technische universität
dortmund

Simulation-based expert prior elicitation: Method and software development

Dissertation submitted in partial fulfillment of the requirements for the degree of *Dr. rer. nat.* at the Faculty of Statistics,
Computational Statistics Working Group,
TU Dortmund University

by

Florence Bockting

19. March 2026

Dean:
Prof. Dr. Philipp Doebler

Reviewer:
Prof. Dr. Paul-Christian Bürkner (Primary Supervisor)
Prof. Dr. Katja Ickstadt

ABSTRACT

This thesis focuses on the specification of *prior distributions* for parameters in Bayesian models. In particular, it addresses the formulation of *informative priors* derived from *expert knowledge*, a research area referred to as *expert prior elicitation*. Expert prior elicitation is particular valuable in settings where data are sparse but substantial domain expertise is available. Typical examples include the development of a new product, the adaptation of interventions to new contexts, or the assessment of risks associated with rare events. Despite its long research history dating back to the 1970s, the practical use of expert prior elicitation remains limited. One major limitation is that many existing methods have been developed as highly specialized, model-dependent solutions, thereby constraining their flexibility and transferability across diverse real-world applications. Furthermore, the lack of a standardized diagnostic framework hampers the systematic comparison of existing methods and the evaluation of prior distributions derived from elicited expert knowledge. In addition, the availability of open-source and interoperable software tools that integrate the expert elicitation workflow within the broader Bayesian workflow remains limited.

This thesis contributes to the field of expert prior elicitation by introducing a novel simulation-based method that builds on recent advances in *predictive* prior elicitation. The proposed method is *hybrid* and *model-agnostic*, developed within a *modular* framework that facilitates flexible adaption and extension. Its performance is systematically evaluated across several simulation studies. In addition, we introduce *elicito*, an open-source Python package that implements the proposed method. The software adheres to the FAIR principles of research software engineering and is distributed via PyPI and conda-forge. Comprehensive documentation, including API references and tutorials, is hosted on a project website. The source code is openly available on GitHub, where it is actively maintained and version-controlled. A continuous integration testing framework is incorporated for quality assurance. Future research directions include validating the method on complex, real-world case studies and expanding its implementation into a fully interoperable software ecosystem for expert prior elicitation.

Keywords: *expert prior elicitation*, *Bayesian workflow*, *prior specification*, *informative priors*, *elicito*

ACKNOWLEDGEMENT

This doctoral thesis would not have been possible without the support of the many people who have accompanied and encouraged me throughout my academic and personal journey.

First and foremost, I would like to thank my parents for their unwavering belief in me and their constant support in all my decisions. Their selfless help and continuous encouragement have enabled me to overcome numerous challenges that would otherwise have been far more difficult to manage.

I would like to thank Prof. Dr. Frank Lammers and Prof. Dr. Georg Felser for their guidance, which played a pivotal role in my decision to pursue an academic career. Their support was important in helping me find the confidence and motivation to keep moving forward. In this context, I would also like to thank Prof. Dr. Jürgen Dix, who has supported me since my bachelor's studies. His advice and our many insightful conversations have played an essential role in shaping both my academic perspective and personal development.

I would like to thank my colleagues, in particular Javier Enrique Aguilar, Luna Fazio, Soham Mukherjee, Maximilian Scholz, and Philipp Reiser. It has been a pleasure to work alongside such a talented, kind, and inspiring team, who have become my academic family. Further, I would like to thank my supervisor Prof. Dr. Paul-Christian Bürkner, whose confidence in me made it possible to pursue a doctorate in computational statistics. I am grateful for the support and guidance throughout this journey.

I would further like to thank all my students who patiently attended my seminars and enriched them through their thoughtful questions. I would like to give special thanks to Lennart Koppe, who wrote an outstanding bachelor's thesis under my supervision. Working with him was a great pleasure, and his many insightful questions helped me gain a deeper understanding of my own research.

Finally, I would like to thank Luna Fazio in particular, for always being there and supporting me during my PhD. Our long conversations have played an important role in shaping my academic and personal development. Having someone with whom I could openly discuss not only scientific matters but also the doubts, uncertainties, and frustrations that inevitably accompany the PhD journey has been truly invaluable.

LIST OF PUBLICATIONS

1. **Bockting, F.**, Radev, S. T., and Bürkner, P. C. (2024). Simulation-based prior knowledge elicitation for parametric Bayesian models. *Scientific Reports, 14*(1), 17330. https://doi.org/10.1038/s41598-024-68090-7

   Contribution of the thesis author: The author of this thesis formulated the method with feedback from Prof. Bürkner and Prof. Radev. She implemented the method in Python, designed and selected the simulation studies, and prepared the first draft of the manuscript with input from Prof. Bürkner and Prof. Radev.

2. **Bockting, F.**, Radev, S. T., and Bürkner, P. C. (2025). Expert-elicitation method for non-parametric joint priors using normalizing flows. *Statistics and Computing, 35*(5), 132. https://doi.org/10.1007/s11222-025-10665-z

   Contribution of the thesis author: The author of this thesis formulated the method with feedback from Prof. Bürkner and Prof. Radev. She implemented the method in Python, designed and selected the simulation studies, and wrote the manuscript. Feedback has been provided by Prof. Bürkner and Prof. Radev.

3. **Bockting, F.** and Bürkner, P. C. (2025). elicito: A Python Package for Expert Prior Elicitation. *arXiv*. https://doi.org/10.48550/arXiv.2506.16830 (in review)

   Contribution of the thesis author: The author of this thesis designed and implemented the method as a Python package. She selected and implemented the simulation study and wrote the manuscript, with feedback on the final version provided by Prof. Bürkner.

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

Basic Notation

$\dot{a}$	The element a is *observed*
$\left\{a^{(s)}\right\}_{s=1}^{S}$	A sequence of S samples with $a^{(s)}$ being the s-th sample
$\tilde{a}$	The element a is *simulated*
$p(\cdot)$	A probability density function
$p(\cdot \mid \cdot)$	A conditional probability density function

Simulation-based Expert Prior Elicitation

δ	The learning rate in the optimization algorithm
$\gamma_{m'}$	The weight of the m'-th component in the multi-objective loss
$\mathcal{D}_{m'}$	The discrepancy measure of the m'-th loss component
$\mathcal{L}(\lambda)$	The multi-objective loss function with respect to λ
E_m	The m-th elicited summary, with indexing starting at 1
T_p	The p-th target quantity defined as random variable with $T_p \sim p_{T_p}$

Normalizing Flow

g_λ^{-1}	The inverse of mapping g_λ
g_λ	An invertible and differentiable mapping
$p(u)$	The base distribution

Numbers

J	The number of model parameters in $\mathcal{M}$
K	The number of model hyperparameters in $\mathcal{M}$
M	The number of elicited summaries

