Workshop No.4 — Systems Engineering Analysis

COVID-19 Global Forecasting

Kaggle Competition

Santiago Vargas Gomez - 20242020139
David Esteban Sanchez Torres - 20221020093
Dilan Guisseppe Triana Jimenez - 20221020100
Daniel Alejandro Castro - 20242020271
Course: Systems Analysis & Design
Professor: Carlos A. Sierra

November 8, 2025

1 Data Preparation

The data preparation stage builds upon the analytical foundations (constraints, sensitivity and chaos) established in Workshop 2 and incorporates the updated ingestion module developed for the current system implementation. With this module update, the goal is to ensure that all datasets used in the forecasting and simulation tasks are validated, cleaned, and standardized according to the system requirements defined earlier, fulfilling the essential requirements for the system's design in order to maintain reliability, traceability, and consistency across all subsequent modules of the architecture.

The ingestion module developed for this workshop serves as the foundation for all subsequent data processing, modeling, and simulation tasks. It achieves its goal by eliminating the structural, temporal, and semantic inconsistencies identified in Workshop 2, where the dataset was shown to contain missing values, irregular cumulative counts, noise introduced by delayed reporting, and schema inconsistencies across regions. Through this section, we will explain in detail how each component of the ingestion code works and how it reduces the system's constraints.

It is important to take into account that the dataset used in this workshop corresponds to the COVID-19 Global Forecasting Week 1 competition, including cumulative confirmed cases and fatalities across multiple countries and regions.

1.1 File Verification and Traceability

The ingestion module begins by verifying the physical existence of the two required input files provided by the Kaggle platform, `train.csv` and `test.csv`. If either file is missing, the system halts execution to prevent downstream failures.

To ensure complete traceability, the module computes a SHA-256 hash of each file using the `file_hash()` function.

1.2 Safe File Loading

The `safe_read_csv()` function loads each dataset with robust parsing rules that minimize data corruption by doing the next steps:

- automatic date parsing when a `Date` column is present,
- standardized interpretation of empty or null values,
- protection against malformed entries by avoiding strict parsing.

These mechanisms directly mitigate the "noisy or inconsistent data" constraint that we observed in Workshop 2 through the analysis of the multiple entries that contained missing or incorrectly encoded values.

1.3 Schema Validation

Once the files are loaded, the `validate_columns()` function ensures that all required columns exist. The system accepts both primary schemas and alternative schemas to accommodate regions where `Id` or `ForecastId` may be missing.

This step is essential given that Workshop 2 identified schema irregularities as a major source of instability: several rows had missing province information, and some datasets omitted identifier fields entirely. By forcing schema consistency at this stage, the system prevents structural failures during modeling and guarantees alignment with the Kaggle submission format.

1.4 Text Normalization and Missing Province Handling

To address missing and inconsistent geographic identifiers, the ingestion module applies a normalization step:

- missing `Province/State` values are replaced with empty strings,
- all text fields are stripped of trailing spaces,
- non-string values are converted to strings.

Workshop 2 revealed that approximately half of the rows contained missing province information, leading to ambiguity in regional grouping. Additionally, the distribution of confirmed cases in the training dataset exhibits strong imbalance, where most observations are close to zero while a small number of regions show extreme values. This reinforces the importance of applying robust cleaning rules before any modeling stage.

