The Game of Life: Analyzing the Impact of Initial Conditions on Game Duration

Group 120, William Brennan, Project Topic 13

Abstract

John Conway's Game of Life, known for its simple rules, produces a wide variety of behaviors, from quick extinction to perpetual patterns. Despite being created in the 1970s, it continues to captivate mathematicians, programmers, and educators due to its insights into complexity from simplicity.

This study analyzes how varying initial setups affect the longevity, population variance, and outcomes of the game. Using Monte Carlo simulation, we test different matrix sizes and initial percentages of alive cells across multiple runs. Key metrics, such as average lifespan, time to extinction, and population stability, are used to evaluate the effects of these starting conditions. The goal is to identify optimal conditions for sustained activity and determine those that lead to short-lived or stagnant evolution. Our findings show a statistically significant difference in the average duration of the game based on initial conditions, highlighting the impact of randomness in complex systems.

Background

Overview

The Game of Life (GoL), created by mathematician John Conway in 1970, is a well-known cellular automaton often described as a "zero-player game."[5]. Unlike traditional games, it requires no user interaction once the initial state is set. No joysticks, buttons, or manuals are necessary—just a grid and a

set of rules. Despite their simplicity, these rules often result in complex and varied patterns. The game's evolution unfolds automatically based on the following rules [3]:

- For a space that is populated
    - Each cell with one or no neighbors dies, as if by solitude
    - Each cell with four or more neighbors dies, as if by population
    - Each cell with two or three neighbors survives
- For a space that is empty or unpopulated
    - Each cell with three neighbors becomes populated

Birthed from a simple set of rules, the GoL can produce a remarkable variety of behaviors, ranging from repeating patterns and moving structures to intricate and chaotic designs. In contrast, the same rules can also result in sequences that completely die out within just a few generations. The next state of an individual cell is determined purely on its current state and the state of each neighbor, evolving as time and generations pass [6]. This endless variability has inspired a passionate community of enthusiasts, mathematicians, and programmers who continue to explore the game's possibilities decades after its creation.

Conway identified three primary classes of stable behavior: still life (unchanging configurations), oscillators (patterns that repeat in cycles), and spaceships (structures that move across the grid). Even today, new structures are being discovered as researchers and hobbyists test an infinite number of initial conditions [5]. Although the Game of Life can generate a remarkable range of patterns—from gliders and blinkers to highly complex structures—the final state of any simulation typically falls into one of three categories: extinction, steady state, or eternal continuation [4].

- Extinction occurs when all cells die out and no further evolution is possible.
- Steady state refers to a configuration that remains constant

- Eternal continuation refers to patterns that evolve indefinitely, often through dynamic, moving structures like gliders, spaceships or oscillators.

One of the most intriguing aspects of the Game of Life is how different initial conditions influence its outcomes. Are certain configurations more likely to produce long-lived patterns, stabilize quickly, or evolve indefinitely? This question continues to inspire experimentation and raises deeper inquiries into complexity, emergence, and predictability within seemingly simple systems. The focus of this work is to simulate and analyze a range of matrix sizes and initial percentages of live cells to understand their effects on the duration and dynamics of the game.

Main Findings

Program Description

This program aims to determine whether varying initial conditions in Conway's Game of Life lead to statistically significant differences in the average number of generations produced. The analysis evaluates multiple matrix sizes, ranging from 10×10 to 100×100 in increments of 10, and initial alive cell densities ranging from 5% to 95% in 5% increments.

For every matrix size and alive cell percentage combination, random initial configurations are generated in each iteration to ensure unbiased simulation runs.

Developed in Python using Visual Studio Code, the program is organized into three main modules:

- **Simulation**: Contains functions that implement the core logic of Conway’s Game of Life.
- **Analysis**: Contains functions that perform statistical analysis on the simulation outputs.
- **Visualization**: Contains functions that generate charts and graphs based on both raw and analyzed data.

The “requirements.txt” file contains the necessary Python packages the program is built on.

One Jupyter Notebook file is used to run the preliminary simulations, analyze variance and confidence intervals, and execute the final simulations using optimized sample sizes. Based on the initial analysis, sample sizes are calculated for each condition to reduce the width of the confidence intervals. These optimized sample sizes are then used in the final simulation phase. The resulting data is analyzed to determine whether a statistically significant difference exists in the average number of generations across the tested conditions.

Code Documentation

Simulation Module

- **generate_rand_matrix(rows, cols, p_ones)**
    - Creates a binary matrix of size (rows × cols) where a proportion p_ones of cells are alive (1) and the rest are dead (0), randomly distributed. Used to simulate initial conditions.
- **get_neighbors(mat)**
    - Builds a dictionary mapping each cell’s coordinates to a list of values of neighbors
- **next_gen(mat)**
    - Generates the next state of the matrix using John Conway’s Game of Life rules. Applies birth, survival, and death logic based on sum of neighbor to return the new matrix.
- **matrix_data_collection(mat, gen_num, run_num, collection_dict, alive_percent, matrix_size)**
    - Records the number of alive cells and simulation description data at a given generation. Updates a shared dictionary with values for later aggregation and analysis.
- **run_game(mat, run_num, matrix_size, master_data_collection_dict, alive_percent, max_steps=1000)**

- Simulates Conway's Game of Life from a given matrix until reaching a steady state, all cells are dead, or the maximum number of steps is reached. Logs generation-level and run-level data into a shared dictionary.

