-of-a-bad-start-in-wordle-how-to-avoid-it-2cdd63f8d599

[10]J., & J. (2023, January 5). *How Many People Play Wordle in 2023? (User & Growth Stats)*. Fiction Horizon. https://fictionhorizon.com/how-many-people-play-wordle/

[11]*R: Calculate readability*. (n.d.). https://search.r-project.org/CRAN/refmans/quanteda.textstats/html/textstat_readability.html

[12]*RPubs - Time Series Analysis - Forecast box office sales*. (2018, October 22). https://rpubs.com/bennylee_uts/431647

[13]Rul, C. V. D. (2021, December 12). *[NLP] Basics: Measuring The Linguistic Complexity of Text*. Medium. https://towardsdatascience.com/linguistic-complexity-measures-for-text-nlp-e4bf664bd660

[14]Russell, T. (n.d.). *What Your Wordle Score Says About You*. The Blue & Gold. https://chambleeblueandgold.com/11679/featured-posts/what-your-wordle-score-says-about-you/

[15]Sensor Tower. (n.d.). *State of Word Games: Rising Revenue, Emerging Markets*. https://sensortower.com/blog/mobile-word-game-category-insights

[16] Statista. (n.d.). *Word Games - Worldwide | Statista Market Forecast*. https://www.statista.com/outlook/dmo/app/games/word-games/worldwide

[17] *What makes a Wordle word hard?* (2021, September 27). David Waldron. https://www.waldrn.com/what-makes-a-wordle-word-hard/

[18] *Word Familiarity and Frequency*. (2018, June 9). DeepAI. https://deepai.org/publication/word-familiarity-and-frequency

