Generative Language Modeling for Causal Inference on Dynamical Data

Jun Won Park[1] and Sanket Rane[1]

[1]Irving Institute for Cancer Dynamics, Columbia University, New York, NY, USA

April 7, 2025

Contents

Abstract

We introduce viaABC (Variational Inference Assisted Approximate Bayesian Computation), a novel likelihood-free parameter estimation framework that integrates Approximate Bayesian Computation (ABC) with Variational Autoencoders (VAEs). Traditional ABC methods rely on carefully chosen summary statistics, which can be challenging to define in high-dimensional settings, often leading to ill-posed inverse problems. viaABC mitigates this limitation by leveraging the representational power of VAE, a generative model, to learn a structured manifold representation of data beyond the constraints of Euclidean space. This approach is particularly advantageous for complex datasets, including hierarchical time series, spatial, spatio-temporal, and stochastic dynamical systems, enabling more robust and scalable parameter inference in likelihood-free settings.

1 Introduction

Mechanistic inference is essential for understanding phenomena across various scientific disciplines, as it enables the mathematical modeling of dynamical systems and the extraction of their underlying parameters. Therefore, accurately inferring model parameters, denoted as θ, from experimentally observed data, y^{obs}, is crucial. In the Bayesian framework, this involves determining the posterior distribution

$$\pi(\theta|y^{obs}) \propto f(y^{obs}|\theta) \cdot \pi(\theta)$$

of the parameters from the data by calculating the likelihood $f(y^{obs}|\theta)$ of such observations under the model with given parameter values. However, deriving the likelihood function theoretically is often intractable, computationally expensive, or too complex for direct optimization. Approximate Bayesian Computation (ABC) was introduced as an alternative framework for approximating posterior distributions when traditional likelihood-based methods are impractical[1]. This approach either employs summary statistics to represent both simulated and observed data using a distance metric to gauge their similarity in the case high-dimensional data or it directly compares the raw simulated and observed datasets. In both cases, ABC aims to identify the parameter values that minimize the discrepancy between the two. Numerous variants of the ABC algorithm have since been developed, including ABC Monte Carlo Markov Chain[2], ABC Sequential Monte Carlo[3], and ABC Population Monte Carlo[4], each of which has demonstrated effectiveness across a wide range of scientific disciplines.

Despite their success, the accuracy and efficiency of existing Approximate Bayesian Computation (ABC) methods heavily depend on the selection of hyperparameters, including summary statistics and distance metrics, which are often data-dependent and require extensive tuning. In particular, reducing data to a few informative summary statistics is a critical step that directly influences the accuracy of the inference. The choice of these statistics must be made carefully, as it determines the quality of the representation and, consequently, the reliability of the results[5]. This challenge is especially pronounced in biological applications, where data is inherently high-dimensional. As the dimensionality increases, identifying a compact yet informative set of summary statistics becomes impractical. Additionally, the choice of distance metric, including how to weight each statistics, also influences the accuracy of the parameter estimation.

To address these limitations, we introduce viaABC, a novel statistical inference framework that leverages techniques from computer vision and natural language processing to enhance approximate posterior inference in mechanistic modeling. By learning a structured representation of data through variational inference, viaABC eliminates the reliance on manually selected summary statistics and distance metrics, thereby improving both the accuracy and robustness of ABC-based inference. Based on the manifold hypothesis that high-dimensional data often reside near low-dimensional manifolds[6], viaABC facilitates inference on complex, high-dimensional datasets, addressing challenges typically encountered in such analyses.

We show that

1. VAE can learn a dense, vector representation of the data, eliminating the need for using summary statistic to represent data
2. our framework is more efficient in terms of acceptance rate
3. our framework is superior in terms of accuracy
4. our framework is more flexible and robust towards experimental and biological noises

Our work will primarily focus on multivariate time-series data. Our method employs self-supervised pre-training on simulated multivariate time-series data. During pre-training, the model captures temporal and cross-channel dependencies, constructing a latent distribution for each time step. This process yields an entangled representation of the time-series data. The pre-trained model then generates latent representations for simulated data, which are subsequently compared to the representation of y^{obs} for parameter inference using sequential ABC.

2 Related Work

2.1 ABC

This section provides a comprehensive review and development of the theoretical foundations of Approximate Bayesian Computation (ABC), with a particular focus on its applications to dynamical systems and statistical parameter inference before introducing the variational inference approach in the context of ABC.