- **sample_runs(number_of_sample_per_matrix_size_and_percent_alive, max_steps)**
  - Executes a fixed number of simulations for every combination of matrix size (10×10 to 100×100) and initial alive percentage (5% to 95%). Returns a DataFrame with merged generation and run data across all trials.
- **final_run(dicts, max_steps)**
  - Runs simulations using optimized sample sizes per condition (matrix size and initial percentage alive), as determined from prior analysis. Returns a DataFrame of merged results for final evaluation.

Analysis Module

- **cohens_f(f_stat, df_groups, df_obs)** [1,7]
  - Calculates Cohen's *f*, a measure of effect size for ANOVA. Takes the F-statistic and degrees of freedom for groups and observations to compute the proportion of variance explained and return the corresponding effect size.
- **anova_analysis(df)** [8]
  - Performs one-way ANOVA for each unique matrix size in the dataset to test if the mean number of generations differs significantly between initial percent alive groups. Also calculates Cohen's *f* for each matrix size and returns a summary DataFrame containing the F-statistic, p-value, and Cohen's f.
- **calc_matrix_size_and_initial_percent_alive_group_stats(df)**
  - Computes descriptive statistics including mean, standard deviation, standard error, and 95% confidence interval for the number of generations per matrix size and initial percent

alive group based on simulation results. Returns a sorted DataFrame by confidence interval width.

- **required_n_per_group(grouped_stats, min_sample_size ,half_width=10, confidence=0.95)** [10]
    - Estimates the sample size required for each group to achieve a confidence interval width no larger than a specified value (default: ±10), given the group's standard deviation and desired confidence level. Returns a dictionary with keys of matrix size and initial percent alive with required sample sizes.
    - Selects the appropriate critical value based on the current sample size:
        - If n >= 30, the z-statistic is used.
        - If n < 30, the t-statistic with appropriate degrees of freedom is used.
- **sample_v_final_stat_comparison(sample, final)**
    - Compares group statistics between sample and final simulation results. Calculates and returns deltas for means, standard deviations, standard errors, and confidence interval widths to observe and evaluate the impact of using optimized sample sizes.

Visualization Module

- ***show_end_state_circle_chart(df)***
    - Creates a donut-style pie chart showing the distribution of how simulations terminated:
        - **dead**– all cells died out,
        - **steady"**– reached a steady state,
        - **max** – hit the maximum number of allowed generations.
- ***show_random_sample_runs(df)***

- Selects 6 random simulation runs with more than one generation and plots line charts of alive cell counts over generations. Subplots are labeled with matrix size, initial alive percentage, and termination reason.

- ***show_heat_map(df)***[9]

  - Generates a heatmap showing the average number of generations until termination for each combination of matrix size and initial percent alive groups
    - matrix size (y-axis)
    - Initial alive cell percentage (x-axis)

- ***show_initial_percent_alive_ci_avg(df)***

  - Calculates and plots the average 95% confidence interval width for each matrix size and initial percent alive group. Gives a sense of variability and across trials per group.

- ***show_ci_delta(df, top_bottom, n)***

  - Plots a bar chart of the top or bottom n matrix size and initial alive percentage group based on largest position of negative change in confidence interval width between sample and final simulation

- ***difference_in_average_ci_percent_alive_groups(***df,df2)
  - Compares the difference between average confidence interval width of the sample outputs (df) to the average confidence interval width of the final outputs (df2)
  - Plots a bar chart for ease of comparison as well as a table of results

This modular approach ensures that each component is reusable and testable, while also allowing for flexible configuration of initial parameters such as matrix size and the percentage of live

cells. Only a subset of the defined functions is required for the simulation to run, enabling customized configurations tailored to specific goals.

Running the Program

Running the Game of Life simulation program is straightforward. The following steps outline the core process to execute both the initial sampling phase and the final simulation using calculated sample sizes:

1. **Set Up the Environment**
    a. Create a new virtual environment using either venv or conda.
    b. Install all required Python packages using the requirements.txt file:
        i. pip install -r requirements.txt
2. **Import Required Functions** From the simulation and analysis modules, import the following functions:
    a. generate_rand_matrix, get_neighbors, next_gen, matrix_data_collection, run_game
    b. sample_runs
    c. calc_matrix_size_and_initial_percent_alive_group_stats
    d. required_n_per_group
    e. final_run
3. **Run the Simulation**
    a. **Initial Sampling:**

    Call sample_runs() with your desired number of samples per group and the maximum number of generations for each run.

        i. sample_data = sample_runs(samples_per_group=10, max_generations=100)

b. **Calculate Group Statistics:**

   Pass the simulation results to calc_matrix_size_and_initial_percent_alive_group_stats() to compute group mean, standard deviation, variance, and 95% confidence intervals.

   i. stats = calc_matrix_size_and_initial_percent_alive_group_stats(sample_data)

c. **Determine Sample Sizes:**

   Feed the statistics into required_n_per_group() to calculate how many runs are needed to tighten the confidence intervals to an acceptable width.

   i. required_sample_sizes = required_n_per_group(stats)

d. **Final Simulation:**

   Run final_run() with the optimized sample sizes to generate final output.

   i. final_run(required_sample_sizes)

4. Optional: Visualizations and Analysis

You can use any of the available visualization and analysis functions by passing in the outputs from either the sample or final simulation. While these are not essential to running the simulation, they are useful for exploring and presenting the results.