M'	The number of loss components in the multi-objective loss
N	The number of observations in a data vector y
P	The number of target quantities
Q_p	The number of elicitation techniques per target quantity
S	The number of samples from a prior distribution

Probability Theory

$p(\theta \mid y, \lambda)$	The posterior distribution
$p(\theta \mid \lambda)$	A parametric prior distribution
$p(y \mid \lambda)$	The marginal data model (a.k.a. evidence, marginal likelihood)
$p(y \mid \theta)$	The data distribution
$p_\lambda(\theta)$	A non-parametric prior distribution

Spaces

Λ	The sample space of hyperparameters
$\mathcal{Y}$	The sample space of the data
$\mathfrak{T}$	The sample space of target quantities with $\mathfrak{T} = \phi_\Theta(\Theta) \cup \phi_\mathcal{Y}(\mathcal{Y})$
ϕ	The function space of transformation functions
Θ	The sample space of parameters

Statistical Model

η	A vector of derived model parameters
λ_k	The element k of hyperparameter vector λ, with indexing starting at 1
$\mathcal{M}$	The Bayesian model (including prior and likelihood)
θ_j	The element j of parameter vector θ, with indexing starting at 1
f^{dat}	A data-transformation function with $f^{\text{dat}} \in \phi_\mathcal{Y}(\mathcal{Y})$
f^{par}	A parameter-transformation function with $f^{\text{par}} \in \phi_\Theta(\Theta)$
y_n	The observation n of data vector y, with indexing starting at 1
z	A derived model outcome

TABLE OF CONTENTS

CHAPTER 1

Background

1.1 Motivation and outline

In this thesis, we examine the specification of *prior distributions* over parameters in a *Bayesian* model using *expert prior elicitation*. A fundamental characteristic of the Bayesian paradigm is the explicit incorporation of probability distributions over model parameters, known as *prior distributions* or simply *priors*. Consequently, the selection of appropriate prior distributions constitutes a critical step in the *Bayesian workflow* (Gelman et al., 2020). An important dimension for classifying various approaches to prior specification is the extent of subject-matter information incorporated into the prior. This spectrum ranges from non-informative priors, which do not encode any subject-matter knowledge, to informative priors that incorporate highly problem-specific information (Gelman and Hennig, 2017). This thesis addresses the construction of *informative priors* derived from subject-matter knowledge elicited from domain *experts*. An entire research domain, referred to as expert prior elicitation (EPE), has been established to study this problem.

In general, EPE refers to the process of translating an individual's knowledge and beliefs about one or more uncertain quantities into a (potentially joint) prior distribution (Garthwaite et al., 2005; Mikkola et al., 2024b). This process integrates two perspectives: that of the individual (i.e., the domain expert), from whom information is to be elicited, and that of the prior distribution, which is to be constructed based on the elicited information. A key challenge in EPE is that specifying a prior distribution imposes particular requirements on the type and quantity of information needed to identify a probability distribution, which may not align with the domain knowledge that an expert is able to provide.

Research on EPE dates back to the 1960s (Winkler, 1967, 1968) and continues to be an active area of investigation (Mikkola et al., 2024b; Falconer et al., 2022; Johnson et al., 2010a). Over the decades, numerous methods have been developed to translate expert knowledge into formal prior distributions. Despite this progress, EPE methods remain underutilized in practical applications (Mikkola et al., 2024b). Several factors contribute to this limited adoption, including the fact that many methods (1) are spe-

cialized for specific types of generative models and cannot be readily extended to other models, (2) require very specific type of expert information that may not be readily interpretable by individual domain experts, and (3) lack accessible, user-friendly software implementations that would make their application more practical.

This thesis contributes to the field of EPE by introducing a simulation-based method for translating expert knowledge into prior distributions. Our proposed method is designed to be flexible and compatible with a wide range of generative models, prior distribution characteristics, and types of expert information. Additionally, we provide an open-source, version-controlled, and user-friendly software implementation to facilitate the adoption of this method in both research and practice. Table 1.1 outlines the structure of this thesis.

Ch.	*Sec.*	*Description*
1	1.1	Introduction to expert prior elicitation (EPE), current challenges, and motivation for the proposed work.
	1.2	*Bayesian inference* and introduction to the notation used throughout the thesis.
	1.3	*Prior specification* and its relation to EPE.
2	2.1	The *EPE process*, including its main stages and the actors involved.
	2.2	*Setting up* the EPE process, including the selection of experts, choice of the generative model and prior characteristics, identification of target quantities, and choice of elicitation techniques.
	2.3	*Extracting knowledge* from domain experts, including the use of elicitation protocols for multi-expert settings.
	2.4	*Translating expert-elicited knowledge* into formal prior distributions. Overview of existing EPE methods and associated challenges.
	2.5	*Evaluating* constructed prior distributions, including criteria such as validity, feasibility, and replicability.
	2.6	*Desiderata* for an ideal EPE method and software.
3	3.1	Contribution 1: A *simulation-based EPE method* for *parametric prior* distributions.
	3.2	Contribution 2: Extending the proposed method to *non-parametric, joint prior* distributions using normalizing flows.
	3.3	Contribution 3: *Implementing a FAIR software* solution for the developed simulation-based EPE method.
4		Discussion of achievements, remaining challenges, and directions for future research.

Table 1.1 Outline of the present thesis. Abbreviations: *Ch.*, chapter; *Sec.*; section.

1.2 Bayes' Rule

In this section, we introduce key concepts of Bayesian inference and establish the general notation adopted throughout the thesis. The data are represented as a vector $y = (y_1, \ldots, y_N)$, where y_n denotes the n-th observation and $y \in \mathcal{Y}$. Similarly, let $\theta = (\theta_1, \ldots, \theta_J)$ denote a vector of parameters, where θ_j denotes the j-th parameter and $\theta \in \Theta$. A (marginal) probability density function is denoted by $p(\cdot)$ and a conditional probability density by $p(\cdot \mid \cdot)$, with the corresponding arguments determined by the context. The terms probability distribution and probability density will be used interchangeably. Along these lines, the joint probability mass or density function of the data and model parameters given some hyperparameter vector $\lambda = (\lambda_1, \ldots, \lambda_K)$ can be factorized as

$$p(y, \theta \mid \lambda) = p(y \mid \theta) p(\theta \mid \lambda) \tag{1.1}$$

where $p(y \mid \theta)$ is referred to as the *sampling* or *data distribution* and $p(\theta \mid \lambda)$ as the *prior distribution*, here parameterized by λ, with λ_k denoting the k-th hyperparameter and $\lambda \in \Lambda$. The joint probability density of data and model parameters in Equation 1.1 is sometimes also referred to as *data model* (Gelman, 2014).