9. Appendix

```
library(stringi)
library(ggplot2)
library(carData)
library(car)
library(hrbrthemes)
library(GGally)
library(readxl)
Problem_C_Data_Wordle <- read_excel("Problem_C_Data_Wordle (version 1).xlsb.xlsx",
                    col_types = c("numeric", "numeric", "text", "numeric", "numeric", "numeric",
                     "numeric", "numeric", "numeric", "numeric", "numeric", "numeric", "numeric",
                     "numeric", "numeric", "numeric", "numeric", "numeric",  "numeric", "numeric"
                    , "numeric", "text"))
#import solution
solution <- read.table("~/solutions.txt", quote="\"", comment.char="")
#data deleted date and ID number
data = Problem_C_Data_Wordle
data = data[,3:20]attach(data)
colnames(data) <- c('word', 'num_report', 'num_hard', 'one', 'two', 'three', 'four', 'five',
                    six', 'seven','sum_of_5+6','sum_of_2+3','none3','none4','mean','medium','sd','skw')
colnames(all_count) <- c("Letter", "Frequency","Rank")
n <- length(data$word) #sample size of given data
n_all <- length(solution[,1]) #sample size of dictionary
#frequency table of letters of given words
count <- matrix(data = c(c('a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z'),
                    rep(0, 26)), nrow = 26, ncol = 2)
count <- data.frame(count)
count[,2] <- as.numeric(count[,2])
alphabet = c('a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z')
for (k in 1:n) {
 for (i in 1:5){
  for (j in 1:26){
   s1 <- stri_sub(data$word[k],i,i)
   if (s1 == alphabet[j]) {
   count[j,2] = count[j,2] + 1
   }
  }
 }
}
all_count <- matrix(data = c(c('a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z'), rep(0, 26)), nrow = 26, ncol =
all_count <- data.frame(all_count)
all_count[,2] <- as.numeric(all_count[,2])
#frequency table of letter from all words
for (k in 1:n_all) {
 for (i in 1:5) {
  for (j in 1:26) {
   s1 <- stri_sub(solution[k,1],i,i)
   if (s1 == alphabet[j]) {
    all_count[j,2] = all_count[j,2] + 1
   }
  }
 }
}
all_count <- cbind(all_count,rank(all_count[,2]))
#change to numerical data
data <- data.frame(data)
data[,2] <- as.numeric(data[,2])
data[,3] <- as.numeric(data[,3])
data[,4] <- as.numeric(data[,4])
data[,5] <- as.numeric(data[,5])
data[,6] <- as.numeric(data[,6])
data[,7] <- as.numeric(data[,7])
data[,8] <- as.numeric(data[,8])
data[,9] <- as.numeric(data[,9])
data[,10] <- as.numeric(data[,10])
data[,18] <- as.numeric(data[,18])

#percentage
perc <- data$num_hard/data$num_report
summary(perc)
plot(num_hard,num_report)
hard_report <- lm(num_report~num_hard)
resi <- rstandard(hard_report)
plot(num_hard, resi, main = 'Standardized Residual for Num_report ~ Num_hard', xlab = 'Number of Hard Mode', ylab = 'Standardized Residual')

#MLR
reg <- lm(num_report~num_hard+one+two+three+four+five+six+seven, data = data)

# ------ Monte Carlo ------- #
counter <- 1
dist_table <- matrix(data=0, nrow = 1, ncol = 7)
len_list = length(solution[,1])
result_matrix <- matrix(data = NA, nrow = 1, ncol = 5) #FIXED
solution_copy <- solution

while (counter < 50){
 #updates
 solution_copy <- solution
 signal <- FALSE #indicate where to stop (ignore the following steps)
 result_matrix <- matrix(data = NA, nrow = 1, ncol = 5) #FIXED
 new_list <- c()

 target_word <- 'eerie'
 #target_num <- sample.int(len_list,1)
 #target_word <- solution[target_num, 1] #get a random word
 #while (is.na(target_word)) {
  #target_num <- sample.int(len_list,1)
  #target_word <- solution[target_num, 1]
 #}

 # ------------TRIAL LOOP------------#
 for (cyc in 1:6) {
 first_num <- sample.int(len_list,1)
 first_word <- solution_copy[first_num, 1]
 while (is.na(first_word)) {
  first_num <- sample.int(len_list,1)
  first_word <- solution_copy[first_num, 1]
 }
 if (first_word == target_word) {
  print(target_word)
  dist_table[1,1] <- dist_table[1,1] + 1
  signal <- TRUE
 }
 if (signal == FALSE) {
  #check letters
  for (i in 1:5) {#for guess
   result_matrix[1, i] <- 0
   for (j in 1:5) {#for target
    if (grepl(substring(first_word,i,i), substring(target_word,j,j))) {
     result_matrix[1,i] <- 1
    }
    if (i == j) {
     if (grepl(substring(first_word,i,i), substring(target_word,j,j)))  {
      result_matrix[1,i] <- 2
     }
    }
   }
  }
  true_letter <- c()
  for(i in 1:5) {
   if(result_matrix[i] == 2) {
    true_letter <- append(true_letter, substring(first_word,i,i))
    true_letter <- append(true_letter, i)
   }
  }
  #delete all the words contain grey letters in dictionary
  for (j in 1:n_all) {#all words in dictionary
   for (i in 1:5) {#loop for letters in result matrix
    if (result_matrix[i] == 0){
     if (is.na(solution_copy[j,1]) == FALSE) {
      if (grepl(substring(first_word,i,i), solution_copy[j,1])) {
       solution_copy[j,1] <- NA
      }
     }
    }
   }
  }
  for (i in 1:len_list) {
```