Output Analysis

Sample Run Output

To better understand the behavior of different matrix sizes and initial percentages of live cells, 15 samples were generated for each combination, with a maximum generation limit of 100. The simulation results were surprising, with an overwhelming majority of runs reaching this maximum. Nearly 70% of all runs terminated due to hitting the generation cap, suggesting that many configurations sustain ongoing activity without settling into a steady state or extinction. Without diving deeper into the group-specific

behavior, the system appears relatively stable based on this initial sample. Figure 1.1 presents the breakdown of end-state results across the 191 independent simulation runs.

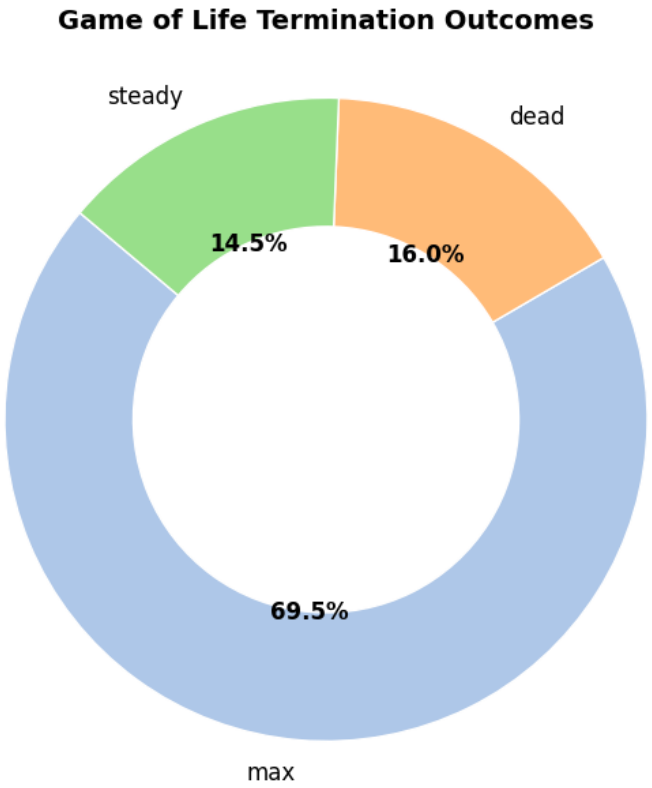

Figure 1.1: Proportion of Termination Outcomes

Taking a small sample of individual runs and visualizing their population over time reveals intriguing patterns. Some samples, as shown in Figure 1.2, demonstrate a relatively sharp decline in live cell count shortly after the simulation begins, eventually stabilizing as the generations progress. Others exhibit minimal change throughout the course of the simulation, suggesting more resilient or balanced configurations. This initial observation could be an indication of clear and distinct behavioral difference between group configurations.

