Automated Statistical Model Discovery with Language Models

Michael Y. Li [1] **Emily B. Fox** [1 2 3] **Noah D. Goodman** [1 4]

Abstract

Statistical model discovery is a challenging search over a vast space of models subject to domain-specific constraints. Efficiently searching over this space requires expertise in modeling and the problem domain. Motivated by the domain knowledge and programming capabilities of large language models (LMs), we introduce a method for *language model driven automated statistical model discovery*. We cast our automated procedure within the principled framework of Box's Loop: the LM iterates between proposing statistical models represented as *probabilistic programs*, acting as a modeler, and critiquing those models, acting as a domain expert. By leveraging LMs, we do not have to define a domain-specific language of models or design a handcrafted search procedure, which are key restrictions of previous systems. We evaluate our method in three settings in probabilistic modeling: searching within a restricted space of models, searching over an open-ended space, and improving expert models under natural language constraints (*e.g.,* this model should be interpretable to an ecologist). Our method identifies models on par with human expert designed models and extends classic models in interpretable ways. Our results highlight the promise of LM-driven model discovery.

1. Introduction

Modeling, or generating a parsimonious but explanatory representation of a complex system, is at the heart of scientific discovery. Model discovery is challenging because it involves searching over a vast space of candidate models subject to domain-specific constraints (*e.g.,* find the best model that remains interpretable to domain experts). Efficiently searching over the space requires extensive human expertise: modelers need broad knowledge of different modeling approaches and must work closely with domain experts to adapt these approaches to a given problem domain. As a concrete example, consider modeling blood-glucose dynamics in Type 1 diabetes (T1D) patients; accurately modeling these dynamics can enable better insulin regulation and reduce complications from the disease. To model these dynamics, modelers need to understand biomedical models that capture blood-glucose dynamics in idealized, lab settings but they also need to understand techniques for adapting these models to handle real data with noise and missingness (Miller et al., 2020). Domain experts play a crucial role in this process: modelers must work closely with clinicians to ensure that the model is consistent with human physiology. As this example illustrates, model discovery can require significant human expertise. Automating this process could accelerate and democratize scientific discovery.

Automated model discovery is not a new ambition. Previous systems have been successfully deployed for discovering physical laws (Bongard & Lipson, 2007; McKinney et al., 2006; Linka et al., 2023), reverse-engineering non-linear dynamical systems (Schmidt & Lipson, 2009), nonparametric regression (Duvenaud et al., 2013) and unsupervised learning (Grosse, 2014). However, in these systems, a human expert had to carefully design a domain specific language (DSL) of models and specify a hand-crafted search procedure for composing models in that DSL. For example, in the Automatic Statistician (Duvenaud et al., 2013; Lloyd et al., 2014), a system for nonparametric regression and time series modeling, human experts defined a DSL of Gaussian process kernels and a search procedure that composes kernels via addition and multiplication. Defining the DSL and the operators for composing models requires significant modeling expertise. These systems also compromised flexibility for automation: rather than choosing a model class best-suited for a problem, experts chose models that compose conveniently. This breaks the core principle of separation of modeling from inference (van de Meent et al., 2021).

As we saw above, there are two key roles in the model discovery pipeline: the domain expert and the modeler. We hypothesize that LMs can supplement these roles and re-

[1]Department of Computer Science, Stanford University [2]Department of Statistics, Stanford University [3]Chan Zuckerberg Biohub – San Francisco [4]Department of Psychology, Stanford University. Correspondence to: Michael Y. Li <michaelyli@stanford.edu>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

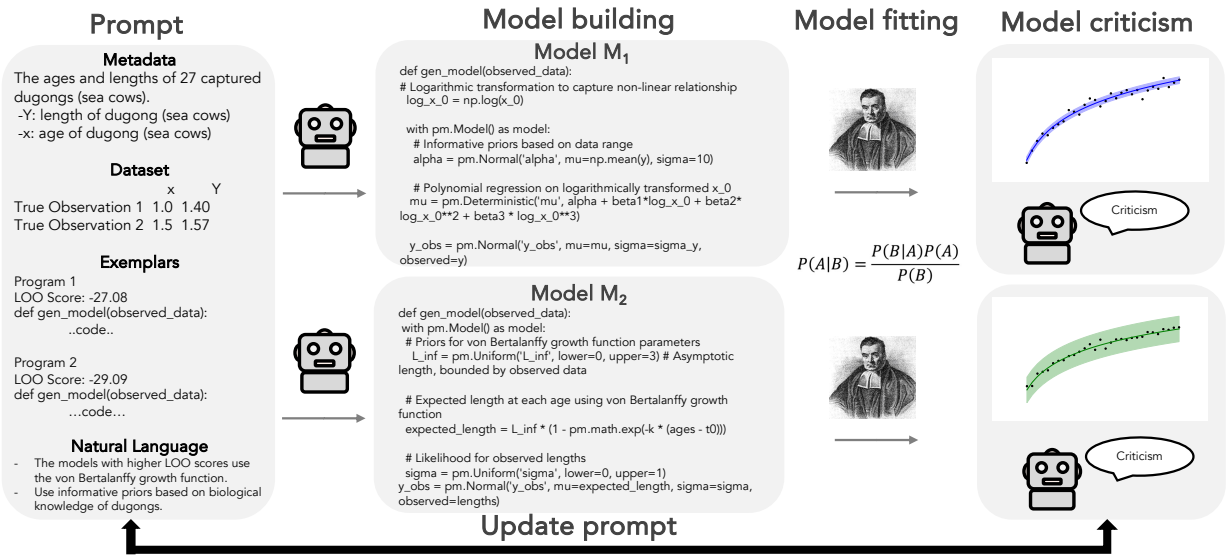

Figure 1. **Language model driven automated model discovery (`BoxLM`).** 1) The prompt for the LM contains the dataset in visual and/or textual form, dataset metadata (*e.g.,* dataset description), the code for previous probabilistic programs, and natural language feedback. 2) Given this, the proposal LM proposes new models expressed as *probabilistic programs*. 3) To fit these programs in a generic way, we leverage probabilistic programming languages and obtain scores and posterior predictive samples. 4) After we fit models, we compute the posterior predictive mean and variance. We provide these statistics to a critic LM which produces natural language feedback to guide the next round of model building. 5) We propagate the best programs, their posterior predictive means and variances, and natural language feedback forward by updating the prompt.

duce the human expertise required in model discovery. Our hypothesis is motivated by recent work exploring LM capabilities. First, LMs have been successfully applied to domains including law (Bommarito & Katz, 2022), medicine (Lee et al., 2023), and mathematics (Wu et al., 2023). This suggests that LMs have broad domain knowledge which may enable them to supplement the domain expert. Second, LMs can reliably write code (Rozière et al., 2023; Chen et al., 2021) which means we do not require a human expert to define a DSL. Instead, we can search over a more open-ended space of models, provided they can be expressed in a generic programming language like Python. Third, LMs have strong inductive reasoning capabilities (*e.g.,* they can generate hypotheses from limited data) (Wang et al., 2024; Qiu et al., 2024). We hypothesize these capabilities may enable them to reason about data (Zhong et al., 2023). These capabilities, in addition to their knowledge of various modeling approaches, may enable the LM to supplement the modeler.

Leveraging LMs for automated model discovery is enticing at a conceptual level. However, in order to deliver on this promising idea, we need to make several important design choices. These design choices should exploit the strengths of LMs but remain grounded in principled statistical modeling. First, we need a generic and flexible *representation* of a statistical model that can be expressed programmatically and is amenable to automated inference (*e.g.,* model fitting). Second, we need a method for *guiding LM proposals through natural language*.

Algorithm 1 Automated Model Discovery with LMs

input dataset $\mathcal{D}$, number of rounds T, k number of exemplars, m number of proposals per round, (optional) warm-start example z_0, function for scoring a program `score`, (optional) function for producing natural language feedback `criticize`
$\mathcal{Z} \leftarrow \emptyset$
while $t < T$ **do**
$\{z_i^t\}_{i=1}^m \sim q_{\text{LM}}(\cdot|\mathcal{Z}, z_0, h^t, \mathcal{D})$
$\{s_i\}_{i=1}^m \leftarrow \texttt{score-all}(\texttt{score}, \{z_i^t\}_{i=1}^m, \mathcal{D})$
$\mathcal{Z} \leftarrow \texttt{select-exemplars}(k, \{z_i^t\}_{i=1}^m, \{s_i\}_{i=1}^m)$
$h^{t+1} \leftarrow \texttt{criticize}(\{z_i^t\}_{i=1}^m, \{s_i\}_{i=1}^m, h^t)$
end while

To fulfill these requirements, we draw on research in probabilistic programming and inductive reasoning with LMs. In particular, we introduce the following method: LMs propose statistical models expressed as *probabilistic programs*, given a dataset and some metadata (*e.g.,* dataset description). We then fit these models using generic probabilistic inference techniques, compute statistics assessing the model fit, summarize findings from these statistics in natural language, and select exemplar models to guide the next round of proposals (Figure 1). Our approach is connected to recent

work on hypothesis search and inductive reasoning with LMs, but targets a fundamentally different problem (Wang et al., 2024; Qiu et al., 2024). While we also leverage the inductive reasoning capabilities of LMs, our focus is on *statistical modeling of noisy real-world data*, while previous work focused on learning deterministic input-output rules that can be implemented with standard Python programs.

