Probabilistic Ancestral Inference from Incomplete Genetic Data

Makenna Worley

January 2026

Abstract

This project investigates the efficacy of probabilistic machine-learning architectures in reconstructing latent states within high-dimensional, stochastic datasets. Using a multi-generational simulation framework as a high-fidelity data engine, the study evaluates how Bayesian inference, Hidden Markov Models (HMMs), and Graph Neural Networks (GNNs) recover missing features from intentionally degraded data. By leveraging a "ground-truth" generator to produce discrete, rule-based hierarchical data, we benchmark model performance against varying levels of controlled data masking.
The research focuses on the technical trade-offs between computational efficiency and inference precision, utilizing a reproducible pipeline that employs JSON-based meta-replay to ensure exact dataset reconstruction for benchmarking. The final deliverable is an interactive web dashboard that visualizes model calibration and robustness, providing a proof-of-concept for applying probabilistic modeling to any domain where hierarchical data is sparse or incomplete.

1 Introduction

The ability to reason under uncertainty remains a central challenge in computational modeling, particularly when dealing with high-dimensional datasets where hierarchical dependencies are obscured by missing or degraded data. While data sparsity often limits the effectiveness of traditional analytical methods, probabilistic machine-learning models provide a framework to infer these hidden variables. This project treats rule-based simulations as a rigorous testing ground for algorithmic robustness, focusing on the mathematical challenge of recovering latent states from controlled, stochastic observations.

Traditional inference approaches often rely on dense or fully observed datasets, making them sensitive to missing information. To address this, we implement a simulation-driven approach that generates "ground-truth" datasets where every internal state is known. We then systematically mask these datasets at varying rates to evaluate how different probabilistic models—including Bayesian inference, Hidden Markov Models (HMMs), and Graph

Neural Networks (GNNs)—perform under increasing levels of uncertainty. By utilizing a discrete, rule-based simulation engine, we ensure that the data contains clear, logical structures that allow for quantitative validation of model accuracy.

The technical foundation of this work is a reproducible data generation pipeline capable of "meta-replay". This system uses JSON-based run metadata and specific random seeds to ensure that any stochastic dataset can be exactly reconstructed for benchmarking purposes. This engineering ensures that performance metrics—such as precision, recall, and calibration—are reflective of the model's architectural strengths rather than noise in the data generator. Through an interactive web dashboard, this project will demonstrate the trade-offs between computational cost and inference precision across multiple probabilistic frameworks.

The core objectives are:

- To design a simulation and inference system that models latent dependencies within hierarchical stochastic data.
- To implement, train, and test probabilistic and machine-learning models (Bayesian, HMM, and GNN) for high-dimensional state reconstruction.
- To develop an interactive web dashboard for visualization, evaluation, and comparison of model performance across varying data degradation thresholds.

If successful, this project would establish a reproducible methodology for state reconstruction under uncertainty, offering a new computational framework for any domain that relies on incomplete or partially observed stochastic data.