ABC Rejection

Approximate Bayesian Computation (ABC) rejection is a method for parameter inference when likelihoods are intractable or computationally expensive to evaluate[1]. The process begins by defining prior distributions for the parameters, a distance metric, and summary statistics. Common distance metrics include L1 and L2 distances, while statistics often include the mean, variance, or combinations of moments. A parameter set is then sampled from the prior distributions and passed through a predefined simulator, a mathematical model describing the system's dynamics. The resulting statistics are computed and compared to those from the observed data. If the distance between the simulated and observed statistics falls within an acceptance threshold, the parameter set is retained for the approximate posterior; otherwise, it is rejected.

Sequential ABC

Approximate Bayesian Computation (ABC) rejection sampling relies on several hyperparameters, including the rejection threshold and the choice of distance metric for comparing simulated and observed data. A small rejection threshold typically results in a low acceptance rate, leading to high computational costs, especially when the prior distribution differs significantly from the posterior[3]. To address this, ABC Population Monte Carlo (ABC-PMC) and ABC Sequential Monte Carlo (ABC-SMC) were introduced, which the idea is to begin with a large threshold and iteratively reduce it. In each iteration, samples, referred to as particles, are drawn from the approximate posterior of the previous iteration, iteratively refining the shape of the posterior distribution and improving sampling efficiency. The key distinction between ABC-PMC and ABC-SMC lies in the choice of the perturbation kernel. ABC-SMC assumes no explicit perturbation kernel, though prior studies have seen success using a uniform distribution. In contrast, ABC-PMC utilizes a multivariate normal distribution for perturbation, allowing for smoother updates and potentially more efficient exploration of the posterior space.

2.2 Masked Modeling

Self-supervised learning (SSL) has demonstrated remarkable success in natural language processing and computer vision, with its applications extending beyond text and images to modalities such as video, audio, and time series[7]. In this publication, we explore the masked modeling approach, a prominent self-supervised learning technique.

The masked modeling approach learns useful representations by masking parts of the input and reconstructing them using the unmasked portions. This technique has been successful in natural language processing, as demonstrated by masked language modeling in BERT, where some tokens are randomly masked, and the model predicts the missing tokens based on the surrounding context[8]. In computer vision, an image is divided into patches, with some randomly masked, and the model reconstructs the original image using the remaining patches[9].

There are several variations of masked modeling. One approach masks portions of the input before passing it through the model, as seen in BERT, while another applies masking to the output of the encoder, as used in MAE. The optimal masking ratio varies depending on the application: in BERT, an effective masking ratio is approximately 15%, whereas in MAE, it reaches as high as 75%[10].

Masked modeling has been adopted into time-series domain for time-series forecasting like TiMAE and TimeMAE[11] [12]. By reconstructing the missing time-stamps from partially masked inputs, the model gathers contextual information to predict the missing regions, enhancing the model's understanding of complex interaction relationships between different dimensions.

2.3 Varitional Autoencoder

Variational inference (VI) has been proposed as a machine learning approach to approximate intractable probability densities, as an alternative method to Markov Chain Monte Carlo (MCMC) sampling.[13] [14]. There are numerous advantages to variational inference: variational inference is typically much faster especially for high-dimensional data and is much easier to scale to large data[14].

Variational autoencoders (VAEs) are deep latent variable models that leverage variational inference for approximate posterior inference. When a neural network is used for the recognition model, an approximation to the intractable true posterior, i.e. $q_\phi(z|x)$, we call it a variational auto-encoder[15].

Variational autoencoder consists of a probabilistic encoder $q_\phi(z|x)$ and a probabilistic decoder $p_\theta(x|z)$.

3 Methodology

3.1 Model Architecture

We extend the transformer-based Masked Autoencoder (MAE), originally developed for computer vision, where it employs the Vision Transformer (ViT) as the backbone model for both the encoder and decoder, to the domain of multivariate time-series data[9,16]. While MAE has been widely employed for image representation learning, its direct application to time-series data presents challenges due to the temporal dependencies and multivariate structure inherent in such data. To address these challenges, we introduce the Time-Series Masked Variational Autoencoder (TSMVAE), an adaptation that integrates the principles of