We evaluate our method in three settings that cover common use cases in probabilistic modeling: searching *within* a DSL, searching over an *open-ended* space of probabilistic models, and improving expert models subject to modeling constraints expressed in natural language. In the first use case, we illustrate that our LM system is effective even in a DSL and matches the performance of the Automatic Statistician (Duvenaud et al., 2013). We then consider the more general setting of automatically constructing probabilistic models for real world data; crucially, we do not require a user to define a DSL and this generality is enabled by our choice to use probabilistic programs. Our method identifies probabilistic programs that match the performance of human expert written programs. In the third setting, we use LMs to improve classic models and illustrate a compelling advantage of using LMs for model discovery: given that certain modeling constraints can be difficult to express formally but easy to express in natural language (*e.g.,* this model should be more physical), we use natural language to guide LMs towards models that balance interpretability and flexibility.

2. Automated Box's Loop with Language Models

We begin with a brief background on the probabilistic modeling paradigm. We then formally introduce our problem setting and describe our approach. For an overview, see Figure 1 and Algorithm 1.

2.1. Background

In probabilistic modeling, our goal is to describe a dataset in terms of unobserved, latent structure. We describe a dataset through a *probabilistic model* which is a joint distribution $p(x, z|\eta)$; here $x = x_{1:N}$ denotes N observed data points, $z = z_{1:M}$ denotes M latent variables, and η corresponds to non-random quantities in the model (Blei, 2014). After specifying a probabilistic model, we fit the model to observed data. Fitting a model involves *inference*, or conditioning on observed data and computing the *posterior distribution* $p(z|x)$. After fitting a model, we perform *model criticism*. In the model criticism step, we evaluate the model by interrogating the posterior. In this work, our model criticism is inspired by a common model criticism technique known as a *posterior predictive check*: to identify discrepancies, samples are drawn from the posterior predictive distribution and statistics of these samples are compared against those of the observed data (Gelman et al., 2013). Model building and model criticism typically take place over multiple iterations in an iterative process known as *Box's Loop* (Box & Hunter, 1962).

There are many different representations of a probabilistic model. Since our goal is automated model discovery, we use *probabilistic programs*. Probabilistic programming languages provide ways to flexibly represent probabilistic models as programs and support generic inference methods for any arbitrary program (Wood et al., 2014; Goodman et al., 2008; van de Meent et al., 2021). In the context of automated model discovery, these are highly desirable properties, since LMs can write code reliably and other representations of probabilistic models can require custom inference methods. Our approach of using LMs to generate probabilistic programs is related to the approach taken by Wong et al. (2023) but is motivated by a different problem. Our emphasis is on using LMs as a tool for developing statistical models of real-world datasets, while their focus was on integrating symbolic methods with LMs.

2.2. Problem formulation

Our framework is motivated by recent work in inductive reasoning with LMs (Qiu et al., 2024; Wang et al., 2024), integrating tools with LMs (Gao et al., 2023), and driving LMs via linguistic feedback (Shinn et al., 2023).

At a high level, we consider a method for *learning probabilistic models from data* that involves two steps: a *model building* step and a *criticism* step[1]. Crucially, by learning a model, we mean searching over a space of model structures and *not just learning the parameters of some fixed model class*. In each step, we leverage LMs. In the proposal step, a *proposal LM* proposes probabilistic programs for a dataset. We then fit these probabilistic programs and evaluate them. In the criticism step, we provide a *critic LM* with programs and statistics assessing model fit (*e.g.,* model criticism statistics) and ask the critic LM to provide feedback to guide the next round of proposals.

We start with a dataset $\mathcal{D} = \{\mathbf{x_i}, y_i\}_{i=1}^n$. Here $\mathbf{x_i} \in \mathbb{R}^d$ are fixed d-dimensional input values (*e.g.,* features) and $y_i \in \mathbb{R}$ are the observations. Let Σ be the vocabulary of the LM. For each dataset, we have an associated metadata set $\mathcal{C} \in \Sigma^{d+1}$, which consists of natural language descriptions of $\mathcal{D}$ (*e.g.,* animal ages vs length) and natural language descriptions of each feature in $\mathcal{D}$ (*e.g.,* length of animal). Context informs how human modelers leverage prior knowledge; for example, if a modeler knows their dataset consists of monthly carbon dioxide measurements over a fifty-year

[1]To avoid overloading the word "model", we will refer to language models as LMs.

time span, they will choose a model that can capture periodicity and a linear trend. Our goal is to find a probabilistic program $z \sim \Sigma^*$ that maximizes some notion of quality, which we take here to be either the log marginal likelihood or expected log predictive density (ELPD) estimated via cross validation (LOO) (Vehtari et al., 2017).

2.3. Approach

Model Building Step In the *model building step*, we automatically generate probabilistic programs for modeling a dataset given information about the dataset and previously proposed programs. In particular, to generate candidates for round t, we sample m probabilistic programs z_i^t from the proposal LM, $q_{\text{LM}}(\cdot)$. In our experiments, we use GPT-4 V (Achiam et al., 2023) (`gpt4-11-06-preview`), which has multimodal capabilities. We leverage *in-context learning*, or LM's ability to learn from examples in a prompt, to guide the LM's proposals based on high-scoring programs in the past (Brown et al., 2020). Specifically, q_{LM} "conditions" on $h^t \in \Sigma^*$, a natural language instruction synthesizing previous modeling approaches and suggesting new approaches, k exemplars $\{z_1, \ldots z_k\}$, and a visual or textual representation of $\mathcal{D}$. Optionally, q_{LM} also conditions on a warm-start expert program z_0:

$$z_i^t \sim q_{\text{LM}}(\cdot | z_0, z_1, \ldots, z_k, h^{t-1}, \mathcal{D}).$$

We run this at a temperature of 0.7. Chain-of-thought reasoning, or generating intermediate reasoning steps, improves the performance of LMs (Wei et al., 2022; Kojima et al., 2022). Motivated by this, we instruct q_{LM} to reflect on the properties of the dataset or plot of the data, sketch a high-level modeling approach, state the hypotheses that it will address before writing a program, and add comments to code that address specific hypotheses.

To create exemplars $z_1, \ldots, z_k$ for round t for q_{LM}, we choose the best k programs among the m proposed programs in round $t-1$.

Model Fitting Step In the *model fitting step*, we fit a probabilistic program to data. This requires us to perform (approximate) inference for generic probabilistic models. To accomplish this, we leverage `pymc` (Abril-Pla et al., 2023), a Python probabilistic programming library. `pymc` automatically assigns a Markov Chain Monte Carlo (MCMC) sampler to perform inference; by default `pymc` uses a Hamiltonian Monte Carlo sampler (Homan & Gelman, 2014). Crucially, by using a probabilistic program, we decouple modeling from inference: the proposal LM's role is to build a good model, while `pymc` takes care of inference. Our approach of offloading computations to an external tool is connected to recent work enhancing LMs by giving them tools (*e.g.,* Python interpreter) (Gao et al., 2023; Schick et al., 2023).

Model Criticism Step In the *criticism step*, we ask the critic LM, p_{LM}, to produce natural language criticism of fitted models; we use this criticism to drive model revision. First, we can obtain a scalar score measuring the model fit (*e.g.,* ELPD LOO) for each proposed model. Second, to enable the critic LM to do something akin to a posterior predictive check, we obtain samples from the posterior predictive distribution (*e.g.,* $p(x'|z,x) = \int_z p(x'|z)p(z|x)$) and then compute summary statistics of these posterior predictive samples (Gelman et al., 2013). For simplicity, we compute the posterior predictive means and variances for each fitted probabilistic program. We then provide p_{LM} with select probabilistic programs, their scores $\mathcal{S}$ (*e.g.,* ELPD LOO), the posterior predictive means and variances $\mathcal{P}$, and the dataset itself $\mathcal{D}$; we explore both visual and textual dataframe based representations of the posterior predictive and the dataset. Finally, we ask p_{LM} to distill this criticism in natural language (*e.g.,* von Bertalanffy growth function with informative priors) which we use to drive the next round of model building (Shinn et al., 2023).