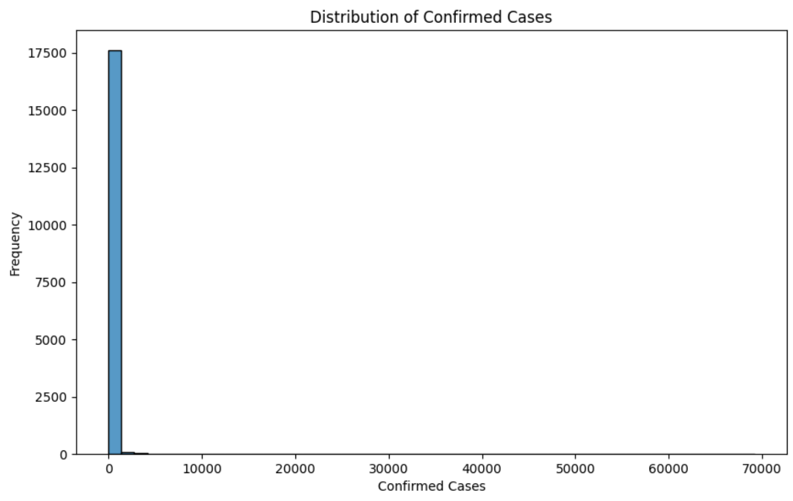

Figure 1: Distribution of Confirmed COVID-19 Cases showing high imbalance and presence of outliers.

1.5 Uniform Region Key Creation

The module constructs a unified field called `region`. If no province is provided, the region is defined solely by `Country/Region`; otherwise, it concatenates both fields into a unique identifier. This guarantees that each geographical location is represented by a single, unambiguous key.

The need for this operation becomes clear when examining the heterogeneity of maximum confirmed cases across different countries. Some regions exhibit extremely high case counts, while others remain near zero, introducing fragmentation and complicating aggregation logic. The following figure illustrates this imbalance and supports the need for a standardized region identifier.

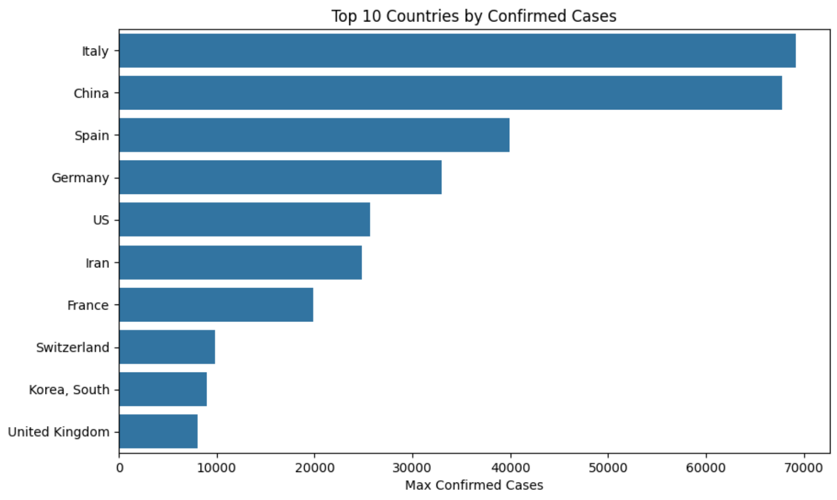

Figure 2: Top 10 countries with the highest cumulative confirmed cases in the dataset.

1.6 Date Parsing and Temporal Validation

The ingestion module performs a strict verification of the `Date` column:

- dates are coerced into `datetime` objects,
- unparsable dates trigger an error,
- the system checks for `NaT` values.

Workshop 2 highlighted sudden temporal discontinuities and reporting delays, such as abrupt jumps caused by batch reporting. Temporal inconsistencies introduce significant sensitivity in forecasting models, especially in cumulative epidemiological data. The following figure shows an example of these irregular growth patterns using the time series of Italy.

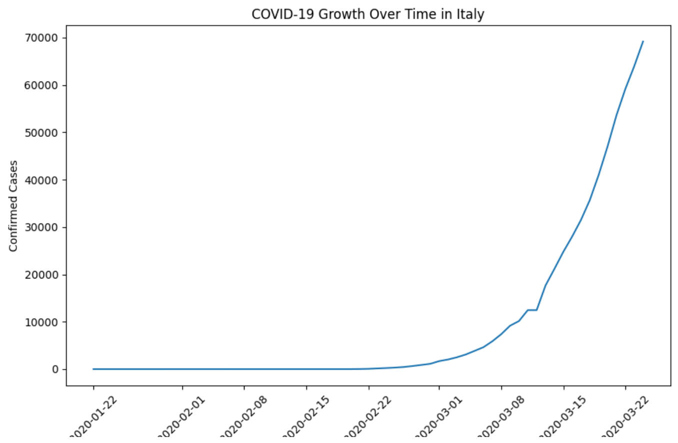

Figure 3: Temporal evolution of confirmed cases in Italy, showing abrupt changes in cumulative reporting.