In Bayesian inference, the primary objective is typically to make *probabilistic* statements about the parameters conditional on the observed data. This conditional distribution, $p(\theta \mid y, \lambda)$, is referred to as the *posterior distribution*. A formal framework for computing this inverse probability is provided by *Bayes' rule* (Robert, 2007)

$$p(\theta \mid y, \lambda) = \frac{p(y \mid \theta, \lambda) p(\theta \mid \lambda)}{p(y \mid \lambda)} = \frac{p(y, \theta \mid \lambda)}{p(y \mid \lambda)}, \tag{1.2}$$

where the normalizing constant $p(y \mid \lambda)$ is known as the *marginal likelihood* or *evidence* and defined as $p(y \mid \lambda) = \int_\Theta p(y \mid \theta, \lambda) p(\theta \mid \lambda) d\theta$ for continuous parameter spaces, and $p(y \mid \lambda) = \sum_\Theta p(y \mid \theta, \lambda) p(\theta \mid \lambda)$ for discrete parameter spaces (Gelman, 2014). Bayes' rule provides a principled approach for updating prior beliefs $p(\theta \mid \lambda)$ to posterior beliefs $p(\theta \mid y, \lambda)$ in light of observed data y. In essence, Bayes' rule formalizes the process of learning from experience, whereby a defining feature of the Bayesian paradigm is the explicit use of a probability distribution over the parameters (Bissiri et al., 2016; Winkler, 1967; Robert, 2007). This directly raises the question of how to specify prior distributions, a question that constitutes the primary focus of this thesis.

1.3 Prior Specification

Specifying prior distributions is a key task in Bayesian inference (Gelman, 2014), yet it also represents one of its greatest challenges (Van Dongen, 2006; Gelman and Hennig,

2017). From an ideal perspective, the prior distribution encodes information beyond the observed data y (Jaynes, 2003). This naturally raises the question: What kind of information should be represented through a probability distribution, and how should it be quantified in form of a prior? Perhaps the least satisfying immediate answer is that there exists no single universal rule for assigning priors (Jaynes, 2003).

However, over time, a variety of approaches for constructing prior distributions have emerged. Historically, these approaches have been broadly categorized into two paradigms: the *objective* and *subjective* Bayesian perspectives. The objective view is commonly associated with the work of Jaynes (2003) and Jeffreys (1961), while the subjective view is rooted in the work of De Finetti (1974) (Kass and Wasserman, 1996; Berger, 2006). The central principle of the subjective approach is that the prior should reflect the *personal beliefs* of the analyst regarding the problem at hand. In contrast, the objective approach aims to construct priors that have minimal impact on the corresponding posterior distribution, thereby allowing the data to dominate the inference (Consonni et al., 2018; Berger, 1985).

However, it has been argued that the distinction between subjective and objective approaches is not entirely meaningful, and might be misleading, as all statistical methods involve subjective elements in their formulation beyond the specification of a prior distribution (Gelman and Hennig, 2017; Berger, 1985). As an alternative, approaches to prior specification can be classified based on the extent to which information from the data distribution is used in the construction of the prior (Gelman et al., 2017; Gelman, 2002). Priors that aim to maximize the use of information derived from the data distribution, effectively minimizing the influence of external information not present in the observed data, are commonly referred to as *diffuse* or *non-informative* (Van De Schoot et al., 2021; Gelman et al., 2017; Seaman et al., 2012). Classical examples include Jeffreys priors (Jeffreys, 1961), reference priors (Bernardo, 1979), and maximum entropy priors (Jaynes, 2003). In contrast, *informative* priors are constructed to incorporate substantial problem-specific knowledge, ideally capturing all relevant information available before observing the data (Gelman et al., 2017; Van De Schoot et al., 2021; Winkler, 1967). This is basically the idealized Bayesian view which assumes that separate information about a parameter θ exists and can be quantified, which is then combined with the data (Berger, 1985). Positioned between the extremes of non-informative and informative priors are *weakly informative* priors. These priors encode general domain knowledge that is applicable across a broad class of problems, while intentionally avoiding the use of highly problem-specific information (Van De Schoot et al., 2021; Gelman et al., 2017). One example for a weakly informative prior is the Penalised Complexity prior proposed by Simpson et al. (2017).

Overall, no universally correct or superior method exists for prior specification (Van De Schoot et al., 2021). The choice of strategy depends largely on the specific character-

istics of the problem and the resources available to the analyst. Relevant considerations include the extent of prior knowledge that is available about the problem, the format in which this knowledge is available, the financial and temporal resources at hand, the amount of data informing the inference, and the expected influence of the prior on the posterior distribution. In this thesis, we focus on the construction of *informative priors*, which naturally raises the question of the appropriate source of prior information. Different sources can be distinguished, giving rise to specific approaches to prior specification. For example, when historical data are available, several data-driven approaches have been proposed, including *power priors* (Ibrahim et al., 2015; Rietbergen et al., 2011), *catalytic priors* (Huang et al., 2020), and, more recently, *primed priors* (Fazio et al., 2024). In addition, meta-analyses, systematic literature reviews, and results from previous empirical studies can be used to construct priors (Mikkola et al., 2024b; Van De Schoot et al., 2018; Choy et al., 2009). The specific information source considered in this thesis is *expert information*, which gives rise to EPE. An approach that involves specifying informative priors based on knowledge or beliefs elicited from subject-matter experts (Kass and Wasserman, 1996).

CHAPTER 2

Expert prior elicitation

The incorporation of expert knowledge into the construction of prior distributions is particularly valuable in contexts where empirical data are scarce but substantial domain expertise is available. For example, in the design of clinical trials for a novel drug, members of the development team may provide assessments of likely efficacy and potential adverse effects based on prior experimental evidence (O'Hagan, 2019). Another example is the case of newly developed procedures for genetically modifying plants to render crops resistant to a specific herbicide, where expert knowledge can be elicited to characterize uncertainties in agricultural risk assessments, including potential adverse effects on cultivation practices, agricultural systems, or public health (Knol et al., 2010).

External knowledge can be used to constrain model behavior, ensuring consistency with known characteristics of the phenomenon under study and with established scientific understanding. This is particularly important for complex, overparameterized, or partially identified models (Johnson et al., 2010b; Morris et al., 2014; Clemen et al., 2000; Manderson, 2023; Gustafson, 2010). The inclusion of prior information can also improve the computation of posterior estimates, rendering otherwise intractable models suitable for inference (Manderson, 2023). The value of integrating expert knowledge into statistical models has long been recognized, and its application spans a wide range of scientific disciplines, including pharmacy (Lesaffre et al., 2020), medicine (Azzolina et al., 2021), and ecology (Choy et al., 2009; Johnson et al., 2010b).