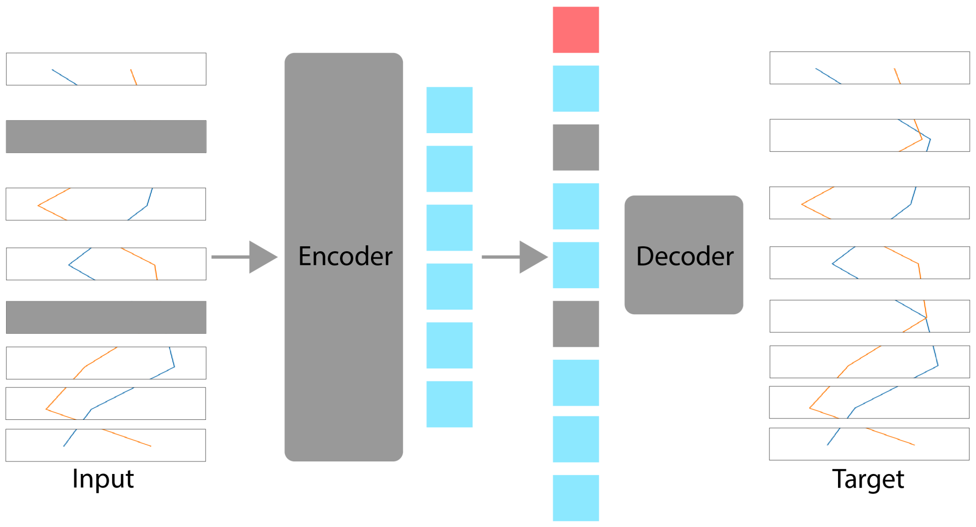

Figure 1: Our TSMVAE architecture. During pretraining, a random subset of temporal patches (e.g., 15%) is masked out. The encoder is applied to the visible patches. The CLS token (red) and Mask tokens (gray) are introduced after the encoder, and the full set of encoded patches along with the mask tokens undergoes the reparametrization trick. This is then processed by a small decoder that reconstructs the original patches. After pretraining, the decoder is discarded, and the encoder is applied to uncorrupted multivariate time series (full sets of patches) for viaABC.

MAE with a variational autoencoder framework tailored for time-series modeling. The intention of this paper is not to introduce the new model architecture.

Similar to all other autoencoders, our model consists of an encoder that maps the observed signal into a latent representation and a decoder that reconstructs the original signal from this latent space. However, TSMVAE diverges from the original MAE in two fundamental ways to better capture the structure of multivariate time-series data:

Temporal Patching: Given a D-dimensional time-series, $\{X_t\}_{t=1}^{T}$, where $X_t \in \mathbb{R}^D$, TSMVAE partitions the sequence along the time axis, forming patches that correspond to different time intervals. Each patch is then projected into a higher-dimensional space via a linear transformation, facilitating the encoding of temporal dependencies.

Variational Latent Space Representation: The encoded output undergoes two additional linear transformations. The first maps the encoded representations into a predefined latent dimension, structuring the latent space effectively. The second transformation introduces variational layers, which impose a probabilistic structure on the latent variables, thereby enabling the learning of meaningful latent distributions.

These modifications enable TSMVAE to effectively learn representations from multivariate time-series data time-stamp wise while leveraging the power of masked self-supervised learning. Figure 1 provides a high-level illustration of our proposed model. For further details on the model architecture, we refer the reader to prior work.

3.2 Pretraining

Suppose we have a mathematical model or data-generating process, denoted as f, which maps a k-dimensional parameter vector to a d-dimensional multivariate time series:

$$f(\theta) = \{X_t\}_{t=1}^{T}, \quad \theta \in \mathbb{R}^k.$$

To construct a training data set, we use Latin hypercube sampling (LHS), a commonly used sampling technique in monte carlo simulations, to generate N parameter sets, each of which is used to simulate an associated multivariate time series . Specifically, we denote the i-th simulated time series as

$$\{X_t^{(i)}\}_{t=1}^{T}, \quad X_t^{(i)} \in \mathbb{R}^d,$$

where each sample represents a realization of the underlying process. This procedure yields a dataset consisting of N multivariate time series samples. To facilitate the study of the model dynamics across a diverse range of parameter configurations, we employ Latin hypercube sampling based on the reasoning that a well-distributed exploration of the parameter sample space is potentially better, thus mitigating the risk of overfitting to any particular mode[17].

During training, Gaussian noise ϵ is dynamically injected at each time step such that

$$\tilde{X}_t = X_t + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \sigma^2).$$

The noise should be dimension-scale dependent. The variational autoencoder (VAE) is trained using noise-injected time series data as input, learning to reconstruct the original denoised data. The purpose of this is to allow the model to learn a representation that understands biological variation. In biology, observational data incorporate both biological variation and experimental variation. By dynamically injecting noise during pre-training, we

In addition to minimizing reconstruction loss, the VAE also minimizes Kullback-Leibler (KL) divergence to regularize the latent space and encourage a structured representation. In this study, we enforce a Gaussian prior on the latent space.