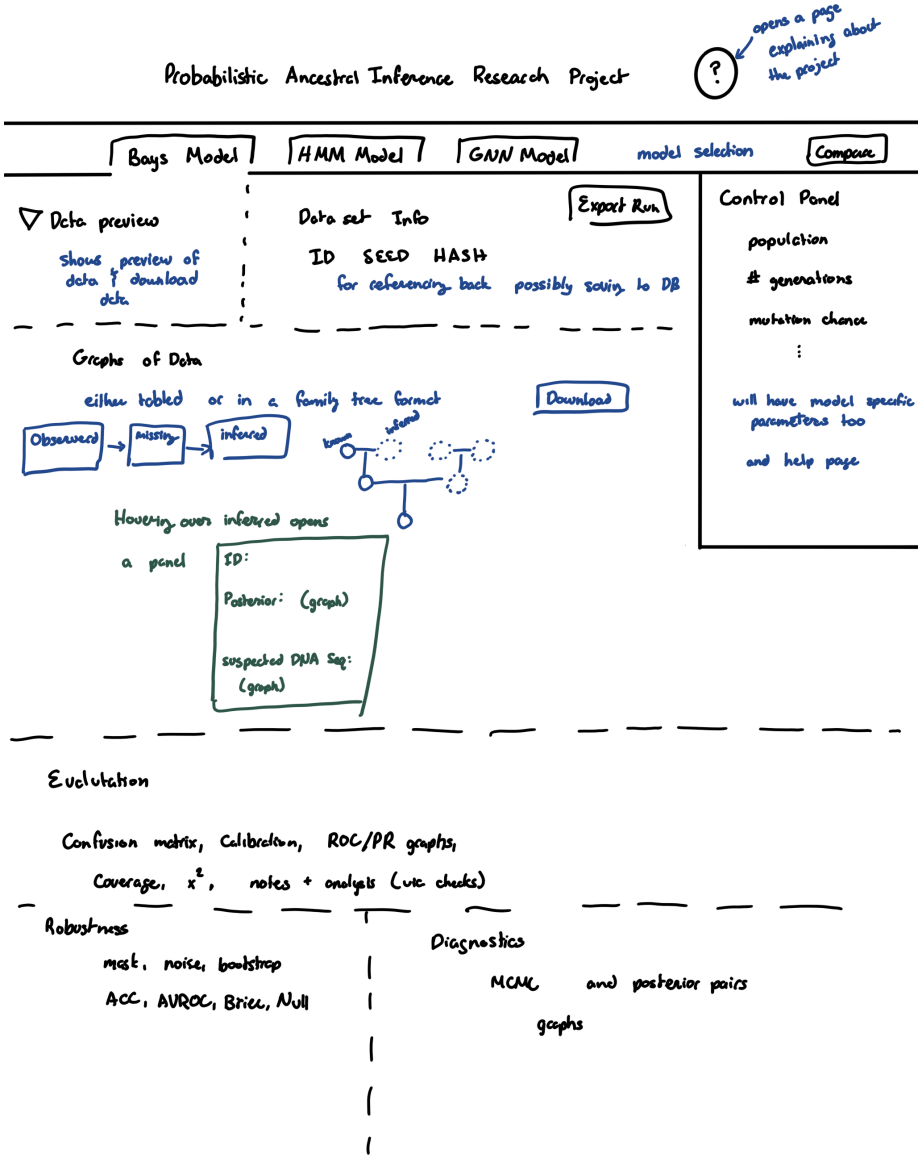

Figure 1: I plan on using a similar design language to my 3D Visualizer.

2 Background

2.1 Scientific Context

Inferring latent states within high-dimensional, stochastic systems is a persistent challenge in machine learning, particularly when datasets exhibit complex hierarchical dependencies. Probabilistic architectures such as Bayesian inference, Hidden Markov Models (HMMs), and Graph Neural Networks (GNNs) offer diverse strategies for modeling uncertainty and state transitions across discrete time steps.

Table 1: Comparison of Probabilistic Inference Architectures for Stochastic Data

Method	Strengths	Limitations	Best Used When
Bayesian Inference	Highly interpretable; incorporates structural priors; models uncertainty explicitly.	High computational cost; slower convergence for large datasets.	Domain-specific knowledge is available and interpretability is critical.
Hidden Markov Model (HMM)	Efficient for sequential linkage; strong theoretical foundation for state transitions.	Assumes independence between hidden states; simplifies non-linear dependencies.	Modeling sequential transitions or temporal linkage across discrete features.
Graph-Based Model	Captures multi-way relationships and complex hierarchical structures; highly scalable.	Requires larger training sets and intensive hyperparameter tuning.	Modeling highly connected networks or population-level dependencies.

Existing inference approaches often struggle with "missingness" in sparse datasets. Traditional maximum-likelihood methods assume near-complete data or well-defined priors, making them sensitive to high masking rates. Similarly, many sequential modeling tools degrade rapidly when internal nodes are unobserved or sparsely sampled. These limitations motivate the development of models that remain robust under controlled data degradation.

Unlike empirical studies constrained by noisy real-world data, this project utilizes a *stochastic data engine* to create a fully controlled testing environment. By using a discrete, rule-based simulation framework, we can generate "ground-truth" matrices where every state is known before applying controlled masking. This allows for a direct evaluation of model recovery performance without the confounding variables found in empirical datasets.

The simulation pipeline leverages the `msprime` framework to generate multi-generational datasets following discrete Mendelian rules. This produces a "diploid dosage matrix" where features represent independent loci and entries represent discrete states. By masking these matrices at varying rates, we systematically measure how effectively different ML architec-

tures reconstruct the latent hierarchy. This modular design ensures that the focus remains on **algorithmic inference accuracy** rather than species-specific modeling.

2.2 Technological Context

Recent advances in simulation frameworks and probabilistic programming make the rigorous benchmarking of machine learning models under uncertainty both feasible and efficient. The `msprime` library serves as a high-performance stochastic engine, enabling the generation of large-scale datasets with discrete, rule-based dependencies. This framework provides fine-grained control over the "stochastic backbone" of the data, allowing for the simulation of complex relational structures while maintaining strict scalability.