The key design choice is which programs to provide to p_{LM} (*e.g., critic exemplars*). Naively, we could provide all proposed programs across all rounds to p_{LM}. However, this list grows each round and has redundancy. Furthermore, LMs struggle to reason over long contexts. We therefore explore a simple approach to selecting exemplars where we provide p_{LM} with the top d programs $\tilde{z}_1, \ldots \tilde{z}_d$ from the current round t. To produce h^{t+1}, we sample:

$$h^{t+1} \sim p_{\text{LM}}(\cdot | \tilde{z}_1, \ldots, \tilde{z}_d, \mathcal{D}, \mathcal{S}, \mathcal{P}).$$

In the appendix, we report additional results for an approach inspired by a state-space update (Baum & Petrie, 1966; Kalman, 1960; Rabiner, 1989). To avoid storing the entire history of proposed programs, we interpret h^{t+1} as a latent state and compute it using the previous state h^t and the new fitted programs $\{z_i\}_{i=1}^m$ at round t:

$$h^{t+1} \sim p_{\text{LM}}(\cdot | z_1, \ldots, z_m, h^t, \mathcal{D}, \mathcal{S}, \mathcal{P}).$$

In practice, we implement this by asking p_{LM} to add and delete hypotheses. We find that these two approaches have similar performances. We run this criticism step at a temperature of 0.0 to limit stochasticity in the criticism produced.

We refer to our full LM-driven automated Box's loop as `BoxLM`.

3. Experiments

3.1. Searching over a DSL: automated Gaussian process kernel discovery

We first evaluate LM's ability to search over a constrained space of models; in some settings, a domain expert may require the modeler to search over a DSL for reasons such as

Dataset	BoxLM+	BoxLM	Periodic	AS	SM	N-BEATS
Air	0.08	0.07	0.15	0.19	**0.06**	0.22
Beer	0.07	0.22	0.15	0.06	**0.05**	0.02
Heart	0.21	0.21	**0.20**	0.21	0.21	0.07
Milk	0.12	0.10	0.10	0.11	**0.09**	0.04
Wine	0.14	0.16	0.21	**0.13**	0.18	0.17
Wool	0.20	0.19	0.19	0.23	**0.13**	0.18

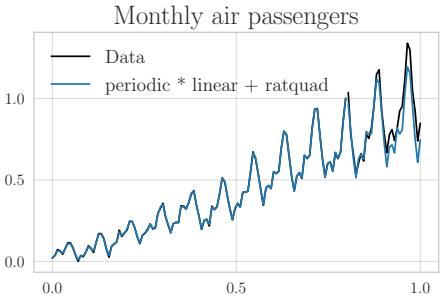

Figure 2. **Test set performance on time series datasets.** Our BoxLM system identifies compositional kernels with performance on par with strong baselines. **(left)** Comparison of BoxLM test mean absolute error (MAE) against Automatic Statistician using greedy search (AS), spectral mixture kernel (SM), periodic kernel (Periodic), and N-BEATS. BoxLM+ searches over an augmented kernel space. We bold the best and underline the second best among the GP methods, treating N-BEATS as a powerful non-GP-constrained baseline. **(right)** Extrapolations from GP with a BoxLM-discovered kernel.

interpretability. In particular, we consider time-series modeling with Gaussian processes (GPs) as in the Automatic Statistician (Duvenaud et al., 2013). GPs are probabilistic models that specify a distribution over functions. GPs are defined by a mean function and a positive-definite kernel function $k(\mathbf{x}, \mathbf{x}')$ that gives the covariance between $f(\mathbf{x})$ and $f(\mathbf{x}')$ as a function of $\mathbf{x}$ and $\mathbf{x}'$. The properties of a GP depend significantly on the kernel and therefore choosing a kernel that reflects domain knowledge (*e.g.,* linearity, periodicity) is a key design choice. One common way to produce a more flexible kernel is to compose kernels via addition and multiplication, leveraging the closure properties of kernels. Here, we evaluate LMs' ability to search over a space of kernels to identify an appropriate composition of kernels.

Setup Mirroring Duvenaud et al. (2013), we define a space of base kernels. In the prompt, we ask q_{LM} to perform one of the three operations (addition, multiplication, and replacement) on one of the in-context exemplar programs. For the first round, we ask q_{LM} to produce an initial guess based on the structure of the dataset, which we provide as a plot. Given a kernel expression, we learn the kernel hyperparameters via gradient-based optimization of the marginal likelihood.

Results We evaluate our method on six common univariate time series datasets. We compare against the Automatic Statistician, a greedy algorithm proposed by Duvenaud et al. (2013) (run until a depth of 10), and two GP baselines: a periodic kernel and a spectral mixture kernel, a strong non-compositional baseline. We also compare against N-BEATS, a strong neural baseline (Wilson & Adams, 2013; Oreshkin et al., 2019). In Figure 2, we compare the mean absolute error (MAE) on held-out test data for all datasets. To account for stochasticity in the Automatic Statistician implementation we used (Saad et al., 2023) and stochasticity in different repetitions of our pipeline, we average the test MAE across three model runs. Our method, denoted LM in the table, matches the performance of the Automatic Statistician, showing that BoxLM can efficiently search over a constrained space of models. We also experiment with augmenting the base kernel space with additional kernels (denoted BoxLM+ in the table). In some cases, this additional flexibility is beneficial. For example, by using the additional kernels, BoxLM+ can much better capture the Australian beer sales dataset than LM; in other cases, the additional flexibility does not appreciably improve performance. On the right panel, we show the extrapolations and identified kernel for the monthly air passenger dataset; BoxLM+ identifies a kernel with a periodic times linear component to capture the increasing amplitude in the data.

3.2. Open-ended probabilistic model discovery for real-world datasets

By integrating LMs into the model discovery process, we can search over a much broader class of models. Here, we explore BoxLMs' ability to automatically construct pymc probabilistic programs (Abril-Pla et al., 2023) for datasets.

Dataset We consider four real world datasets from the Stan PosteriorDB dataset (Magnusson et al., 2023): (1) a dataset consisting of average improvements in SAT scores after an SAT improvement program across eight different high schools, (2) a dataset consisting of ages of twenty-seven dugongs and their lengths, (3) a surgical dataset consisting of mortality rates in twelve hospitals performing cardiac surgery on babies, and (4) a dataset of peregrine population counts in the French Jura from 1964 to 2003. Each dataset has an associated human expert written probabilistic program in Stan, which we translate into pymc. These expert programs are generally open-source contributions from the Stan developer community. The datasets cover

common modeling motifs, such as hierarchical modeling and regression.

Ablations To study how domain knowledge affects the LM's modeling capabilities and to mitigate concerns about dataset leakage, we consider two ablations. In the first ablation, we construct a *simulated* analog for each dataset: for each dataset, we generate synthetic observations by sampling from the prior of a generative model or by sampling from a model with fixed parameters. We denote these datasets as *simulated* datasets. For example, to generate a simulated dataset for the `eight schools` dataset, we sample from the prior of the human expert program. In some cases, such as the `dugongs` dataset, the prior distributions over parameters are highly unconstrained and we therefore fix values for the parameters instead. For example, to generate a simulated analog of the `dugongs` dataset, we fix the values of the parameters α, β, γ used in the expert model $y_i = \alpha - \beta\gamma^{x_i} + \epsilon_i$ where $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$. Since it is unlikely that these simulated datasets appeared in the training data, this ablation helps mitigate concerns about dataset leakage.

In the second ablation, we remove the dataset *metadata* from the LM prompt. In particular, we remove the dataset description, replace the column names with domain-agnostic column names (*e.g.,* x_0, y), and replace the axis labels with uninformative labels. We denote these datasets as *no metadata* datasets. By removing this metadata, we can characterize how LMs use domain knowledge.

Quantitative Results In Table 1, we compare the performance of our method across different datasets and ablations relative to the expert programs. We emphasize that the expert programs are strong baselines, especially for the simulated datasets where the expert programs generated the data. We highlight numbers corresponding to significant differences, where significance is defined as an ELPD LOO difference of greater than four times the standard error estimate.

`BoxLM` reliably identifies programs on par with expert programs; we validate convergence using standard Markov Chain Monte Carlo diagnostics. Interestingly, removing metadata or switching to simulated datasets does not generally reduce performance relative to the expert program. The main exception is the `surgical` dataset. By replacing the labels with x_0 and y, `BoxLM` formulates this problem as a regression problem instead of using a hierarchical model like the expert model. For the `peregrine` dataset, removing metadata improves performance. We discuss this further in the next section.