2.1 EPE process

EPE is a structured process for translating an individual's knowledge and beliefs about one or more uncertain quantities into a (joint) probability distribution (Garthwaite et al., 2005; Mikkola et al., 2024b; Kuhnert et al., 2010; O'Hagan et al., 2006). As outlined by Garthwaite et al. (2005), the EPE *process* involves four key stages:

1. **Setup stage**: The problem is defined, suitable expert(s) selected, and the quantities to be elicited (in the following referred to as *target quantities*) identified;

2. **Elicitation stage**: The target quantities are queried from the expert(s) using spe-

cific elicitation techniques, yielding a set of *expert-elicited summaries*;

3. **Fitting stage**: The expert-elicited summaries are used to fit a (joint) probability distribution that serves as the specified prior;

4. **Evaluation stage**: Finally, the adequacy of the learned prior distribution is assessed in collaboration with the expert, and adjustments in previous stages are made if necessary.

Within this process, three different roles can be differentiated (Falconer et al., 2022): The *domain expert* is an individual with extensive knowledge of a substantive area from whom prior information is to be elicited. The domain expert may or may not have formal training in statistics. The *analyst* is the statistical expert responsible for formulating the statistical model and translating the expert's knowledge into a prior distribution that can be incorporated into the probabilistic model. In some cases, it is useful to further distinguish the role of the *facilitator*, who guides the elicitation process, provides training and interacts directly with the domain expert to obtain the necessary information (Morris et al., 2014). Depending on the context and complexity of the elicitation task, the roles of the analyst and the facilitator may be assumed by the same individual.

Each stage of the EPE *process* is associated with a specific set of tasks. For a given problem setting, the subset of *relevant* tasks defines the corresponding EPE *workflow*. An EPE *method* typically refers to the algorithm employed in the fitting stage, that is, the process of translating expert knowledge into corresponding prior distributions. In this thesis, we focus on methodological advancements in EPE methods. Since the input-output structure of an EPE method is inherently shaped by the upstream and downstream tasks within the overall EPE workflow, a thorough understanding of the entire workflow is essential. Although a comprehensive review of all tasks involved in the EPE workflow lies beyond the scope of this thesis, and is already well-documented in the literature (see e.g., Mikkola et al., 2024b; Garthwaite et al., 2005; Falconer et al., 2022; O'Hagan et al., 2006), the following sections provide an overview of selected components that are of particular relevance to the development of EPE methods.

2.2 Setting up EPE

In the setup stage, key decisions are made that collectively define the EPE workflow. These include the selection of experts (Section 2.2.1); the specification of a generative model, encompassing both the data distribution and the characteristics of the prior distributions (Section 2.2.2); the identification of target quantities (Section 2.2.3); and the choice of elicitation techniques (Section 2.2.4). In the following, each of these aspects is discussed in greater detail.

2.2.1 Expert selection

The process of expert selection involves determining the appropriate number of experts and specifying their required expertise and background. The *expertise* of the selected experts influences the choice of target quantities, particularly their interpretability, and is discussed in greater detail in Section 2.2.3. The *number* of experts can range from a single individual to multiple experts. In many cases, eliciting prior knowledge from multiple experts is preferable, as it can reduce the influence of cognitive biases in individual judgments, allow assessment of variability in responses, and thus provide a more representative characterization of prior knowledge within a research field (Falconer et al., 2022; Stefan et al., 2022; Kuhnert et al., 2010). Different recommendations have been made regarding the optimal number of experts (see for review e.g., Stefan et al., 2022). In practice, however, the choice is often constrained; for example, in highly specialized fields, the available pool of experts may be limited (Stefan et al., 2022). When multiple experts are involved, an additional decision concerns the method for aggregating the elicited information. Possible approaches include *behavioral aggregation* (i.e., reaching a consensus among experts), *mathematical aggregation* (i.e., combining multiple elicited values using quantitative methods), or a combination of both (see for review, e.g., O'Hagan et al., 2006; O'Hagan, 2019; Falconer et al., 2022; Clemen and Winkler, 1999; Winkler, 1968; Johnson et al., 2010a; Rougier et al., 2013; Williams et al., 2021).

2.2.2 Model selection

The formulation of the *statistical model* $\mathcal{M}$ involves specifying both the functional form of the data distribution and the characteristics of the prior distribution(s) over the model parameters. Two key characteristics of priors are their *dimensionality* and their *parametric nature*, that is, whether they are *parametric* or *non-parametric* (Mikkola et al., 2024b).

Parametric nature When a *parametric* family is specified for the prior distribution(s), $p(\theta \mid \lambda)$, the individual members of the family are characterized by the model hyperparameter vector λ. In this setting, the objective of an EPE method is to find hyperparameter values that accurately represent the expert's beliefs (Garthwaite et al., 2005). The selection of a parametric family typically represents a compromise between conceptual, problem-specific considerations and mathematical convenience. In particular, earlier work in EPE, conducted when computational resources were limited, focused primarily on conjugate priors due to their mathematical tractability (Mikkola et al., 2024b; Gribok et al., 2004; Choy et al., 2009). Although a conjugate prior is unlikely to fully capture an expert's beliefs, the hope is that such a choice will nonetheless

reflect the most important features of the expert's opinion (Al-Awadhi and Garthwaite, 1998). However, sometimes the use of more flexible prior distributions beyond the natural conjugate family is desirable. A major challenge associated with non-conjugate priors is that, in general, the resulting problem becomes analytically intractable (Gribok et al., 2004), which has led to the development of numerous *model-specific* EPE methods in the past (see Section 2.4 for further discussion).

An even higher degree of flexibility can be achieved through the use of *semi-* or *non-parametric* priors, $p_\lambda(\theta)$, which reduce or avoid imposing specific assumptions about the functional form of the prior distribution (Garthwaite et al., 2005). Oakley and O'Hagan (2007) argue that a non-parametric approach is preferable because it explicitly quantifies the uncertainty arising from the use of a finite set of expert-elicited summaries (discussed further in Section 2.2.3). Although the increased flexibility offered by non- (or semi-) parametric priors is appealing, it requires more informative input, both in quantitative and qualitative terms. This includes, among others, eliciting a larger number of increasingly specialized target quantities from the domain expert to adequately constrain the model behavior. A requirement that usually conflicts with the principle that target quantities should remain easily interpretable for the domain expert (Mikkola et al., 2024b). The increased flexibility also places additional demands on the analyst, requiring further decisions during the model-building stage to impose appropriate structure on the problem (Bornkamp and Ickstadt, 2009).

Dimensionality Another relevant property of prior distributions is their *dimensionality*. The majority of EPE methods concentrate on *univariate* prior distributions, assuming independence among the model parameters. This approach is often preferred because it relies exclusively on univariate elicitation techniques and on target quantities that are more easily interpretable by domain experts (Garthwaite et al., 2005).