Team Control Number
2322810

```
      }
    }
  }
 }
 for (i in 1:len_list) {
  for (j in 1:5) {
   if (result_matrix[j] == 1) {
    if (is.na(solution_copy[i,1]) == FALSE && is.na(first_word) == FALSE) {
     if (substring(solution_copy[i,1],i,i) == substring(first_word,j,j)) {
      solution_copy[i,1] <- NA
     }
    }
   }
  }
 }
 #secure words with letters in right position
 new_list <- c()
 if (length(true_letter) != 0) {
  for (k in 1:n_all) {
   if (is.na(solution_copy[k,1]) != TRUE) {
    len_true <- length(true_letter)/2
    if (len_true == 1) {
     p1 <- as.numeric(true_letter[2])
     if (grepl(substring(solution_copy[k,1],p1,p1), true_letter[1])) {
      new_list <- append(new_list, solution_copy[k,1])
     }
    }
    if (len_true == 2) {
     p1 <- as.numeric(true_letter[2])
     p2 <- as.numeric(true_letter[4])
     if (grepl(substring(solution_copy[k,1],p1,p1), true_letter[1]) &&
         grepl(substring(solution_copy[k,1],p2,p2), true_letter[3])) {
      new_list <- append(new_list, solution_copy[k,1])
     }
    }
    if (len_true == 3) {
     p1 <- as.numeric(true_letter[2])
     p2 <- as.numeric(true_letter[4])
     p3 <- as.numeric(true_letter[6])
     if (substring(solution_copy[k,1],p1,p1) == true_letter[1] &&
        substring(solution_copy[k,1],p2,p2) == true_letter[3] &&
        substring(solution_copy[k,1],p3,p3) == true_letter[5]) {
      new_list <- append(new_list, solution_copy[k,1])
     }
    }
    if (len_true == 4) {
     p1 <- as.numeric(true_letter[2])
     p2 <- as.numeric(true_letter[4])
     p3 <- as.numeric(true_letter[6])
     p4 <- as.numeric(true_letter[8])
     if (substring(solution_copy[k,1],p1,p1) == true_letter[1] &&
        substring(solution_copy[k,1],p2,p2) == true_letter[3] &&
        substring(solution_copy[k,1],p3,p3) == true_letter[5] &&
        substring(solution_copy[k,1],p4,p4) == true_letter[7]) {
      new_list <- append(new_list, solution_copy[k,1])
     }
    }
    if (len_true == 5) {
     p1 <- as.numeric(true_letter[2])
     p2 <- as.numeric(true_letter[4])
     p3 <- as.numeric(true_letter[6])
     p4 <- as.numeric(true_letter[8])
     p5 <- as.numeric(true_letter[10])
     if (substring(solution_copy[k,1],p1,p1) == true_letter[1] &&
        substring(solution_copy[k,1],p2,p2) == true_letter[3] &&
        substring(solution_copy[k,1],p3,p3) == true_letter[5] &&
        substring(solution_copy[k,1],p4,p4) == true_letter[7] &&
        substring(solution_copy[k,1],p5,p5) == true_letter[9]) {
      new_list <- append(new_list, solution_copy[k,1])
     }
    }
   }
  }
  len_new_list <- length(new_list)