Qualitative Analysis: Language Models are Domain Experts Earlier, we asked whether LMs can reliably play the role of a domain expert. In our ablations, we study how metadata influences the model search process by examining how removing metadata changes the programs proposed by `BoxLM`. In Figure 3, we plot the model fit and list the corresponding `BoxLM` programs in the figure caption. For the `peregrine` dataset, when given all the metadata, `BoxLM` models this dataset using a logistic growth model, which is motivated by the problem domain. However, logistic growth models cannot capture the initial decline in population. Interestingly, when we remove all metadata, `BoxLM` performs better, because it uses a quartic polynomial to model the data. This closes the gap in performance with the expert program. This highlights how prior knowledge can have a (sometimes overly) strong influence on the modeling choices of `BoxLM`. We observe a similar trend with the `dugongs` growth curve dataset. When given metadata, `BoxLM` uses a von Bertalanffy growth function (von Bertalanffy, 1949), which is commonly used to model animal growth. When we remove all metadata, `BoxLM` uses a polynomial function with a logarithmically transformed value for the inputs. Here, both modeling approaches fit the data well. Interestingly, `BoxLM` sets the prior parameters in these models in a data-informed way. For example, `BoxLM` sets the asymptotic length in the von Bertanlanffy function based on the observed lengths in the dataset. See code snippet in Figure 7 of the Appendix. For the eight schools dataset, `BoxLM` identifies a hierarchical model even without metadata, illustrating that model criticism can compensate for lack of domain knowledge. We illustrate the improvement round to round in Figure 6.

Dataset	Expert	LM
Eight schools	-30.70	-30.42
Eight schools sim	-18.09	-18.31
Eight schools sim no metadata	-18.09	-16.36
Dugongs	22.52	23.40
Dugongs sim	50.04	57.40
Dugongs sim no metadata	50.04	55.24
Surgical	-40.29	-38.03
Surgical sim	-39.80	-38.38
Surgical sim no metadata	**-39.80**	-63.72
Peregrine	**-142.19**	-173.11
Peregrine sim	**-130.48**	-179.06
Peregrine sim no meta	-130.48	-136.39

Table 1. **Comparison of `BoxLM` programs against expert programs** We perform this comparison across four different datasets and two different ablations that replace observations with synthetic observations and remove all metadata. We report the expected predictive log density estimated via leave-one-out cross validation. We bold statistically significant differences and underline non-significant differences. LM programs match the performance of human expert programs on 9/12 datasets.

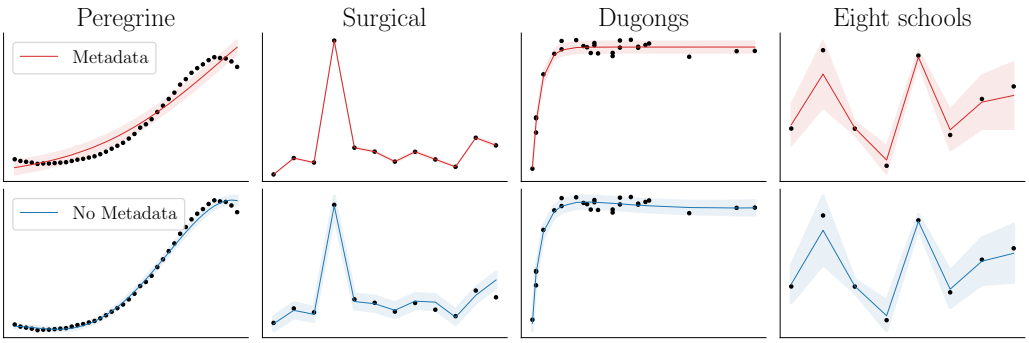

Figure 3. **Domain knowledge shapes BoxLM modeling approaches.** In the top row, we keep all metadata (*e.g.,* dataset description). In the bottom row, we remove metadata that reveals information about the domain. This leads to qualitatively different approaches for three datasets; for eight schools, BoxLM discovers a hierarchical model even without metadata. We list the corresponding programs for these different ablations: **(top row)** $\frac{K}{(1+((K-P_0)/P_0)\exp(-rt))}$, $\text{BetaBin}(n, \alpha, \beta)$, $L_{\text{inf}}(1 - \exp(-k(\text{age} - t_0)))$; **(bottom row)** $a + bx_0 + cx_0^2 + dx_0^3 + ex_0^4$, $\alpha + \beta x_0 + \gamma x_0^2$, $\alpha + \beta_1 \log x_0 + \beta_2 \log x_0^2 + \beta_3 \log x_0^3$.

3.3. Improving classic models under modeling constraints

In the previous experiment, we explored BoxLM's ability to identify models *tabula rasa* (*e.g.,* without any initial seed model). However, in many scientific settings, we begin with a well-known model and are tasked with improving it. Here, we explore BoxLM's ability to improve upon a Lotka-Volterra model of predator-prey dynamics. In addition, a crucial component of model discovery is respecting "soft" modeling constraints that are easy to express in natural language but hard to formalize (*e.g.,* ecologists should think this is a plausible model). We therefore illustrate another advantage of BoxLM: we can express these modeling constraints in natural language and use them to drive LM proposals.

Dataset To create our dataset, we simulate data from the following "perturbed" Lotka-Volterra dynamics

$$\frac{db}{dt} = \alpha b - \beta bc$$
$$\frac{dc}{dt} = -\gamma c + \delta bc^{0.95}.$$

Setup BoxLM is tasked with implementing an ODE model in Python using Jax (Bradbury et al., 2018). Estimating the parameters of Lotka-Volterra models via Bayesian inference is challenging. We instead learn the parameters via gradient descent which can be straightforwardly implemented using modern automatic differentiation libraries (Chen et al., 2018; Kidger, 2021); in particular, we use diffrax (Kidger, 2021), a Jax-based ODE library that supports learning ODE parameters via backpropagation. We consider three variations: *warm-start with constraints* (WS-Constraint), *warm-start with no constraints* (WS-No Constraint), and *no warm-start* (No-WS) that differ in their initial seed program and the initial instructions which express modeling constraints. In all variations, we provide the LM with a scatter plot of training datapoints. In the *no warm-start* variation, we provide the LM with an implementation of standard Lotka-Volterra dynamics using diffrax and the predictions obtained from fitting standard Lotka-Volterra to the training data. In both *warm-start* variations, we provide the LM with (1) an implementation of a hybrid neural ODE that introduces an additive correction term to the predator dynamics, parameterized by a multilayer perceptron (MLP), and (2) the predictions obtained from fitting this model to the training data. In the constrained warm-start variation, we ask the LM to produce a hybrid neural ODE model that is interpretable to an ecology expert who suggested a Holling's type II response (Rosenzweig & MacArthur, 1963). In the unconstrained warm-start variation, we provide the same seed program but do not impose this additional interpretability constraint. For models with both neural and physical components, we employ a two-stage learning procedure so that the neural component does not dominate the dynamics; see Appendix C for details.

Results In Figure 4 (left), we plot the predictions obtained from integrating the learned dynamics for an ODE proposed by BoxLM, the training data points generated from the true dynamics, and the predictions from the standard Lotka-Volterra model. We fit free parameters to the training data via gradient descent. The grey region indicates the extrapolation region.

BoxLM can significantly improve upon the standard Lotka-Volterra model by introducing corrections to the dynamics (Figure 4). The standard Lotka-Volterra model cannot capture the decreasing amplitude in the data; furthermore, there is a slight phase shift relative to the training datapoints. In contrast, BoxLM identifies an ODE that captures these properties and extrapolates accurately. In Figure 4 (right), we compare these programs against a neural ODE base-

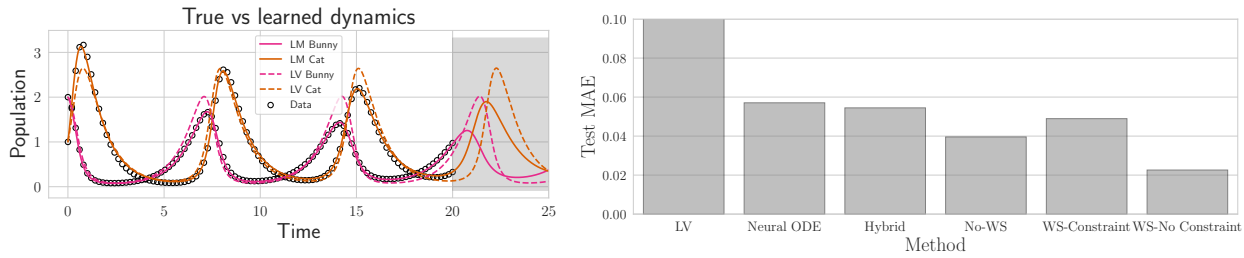

Figure 4. **Correcting misspecified Lotka-Volterra dynamics.** `BoxLM` can introduce corrections to standard Lotka-Volterra dynamics (no warm-start) and a hybrid neural ODE approach (warm-start) that outperform several baselines. **(left)** `LM`-proposed model predictions on training data and extrapolations (grey region). **(right)** Test MAE of `LM` models (`No-WS`, `WS-Constraint`, and `WS-No Constraint`) compared to the standard Lotka-Volterra model `LV`, a `Neural ODE`, and a hybrid Neural ODE model with a multiplicative correction to the prey-predator dynamics (`Hybrid`).