Objective function: Like any other VAE, the objective function optimizes the Evidence Lower Bound (ELBO). We define the loss function as a combination of reconstruction loss and KL divergence, applied at each time step. Since we employ the VAE in a time-step-wise manner, the loss function for each time step t consists of the reconstruction loss and the Gaussian KL divergence loss:

$$L_t = \frac{1}{2} MSE(x_t, \hat{x}_t) + \beta D_{KL}(q_\phi(z_t|x_t)||p_\theta(z_t))$$

Aggregating the losses over the entire time sequence, the total loss is given by:

$$\begin{aligned} L &= \frac{1}{T}\sum_{t=1}^{T} L_t \\ &= \frac{1}{T}\left(\frac{1}{2N \cdot D}\sum_{i=1}^{N}\sum_{j=1}^{d}(x_{t,j}^{(i)} - \hat{x}_{t,j}^{(i)})^2 + \beta D_{KL}(q_\phi(z_t|x_t)||p_\theta(z_t))\right) \end{aligned}$$

where β is a hyper-parameter that balances the trade-off between latent channel capacity and independence constraints with reconstruction accuracy[18].

Scaling: In biological data, multivariate time series often exhibit heterogeneous magnitudes, with each dimension varying significantly in scale. This variation presents optimization challenges for deep learning models, making normalization a crucial pre-processing step. In the case of TSMVAE, the goal of normalization is to ensure that values are scaled appropriately for deep learning training. This helps prevent issues such as vanishing or exploding gradients and mitigates the risk of the model disproportionately focusing on reconstructing one dimension while neglecting others, which could lead to under-fitting.

A widely used normalization approach involves applying an affine transformation to the time series, i.e.. $\tilde{x} = (x_i - m)/s$[19 20]. Different choices of m and s result in various well-known normalization techniques such as mean scaling, standard scaling, and min-max scaling.

In our study, we adopt mean scaling as the normalization technique, as it has demonstrated effectiveness in deep learning models frequently applied to practical time-series tasks[21 19 20]. Mean scaling normalizes each dimension of the time series by dividing its values by the absolute mean, setting $m = 0$ and $s = \frac{1}{T}\sum_{t=1}^{T}|x_i|$ in the above affine transformation.

Previous studies have addressed the challenge of heterogeneous magnitudes in time-series data using mean scaling, which enables deep learning models to learn scale-invariant patterns while preserving zero values[20]. While this property facilitates stable model training, it introduces an additional challenge in our work. Specifically, if the priors are non-informative or poorly specified, viaABC may accept particles that generate multivariate time series with differing absolute magnitudes but similar scaled representations compared to the observational data. This could lead to misleading inferences, as viaABC compares its similarity to the scaled observational data in latent space.

Special Token: The TSMVAE model has a special CLS token, which will be used to output a latent vector representation of a multi-variate time-series data. This is often known as the CLS pooling.

3.3 viaABC

Following the notation of[22] and[23], the resulting ABC posterior can be written as

$$\pi_\epsilon(\theta|y_{obs}) = \int [\frac{f(y_{prop}|\theta)\pi(\theta)\mathbb{1}_{A_{\epsilon,y_{obs}}}(y_{prop})}{\int_{A_{\epsilon,y_{obs}}\times\Theta} f(y_{prop}|\theta)\pi(\theta)dy_{prop}d\theta}]d_{y_{prop}}$$

where $\mathbb{1}_{A_{\epsilon,y_{obs}}}(\cdot)$ is the indicator function for the set $A_{\epsilon,y_{obs}} = \{y_{prop}|\rho(s(y_{obs}), s(y_{propr})) \leq \epsilon\}$

viaABC follows the Adaptive Approximate Bayesian Sequential Monte Carlo (aABC-SMC) algorithm, a variant of ABC-SMC that incorporates adaptive thresholds. However, it introduces two key modifications:

1) Instead of comparing distances in the original data manifold using predefined summary statistics, viaABC encodes the data into latent vectors and evaluates distances within the latent space. The TSVMAE model incorporates a CLS token, which facilitates the aggregation of contextual information and allows the representation of a multivariate time series as a vector. Furthermore, the latent representations at each time step can be used to represent the multivariate time series as a matrix, or they can be aggregated into a vector by taking the mean. These latent representations enable the calculation of the distance between the simulated data and the observational data in the latent space.