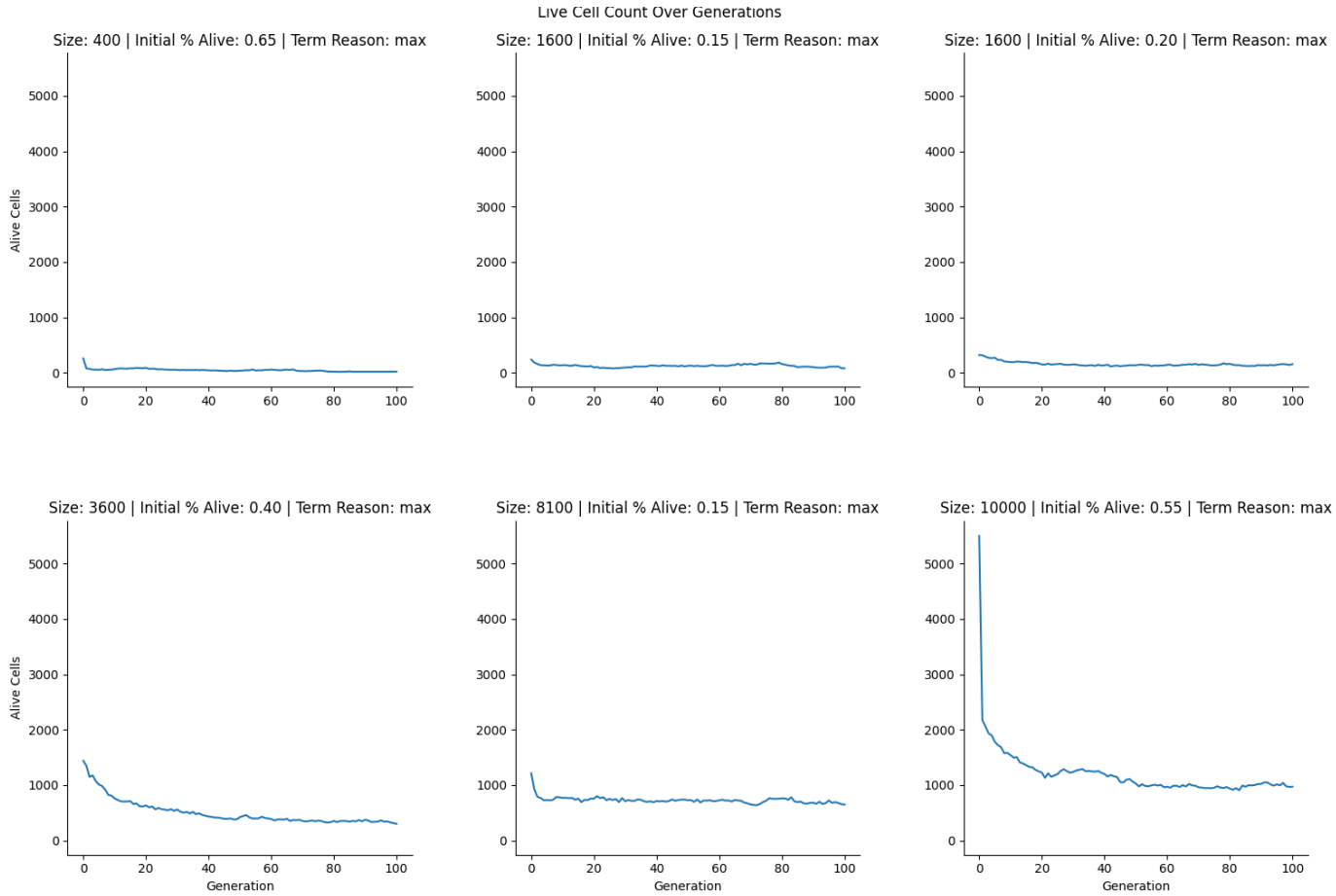

Figure 1.2: Sample of population behavior

Taking a closer look at the group level statistics, the samples reveal notable variance and wide confidence intervals across several configurations. Preliminary analysis suggests a trend where groupings with either very high or very low initial percentages of live cells tend to exhibit greater variability. This may be explained by two distinct phenomena: patterns may die out quickly due to insufficient neighboring cells to sustain growth; emergent structures may continue evolving for longer periods due to ample space and fewer early collisions, allowing for prolonged transformation and survival. These behaviors could contribute to the observed variability within those groups.

Matrix Size	Initial % Alive	Mean	Standard Deviation	Sample Size	Standard Error	95% Confidence Interval
1600	0.05	43.06	48.15	15	12.43	26.66
2500	0.05	62.6	47.42	15	12.24	26.26
10000	0.85	43.93	47.40	15	12.24	26.25
2500	0.8	45.4	46.25	15	11.94	25.61
4900	0.8	69.33	44.89	15	11.59	24.86

3600	0.8	43.26	44.60	15	11.51	24.70

Table 1.1: Showing the top 6 groups with the widest confidence interval

The theory of high and low initial percentage of alive cells effecting variability seems to hold, until we reach an initial percentage of 85% and greater. Above these values, variability seems to sharply decline, as shown in Figure 1.3. Infact, when looking at the average 95% confidence interval width by initial percentage alive groupings, there seems to be a clear correlation between confidence interval width and initial percent alive groupings. To get a more accurate gauge of the impact of our test groupings have on the length of the simulation, we will attempt to tighten the confidence interval for groups with high variance.

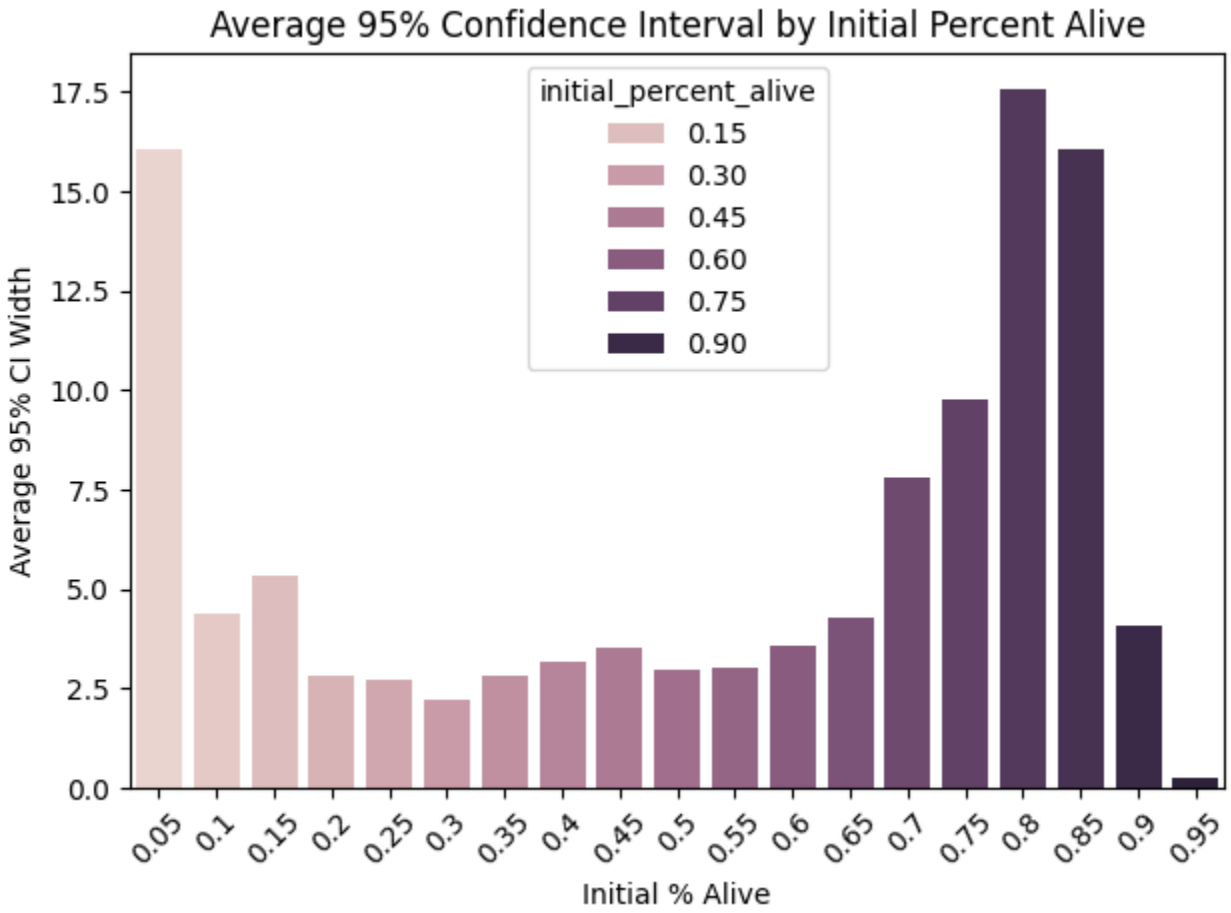

Figure 1.3: Average Confidence Interval Width by Initial Percent Alive

As matrix sizes increase, the likelihood that a simulation reaches the maximum generation limit also rises. For example, the average number of generations for most 10x10 matrix configurations is 100,