However, for complex and high-dimensional problems, it is generally insufficient to consider only marginal priors; instead, the joint behavior of the parameters must be taken into account (Gelman et al., 2017). In some cases, the use of independent priors may even be problematic. For example, weakly informative independent priors can, when applied to a large number of parameters, collectively become strongly informative (Gelman et al., 2020). Joint priors provide a means to control the overall complexity of large parameter sets and to generate more sensible prior predictions, which may be difficult or impossible to obtain when using independent priors (Gelman et al., 2020). While the use of joint priors may be conceptually advantageous, the elicitation task becomes inherently more complex (Garthwaite et al., 2005). In particular, specifying a joint distribution requires information on correlations and covariances (Mikkola et al., 2024b; Garthwaite et al., 2005; Clemen et al., 2000). However, individuals often display systematic biases when making joint probability assessments; for example, they tend to

overestimate the probability of conjunctive events and underestimate the probability of disjunctive events (Garthwaite et al., 2005).

Several approaches have been proposed to address the complexity of eliciting joint priors: In some cases, the model can be reformulated such that the quantities of interest become independent (Garthwaite et al., 2005; O'Hagan, 2019, 2012). However, finding such transformations is not always possible, thus alternative approaches are required to handle joint priors. Gaussian copulas have been proposed as a means of decomposing a multivariate joint prior into its marginal priors and a corresponding correlation matrix. Although assessing dependence is a non-trivial task for individuals, Clemen et al. (2000) demonstrated that individuals are capable of making direct judgments about correlations. Others have suggested the use of pairwise concordance probabilities or partial rank correlation coefficients (O'Hagan, 2019). However, it has also been argued that the direct assessment of correlation may be an unreliable method for quantifying expert beliefs (Garthwaite et al., 2005). Alternative strategies include defining a joint prior over a subset of parameters that control a specific aspect of the model (Simpson et al., 2017), or over a set of meaningful summaries of the parameters (Seaman et al., 2012). However, these approaches present their own challenges, as it is often difficult to anticipate the broader implications of such specifications within the overall analysis (Seaman et al., 2012).

In addition to facilitating control over the joint behavior of model parameters, joint priors offer the advantage of reducing the number of hyperparameters in a model. Consequently, fewer hyperparameters need to be elicited, thereby reducing the elicitation burden on domain experts. This motivation underpins the concept of joint hyperparameters, wherein individual priors depend on a smaller, potentially structured set of shared hyperparameters (Mikkola et al., 2024b). Hierarchical priors naturally embody this principle, as they are defined in terms of higher-level parameters that govern the distributions of lower-level parameters (Bürkner, 2017). Common examples include shrinkage priors and their recent generalizations, which aim to develop parameterizations with more intuitive hyperparameters (Piironen and Vehtari, 2017; Aguilar and Bürkner, 2023, 2025; Zhang et al., 2022).

2.2.3 Selection of target quantities

The selection of target quantities is guided by the question *what* information shall be elicited from a domain expert. A *target quantity* refers to a specific aspect of the statistical model $\mathcal{M}$, such as a model parameter, outcome variable, or any quantity derived from them. It serves as a bridge between the model hyperparameters that define the prior distributions and the expert's elicited knowledge. In practice, experts are typically queried with respect to a *set of target quantities*. This set is defined with the aim of

capturing all relevant aspects of the model. The selection of target quantities is guided by two primary criteria: *informativeness* and *interpretability*.

Informativeness *Informativeness* is defined from the perspective of the model and refers to the extent to which the set of target quantities allows to identify the model hyperparameters λ. Let A denote the subset of hyperparameter values that do not induce a distinctive distribution of the observable data y. Formally, $A \subset \Lambda$ where $A = \{\lambda \in \Lambda : p(y \mid \lambda) = p(y \mid \lambda^*) \text{ for some } \lambda^* \in \Lambda \backslash \{\lambda\}\}$. The model $\mathcal{M}$ is said to be *fully identified* if $A = \emptyset$, *essentially identified* if $A \neq \emptyset$ but $p_\Lambda(A) = 0$, and *nonidentified* if $A \neq \emptyset$ and $p_\Lambda(A) > 0$ (Gustafson, 2005). Due to the probabilistic nature of this definition, it is non-trivial to determine for a specific model whether a selected set of target quantities uniquely identifies the model $\mathcal{M}$, particularly when the target quantities refer to the outcome variable.

In fact, when the target quantities refer to the outcome variable or a quantity derive from it, the hyperparameters λ are typically not identifiable, as it is generally impossible in these situations to disentangle aleatoric uncertainty, arising from the data, from epistemic uncertainty, arising from the model (da Silva et al., 2023; Manderson and Goudie, 2024; Perepolkin et al., 2024). For the present discussion, it is useful to further introduce the notion of *partially identified* models. A model is said to be partially identified if the hyperparameters λ cannot be uniquely determined, but the possible set of values, referred to as the *identification region*, is smaller than the a priori set of possible values (Gustafson, 2010). This scenario is likely the most common in the context of EPE and naturally raises the follow-up question: How large is the identification region?

A relatively naive yet straightforward approach to exploring the identification region involves repeatedly applying the fitting procedure defined by the EPE method to a fixed set of expert-elicited target quantities. In the case of non- or partial identifiability, repeated fitting yields different prior distributions, as defined by hyperparameters λ, each representing a sample from the identification region. Analyzing these samples offers insight into the range of prior distributions that align with a given set of target quantities. A more principled approach to investigate the identification region is to frame the problem as a Bayesian inference task by specifying a prior distribution over the hyperparameter and computing the corresponding posterior distribution. Regardless of the chosen approach, if the identification region contains prior distributions that the expert or analyst deems unrealistic, the subsequent question is how the identification region can be reasonably constrained.

One option for constraining the identification region is to regularize the loss function of the EPE method used to fit the prior distributions. Regularity conditions assumed, this approach is effectively equivalent to placing a prior distribution over the hyperparameter (Luo et al., 2023). Manderson and Goudie (2024) present an exam-

ple of extending the loss function through a regularization term, designed to encode a preference for maximizing prior uncertainty. Specifically, given two indistinguishable hyperparameter vectors, λ' and λ'', the vector that produces a larger variance for $p(\theta \mid \lambda)$ is preferred. Another form of loss regularization is to expand or modify the set of target quantities to increase their informativeness. However, this strategy demands additional resources, both in terms of the quantity and quality of information elicited from experts. In practice, the willingness of experts to contribute their time imposes limits on the number of quantities that can be realistically elicited (O'Hagan, 2019). Furthermore, there is a trade-off between the informativeness of a target quantity and its interpretability for domain experts (da Silva et al., 2023).