 }
 counter = counter+1
}
}
# ------ ENDS ------ #
plot(Problem_C_Data_Wordle$...1[2:360],num_hard , xlab = 'Date', ylab = 'Number of Hard Mode')
abline(v=44592, col = 'green', lwd = 2)
#num report against date
plot(Problem_C_Data_Wordle$...1[2:360], num_report)
plot(Problem_C_Data_Wordle$...1[2:360], log(num_report))
```

```
#regression for difficulty
#y <- skewness
#x1 <- frequency of single letter in all words
#x2 <- continuous two same letters; 1 <- there is consecutive same letter; 0<- no consecutive same letter
#x3 <- 
#x3 <- textstat_readability

#y <- skw
y <- none1

x1 <- c()
for (i in 1:n) {
 score1 <- all_count$`rank(all_count[, 2])`[all_count$X1==substring(word[i],1,1)]
 score2 <- all_count$`rank(all_count[, 2])`[all_count$X1==substring(word[i],2,2)]
 score3 <- all_count$`rank(all_count[, 2])`[all_count$X1==substring(word[i],3,3)]
 score4 <- all_count$`rank(all_count[, 2])`[all_count$X1==substring(word[i],4,4)]
 score5 <- all_count$`rank(all_count[, 2])`[all_count$X1==substring(word[i],5,5)]
 sumofeach <- sum(score1, score2, score3, score4, score5)
 sumofeach <- sumofeach/5
 x1 <- append(x1, sumofeach)
}
x1 #frequency mean
x2 <- c() #2 set of 2, despite consecutive or not
for (i in 1:n) {
 tab <- table(rle(strsplit(word[i],"")[[1]]))
 wlen <- length(tab[1,])
 count_for_repeat <- 0
 for (j in 1:wlen) {
  if (tab[1,j] == 2) {
   count_for_repeat = count_for_repeat + 1
  }
  if (length(tab[,1]) > 1) {
   if (tab[2,j] == 1) {
    count_for_repeat = count_for_repeat + 1
   }
  }
 }
 if (count_for_repeat == 2) {
  x2 <- append(x2, 1)
 } else {
  x2 <- append(x2, 0)
 }
}
x3 <- c()
for (i in 1:n) {
 tab <- table(rle(strsplit(word[i],"")[[1]]))
 wlen <- length(tab[1,])
 count_for_repeat <- 0
 for (j in 1:wlen) {
  if (tab[1,j] == 3) {
   count_for_repeat = count_for_repeat + 1
  }
  if (length(tab[,1]) > 1) {
   if (tab[1,j] == 1 && tab[2,j] == 1) {
    count_for_repeat = count_for_repeat + 1
   }
  }
 }
 if (count_for_repeat == 1) {
  x3 <- append(x3, 1)
 } else {
  x3 <- append(x3, 0)
 }
}
x4 <- c() #consecutive letters
for (i in 1:n) {
 lengs <- length(table(rle(strsplit(word[i],"")[[1]]))[,1])
 if (lengs > 1) {
  if (x3[n] == 1 || x2[n] == 1) {
   x4 <- append(x4, 0)
  } else {
   x4 <- append(x4, 1)
  }
 } else {
  x4 <- append(x4, 0)
 }
}
x5 <- c() #2 same letter not consecutive
for (i in 1:n) {
 lengs <- length(table(rle(strsplit(word[i],"")[[1]]))[,1])
 wlen <- length(table(rle(strsplit(word[i],"")[[1]]))[1,])
 count_for_repeat <- 0
 if (lengs == 1) {
  for (j in 1:wlen) {
   if (table(rle(strsplit(word[i],"")[[1]]))[1,j] == 2) {
```