the maximum allowed, across a majority of initial percent alive groupings. This suggests that larger matrices are more likely to sustain ongoing activity and avoid both extinction and stabilization when compared to smaller matrices which tend to terminate more quickly. As shown in figure 1.4.1 and 1.4.2, you can see the clear increase in average number of generations as matrix size increases.

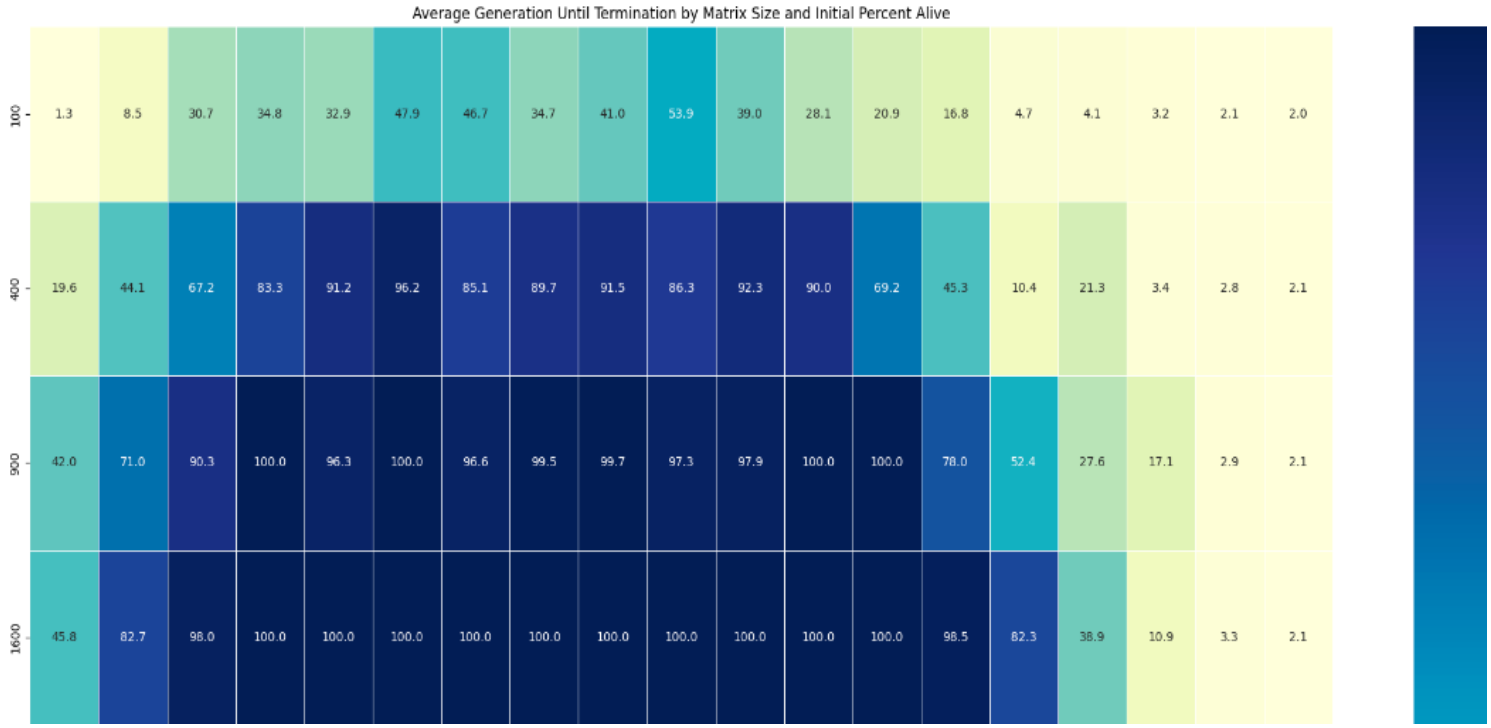

Figure 1.4.1: Average Generation Length by Matrix Size and Initial Percent Alive (truncated due to size)