Interpretability *Interpretability* reflects the expert's perspective and refers to the extent to which a target quantity can be meaningfully understood and assessed by a domain expert. As such, it directly influences the *validity* of an EPE method, which can only be as good as the quality of its inputs (Johnson et al., 2010b). A commonly discussed distinction in relation to interpretability is whether a target quantity refers to the parameter space, such as a regression coefficient in a linear model, or to the outcome (or observable) space, that is, the dependent variable conditional on various values of the predictor variable (Kadane and Wolfson, 1998). The former approach is referred to as *structural* elicitation, whereas the latter is referred to as *predictive* elicitation in the EPE literature (Kadane and Wolfson, 1998; Falconer et al., 2022).

Predictive elicitation is generally recommended over structural elicitation, as it does not require experts to interpret the meaning of model parameters and additionally allows for model-agnostic EPE methods (see Section 2.4 for further discussion, Kadane et al., 1980). The difficulty faced by experts, regardless of their statistical expertise, in interpreting model parameters, whose meaning may even depend on some non-trivial link function, has been widely recognized (Kadane et al., 1980; Bedrick et al., 1996). However, we believe that eliciting knowledge about model parameters should not be categorically avoided. The suitability of structural versus predictive elicitation is highly context dependent and varies according to the specific problem setting and the type of expert(s) (Denham and Mengersen, 2007). For instance, experts may possess prior knowledge about model parameters derived from previous experiments, meta-analyses, or related empirical studies, and such information should not be disregarded when available.

2.2.4 Selection of elicitation techniques

Elicitation techniques specify *how* a target quantity should be elicited from an expert. This process involves (1) identifying appropriate summary measures that take the target quantity as input and return a summary value (hereafter referred to as *elicited summaries*), and (2) selecting a suitable assessment tool that specifies the *response format*

for the elicitation task.

Elicited summaries Experts are usually not asked to specify the full distribution of a target quantity directly. Instead, they provide a small set of summaries that capture the key aspects of their beliefs or knowledge about the target quantity (Morris et al., 2014; O'Hagan and Oakley, 2004). Commonly used summaries to obtain a measure of central tendency are the median or mode (Johnson et al., 2010a). However, using the mean is generally discouraged, as human judgments of the mean are often biased when the underlying distribution is skewed (Garthwaite et al., 2005). In general, experts should not be asked to provide information on statistical moments directly. For instance, eliciting the variance as a measure of variability is not recommended, as it can be difficult for experts to interpret. Instead, eliciting credible intervals is advised (Garthwaite et al., 2005).

A common approach to eliciting the variability of a target quantity is to ask experts to specify a plausible range of values by providing lower and upper bounds, such that they would consider it highly unlikely for the true value to fall outside this interval (O'Hagan, 2019; O'Hagan, 2005). Subsequently, variability can be assessed in greater detail. Two methods commonly used in expert elicitation are the *fixed interval* and *variable interval* methods. In the fixed interval method, the facilitator or analyst partitions the range of possible values of the target quantity into intervals, and the expert assesses the probability that the quantity will fall within each interval. In the variable interval (or bisection) method, the expert is asked sequentially to provide estimates of the median as well as the lower and upper quartiles (Garthwaite et al., 2005; O'Hagan, 2019). Variants of the bisection method include, for example, the *tertile method*, in which the expert specifies the median as well as the 33rd and 66th percentiles (Morris et al., 2014). However, it should be avoided to elicit percentiles that correspond to the extreme tails of a distribution (e.g., the 1st or 99th percentiles), as individuals exhibit difficulty in reasoning about highly improbable events and tend to show overconfidence (Garthwaite et al., 2005; Morris et al., 2014). Other methods include the probability method to elicit probability judgments from the expert (Morris et al., 2014). Finally, the different methods can also be combined into hybrid approaches, for instance, asking the expert to first specify the median and then provide probabilities for two additional intervals (Morris et al., 2014).

Response formats The use of an appropriate assessment tool can greatly facilitate the extraction of expert knowledge. Common response formats include direct assessment of the estimate (e.g., in tabular form), sliders or visual analog scales, sketching a graph, and the roulette method (also called the "bins and chips" method; Johnson et al., 2010a; Oakley et al., 2010; Johnson et al., 2010b). The roulette method is a variant of the fixed

interval approach, in which the expert distributes a fixed number of chips across pre-specified bins. A key advantage of this method is that the resulting allocation provides an immediate graphical representation of the expert's belief (Oakley et al., 2010).

2.3 Eliciting summaries from experts

During the *elicitation stage*, the expert is asked to provide the set of elicited summaries. A key challenge at this stage is to extract expert knowledge while minimizing the influence of *heuristics and biases* that affect human judgment under uncertainty (Tversky and Kahneman, 1974). The heuristics and biases most commonly discussed in EPE include availability, anchoring, overconfidence, range-frequency, representativeness and expert fatigue (for further discussion, see e.g., Falconer et al., 2022; O'Hagan, 2019; O'Hagan et al., 2006; European Food Safety Authority, 2014).

Substantial research in EPE has focused on designing *elicitation protocols* to minimize judgmental biases and heuristics during expert elicitation (O'Hagan et al., 2006; O'Hagan, 2019). While many decisions in the EPE process are context-dependent, such as the selection of target quantities or elicited summaries, other, more general workflow decisions can be standardized within elicitation protocols (European Food Safety Authority, 2014). Examples of such workflow decisions include the sequencing of questions, the training of experts, the scheduling of group interactions among experts, and the timing and manner of feedback provision. The European Food Safety Authority (2014) has recommended three elicitation protocols for scenarios involving multiple experts: the *Cooke* protocol (Cooke, 1991), the *Sheffield* protocol (Gosling, 2018), and the *Delphi* method (Rowe and Wright, 1999). Their main differences concern the elicited summaries employed, the sequence of questions, the method used for expert aggregation (i.e., behavioral or mathematical), and whether *seed variables* are incorporated to assess expert calibration (O'Hagan, 2019; Falconer et al., 2022). Introduction material on these protocols is provided among others in Sahlin et al. (2024); European Food Safety Authority (2014); O'Hagan (2019), and Williams et al. (2021).

EPE can be conducted on-site through interviews or remotely via telephone or video conferencing, and either synchronously in face-to-face settings or asynchronously through self-paced questionnaires (Johnson et al., 2010a; Stefan et al., 2022). Although the use of questionnaires may appear advantageous due to experts' time constraints, the quality of results is generally lower compared to face-to-face settings guided by a skilled facilitator (O'Hagan, 2019; O'Hagan, 2005). In multiple-expert elicitation, a further distinction can be made between interviewing experts individually or collectively. Collective interviews are often preferred because they facilitate knowledge sharing; however, they also carry the risk of undesirable group dynamics, such as groupthink and social conformity (Stefan et al., 2022). In such cases, a skilled facilitator is essential

to guide the group through the elicitation process (O'Hagan, 2005).