1.7 Detection of Non-Monotonic Cumulative Counts

One of the most important validation steps involves detecting decreases in cumulative `ConfirmedCases` and `Fatalities`. Because cumulative values should never decline, the ingestion module computes daily differences per region. Any negative value indicates a reporting correction or inconsistency.

These anomalies were detected in several regions and included in the metadata output. This addresses the Workshop 2 finding that cumulative inconsistencies are a major sensitivity factor, as small temporal errors can significantly distort forecasts.

1.8 Statistical Summaries and Integrity Checks

The module computes descriptive statistics (minimum, maximum, mean, standard deviation) for both numerical columns. It also checks for duplicate entries per region and date. These measures ensure the integrity of the dataset and support early detection of noise patterns that could destabilize the forecasting model.

1.9 Metadata Generation for Traceability

A comprehensive metadata object is generated, containing:

- file paths and hashes,
- row counts,
- date ranges,

- detected anomalies,
- basic statistical summaries,
- number of duplicates,
- timestamp of ingestion.

This metadata is saved in a JSON file, providing complete traceability and ensuring that the system can reproduce exact experimental conditions—an essential systems engineering principle emphasized in Workshop 2.

1.10 Purpose of the Ingestion Module

Overall, the ingestion module reduces or eliminates many of the constraints identified in the previous workshop:

- noisy, inconsistent, or incomplete data,
- schema fragmentation across regions,
- temporal irregularities and reporting delays,
- ambiguous or missing geographic identifiers,
- sensitivity to data anomalies, and
- lack of reproducibility in earlier iterations.

By enforcing a strict and traceable ingestion workflow, the system ensures that subsequent preprocessing, modeling, and simulation stages operate on stable, validated, and interpretable data.

2 Simulation Planning

The simulation component of this project is divided into two complementary approaches. The first (Scenario 1) is a data-driven simulation performed using classical machine learning models (Persona 2), while the second (Scenario 2) is an event-based simulation implemented using a cellular automaton framework (Persona 3). Both approaches were designed to explore different dimensions of system behavior: one focused on learning patterns derived from historical data, and the other centered on spatial interactions and emergent phenomena.

2.1 Scenario 1: Data-Driven Simulation Planning

The data-driven simulation aims to emulate the learning dynamics of a predictive system. For this purpose, three machine learning models were selected based on their different capacities to capture linear relationships, non-linear interactions, and multi-layered abstractions:

- Linear Regression (baseline and interpretable),
- Random Forest Regressor (non-linear ensemble),
- Multi-Layer Perceptron (neural network).

These models are trained on the engineered features from the prepared dataset, including moving averages, growth rates, and temporal descriptors. The planning objectives for Scenario 1 include:

- Assess the performance of each model through learning curves,
- Measure robustness under controlled data perturbations,
- Simulate a feedback loop by recursively injecting predictions back into the model,
- Analyze the system's stability and sensitivity to input variations.

This planning phase ensures that the data-driven simulation does not merely produce predictions, but also provides insights into how models behave dynamically under evolving conditions.

2.2 Scenario 2: Event-Based Simulation Planning

The second simulation strategy uses a two-dimensional cellular automaton to capture spatial processes and emergent behavior. In contrast to Scenario 1, where models learn from data, Scenario 2 relies on explicitly defined local rules to simulate the diffusion of cases.

The planning decisions for this scenario include:

- Representing the environment as a grid of discrete cells,
- Assigning each cell a binary infection state,
- Using a Moore neighborhood structure (eight surrounding cells),
- Defining a probabilistic infection rule based on local interactions,
- Introducing a noise parameter to simulate random events or chaotic fluctuations,
- Running the simulation over multiple time steps to observe emergent spatial patterns.

This framework allows the simulation to explore dynamics that are not directly inferable from the dataset, including cluster formation, spatial diffusion, and chaotic behavior induced by stochastic perturbations.