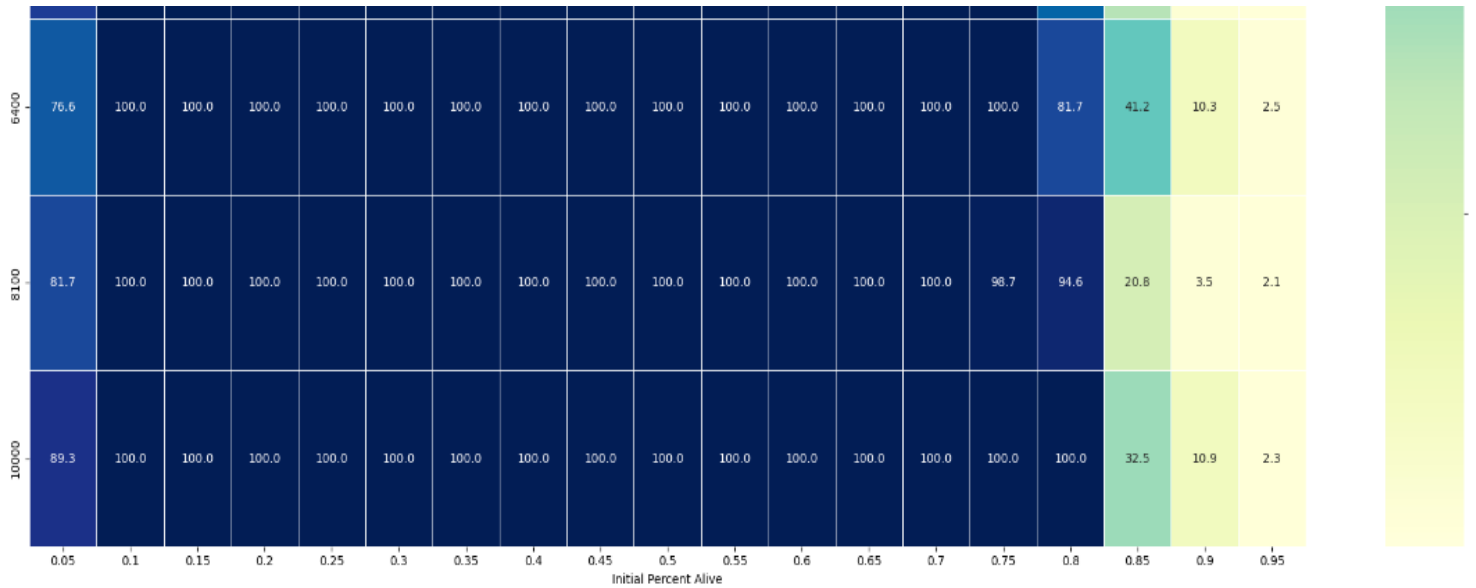

Figure 1.4.2: Average Generation Length by Matrix Size and Initial Percent Alive (truncated due to size)

To reduce the width of the confidence intervals and obtain a more accurate comparison between groupings, we use the sample size estimation formula for a population mean. This approach allows us to determine how many simulation runs are required per group to achieve a desired level of precision. The following formula is used to determine the optimal sample size, using the t-statistic if the sample size is below 30 and the z-statistic if above. With the half-length being the desired half-width confidence interval for the group. [10]

- n = ((z or t)*standard deviation/half length)^2

To retain consistency across groupings—particularly those with tighter confidence intervals in the initial sample—we will collect the same number of samples for each group as used in the original simulation run. With the updated sample sizes determined per group, we will rerun the final simulation to assess whether the confidence interval widths have decreased. Ultimately, this will allow us to compare the groupings more accurately and evaluate whether there is a statistically significant difference between matrix size and initial percent alive combinations.

Matrix Size, Initial % Alive Group	Required Sample Size
4900, .85	109
10000, 0.85	108
900, 0.75	104
2500, 0.05	100
1600, 0.8	97
3600, 0.05	95
100, 0.15	94

Table 1.2: showing required number of samples need per group to tighten confidence interval (truncated due to size)

Final Run Output

Increasing sample size for groups with wide confidence intervals resulted in reduced variance. Of the 190 matrix size and initial percent alive groupings, 67 saw a tighter confidence interval when compared to the initial sample, 95 groups experienced no change and 28 saw a wider confidence interval. Groups whose confidence interval widen are of much intrigue. One potential explanation is that

the initial sample size was insufficient to capture the full range of variance for those specific groupings. If the variance was not accurately reflected in the first set of samples, additional data might have revealed previously unobserved variability, causing the confidence intervals to widen. This suggests that the initial sample may have failed to represent the true distribution of the data for certain groupings, leading to a less accurate estimate of the population parameters. As shown in Tables 2.1 and 2.2, there does not appear to be a clear pattern based on either matrix size or initial percent alive, suggesting that the widening of confidence intervals may not be systematically related to these factors.