To support elicitation protocols, several software tools have been developed. These include SHELF (Oakley, 2025), designed for the Sheffield protocol, along with its browser-based extension MATCH (Morris et al., 2014) for remote EPE. Another example is the closed-source software EXCALIBUR (LightTwist Software, 2007), developed to support Cooke's protocol. However, despite the availability of such tools, their usability and intuitiveness have been criticized, and further development remains necessary (Johnson et al., 2010a).

2.4 Translating expert-elicited summaries to priors

Once the set of expert-elicited summaries has been obtained, it can be employed in the *fitting stage* to construct a corresponding prior distribution for the parameters in a Bayesian model. An *EPE method* specifies the systematic procedure by which the elicited summaries are converted into corresponding prior distributions. The complexity of an EPE method can vary considerably. In the simplest case, analytical solutions exist for conjugate models. When elicited summaries map directly onto model parameters, simple least-squares fitting may be sufficient. More complex scenarios arise when the summaries correspond to the outcome variable rather than parameters. In such cases, an additional transformation step is required, often followed by the application of general optimization techniques. A key determinant of the complexity is the extent to which the EPE method can accommodate different configurations specified during the setup stage. This flexibility involves assumptions about the model (i.e., data distribution and prior characteristics), the target quantities, and the elicitation techniques. Table 2.1 summarizes how EPE methods can be characterized according to the assumptions specified during the setup stage of the EPE process.

Task in setup stage	*Characterization of an EPE method*
Defining the data model	model-specific vs. model-agnostic
Defining prior characteristics	parametric vs. non-parametric independent vs. joint
Selecting the target quantities	structural vs. predictive vs. hybrid
Selecting the elicitation techniques	fixed vs. flexible

Table 2.1 Characterizing an EPE method based on the assumptions made during the setup stage of the elicitation process

Considerable work has focused on developing *structural* EPE methods based on independent, parametric priors (Garthwaite et al., 2005; Mikkola et al., 2024a,b). These methods are inherently model-specific, and consequently, structural EPE methods have been proposed for a variety of model classes, including linear regression (Ibrahim, 1997), logistic regression (Bedrick et al., 1997), hierarchical models (Hem et al., 2024),

and autoregressive time-series models (Jarociński and Marcet, 2019). Additional examples are provided in Manderson (2023), and a comprehensive review is available in the supplementary material of Mikkola et al. (2024b). In addition to work on independent, parametric priors, there is a longstanding line of research on EPE methods that employ more flexible, non-parametric joint priors. For example, Gaussian process models have been proposed to infer unknown prior functions from expert-provided summaries (Oakley and O'Hagan, 2007; Gosling et al., 2007; Moala and O'Hagan, 2010). Clemen et al. (2000) proposed the use of Gaussian copulas to decompose a multivariate joint prior into its marginal priors and a corresponding correlation matrix. More recently, Mikkola et al. (2024a) introduced a method for inferring prior distributions from preferential comparisons using normalizing flows. Perepolkin et al. (2024) suggested employing quantile-parameterized distributions to represent flexible priors, and Bornkamp and Ickstadt (2009) provided a semi-parametric alternative by modeling prior distributions as linear combinations of B-splines. Despite these advances, methodological and practical challenges associated with non-parametric (joint) priors have limited their broader adoption in EPE as discussed in Section 2.2.2.

Beyond structural EPE methods, a complementary line of research has concentrated on the development of *predictive* EPE methods, with an emphasis on independent, parametric priors. Although predictive EPE methods offer the potential to be model-agnostic, most approaches proposed to date remain model-specific (Mikkola et al., 2024b). Only recently, several predictive EPE methods have been proposed with the aim of reducing model dependence, including Hartmann and Agiashvili (2020), Manderson and Goudie (2024), and da Silva et al. (2023).

With respect to the choice of elicitation techniques, most EPE methods employ only one specific approach, typically quantile-based elicitation. An exception is the method proposed by da Silva et al. (2023), which accommodates multiple types of elicitation techniques. A similar pattern emerges for the choice of target quantities: most EPE methods focus exclusively on either structural elicitation (i.e., querying experts about model parameters) or predictive elicitation (i.e., querying experts about observables), while hybrid approaches remain largely unexplored.

2.5 Evaluating the elicitation process

In the *evaluation stage*, the probability distribution obtained from the fitting stage are further assessed. This raises the question of how to define a "good" prior. Within the context of EPE, we regard a *good* prior as a probability distribution that accurately approximates the expert's knowledge and beliefs. Given that expert knowledge cannot be accessed directly and is instead inferred from elicited summaries, the present definition warrants further refinement: a prior can be considered good if it *faithfully* reflects

the expert-elicited summaries, provided that these summaries adequately capture the relevant aspects of the expert's knowledge and beliefs (cf. *validity*).

Following Manderson and Goudie (2024), we define *faithfulness* as the extent to which the learned prior accurately represents the expert-elicited summaries. This consideration is particularly important for target quantities defined in the outcome space, which additionally include a translation step from summaries to model parameters (Manderson and Goudie, 2024). A key determinant of the *validity* of the learned prior is whether the chosen set of summaries (i.e., target quantities together with the elicitation techniques) provides a sufficiently informative representation of the expert's knowledge and beliefs. The critical question, then, is how one can assess whether the elicited summaries are indeed *sufficiently* informative (O'Hagan and Oakley, 2004)? One approach to evaluating the alignment of the selected target quantities with the expert's knowledge is to ask the experts directly whether the set of elicited summaries captures all relevant aspects of the problem or whether important aspects are missing (*face validity*, Johnson et al., 2010b). While this approach can provide first insights into whether important thematic concepts have been omitted, it does not provide information on whether the target quantities are sufficiently informative to identify the model hyperparameters λ.

In certain (simple) cases, sufficient statistics can be derived, thereby guiding the choice of elicited summaries. However, when sufficient statistics cannot be derived, other approaches are needed. One approach, already discussed in Section 2.2.3, is to repeatedly fit a prior to the set of expert-elicited summaries (*replicability*, Manderson and Goudie, 2024). If the hyperparameters are only partially identifiable, this procedure yields a set of priors reflecting samples from the identification set. Partial identifiability, and consequently the existence of a set of priors that equally well reflect the expert-elicited summaries, should not necessarily be regarded as a deficiency of an EPE method. Winkler (1967) argues that an expert does not possess a *unique* true prior distribution, but rather that there exists a set of equally valid priors. Even though O'Hagan (1988) assumes the existence of a true prior distribution, provided that an individual is capable of perfectly accurate self-assessment of their beliefs, O'Hagan and Oakley (2004) acknowledge that any prior implied by an expert's knowledge can only be elicited imperfectly. In light of this, it is reasonable to assume that there does not exist a single ideal prior; rather, the objective is to identify a set of suitable priors. If this set includes priors that the expert or analyst regards as unrealistic, it suggests that the chosen summaries do not adequately capture all relevant aspects of the model and therefore require refinement. This point highlights the iterative nature of EPE and the critical role of providing *feedback*. Feedback involves presenting the implications of the judgments made, allowing the expert to confirm or reject whether the provided implications accurately reflect their beliefs (Garthwaite et al., 2005). Implementing this step necessitates the development of suitable tools for visualization and other forms of

informative feedback for the expert.