The data generation pipeline is designed for high-fidelity reproducibility. By utilizing JSON-based metadata and specific random seeds, the system ensures that every "truth" dataset and its corresponding "masked" version can be exactly reconstructed for comparative model analysis. This reproducible foundation is critical for validating the precision of probabilistic inference models.

For statistical modeling, the project employs `PyMC` (utilizing the `PyTensor` backend) to implement a Bayesian framework that models data uncertainty explicitly through probabilistic priors. Sequential dependencies within the data are addressed using a Hidden Markov Model (HMM) developed via `pomegranate` or `hmmlearn`, which provides flexible APIs for representing state transitions. The graph-based approach utilizes `PyTorch Geometric` to treat data relationships as nodes and edges within a dynamic graph architecture, enabling the capture of multi-way dependencies.

The system architecture follows modern web and data engineering standards. Model training and evaluation outputs are served through a `FastAPI` backend and visualized via a custom `React` dashboard. This interface allows for the interactive exploration of model accuracy metrics, calibration curves, and reconstruction uncertainty. The entire environment is containerized with `Docker` to ensure modular deployment and cross-platform portability.

These tool choices were made for their open-source accessibility, active research communities, and demonstrated performance in genetics, simulation, and AI applications.

I bring direct technical experience to this stack:

- **Frontend Development**: Experience with `React` and `ReactFlow` from the CircuitCraft: a summer research project, where I helped build an interactive circuit-drawing application using `ReactFlow`.

- **Backend Engineering**: Previous implementation of `FastAPI` in the MVP of my differential equations project, which integrated mathematical modeling with a web-based backend.
- **Data Management**: Currently validating a `RSQL` database to manage the storage and retrieval of simulation metadata and model runs.

Through this project, I aim to transition my skills from general web engineering into advanced probabilistic modeling and simulation design. I will focus on implementing and calibrating Bayesian and HMM architectures within a controlled stochastic environment. This work bridges the gap between data engineering and applied machine learning, focusing on the practical challenges of model interpretability and recovery performance in sparse data regimes.

3 Proposed Work

3.1 Proof of Concept

3.1.1 Overview

The core proof of concept (PoC) for this project will be a functional pipeline capable of reconstructing latent states within hierarchical stochastic datasets. Using rule-based simulations to generate ground-truth data, the PoC will demonstrate that a Bayesian inference engine can accurately recover features intentionally removed via a controlled masking process. At minimum, the PoC will validate that the model can handle a specific "masking rate" threshold and visualize the reconstruction accuracy and posterior uncertainty through a web-based interface. This deliverable establishes the technical feasibility of using probabilistic architectures to solve state-recovery problems in sparse data regimes.

3.1.2 Specific Tasks

1. Generate Synthetic Datasets: Use `msprime` to simulate high-fidelity stochastic data with discrete hierarchical dependencies.
2. Implement Masking Framework: Develop a systematic process to mask data entries (e.g., 20% masking rate) to emulate incomplete observations while preserving "truth" for validation.
3. Develop Bayesian Inference Engine: Train a baseline Bayesian model using `PyMC` to infer masked values based on hierarchical priors.
4. Dataset Stratification: Produce independent training, validation, and testing datasets to ensure statistical robustness and prevent overfitting.

5. Deploy Lightweight Dashboard: Build a `Streamlit` interface for real-time visualization of model performance and data degradation metrics.

6. Quantify Recovery Performance: Evaluate accuracy using statistical metrics, including chi-square tests and likelihood-based confidence intervals.

3.1.3 Rationale

The PoC focuses on the mathematical recovery of masked data points rather than domain-specific predictions. By working in a controlled simulation environment, we can perform quantitative validation against an absolute ground truth—a process often impossible with empirical data. Proving that a Bayesian model can successfully resolve these latent states validates the core logic of the inference framework before expanding to more complex architectures like HMMs or GNNs.

3.1.4 Deployment

The PoC will be deployed as a local `Streamlit` application to facilitate rapid demonstration and iterative testing. Simulation outputs, including truth and observed matrices, will be stored as `.csv` files and managed via JSON metadata to ensure "meta-replay" capability. This local deployment provides a streamlined interface for loading diverse datasets, executing inference runs, and monitoring performance without the overhead of external infrastructure, serving as a modular foundation for the final web application.