LM proposed Lotka-Volterra programs

```
def no_warm_start(t, y, _coeffs):
  # Hypothesis 1: Logistic growth for prey with carrying capacity kappa
  # Hypothesis 2: Saturation effect in predation rate with parameter psi
  db_dt = alpha * b * (1 - b / kappa) - beta * b * c / (1 + psi * b)
  dc_dt = -gamma * c + delta * b * c - epsilon * c**2
```

```
def warm_start_constrained(t, y, _mlp):
  inputs = jnp.array([b])  # Input is prey density (bunny population)
  # Neural network learns a modifier for the handling time based on prey density
  handling_time_modifier = _mlp(inputs)
  # Modulate the predation rate beta in a non-linear manner according to handling time
  db_dt = alpha * b - beta * b * c / (1 + handling_time_modifier[0] * b)
  dc_dt = -gamma * c + delta * b * c / (1 + handling_time_modifier[0] * b)
```

```
def warm_start_unconstrained(t, y, _mlp):
  inputs = jfu.ravel_pytree((b, c))[0]
  mlp_output = _mlp(inputs)
  # Fine-tuned scaling of the MLP output to match the amplitude of data more closely
  db_dt = alpha * b - beta * b * c + 0.02 * mlp_output[0] # Reduced scaling for bunnies
  dc_dt = -gamma * c + delta * b * c + 0.06 * mlp_output[1] # Increased scaling for cats
```

Figure 5. **`BoxLM` can propose corrections to ODEs. (top)** In the no warm-start (`No-WS`) variation, `BoxLM` introduces corrections informed by domain knowledge of predator-prey models (carrying capacity, predation saturation). **(middle)** When prompted to introduce neural networks in an interpretable way (`WS-Constraint`), one strategy `BoxLM` proposes is to make the handling time parameter depend non-linearly on the prey density, extending a traditional approach to modeling predation saturation. **(bottom)** When prompted to introduce neural networks without constraints (`WS-No Constraint`), `BoxLM` introduces additive MLP-parameterized corrections and adjusts the scaling factors.

line, and a hybrid neural ODE baseline that incorporates a multiplicative correction (parameterized by an MLP) to the predator-prey interaction term in the predator equation. See Section C of the Appendix for details on these baselines. For the `LM` variations, we report the average test MAE across three runs. In Figure 4, we see that *all* `BoxLM` variations outperform the baselines.

In Figure 5, we present code snippets corresponding to representative programs proposed in the constrained and unconstrained variations. These snippets show how natural language constraints can guide `BoxLM` towards more flexible models that retain interpretability. For the variation with no natural language constraints (WS-No Constraint), `BoxLM` takes a purely empirical approach. In particular, `BoxLM` adjusts the scaling terms on the additive `MLP` correction

term. For the WS-Constraint variation, BoxLM proposes a hybrid approach integrating the neural approach in the prompt with classic models in the literature; importantly, even though BoxLM is asked to balance interpretability and flexibility, BoxLM still identifies programs that outperform the neural ODE and standard Lotka-Volterra baselines. One approach BoxLM proposes is an extension of the Rosenzweig and MacArthur model with a Holling's type II functional response (Rosenzweig & MacArthur, 1963) to allow a static parameter to depend dynamically on the prey density: BoxLM models the handling time, or the time a predator spends "processing" a prey, as a nonlinear function of the prey density via an MLP. These results show how we can use natural language to drive BoxLM towards models that balance flexibility and interpretability.

4. Conclusion

We introduced a method for leveraging LMs for automated model discovery. Our method can identify models that perform favorably against strong baselines and improve upon expert models. We also studied how domain knowledge and natural language constraints influence our system. Altogether, our results highlight the compelling advantages of LM-driven statistical model discovery.

Our work has important limitations that motivate future research. First, we focused on modeling static datasets. An interesting direction could be leveraging LMs for active data collection. Second, since our tasks were restricted to one-dimensional datasets, simple model criticism statistics were sufficient and therefore decided in advance (residuals, posterior predictive mean). Another interesting future direction could be fully automating the criticism step. Finally, while in-context learning was effective in our tasks, we could explore finetuning techniques for training a language model to produce better probabilistic programs.

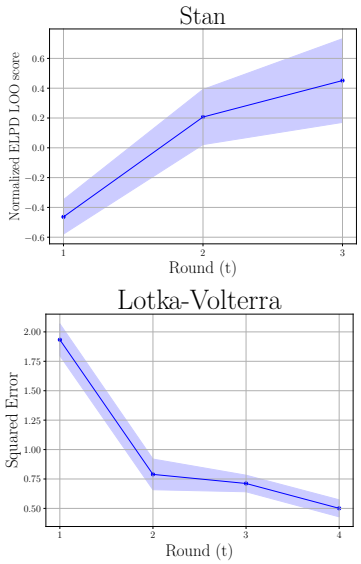

Figure 6. **Round-to-round improvement.** Model revision leads to improvements on average. (Improvement is not necessarily monotonic for a given dataset and run.) **(top)** ELPD LOO score vs. round for Stan experiments. We normalize the ELPD LOO scores across programs proposed for 3 rounds of Box's loop and average across datasets for the no metadata condition. Larger is better and error bars correspond to standard error. **(bottom)** Squared error vs round for LV experiment. We report the squared error averaged across three different runs, for the warm-start, no constraint condition; smaller is better.

Acknowledgements

This work was supported in part by AFOSR Grant FA9550-21-1-0397, ONR Grant N00014-22-1-2110, and an NSF Expeditions Grant, Award Number (FAIN) 1918771. EBF is a Chan Zuckerberg Biohub – San Francisco Investigator.

We thank Eric Zelikman and Neil Band for detailed feedback on this paper. We also thank Omar Shaikh and Jensen Gao for helpful discussions.

Impact Statement

This paper presents work whose goal is to partially automate statistical modeling. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Abril-Pla, O., Andreani, V., Carroll, C., Dong, L., Fonnesbeck, C. J., Kochurov, M., Kumar, R., Lao, J., Luhmann, C. C., Martin, O. A., Osthege, M., Vieira, R., Wiecki, T., and Zinkov, R. PyMC: a modern, and comprehensive probabilistic programming framework in python. *PeerJ Computer Science*, 9, 2023.