2) The initialization process leverages training data to accelerate convergence. In the aABC-PMC initialization, the algorithm samples kN particles, then selects the top N particles based on distance. In viaABC, we adapt this by calculating the distance between the observational data and all training samples, and selecting the top N particles based on this distance. This modification is similar to aABC-SMC when

$k = \frac{N_{training}}{N}$. A critical distinction is that in viaABC, the training samples are drawn using Latin Hypercube Sampling over the entire parameter domain, whereas aABC-PMC relies on pre-defined priors.

Algorithm 1 viaABC

```
 1: Given a pre-trained Encoder of a VAE, T, N, and k
    Initialization ($t = 1$):
 2: for $i = 1$ to $kN$ do
 3:     Sample $\theta_i^{(1)} \sim \pi(\theta)$
 4:     Simulate $y^* \sim f(x \mid \theta_i^{(1)})$
 5:     Pre-process $y^*$
 6:     $z^* = Encoder(y^*)$
 7:     Record $\rho(z_{obs}, z^*) = d_i$
 8: end for
 9: Let $D = \{d_i\}_{i=1}^{kN}$ and reindex the sequence D in monotone increasing order
10: Re-index the initial particles $\Theta_1 = \{\theta_j^1\}_{j=1}^{kN}$ based on D's index
11: Set $\epsilon_1 = d_N$
12: Set initial particles to be the first N elements of $\Theta_1$, e.g. $\{\theta_j^1\}_{j=1}^{N}$
    Iterations ($2 \leqslant t \leqslant T$):
13: for $t = 2$ to $T$ do
14:     Take $\tau_{t-1}^2$ as twice the weighted empirical variance of the $\{\theta_i^{(t-1)}\}_{i=1}^{N}$
15:     for $i = 1$ to $N$ do
16:         repeat
17:             Sample $\theta_i^*$ from $\{\theta_j^{(t-1)}\}_{j=1}^{N}$ with probabilities $\{\omega_j^{(t-1)}\}_{j=1}^{N}$
18:             Sample $\theta_i^{(t)} \mid \theta_i^* \sim \mathcal{N}(\theta_i^*, \tau_{t-1}^2)$
19:             Simulate $y^{**} \sim f(y \mid \theta_i^{(t)})$
20:             Pre-process $y^{**}$
21:             $z^{**} = Encoder(y^{**})$
22:         until $\rho(z_{obs}, z^{**}) \leqslant \epsilon_{t-1}$
23:         Record $\rho(z_{obs}, z^{**}) = d_i^{(t)}$
24:         Set $\omega_i^{(t)} \propto \pi(\theta_i^{(t)}) / \sum_{j=1}^{N} \omega_j^{(t-1)} \phi\{\tau_{t-1}^{-1}(\theta_i^{(t)} - \theta_j^{(t-1)})\}$
25:     end for
26:     $\hat{c}_t = \sup_\theta \frac{\pi_{\epsilon_t}(\theta)}{\pi_{\epsilon_{t-1}}(\theta)}$
27:     $q_t = \frac{1}{\hat{c}_t}$
28:     $\epsilon_t = $ Quantile$(\{d_i^t\}_{i=1}^{N}, q_t)$
29: end for
```

Stoping Rule: A few ideas have been introduced to determine how to stop in a sequential ABC algorithm. Ishida et al.[24] proposed a stopping rule once the acceptance rate of the algorithm falls below a certain threshold which is estimated based on the sequential ABC posterior distributions. Simola et. al[23] proposed using a stopping rule based on estimated quantile of the ABC posterior distributions where $q_t > 0.99$ and $t \geq 3$. In our work, we adopt the same stopping rule proposed by Simola et al.

4 Experiments

4.1 Main Properties

Masking ratio. Figure shows the influence of the masking ratio. The ratio of 15% seems to work well, which is in agreement with the behavior of BERT[8], whose masking ratio is 15%.

Decoder design. In accordance with the original MAE paper, our TSMVAE decoder can be flexible designed, as studied in Table. Table varies the decoder depth and dimension. A sufficiently deep decoder is important for denoising.

Data augmentation. Table studies the influence of different noise level injected used in our TSMVAE pre-training. Our TSMVAE works well using Gaussian noise. needs to be data dependent and channel dependent.