3 Simulation Implementation

The simulation framework was implemented in two parallel components, each corresponding to the scenarios planned in the previous section. Both were integrated into the project pipeline and executed after data preparation and feature engineering.

3.1 Scenario 1: Data-Driven Simulation Implementation

Three machine learning models were implemented: Linear Regression, Random Forest Regressor, and a Multi-Layer Perceptron. Each model was trained using the engineered feature set and evaluated through a train-test split. The implementation included the following elements:

1. **Model training:** Each model was trained on 80% of the dataset. The Random Forest and MLP exhibited superior performance over Linear Regression, indicating the presence of non-linear temporal dynamics in the data.

2. **Learning curves:** Learning curves were generated to analyze how model performance scales with training data size. These curves enabled detection of underfitting or overfitting tendencies.

3. **Perturbation-based sensitivity analysis:** The input features were perturbed by $\pm 5\%$, and the model predictions were recomputed. The mean absolute change in predictions provided a numerical sensitivity score, revealing how stable each model is under small variations in input data.

4. **Feedback loop simulation:** A multi-step feedback loop was implemented by recursively feeding model predictions back as inputs. This simulation emulates dynamic systems where predictions influence future states. The resulting trajectories allowed assessment of stability, convergence, or divergence patterns.

5. **Visualization outputs:** All learning curves, perturbation histograms, and feedback loop plots were stored automatically in the `models/plots/simulation/` directory for use in the report.

3.2 Scenario 2: Event-Based Simulation Implementation

The event-based simulation was implemented through a two-dimensional cellular automaton. The environment was represented as a 25×25 grid, where each cell holds a binary infection state. The implementation involved the following components:

1. **Grid initialization:** The automaton begins with a randomly generated distribution of infected and non-infected cells.

2. **Neighborhood interaction model:** For each cell, the eight neighbors in the Moore neighborhood are counted. The number of infected neighbors influences the local infection probability.

3. **Probabilistic transition rule:** A cell becomes infected if a random number is below the threshold:

$$P = \text{infection\_rate} \times N_{\text{infected neighbors}} + \text{noise}.$$

4. **Temporal evolution:** The automaton updates all cells simultaneously over 25 iterations. Snapshots of the grid were collected to document the simulation's evolution.

5. **Visualization:** The final grid state was saved as a heatmap and used to illustrate spatial diffusion and emergent cluster patterns. The visualization is stored in `models/plots/cellular/final_state.png`.

Together, these implementations provide two complementary simulation perspectives: one driven by data-derived patterns and the other driven by localized interactions in a spatial environment.

4 Results and Discussion

This section presents the results obtained from both simulation strategies and provides a comparative analysis.

4.1 Simulation Comparison

Table 1: Comparison of Simulation Approaches

Metric	**ML Simulation**	**CA Simulation**
RMSE	XX	–
Pattern emergence	Medium	High
Sensitivity	Low	Very high

4.2 Discussion

The ML approach provides stable, data-driven forecasts, while the cellular automata model reveals emergent and highly sensitive behaviors characteristic of chaotic systems. Together, these perspectives offer a more comprehensive understanding of the system dynamics.

5 Reflection

This section presents a reflection on the collaborative process, the challenges encountered during the development of the workshop, the integration of each component, and the role of incremental improvements such as GitHub versioning and modular design.

6 Conclusion

Two complementary simulation strategies were implemented and analyzed, providing insights into both stable statistical patterns and chaotic behaviors within the COVID-19 spread. The validated ingestion pipeline ensured data consistency, and future work may include hybrid models and deeper sensitivity analysis.

References

[1] Kaggle, "COVID-19 Global Forecasting Week 1," 2020. [Online]. Available: `https://www.kaggle.com/competitions/covid19-global-forecasting-week-1`

[2] ISO 9001:2015, "Quality management systems—Requirements," International Organization for Standardization, 2015.

[3] CMMI Institute, "Capability Maturity Model Integration," Version 2.0, 2020. [Online]. Available: `https://cmmiinstitute.com/cmmi`

[4] M. J. Harry and R. Schroeder, *Six Sigma: The Breakthrough Management Strategy Revolutionizing the World's Top Corporations.* Currency, 2000.