3.2 Full Project Development

3.2.1 Specific Tasks

1. Scale Simulation Pipeline: Extend the `msprime` engine to support larger dataset dimensions and more complex hierarchical parameters.

2. Implement Comparative Architectures: Develop and tune multiple probabilistic models—Bayesian, HMM-based, and Graph Neural Networks (GNN)—to reconstruct latent states from masked data.

3. Systematic Robustness Testing: Regenerate training, validation, and testing datasets under varying "masking rate" conditions to benchmark model degradation.

4. Architect Model-Serving Backend: Integrate a `FastAPI` backend to manage model inference, data retrieval, and storage of run metadata.

5. Develop Interactive Analytics Dashboard: Build a comprehensive `React` frontend for comparative performance visualization, utilizing dynamic graphing for accuracy metrics.

6. Advanced Statistical Validation: Validate reconstruction performance using chi-square, likelihood-ratio, and calibration metrics across all model types.

7. Containerization: Package the entire stack using `Docker` to ensure environment parity and portability for future research.

3.2.2 Rationale

Stochastic datasets are frequently incomplete, but probabilistic models are specifically designed to reason under such uncertainty. Using simulated data provides absolute control over noise and masking parameters, enabling the reproducible experimentation required for rigorous model validation. While biological rules provide the logical backbone for the simulation, the primary goal is testing computational robustness in a scalable environment. Employing independent datasets mitigates overfitting risks, while the `FastAPI--React` architecture ensures the system is maintainable and scalable for broader data engineering applications. These decisions align with modern research software standards, moving the project from a localized prototype to a production-ready analytical tool.

3.3 Plan of Work

1. Data Pipeline Engineering: Finalize the `msprime` pipeline for generating ground-truth matrices. Implement seed-based "meta-replay" to ensure that any stochastic run can be perfectly reconstructed during the evaluation phase.

2. Model Benchmarking: Implement and optimize the three core model types—Bayesian, HMM, and Graph-based. Focus on optimizing runtime for high-dimensional matrices and documenting architectural hyper-parameters.

3. Robustness Profiling: Introduce controlled noise and variable masking thresholds to the datasets. Record error bounds and performance metrics (Precision, Recall, Calibration) to establish the limits of each model's reconstruction ability.

4. Full-Stack Integration: Connect the `FastAPI` endpoints to the `React` dashboard to allow for dynamic querying of model results and visualization of confidence intervals.

5. Optimization and Deployment: Identify and resolve performance bottlenecks in the inference engine. Finalize the `Docker` configuration and complete documentation to ensure the project is fully reproducible by other researchers.

Absolute Minimum: A functional stochastic data engine using `msprime` to generate truth and masked datasets, integrated with a baseline Bayesian inference model and a `Streamlit` dashboard for reporting recovery statistics.

Expected: All features of the Absolute Minimum plus a robust validation pipeline and a full-stack `FastAPI` and `React` dashboard. This tier includes a comparative analysis between two distinct architectures: Bayesian inference and Hidden Markov Models (HMM).

Aspirational: All features of the Expected tier with the addition of a Graph Neural Network (GNN) model to evaluate how graph-based learning handles multi-way dependencies. This will include a comprehensive statistical suite comparing the reconstruction efficiency, accuracy, and computational overhead across all three probabilistic models.

3.4 Preliminary Work

Initial research and engineering have established a high-fidelity data generation pipeline capable of producing discrete dosage matrices from rule-based simulations. This preliminary work identified an effective toolchain—`FastAPI` for model serving, `React` for interactive visualization, and `msprime` for stochastic data generation—and validated a "meta-replay" strategy using JSON metadata to ensure 100

4 Timeline

Table 2: Project Timeline

Phase	Dates	Time est.	Focus	Deliverables
Phase 0: System Integration	Oct 2025	20 hrs	Finalize `data_generation.py` for dosage matrix production. Integrate `RSQL` for metadata storage and seed-based replay tracking.	Reproducible data engine with JSON meta-replay and `RSQL` schema validation.
Phase 1: Proof of Concept	Nov 2025	70 hrs	Implement baseline Bayesian model using `PyMC`. Test recovery accuracy on masked hierarchical datasets.	Inference MVP achieving $\geq$70% state recovery accuracy and initial `Streamlit` UI.
Phase 2: Expansion	Dec–Jan 2026	90 hrs	Implement HMM-based inference. Benchmark both models across variable masking rates and analyze calibration error.	Comparative analysis suite with performance metrics for Bayesian and HMM architectures.
Phase 3: Web Application	Feb–Mar 2026	70 hrs	Develop `FastAPI` model-serving endpoints. Migrate UI to `React` for interactive uncertainty visualization.	Production-ready web dashboard with dynamic data querying and multi-model comparison.
Phase 4: Code Freeze	Apr 2026	40 hrs	Finalize `Docker` containerization. Perform system-wide latency optimization and complete documentation for reproducibility.	Containerized build, comprehensive performance report, and public-facing project documentation.