Beta Table studies the influence of different beta levels used in our TSMVAE pre-training. Based on our study, a beta value of 0.001

5 Dataset

Deterministic Lotka-Volterra model

The Lotka-Volterra (LV) model describes the interaction between predators and prey[25 26]. This deterministic model, which represents the populations of prey x and predators y, is expressed as a system of nonlinear differential equations:

$$\begin{aligned}\frac{dx}{dt} &= ax - cxy \\ \frac{dy}{dt} &= bxy - dy\end{aligned} \tag{1}$$

where a denotes the intrinsic growth rate of the prey, d is the mortality rate of the predators, c represents the predation rate coefficient, and b characterizes the efficiency with which consumed prey are converted into predator offspring. In this study, we set $c = d = 1$ and aim to infer the parameters $\theta = (a, b)$ from prior distributions where $a, b \sim \text{Uniform}(-10, 10)$. Following the methodology of Toni etal.[3], we solve the system of ordinary differential equations with the initial condition $(x(0), y(0)) = (1, 0.5)$ using $\theta = (1, 1)$. The state variables (x, y) are sampled at eight distinct time points between $t = 0$ and $t = 15$. Subsequently, normally distributed noise $\epsilon_t \sim \mathcal{N}(0, 0.5^2)$ is added to each data point, thereby forming the observed dataset $s^{\text{obs}} = \{x_1, y_1, \ldots, x_8, y_8\}$

To evaluate viaABC's performance, we compare the final posterior distributions with ABC-DRF, ABC-SMC, and ABC-SMC-DRF. Algorithm drf is blah blah. Talk about table sizes, different kernel in each algorithm. We impose the series of tolerance thresholds $\epsilon_1 = 0.3, \epsilon_2 = 0.2, \epsilon_1 = 0.1, \epsilon_4 = 0.04$. $\epsilon_4 = 0.04$ is the cosine dissimilarity between the ground truth and the observed data. To produce 1,000 accepted particles, viaABC requires $N = 34,474$ simulations.

Stochastic Susceptible-Infected-Recovered model

The stochastic Susceptible-Infected-Recovered (SIR) model is a fundamental framework in epidemiology for capturing the spread of infectious diseases within a population[?] . Unlike its deterministic counterpart, which employs ordinary differential equations, the stochastic SIR model incorporates intrinsic randomness in disease transmission and recovery, making it particularly suitable for modeling outbreaks in finite populations. The system tracks the evolution of three state variables: S, I, and R, representing the number of susceptible, infected, and recovered individuals, respectively. The model dynamics are governed by the following reaction scheme:

$$\begin{aligned}S + I &\xrightarrow{\beta} 2I \\ I &\xrightarrow{\gamma} R\end{aligned}$$

where β denotes the transmission rate per susceptible-infected pair, and γ represents the recovery rate of infected individuals. To capture the stochastic nature of infection and recovery events, we employ the Gillespie algorithm[27], which simulates the discrete event-driven evolution of the system.
In this study, we set $\beta = 2$ and $\gamma = 0.5$ and simulate the epidemic dynamics over a time span of $T = 15$ days, starting from an initial condition of $(S(0), I(0), R(0)) = (290, 10, 0)$ in a closed population of size $N = 300$. The stochastic trajectories are recorded at thirty-five distinct time points, and to account for observational uncertainty, normally distributed noise $\epsilon_t \sim \mathcal{N}(0, 1^2)$ is added to each recorded data point. The resulting dataset, denoted as $s^{\text{obs}} = \{S_1, I_1, R_1, \ldots, S_{35}, I_{35}, R_{35}\}$ serves as the empirical basis for inference in this study. Since events in the Gillespie algorithm occur at irregular time intervals, we interpolate the simulated trajectories to align their time indices with those of the observed data, ensuring comparability in the inference process.