Achiam, O. J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., Avila, R., Babuschkin, I., Balaji, S., Balcom, V., Baltescu, P., Bao, H., Bavarian, M., Belgum, J., Bello, I., Berdine, J., Bernadett-Shapiro, G., Berner, C., Bogdonoff, L., Boiko, O., Boyd, M., Brakman, A.-L., Brockman, G., Brooks, T., Brundage, M., Button, K., Cai, T., Campbell, R., Cann, A., Carey, B., Carlson, C., Carmichael, R., Chan, B., Chang, C., Chantzis, F., Chen, D., Chen, S., Chen, R., Chen, J., Chen, M., Chess, B., Cho, C., Chu, C., Chung, H. W., Cummings, D., Currier, J., Dai, Y., Decareaux, C., Degry, T., Deutsch, N., Deville, D., Dhar, A., Dohan, D., Dowling, S., Dunning, S., Ecoffet, A., Eleti, A., Eloundou, T., Farhi, D., Fedus, L., Felix, N., Fishman, S. P., Forte, J., Fulford, I., Gao, L., Georges, E., Gibson, C., Goel, V., Gogineni, T., Goh, G., Gontijo-Lopes, R., Gordon, J., Grafstein, M., Gray, S., Greene, R., Gross, J., Gu, S. S., Guo, Y., Hallacy, C., Han, J., Harris, J., He, Y., Heaton, M., Heidecke, J., Hesse, C., Hickey, A., Hickey, W., Hoeschele, P., Houghton, B., Hsu, K., Hu, S., Hu, X., Huizinga, J., Jain, S., Jain, S., Jang, J., Jiang, A., Jiang, R., Jin, H., Jin, D., Jomoto, S., Jonn, B., Jun, H., Kaftan, T., Kaiser, L., Kamali, A., Kanitscheider, I., Keskar, N. S., Khan, T., Kilpatrick, L., Kim, J. W., Kim, C., Kim, Y., Kirchner, H., Kiros, J. R., Knight, M., Kokotajlo, D., Kondraciuk, L., Kondrich, A., Konstantinidis, A., Kosic, K., Krueger, G., Kuo, V., Lampe, M., Lan, I., Lee, T., Leike, J., Leung, J., Levy, D., Li, C. M., Lim, R., Lin, M., Lin, S., Litwin, M., Lopez, T., Lowe, R., Lue, P., Makanju, A. A., Malfacini, K., Manning, S., Markov, T., Markovski, Y., Martin, B., Mayer, K., Mayne, A., McGrew, B., McKinney, S. M., McLeavey, C., McMillan, P., McNeil, J., Medina, D., Mehta, A., Menick, J., Metz, L., Mishchenko, A., Mishkin, P., Monaco, V., Morikawa, E., Mossing, D. P., Mu, T., Murati, M., Murk, O., M'ely, D., Nair, A., Nakano, R., Nayak, R., Neelakantan, A., Ngo, R., Noh, H., Long, O., O'Keefe, C., Pachocki, J. W., Paino, A., Palermo, J., Pantuliano, A., Parascandolo, G., Parish, J., Parparita, E., Passos, A., Pavlov, M., Peng, A., Perelman, A., de Avila Belbute Peres, F., Petrov, M., de Oliveira Pinto, H. P., Pokorny, M., Pokrass, M., Pong, V. H., Powell, T., Power, A., Power, B., Proehl, E., Puri, R., Radford, A., Rae, J., Ramesh, A., Raymond, C., Real, F., Rimbach, K., Ross, C., Rotsted, B., Roussez, H., Ryder, N., Saltarelli, M. D., Sanders, T., Santurkar, S., Sastry, G., Schmidt, H., Schnurr, D., Schulman, J., Selsam, D., Sheppard, K., Sherbakov, T., Shieh, J., Shoker, S., Shyam, P., Sidor, S., Sigler, E., Simens, M., Sitkin, J., Slama, K., Sohl, I., Sokolowsky, B. D., Song, Y., Staudacher, N., Such, F. P., Summers, N., Sutskever, I., Tang, J., Tezak, N. A., Thompson, M., Tillet, P., Tootoonchian, A., Tseng, E., Tuggle, P., Turley, N., Tworek, J., Uribe, J. F. C., Vallone, A., Vijayvergiya, A., Voss, C., Wainwright, C., Wang, J. J., Wang, A., Wang, B., Ward, J., Wei, J., Weinmann, C., Welihinda, A., Welinder, P., Weng, J., Weng, L., Wiethoff, M., Willner, D., Winter, C., Wolrich, S., Wong, H., Workman, L., Wu, S., Wu, J., Wu, M., Xiao, K., Xu, T., Yoo, S., Yu, K., Yuan, Q., Zaremba, W., Zellers, R., Zhang, C., Zhang, M., Zhao, S., Zheng, T., Zhuang, J., Zhuk, W., and Zoph, B. GPT-4 Technical Report. 2023.

Baum, L. E. and Petrie, T. Statistical inference for probabilistic functions of finite state markov chains. *The annals of mathematical statistics*, 37(6):1554–1563, 1966.

Blei, D. M. Build, compute, critique, repeat: Data analysis with latent variable models. *Annual Review of Statistics and Its Application*, 1(1):203–232, 2014.

Bommarito, M. J. and Katz, D. M. GPT Takes the Bar Exam. *ArXiv*, abs/2212.14402, 2022.

Bongard, J. C. and Lipson, H. Automated reverse engineering of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 104:9943 – 9948, 2007.

Box, G. E. P. and Hunter, W. G. A useful method for model-building. *Technometrics*, 4:301–318, 1962.

Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., and Zhang, Q. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/google/jax.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. Language models are few-shot learners. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS'20, Red Hook, NY, USA, 2020. Curran Associates Inc. ISBN 9781713829546.

Chen, M., Tworek, J., Jun, H., Yuan, Q., Ponde, H., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., Ray, A., Puri, R., Krueger, G., Petrov, M., Khlaaf, H., Sastry, G., Mishkin, P., Chan, B., Gray, S., Ryder, N., Pavlov, M., Power, A., Kaiser, L., Bavarian, M., Winter, C., Tillet, P., Such, F. P., Cummings, D. W., Plappert, M.,

Chantzis, F., Barnes, E., Herbert-Voss, A., Guss, W. H., Nichol, A., Babuschkin, I., Balaji, S., Jain, S., Carr, A., Leike, J., Achiam, J., Misra, V., Morikawa, E., Radford, A., Knight, M. M., Brundage, M., Murati, M., Mayer, K., Welinder, P., McGrew, B., Amodei, D., McCandlish, S., Sutskever, I., and Zaremba, W. Evaluating large language models trained on code. *ArXiv*, abs/2107.03374, 2021.

Chen, R. T. Q., Rubanova, Y., Bettencourt, J., and Duvenaud, D. Neural ordinary differential equations. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, pp. 6572–6583, Red Hook, NY, USA, 2018. Curran Associates Inc.

Duvenaud, D., Lloyd, J., Grosse, R., Tenenbaum, J., and Zoubin, G. Structure discovery in nonparametric regression through compositional kernel search. In Dasgupta, S. and McAllester, D. (eds.), *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pp. 1166–1174, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR.

Gao, L., Madaan, A., Zhou, S., Alon, U., Liu, P., Yang, Y., Callan, J., and Neubig, G. PAL: program-aided language models. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023.

Gelman, A. and Rubin, D. B. Inference from iterative simulation using multiple sequences. *Statistical Science*, 7(4): 457–472, 1992. ISSN 08834237.

Gelman, A., Carlin, J. B., Stern, H. S., Dunson, D. B., Vehtari, A., and Rubin, D. B. Bayesian data analysis, third edition. 2013.

Goodman, N. D., Mansinghka, V. K., Roy, D. M., Bonawitz, K., and Tenenbaum, J. B. Church: a language for generative models. In *Conference on Uncertainty in Artificial Intelligence*, 2008.

Grosse, R. B. Model selection in compositional spaces. 2014.

Homan, M. D. and Gelman, A. The no-u-turn sampler: adaptively setting path lengths in hamiltonian monte carlo. *J. Mach. Learn. Res.*, 15(1):1593–1623, jan 2014. ISSN 1532-4435.

Kalman, R. E. A new approach to linear filtering and prediction problems. *ASME Journal of Basic Engineering*, 82(1):35–45, March 1960.

Kidger, P. *On Neural Differential Equations*. PhD thesis, University of Oxford, 2021.

Kojima, T., Gu, S. S., Reid, M., Matsuo, Y., and Iwasawa, Y. Large language models are zero-shot reasoners. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K. (eds.), *Advances in Neural Information Processing Systems*, 2022.

Lee, P., Bubeck, S., and Petro, J. Benefits, Limits, and Risks of GPT-4 as an AI Chatbot for Medicine. *New England Journal of Medicine*, 388(13):1233–1239, 2023. doi: 10.1056/NEJMsr2214184. PMID: 36988602.

Linka, K., St. Pierre, S. R., and Kuhl, E. Automated model discovery for human brain using constitutive artificial neural networks. *Acta Biomaterialia*, 160:134–151, 2023. ISSN 1742-7061. doi: https://doi.org/10.1016/j.actbio.2023.01.055.

Lloyd, J. R., Duvenaud, D. K., Grosse, R. B., Tenenbaum, J. B., and Ghahramani, Z. Automatic construction and natural-language description of nonparametric regression models. In *AAAI Conference on Artificial Intelligence*, 2014.

Magnusson, M., Bürkner, P., and Vehtari, A. posteriordb: a set of posteriors for Bayesian inference and probabilistic programming, October 2023.

McKinney, B. A., Crowe, J. E., Voss, H. U., Crooke, P. S., Barney, N., and Moore, J. H. Hybrid grammar-based approach to nonlinear dynamical system identification from biological time series. *Phys. Rev. E*, 73:021912, Feb 2006. doi: 10.1103/PhysRevE.73.021912.

Miller, A. C., Foti, N. J., and Fox, E. Learning insulin-glucose dynamics in the wild. In *Proceedings of the 5th Machine Learning for Healthcare Conference*, volume 126 of *Proceedings of Machine Learning Research*, pp. 172–197. PMLR, 07–08 Aug 2020.

Oreshkin, B. N., Carpov, D., Chapados, N., and Bengio, Y. N-beats: Neural basis expansion analysis for interpretable time series forecasting. *ArXiv*, abs/1905.10437, 2019.

Qiu, L., Jiang, L., Lu, X., Sclar, M., Pyatkin, V., Bhagavatula, C., Wang, B., Kim, Y., Choi, Y., Dziri, N., and Ren, X. Phenomenal yet puzzling: Testing inductive reasoning capabilities of language models with hypothesis refinement. In *The Twelfth International Conference on Learning Representations*, 2024.

Rabiner, L. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989. doi: 10.1109/5.18626.

Rosenzweig, M. and MacArthur, R. Graphical representation and stability conditions of predator-prey interaction. *American Naturalist*, 97:209–223, 1963.