Team Control Number
2322810

```
    if (table(rle(strsplit(word[i],"")[[1]]))[1,i] == 2) {
      count_for_repeat = count_for_repeat + 1
    }
   }
   if (count_for_repeat == 1) {
    if (x3[n] == 1 || x2[n] == 1 || x4[n] == 1) {
     x5 <- append(x5, 0)
    } else {
     x5 <- append(x5, 1)
    }
   } else {
    x5 <- append(x5, 0)
   }
  } else {
   x5 <- append(x5, 0)
  }
}

regre <- lm(y~x1+x2+x3+x4+x5)
summary(regre)
avPlots(regre)
ggpairs(data.frame(y,x1,x2,x3,x4,x5))


#NQQ plot indicating not normal -> why?
qqnorm(num_report, pch = 1, frame = FALSE)
qqline(num_report, col = "steelblue", lwd = 2)
qqnorm(num_hard, pch = 1, frame = FALSE)
qqline(num_hard, col = "steelblue", lwd = 2)


#difficulty for ocean
oceansum <- 0
for (i in 1:5) {
 numb <- all_count$`rank(all_count[, 2])`[all_count$X1 == substring('ocean',i,i)]
 oceansum = oceansum + numb
}
oceansum/5
 
regre$coefficients[1] + (oceansum/5)*regre$coefficients[2]

barplot(c(0.32, 8.59, 24.64,37.38,19.19,8.98,0.92),main = 'Monte-Carlo Simulation Percentage Distribution', xlab = 'Number of Tries', names.arg
c('1','2','3','4','5','6','X'))
barplot(c(0.47, 5.78, 22.64,32.95,23.71,11.61,2.82),main = 'Average Percentage Distribution from Real Data', xlab = 'Number of Tries', names.ar
c('1','2','3','4','5','6','X'))
eeriesum <- 0
for (i in 1:5) {
 numb <- all_count$`rank(all_count[, 2])`[all_count$X1 == substring('eerie',i,i)]
 eeriesum = eeriesum + numb
}
eeriesum/5

elatesum <- 0
for (i in 1:5) {
 numb <- all_count$Rank[all_count$Letter == substring('elate',i,i)]
 elatesum = elatesum + numb
}
elatesum/5
regre$coefficients[1] + regre$coefficients[2]*(elatesum/5)
```

```
os.chdir('/Users/nguyencaogiakhanh/Desktop/2023_MCM-ICM_Problems')

# response = requests.get('https://reqbin.com/echo', timeout=100)
# print(response.status_code)

df = pd.read_excel('Problem_C_Data_Wordle.xlsx')
# df = df.iloc[1:332, 1:]
df = df.iloc[1:,1:]
df.columns = ['date','fre','word','num_port','num_hard', 'one', 'two', 'three', 'four', 'five', 'six', 'seven']
df.set_index('date')

# df[['date', 'num_port']].plot()
num_port = df['num_port'].values
# *225/ sum(df['num_port'])
# num_port = num_port.values
# [a,b] = scipy.stats.beta.fit(num_port)
# print(a)
# print(b)
# df.fit(num_port)
# @timeout(100)
# num_port = num_port.float()
# x = df['fre'].values
# y = df['num_port'].values

def func(x, a, b,c,d):
    return ((a/(x+b))+c)

def erf(x, a, b, c, d):
    return d + 0.5*c*(1 + special.erf(a*(x-b)))

x = df['fre'].astype(float)
x = x.values[1:]

y = df['num_port'].astype(float)
y = y.values[1:]

print(np.quantile(x, 0.3))
print(np.quantile(y, 0.3))
# plt.plot(x, y)

# 1
A_0=1.11295300e+07
B_0=-1.99542359e+02
C_0=1.2
D_0 = 1

popt,pcov = curve_fit(func,x[1:251],y[1:251],p0=[A_0,B_0,C_0, D_0])
print(pcov)
print(popt)
residuals = y- func(x, *popt)



popt1,pcov1 = curve_fit(func,x[252:331],y[252:331],p0=[A_0,B_0,C_0, D_0]) #(70th percentile)
print(pcov1)
print(popt1)

popt2,pcov2 = curve_fit(func,x[332:],y[332:],p0=[A_0,B_0,C_0, D_0])
print(pcov2)
print(popt2)

# print(parameters)
# print(popt,pcov)
plt.plot(x,y,label='Data')
plt.plot(x,func(x, *popt),'.',label='Fit')

plt.plot(x,func(x, *popt1),'.',label='Fit1')

plt.plot(x,func(x, *popt2),'.',label='Fit2')

# plt.plot(x, (1.11295300e+07/(x+-1.99542359e+02)), 'o', label = 'try')
plt.legend()
plt.show()



# pltt.plot(0.0003173298898368*np.exp(0.999784739819231*x) -0.0003173298889815032*np.exp(0.999784739819239*x))

# rng = np.random.default_rng()
# y_noise = 0.2 * rng.normal(size=x.size)
# yd = y + y_noise
# plt.plot(x, y, 'b-', label='data')
# popt, pcov = curve_fit(func, x, y)
# popt
# parameters, covariance = curve_fit(func, x, y)
# fit_A = parameters[0]
# fit_B = parameters[1]
# print(fit_A)
# print(fit_B)


# f = Fitter(num_port, distributions='decay exponential')
# f.fit()
# f.summary()

# df.beta.fit(num_port)
# plt.hist(data.iloc)

# hypothesis test for frequency used in real life
# maybe: harder the word, if solved, get posted
```