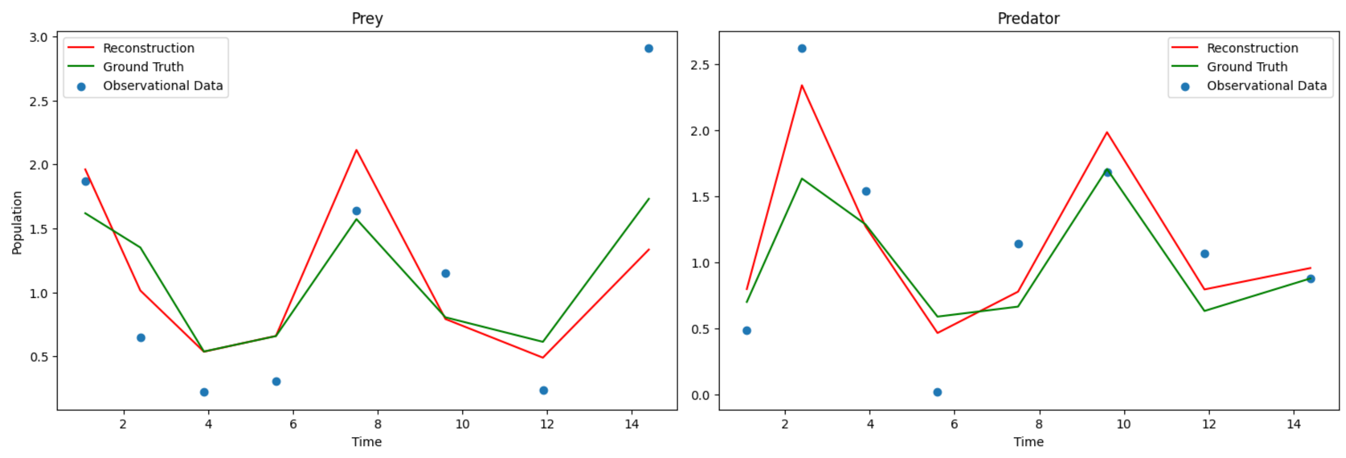

Figure 2: Reconstruction of observational data, the ground truth and the observational data of the Lotka-Volterra system.

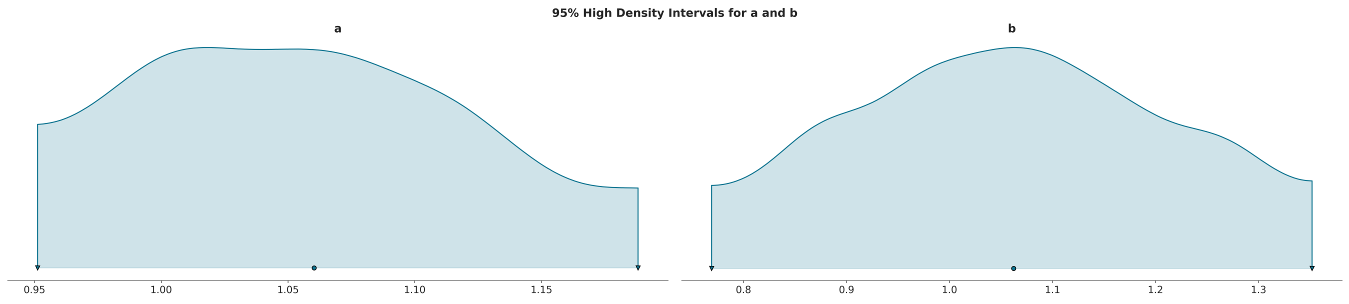

Figure 3: Density plot of the final iteration of viaABC. The estimated 95% highest posterior density (HPD) intervals for a and b are [0.9510, 1.1883] and [0.7685, 1.3525], respectively.

6 Results

6.1 Deterministic Lotka-Volterra model

Statistics	viaABC	ABC-DRF	ABC-SMC	ABC-SMC-DRF
$\mathbb{E}$(a)	**1.0641**	0.7562	1.2912	1.1215
Var(a)	**0.0053**	0.5953	0.1104	0.0333
$\mathbb{E}$(b)	1.0571	1.3092	**1.0269**	0.9704
Var(b)	**0.0299**	0.3593	0.1049	0.0313

Table 1: Means and variances of marginal posterior distributions for the deterministic Lotka-Volterra model from viaABC, ABC-DRF, ABC-SMC and ABC-SMC-DRF. The best result for each statistic across different algorithms is in bold (E(a), E(b) closest to true values (a, b) = (1, 1), and lowest variance in each statistic). Results are taken from Dinh et al.[28].

6.2 Stochastic Susceptible-Infected-Recovered model

Figure Something: a) Predicted trajectories of stochastic sir and the observed data points. (b) Final approximate posterior distribution of parameters inferred by the ABC-SMC sampler.