Rozière, B., Gehring, J., Gloeckle, F., Sootla, S., Gat, I., Tan, X., Adi, Y., Liu, J., Remez, T., Rapin, J., Kozhevnikov, A., Evtimov, I., Bitton, J., Bhatt, M. P., Ferrer, C. C.,

Grattafiori, A., Xiong, W., D'efossez, A., Copet, J., Azhar, F., Touvron, H., Martin, L., Usunier, N., Scialom, T., and Synnaeve, G. Code Llama: Open Foundation Models for Code. *ArXiv*, abs/2308.12950, 2023.

Saad, F. A., Patton, B., Hoffman, M., Saurous, R. A., and Mansinghka, V. K. Sequential monte carlo learning for time series structure discovery. In *International Conference on Machine Learning*, 2023.

Schick, T., Dwivedi-Yu, J., Dessì, R., Raileanu, R., Lomeli, M., Zettlemoyer, L., Cancedda, N., and Scialom, T. Toolformer: Language models can teach themselves to use tools. *ArXiv*, abs/2302.04761, 2023.

Schmidt, M. and Lipson, H. Distilling free-form natural laws from experimental data. *Science*, 324(5923):81–85, 2009. doi: 10.1126/science.1165893.

Shinn, N., Cassano, F., Gopinath, A., Narasimhan, K. R., and Yao, S. Reflexion: language agents with verbal reinforcement learning. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

van de Meent, J.-W., Paige, B., Yang, H., and Wood, F. An introduction to probabilistic programming, 2021.

Vehtari, A., Gelman, A., and Gabry, J. Practical Bayesian model evaluation using leave-one-out cross-validation and WAIC. *Statistics and Computing*, 27(5): 1413–1432, sep 2017. ISSN 0960-3174. doi: 10.1007/s11222-016-9696-4.

von Bertalanffy, L. Problems of organic growth. *Nature*, 163:156–158, 1949.

Wang, R., Zelikman, E., Poesia, G., Pu, Y., Haber, N., and Goodman, N. D. Hypothesis search: Inductive reasoning with language models. In *The Twelfth International Conference on Learning Representations*, 2024.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., brian ichter, Xia, F., Chi, E. H., Le, Q. V., and Zhou, D. Chain of thought prompting elicits reasoning in large language models. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K. (eds.), *Advances in Neural Information Processing Systems*, 2022.

Wilson, A. G. and Adams, R. P. Gaussian process kernels for pattern discovery and extrapolation. In *Proc. ICML*, 2013.

Wong, L. S., Grand, G., Lew, A. K., Goodman, N. D., Mansinghka, V. K., Andreas, J., and Tenenbaum, J. B. From Word Models to World Models: Translating from Natural Language to the Probabilistic Language of Thought. *ArXiv*, abs/2306.12672, 2023.

Wood, F., van de Meent, J. W., and Mansinghka, V. A new approach to probabilistic programming inference. In *Proceedings of the 17th International conference on Artificial Intelligence and Statistics*, pp. 1024–1032, 2014.

Wu, Y., Jia, F., Zhang, S., Li, H., Zhu, E., Wang, Y., Lee, Y. T., Peng, R., Wu, Q., and Wang, C. An Empirical Study on Challenging Math Problem Solving with GPT-4, 2023.

Zhong, R., Zhang, P., Li, S., Ahn, J., Klein, D., and Steinhardt, J. Goal driven discovery of distributional differences via language descriptions. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

A. Gaussian Process experiments

In the prompt, we ask the LM to use the following operations.

1. Replace a subexpression $\mathcal{S}$ with $\mathcal{S} + \mathcal{B}$, where $\mathcal{B}$ is any base kernel.
2. Replace a subexpression $\mathcal{S}$ with $\mathcal{S}x\mathcal{B}$, where $\mathcal{B}$ is any base kernel.
3. Any base kernel $\mathcal{B}$ may be replaced with any other base kernel family.

LM hyperparameters We provide the LM with the following kernels: Exponentiated Quadratic, Periodic, Linear, and Polynomial. We run our pipeline for two rounds with three proposals each round. We use a temperature of 0.2 for the Proposal LM and temperature of 0.0 for the Critic LM. We use three in-context exemplars. Our Critic LM conditions on the best twelve programs so far.

In the augmented variation, we also provide the LM with the following additional kernels: Matern32, Matern52, Cosine, and the Rational Quadratic kernel. In the augumented variation, we run our pipeline for three rounds with eight proposals each round. We use a temperature of 0.7 for the Proposal LM and temperature of 0.0 for the Critic LM. We use three in-context exemplars. Our Critic LM conditions on the best twelve programs so far.

The marginal likelihood can be multimodal in the parameters of the periodic kernel. Therefore, following Duvenaud et al. (2013), if the proposed kernel has periodic components, we initialize the period at five different initial values, optimize the marginal likelihood starting from those different initializations, and choose the kernel hyperparameters with the highest marginal likelihood across those initializations.

Spectral Mixture kernel We use a GP with a spectral mixture kernel (Wilson & Adams, 2013) with 5 mixture components. For each dataset, we randomly initialize the parameters of the mixture and choose the kernel hyperparameters with the highest log marginal likelihood across five random initializations.

B. Stan Experiments

Eight Schools Dataset This dataset consists of eight observations: the estimated treatment effect of a SAT coaching program and the standard error of the treatment effect.

Peregrine dataset This dataset consists of peregrine population counts in the French Jura from 1964 to 2003 (40 observations in total).

Dugongs Dataset The ages and lengths of 27 captured dugongs (sea cows).

Surgical Dataset The mortality rates in 12 hospitals performing cardiac surgery on babies.

LM hyperparameters We run our pipeline for three rounds with eight proposals each round. We use a temperature of 0.7 for the Proposal LM and temperature of 0.0 for the Critic LM. We use three in-context exemplars. Our Critic LM conditions on the best twelve programs so far.

Markov Chain Monte Carlo diagnostics We evaluated the fidelity of the learned posteriors using the Gelman-Rubin $\hat{R}$ diagnostic (Gelman & Rubin, 1992) and by examining the Bulk Effective Sample Size (ESS). In particular, the programs reported in the table all had $\hat{R} \leq 1.01$ and mean bulk ESS $>= 400$ per chain.

C. Lotka-Volterra

Dataset To create our dataset, we simulate data from the following "perturbed" Lotka-Volterra dynamics

$$\frac{db}{dt} = \alpha b - \beta bc \tag{1}$$

$$\frac{dc}{dt} = -\gamma c + \delta bc^{0.95} \tag{2}$$

Language model produced pymc code snippet for modeling Peregrine population count

```
def dugongs_gen_model():
  # Priors for von Bertalanffy growth function parameters
  L_inf = pm.Uniform('L_inf', lower=0, upper=3) # Asymptotic length, bounded by
    observed data
  k = pm.Uniform('k', lower=0, upper=1) # Growth coefficient, bounded by reasonable
    values
  t0 = pm.Uniform('t0', lower=-5, upper=5) # Theoretical age at zero length
  # Expected length at each age using von Bertalanffy growth function
  expected_length = L_inf * (1 - pm.math.exp(-k * (ages - t0)))
  # Likelihood for observed lengths
  sigma = pm.Uniform('sigma', lower=0, upper=1) # Standard deviation of observed
    lengths around the mean
  y_obs = pm.Normal('y_obs', mu=expected_length, sigma=sigma, observed=lengths)
```

Figure 7. **LM proposes programs informed by domain knowledge** LM chooses a model informed by the domain (animal length vs age) and sets the priors based on the dataset (*e.g.,* the largest length in the dataset is smaller than 3).

We set $\alpha = 0.9, \beta = 1.1, \delta = -1.2, \gamma = 2.1$. The parameter α characterizes the prey's maximum growth rate and β controls how the predator population modulate the growth rate. The parameter γ characterizes the prey's maximum death rate and δ controls how the predator's growth rate depends on the prey population density. In contrast to the standard Lotka-Volterra dynamics, we raise c to a fractional power.

We now describe the various baselines we compare against in Section 3.3.

Standard Lotka-Volterra We fit the free parameters of the standard Lotkva-Volterra differential equations.

$$\frac{db}{dt} = \alpha b - \beta bc \tag{3}$$

$$\frac{dc}{dt} = -\gamma c + \delta bc \tag{4}$$

Neural ODE baseline We parameterize the predator and prey equations with an MLP. We run a hyperparameter search over four widths (4, 8, 16, 32) and 3 depths (1,2,4). We use a learning rate of 3e-3 and train using full-batch gradient descent with Adam for 1500 iterations.