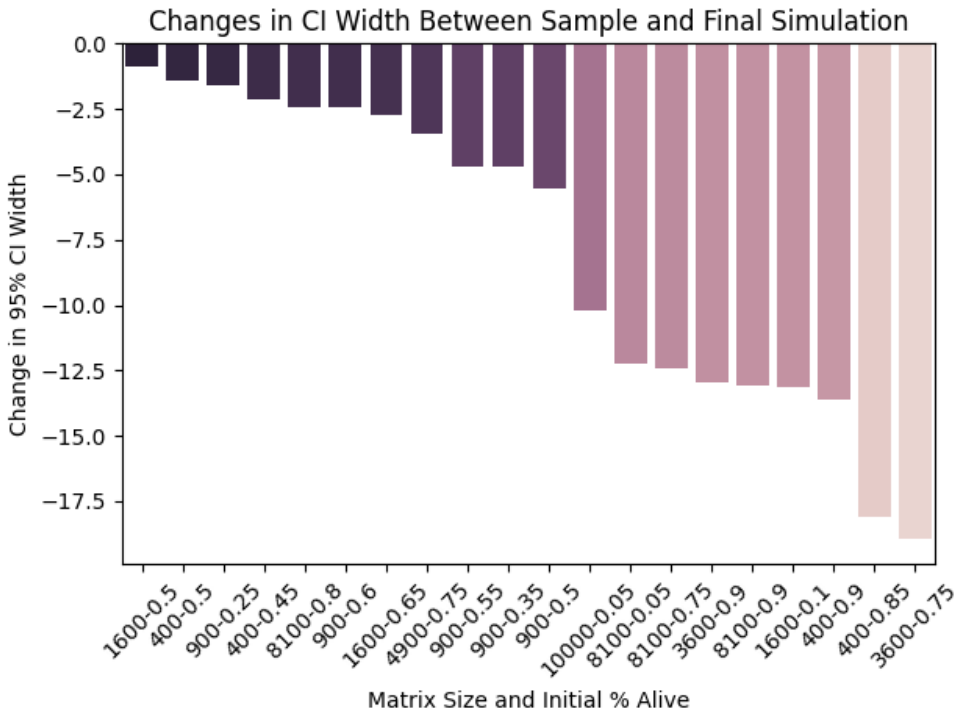

Figure 2.1: Shows 20 largest changes by grouping between sample run CI and final run CI. Calculate by (sample ci – final ci)

Matrix Size	Count of Groups
100	3
10000	1
1600	3
2500	1
3600	3
400	4

4900	3
6400	1
8100	4
900	5

Table 2.1: Count of matrix size groups with increase ci width

Initial Percent Alive	Count of Groups
0.05	3
0.1	1
0.25	1
0.35	1
0.45	1
0.5	3
0.55	1
0.6	1
0.65	1
0.75	3
0.8	2
0.85	2
0.9	5
0.95	3

Table 2.2: Count of initial percent alive groups with increase ci width

Overall, the variance within the system decreased as a result of the more optimized sample sizes. As shown in Figure 2.2, each initial percent alive grouping experienced a tighter confidence interval, with the exception of the 95% group, which showed a marginal increase of just 0.004. These results suggest that the system has become more stable and that overall variance has been reduced. With this improvement in precision, we are now positioned to conduct an ANOVA to determine whether there are statistically significant differences between groups, particularly when analyzing which conditions are associated with longer durations before extinction or the emergence of steady-state structures.

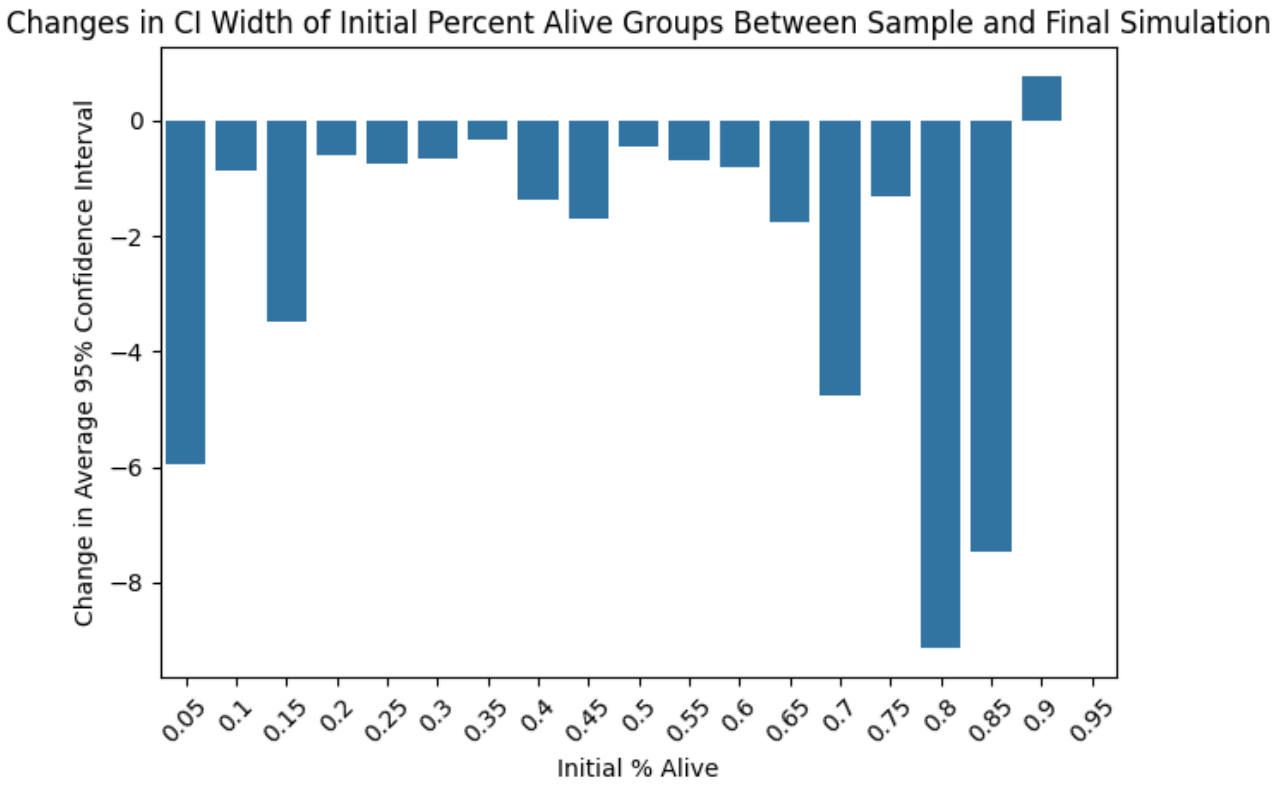

Figure 2.2: Change in average 95% CI of initial percent alive groups between sample and final simulation. Calculated by (final average ci - sample average ci)

Conducting a one-way ANOVA analysis will help us determine if varying the initial starting conditions of the simulations influences the duration of each run [8]. ANOVA analysis was conducted for each matrix size to determine if initial percentage alive has a significant impact. Table 2.3 summarizes the results from the ANOVA analysis, and the results are conclusive.