7 Conclusion

8 Discussion

Cited literature

1. Simon Tavaré, David J Balding, Robert C Griffiths, and Peter Donnelly. Inferring coalescence times from dna sequence data. *Genetics*, 145(2):505–518, 1997.
2. Paul Marjoram, John Molitor, Vincent Plagnol, and Simon Tavaré. Markov chain monte carlo without likelihoods. *Proceedings of the National Academy of Sciences*, 100(26):15324–15328, 2003.
3. Tina Toni, David Welch, Natalja Strelkowa, Andreas Ipsen, and Michael PH Stumpf. Approximate bayesian computation scheme for parameter inference and model selection in dynamical systems. *Journal of the Royal Society Interface*, 6(31):187–202, 2009.
4. Christian P Robert, Mark A Beaumont, Jean-Michel Marin, and Jean-Marie Cornuet. Adaptivity for abc algorithms: the abc-pmc scheme. *arXiv preprint arXiv:0805.2256*, 2008.
5. Mattias Åkesson, Prashant Singh, Fredrik Wrede, and Andreas Hellander. Convolutional neural networks as summary statistics for approximate bayesian computation. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 19(6):3353–3365, 2021.
6. Charles Fefferman, Sanjoy Mitter, and Hariharan Narayanan. Testing the manifold hypothesis. *Journal of the American Mathematical Society*, 29(4):983–1049, 2016.
7. Randall Balestriero, Mark Ibrahim, Vlad Sobal, Ari Morcos, Shashank Shekhar, Tom Goldstein, Florian Bordes, Adrien Bardes, Gregoire Mialon, Yuandong Tian, et al. A cookbook of self-supervised learning. *arXiv preprint arXiv:2304.12210*, 2023.
8. Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pages 4171–4186, 2019.
9. Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16000–16009, 2022.
10. Yuchong Yao, Nandakishor Desai, and Marimuthu Palaniswami. Masked contrastive representation learning. *arXiv preprint arXiv:2211.06012*, 2022.
11. Zhe Li, Zhongwen Rao, Lujia Pan, Pengyun Wang, and Zenglin Xu. Ti-mae: Self-supervised masked time series autoencoders. *arXiv preprint arXiv:2301.08871*, 2023.
12. Mingyue Cheng, Qi Liu, Zhiding Liu, Hao Zhang, Rujiao Zhang, and Enhong Chen. Timemae: Self-supervised representations of time series with decoupled masked autoencoders. *arXiv preprint arXiv:2303.00320*, 2023.
13. Michael I Jordan, Zoubin Ghahramani, Tommi S Jaakkola, and Lawrence K Saul. An introduction to variational methods for graphical models. *Machine learning*, 37:183–233, 1999.
14. David M. Blei, Alp Kucukelbir, and Jon D. McAuliffe. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877, April 2017.
15. Diederik P Kingma, Max Welling, et al. Auto-encoding variational bayes, 2013.

16. Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

17. Michael D McKay, Richard J Beckman, and William J Conover. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 42(1):55–61, 2000.

18. Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. In *International conference on learning representations*, 2017.

19. Stephan Rabanser, Tim Januschowski, Valentin Flunkert, David Salinas, and Jan Gasthaus. The effectiveness of discretization in forecasting: An empirical study on neural time series models. *arXiv preprint arXiv:2005.10111*, 2020.

20. Abdul Fatir Ansari, Lorenzo Stella, Caner Turkmen, Xiyuan Zhang, Pedro Mercado, Huibin Shen, Oleksandr Shchur, Syama Rangapuram, Sebastian Pineda Arango, Shubham Kapoor, et al. Chronos: Learning the language of time series. 2024.

21. David Salinas, Valentin Flunkert, Jan Gasthaus, and Tim Januschowski. Deepar: Probabilistic forecasting with autoregressive recurrent networks. *International journal of forecasting*, 36(3):1181–1191, 2020.

22. Jean-Michel Marin, Pierre Pudlo, Christian P Robert, and Robin J Ryder. Approximate bayesian computational methods. *Statistics and computing*, 22(6):1167–1180, 2012.

23. Umberto Simola, Jessi Cisewski-Kehe, Michael U Gutmann, and Jukka Corander. Adaptive approximate bayesian computation tolerance selection. *Bayesian analysis*, 16(2):397–423, 2021.

24. Emille EO Ishida, Sandro DP Vitenti, Mariana Penna-Lima, Jessi Cisewski, Rafael S de Souza, Arlindo MM Trindade, Ewan Cameron, Vinicius C Busti, COIN collaboration, et al. Cosmoabc: likelihood-free inference via population monte carlo approximate bayesian computation. *Astronomy and Computing*, 13:1–11, 2015.

25. Alfred James Lotka. *Elements of physical biology*. Williams & Wilkins, 1925.

26. Vito Volterra. Variations and fluctuations of the number of individuals in animal species living together. *ICES Journal of Marine Science*, 3(1):3–51, 1928.

27. Daniel T Gillespie. Exact stochastic simulation of coupled chemical reactions. *The journal of physical chemistry*, 81(25):2340–2361, 1977.

28. Khanh N Dinh, Zijin Xiang, Zhihan Liu, and Simon Tavaré. Approximate bayesian computation sequential monte carlo via random forests. *arXiv preprint arXiv:2406.15865*, 2024.