Hybrid Neural ODE baseline We implement a Hybrid Neural ODE baseline that introduces a correction to the predator-prey interaction term. Note that, we follow a two-stage "boosting" type procedure to fit the parameters of the MLP. First, we fit the free parameters $\alpha, \beta, \gamma, \delta$ to the data. We then *freeze* those parameters and fit the MLP parameters. Without this two-staged approach, the MLP term can dominate the dynamics. The MLP term has one layer and four hidden units and we train the MLP with full batch gradient descent with Adam using a learning rate of 3e-3.

$$\frac{db}{dt} = \alpha \cdot b - \beta \cdot b \cdot c \tag{5}$$

$$\frac{dc}{dt} = -\gamma \cdot c + \delta \cdot b \cdot (c + 0.1 \cdot \text{mlp}(b, c)) \tag{6}$$

In the warm-start variations, we provide the LM with an initial hybrid Neural ODE baseline that introduces an additive correction to prey equation. The MLP term has one layer and four hidden units.

$$\frac{db}{dt} = \alpha \cdot b - \beta \cdot b \cdot c + 0.1 \cdot \text{mlp}(b, c) \tag{7}$$

$$\frac{dc}{dt} = -\gamma \cdot c + \delta \cdot b \cdot c \tag{8}$$

LM hyperparameters We run our pipeline for four rounds with twelve proposals each round. We use a temperature of 0.7 for the Proposal LM and temperature of 0.0 for the Critic LM. We use three in-context exemplars. Our Critic LM conditions on the best twelve programs so far.

D. Failure rates of GPT-4 V proposed programs

Model	Percent successfully scored
GPT-4 textual dataframe	78%
GPT-4 vision only	70%
GPT-3.5	76%

Table 2. Percentage of `pymc` programs that we can successfully perform inference in for Stan experiments.

E. Additional ODE results

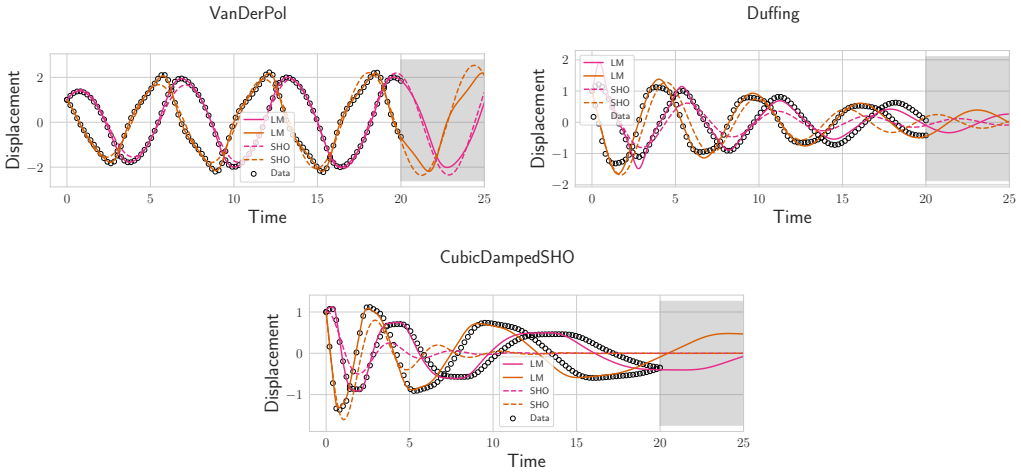

Figure 8. **Modeling nonlinear ODEs.** LM can introduce polynomial corrections to simple harmonic oscillator (SHO) that provide a better fit. Grey region indicates extrapolation region.

We evaluated on three additional ODEs (nonlinear oscillators) from this github repository `pysindy`. These ODEs include a Duffing ODE, Van Der Pol oscillator, and a Cubic Harmonic Damped oscillator. We give `BoxLM` a simple harmonic oscillator to start and ask it to introduce polynomial corrections to improve the fit. Consistent with Section 3.3, our approach can generally improve upon a baseline ODE and extrapolate accurately into the test region.

F. Visual Interface/GPT-3.5 Ablations

Context length limits are a potential limitation if we provide the dataset in textual format in the prompt. We therefore experiment with a visual-only variation. We remove all textual representations of datasets and model criticism statistics from the prompt and only provide visual plots of these datasets and statistics. We show this visual-only variation does not harm the performance relative to textual variation and should not suffer as much from context length limits as the dataset grows larger. We also show we can obtain similar results with GPT-3.5 Turbo, which is significantly less costly than GPT-4.

G. State-space model hidden state update experiments

We present the same results from the main text but take a state-space update inspired approach to computing the natural language criticism h^t.

Dataset	GPT-4 Text	GPT-4 Visual Only	GPT-3.5	Expert
Eight schools	-30.17	-30.40	-30.44	-30.70
Dugongs	22.61	23.76	21.01	22.52
Surgical	-37.36	-38.54	-42.2	-40.29
Peregrine	-164.69	-143.45	-161.14	-142.19

Table 3. **Vision interface and model type ablations.** Comparison of GPT-4 with textual representation of data and model criticism statistics in prompt, GPT-4 with only visual representations (GPT-4 Visual Only) of data and model criticism statistics (*e.g.,* only plots), and GPT 3.5 Turbo against expert programs. The visual-only variation (GPT-4 Visual-Only) performs comparably to the textual variation (GPT-4 Text)and outperforms the textual variation on the Peregrine dataset. GPT-3.5 performs slightly worse on some datasets but comparably on most. We report the expected predictive log density (LOO).

Dataset	BoxLM+ State Space	BoxLM State Space	Periodic	AS	SM	N-BEATS
Air	**0.04**	0.34	0.15	0.19	<u>0.06</u>	0.22
Beer	0.07	0.41	0.15	<u>0.06</u>	**0.05**	0.02
Heart	**0.20**	0.25	**0.20**	<u>0.21</u>	<u>0.21</u>	0.07
Milk	**0.06**	0.11	<u>0.10</u>	0.11	<u>0.09</u>	0.04
Wine	<u>0.17</u>	<u>0.17</u>	0.21	**0.13**	0.18	0.17
Wool	0.24	<u>0.15</u>	0.19	0.23	**0.13**	0.18

Figure 9. **Test set performance of `BoxLM` with state-space updated on time series datasets.** Performance of `BoxLM` system with state space update for model criticism. Comparison of `BoxLM` test mean absolute error (MAE) against Automatic Statistician using greedy search (`AS`), spectral mixture kernel (`SM`), periodic kernel (`Periodic`), and `N-BEATS`. `BoxLM+` searches over an augmented kernel space. We bold the best and underline the second best among the GP methods.

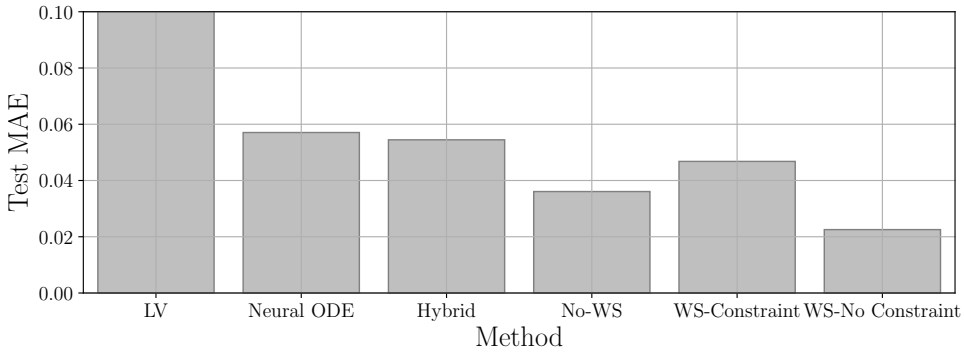

Figure 10. **Correcting misspecified Lotka-Volterra dynamics (`BoxLM` with state-space updates).** `BoxLM` can introduce corrections to standard Lotka-Volterra dynamics (no warm-start) and a hybrid neural ODE approach (warm-start) that outperform several baselines. Test MAE of `LM` models (`No-WS`, `WS-Constraint`, and `WS-No Constraint`) compared to the standard Lotka-Volterra model `LV`, a `Neural ODE`, and a hybrid Neural ODE model with a multiplicative correction to the prey-predator dynamics (`Hybrid`).

Dataset	Expert	LM
Eight schools	-30.70	-30.17
Eight schools sim	-18.09	-18.39
Eight schools sim no metadata	-18.09	-18.90
Dugongs	22.52	22.61
Dugongs sim	50.04	57.4
Dugongs sim no metadata	**50.04**	26.68
Surgical	**-40.29**	-37.36
Surgical sim	-39.80	-38.45
Surgical sim no metadata	**-39.80**	-58.72
Peregrine	**-142.19**	-164.69
Peregrine sim	**-130.48**	-177.15
Peregrine sim no meta	-130.48	-127.32

Table 4. **Comparison of BoxLM with state-space update programs against expert programs** We perform this comparison across four different datasets and two different ablations that replace observations with synthetic observations and remove all metadata. We report the expected predictive log density estimated via leave-one-out cross validation. We bold statistically significant differences and underline non-significant differences.