Matrix Size	F-Statistic	P Value	Cohen's F
100	10.55	6.84E-27	0.51
400	22.48	4.60E-57	0.78
900	33.33	4.45E-74	1.09
1600	31.33	2.14E-70	1.06
2500	32.57	7.08E-71	1.12
3600	20.28	6.82E-47	0.91
4900	33.86	8.83E-74	1.12

6400	38.65	1.26E-77	1.28
8100	34.41	1.68E-66	1.32
10000	73.92	6.51E-119	1.76

Table 2.3: ANOVA analysis results for matrix size groupings

For every matrix size test, the ANOVA analysis reveals significant differences in the impact initial percent of alive cells has on the duration. Each of the group's p-value is far less than .01, which indicates differences between duration of the simulation are very likely to occur from the starting conditions rather than by chance. The positive correlation between matrix size and F-statistic tells us that the difference between the initial percentage alive group averages of simulation duration become more apparent as matrix size increases [11]. Lastly, the interpretation of Cohen's F helps us measure the magnitude of the effect between the different groups. A value above .4 generally indicates there is a very large and substantial difference between the average duration between our groupings [1,12].

Conclusion

This project implements John Conway's Game of Life under various starting conditions of different matrix sizes and how many cells are initially alive. Through initial sampling, we were able to better understand how each configuration of the game behaved by analyzing variance, duration means and 95% confidence intervals. Once behavior was better understood, we utilized sample size optimization to determine the number of samples needed to reduce confidence interval width. Once obtained, we ran a final simulation and performed ANOVA analysis to determine if there was a significant difference between our groupings when analyzing average simulation duration.

The ANOVA results reveal that there are indeed significant differences between the groups, suggesting that initial conditions, such as matrix size and the percentage of alive cells, have a substantial effect on the duration of the simulation before steady-state or extinction is reached. This conclusion emphasizes the importance of initial conditions in shaping the long-term behavior of the system.

Challenges

The primary challenge in this project was identifying which metrics would be most meaningful for analysis and determining how to implement the appropriate statistical methods. As a developer with a stronger background in programming than in mathematics, I found the statistical analysis portion required significantly more time, effort, and research to ensure accuracy and relevance.

Another notable challenge during the early stages of simulation development. Initial versions of the code were not optimized for performance, leading to long testing times and difficulty in making rapid enhancements. Wasteful loops and inefficient indexing caused very long run times. Refactoring for efficiency was necessary to allow for larger-scale simulations and iterative development. Although not fully optimized, the current code is a vast improvement from the original version. Additionally, I would have increased the initial sample sizes used in the experiment to better gauge group variance. I believe doing so would have resulted in more compelling conclusions.

Future

I believe this project could be expanded in several directions. Investigating further into population size metrics, shape and structure analysis as well as exploring larger matrix sizes would all add to the depth of the overall analysis. Diving into predictive tools such as linear and logistic regression based on starting conditions would also be an interesting development. Additionally, identifying certain starting conditions that can be used to model real world systems for various industries. For example, creating a

model for how energy is transferred in grid networks to try to identify weak points and potential system improvements.

Works Cited

1. Northwestern University. (n.d.). *Sample size and power* [PDF]. Preventive Medicine, Northwestern University. https://www.preventivemedicine.northwestern.edu/docs/applied-statistics-presentation-materials/sample-size-and-power-presentation.pdf
2. IBM. (n.d.). *Monte Carlo simulation*. IBM. https://www.ibm.com/think/topics/monte-carlo-simulation
3. Zimmer, C. (2020, December 28). *How the Game of Life explained the universe*. The New York Times. https://www.nytimes.com/2020/12/28/science/math-conway-game-of-life.html
4. Lipa, M. (n.d.). *Conway's Game of Life: Cellular Automata*. Cornell University. http://pi.math.cornell.edu/~lipa/mec/lesson6.html
5. Hu, Y., Li, J., Liu, H., Li, J., & Wang, K. (2016). *Exploring the computational complexity of the Game of Life and its variants*. NCBI. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4905947/
6. Benassi, M., Cagnoni, S., & Lanzi, P. L. (2016). *Evolving interesting initial conditions for cellular automata of the Game of Life type*. ResearchGate. https://www.researchgate.net/publication/286938720_Evolving_Interesting_Initial_Conditions_for_Cellular_Automata_of_the_Game_of_Life_Type
7. Jim. (n.d.). *Cohen's d: Effect size*. Statistics by Jim. https://statisticsbyjim.com/basics/cohens-d/
8. Scribbr. (2023). *One-way ANOVA: Definition, examples, and how to run it*. https://www.scribbr.com/statistics/one-way-anova/
9. DataCamp. (2023). *Seaborn heatmaps: A detailed guide*. https://www.datacamp.com/tutorial/seaborn-heatmaps
10. GeeksforGeeks. (2023, February 2). *Sample size formula*. https://www.geeksforgeeks.org/sample-size-formula/
11. Elsevier. (n.d.). *F-statistics*. ScienceDirect Topics. Retrieved April 22, 2025, from https://www.sciencedirect.com/topics/mathematics/f-statistics
12. Statistics How To. (n.d.). *Cohen's f statistic: Definition, formulas, and examples*. Statistics How To. Retrieved April 19, 2025, from https://www.statisticshowto.com/cohens-f-statistic-definition-formulas/