Total estimated effort: approximately 290 hours.

5 Evaluation

Evaluation will focus on the following quantitative and qualitative dimensions:

- Reconstruction Accuracy: Measuring the delta between predicted values and the known "truth" dataset using precision, recall, and F1-scores. Statistical significance will be validated via chi-square and likelihood-ratio tests to ensure inferred states are mathematically sound.

- Model Calibration: Evaluating the reliability of probabilistic outputs. We will assess whether the model's reported confidence intervals align with actual recovery rates, ensuring the system does not exhibit overconfidence in high-uncertainty regimes.
- System Performance: Benchmarking the runtime efficiency of the inference engines as data dimensionality (number of loci and population size) increases.
- Computational Robustness: Testing the "break point" of each architecture by measuring accuracy decay across a spectrum of controlled masking rates (e.g., 10
- Usability and Transparency: Ensuring the `React` dashboard effectively visualizes latent state uncertainty and provides clear access to the "meta-replay" logs for auditability.

Ethical Considerations and Interpretability:

- Data Privacy: The use of synthetic, rule-based datasets eliminates the privacy risks associated with empirical human or ecological data while providing a high-fidelity environment for algorithmic testing.
- Algorithmic Interpretability: Emphasis is placed on Bayesian and HMM architectures specifically for their transparency, preventing the "black box" outcomes often associated with deep learning in sensitive research contexts.
- Bias Mitigation: By using a controlled generator (`msprime`), we avoid the sampling biases inherent in real-world observations, allowing for an objective assessment of model performance.

The expected outcome is a validated computational framework demonstrating that probabilistic models can accurately reconstruct latent states in hierarchical stochastic systems. This project establishes a reproducible methodology for state inference that is applicable across diverse fields—from conservation modeling to general data science—wherever researchers must reason under significant data uncertainty.

Conclusion

High-dimensional, hierarchical datasets are frequently constrained by missing or sparsely sampled observations, limiting the ability to model complex dependencies and predict future states with confidence. This project addresses that challenge by developing a simulation-driven framework that evaluates the robustness of probabilistic architectures—specifically Bayesian models, Hidden Markov Models, and graph-based methods—in recovering latent states from intentionally degraded datasets. By leveraging the `msprime` engine to generate

reproducible, ground-truth data, the system enables rigorous benchmarking of how different probabilistic models perform under varying levels of uncertainty and masking.

The proposed solution integrates rule-based simulation, statistical modeling, and interactive visualization to create a reproducible toolset for understanding how data incompleteness affects inference accuracy. If successful, this project will deliver a validated prototype demonstrating that probabilistic models can accurately reconstruct missing information even in sparse data regimes. This is achieved through an engineering-heavy pipeline featuring seed-based "meta-replay" and automated dosage matrix generation.

Such a result would provide a methodological foundation for future work in computational modeling, supporting applications in fields as diverse as population history, conservation biology, and biomedical trait prediction. More broadly, this project contributes a scalable, interpretable, and ethically grounded approach to reasoning under uncertainty—offering new possibilities for research settings where complete data is unattainable.

6 References

References

[1] Kelleher, J., Etheridge, A. M., & McVean, G. (2018). *Efficient coalescent simulation and genealogical analysis for large sample sizes. PLOS Computational Biology, 14*(5), e1006581.

[2] Salvatier, J., Wiecki, T., & Fonnesbeck, C. (2016). *Probabilistic programming in Python using PyMC3. PeerJ Computer Science.*

[3] Paszke, A., et al. (2019). *PyTorch: An imperative style, high-performance deep learning library. NeurIPS.*

[4] Browning, S. R., & Browning, B. L. (2007). *Rapid and accurate haplotype phasing and missing-data inference for whole-genome association studies by use of localized haplotype clustering. The American Journal of Human Genetics, 81*(5), 1084–1097.

[5] FastAPI Documentation (2024). *Modern web framework for building APIs with Python 3.7+.*

[6] React Documentation (2024). *React: A JavaScript library for building user interfaces.*