As the result is typically a set of priors, a natural follow-up question is whether one should select a single distribution from this set or, alternatively, employ model-averaging techniques. The appropriate choice is likely context-dependent. Applying averaging techniques across multiple priors to obtain a single prior raises the question of whether the resulting average still represents the expert's beliefs (O'Hagan and Oakley, 2004). Conversely, one could argue that if the set of priors indeed consists of *good* priors, they should not differ substantially from one another, implying that any variation is minor and that the effect of averaging is negligible (O'Hagan and Oakley, 2004). For this discussion, it would be useful to define a metric that quantifies the impact of selecting a specific prior distribution on the subsequent analysis, for instance by evaluating the sensitivity and robustness of the resulting Bayesian inference task (Mikkola et al., 2024b).

Since the core of EPE lies in interaction with domain experts, an additional important criterion for evaluating an EPE method is its *feasibility*. Feasibility includes factors such as cost, need for equipment, completion time, and ease of use (Johnson et al., 2010a; O'Hagan, 2019; Johnson et al., 2010b). To improve feasibility, experts should receive training, the set of elicited summaries should be kept to a minimum, and, in general, EPE methods should be sample-efficient (Garthwaite et al., 2005; Mikkola et al., 2024b). However, clear criteria for quantifying feasibility are still lacking (Mikkola et al., 2024b).

The preceding discussion highlights preliminary considerations for assessing the learned prior distribution and for evaluating an EPE method. However, when taking a broader view of the EPE workflow, it becomes evident that a comprehensive evaluation framework is still lacking (Mikkola et al., 2024b). Establishing such a framework would not only benefit the assessment of individual EPE methods (for example, through a standardized set of diagnostics) but also facilitate more meaningful comparisons across different methods. In this context, the field could further benefit from the adoption of benchmark datasets, as is common practice in machine learning (Mikkola et al., 2024b). A further interesting avenue for future research would be to more closely integrate the evaluation of EPE methods, within the broader elicitation process. In psychology, questionnaires often include items specifically designed for data quality assurance; similar techniques could be incorporated into elicitation protocols. One example is Cook's protocol, which uses a set of *seed* variables to assess how well-calibrated experts are with respect to the elicitation task (European Food Safety Authority, 2014). These seed variables are selected to resemble the target quantities of interest but have known values to the facilitator. The expert's performance on these seed variables is then considered indicative of the expected quality of their judgments on the target quantities (O'Hagan, 2019).

2.6 Desiderata for an EPE method and software

The preceding discussion identified several challenges associated with the development of EPE methods. Furthermore, these challenges arise not only from the methods themselves but also from their integration into the EPE workflow, which is itself embedded within the broader Bayesian workflow. Figure 2.1 presents a schematic overview of the relationships among these concepts.

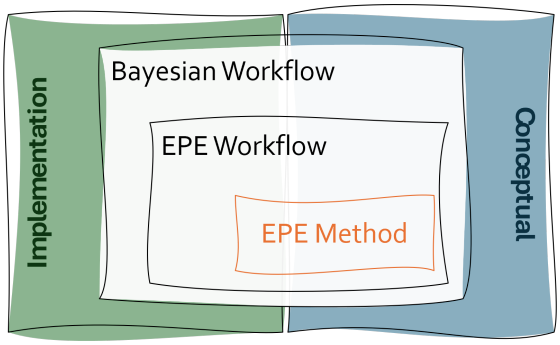


Fig. 2.1 Conceptual overview situating an EPE method within the EPE workflow which is itself a subworkflow of the broader Bayesian workflow. Requirements and challenges associated with integrating these components refer to both conceptual and implementation aspects.

The *EPE workflow* encompasses the entire EPE process: from defining the generative model and selecting summary statistics, through the elicitation stage, to model fitting and subsequent evaluation. A feedback cycle is typically incorporated into this workflow to iteratively align the fitted prior distributions with the elicited expert knowledge. The EPE workflow can be regarded as a subworkflow within the overarching *Bayesian workflow*, specifically associated with the prior specification stage. The *EPE method* corresponds to the fitting stage of the EPE workflow with the task of translating the expert-elicited summaries into corresponding prior distributions.

Developing a conceptual understanding of the relationships among these concepts facilitates the identification and structuring of challenges associated with EPE methods and their implementation. Based on this analysis, we derived a set of desiderata which we present in Table 2.2 and 2.3. The list is not intended to be exhaustive or mutually exclusive but represents an initial framework that will require refinement and critical evaluation by the research community. Within this thesis, its primary function is to structure our research process, clarify objectives, and guide methodological development. The desiderata will be addressed and discussed at relevant points in the subsequent chapters.

Desid.	*An ideal EPE* method ...
D-M1	accommodates a *flexible definition of target quantities*, supporting quantities defined in both the parameter space and the observable space
D-M2	accommodates a *flexible range of elicitation techniques*, such as moments, quantiles, and distributions
D-M3	is *agnostic to the model formulation*
D-M3.1	is *agnostic to the functional form of the data distribution*
D-M3.2	is *agnostic to characteristics of the prior*
D-M4	is *computationally efficient*, enabling real-time feedback during the elicitation process
D-M5	*integrates into the broader EPE workflow*
D-M6	*propagates total uncertainty from the elicitation process* into the resulting prior distributions
D-M7	always *returns a learned prior distribution*, regardless of how limited the input information is
D-M8	detects *incoherent input information* and reconciles inconsistencies where possible or provides feedback on the incoherence
D-M9	*returns the same set of learned priors when fitted to the same set of expert-elicited summaries*, thereby ensuring reproducibility

Table 2.2 List of desiderata for an ideal EPE method. Abbreviation: Desid., Desideratum.

Desid.	*An ideal EPE* software . . .
D-S1	is developed in accordance with the *FAIR* principles (*F*indable, *A*ccessible, *I*nter-operable, *R*eusable)
D-S2	is *functional correct* in that it produces correct results for every run
D-S3	provides *interfaces* compatible with expert-friendly elicitation tools accommodating different response formats
D-S4	*can be used to support an elicitation protocol* by enabling immediate fitting of prior distributions to elicited summaries, providing informative visual feedback and diagnostics, allowing straightforward modification of inputs, and facilitating efficient re-fitting
D-S5	is *modular, open-source, and version-controlled*, thereby facilitating community-driven development, ease of modification, integration of extensions, and transparency
D-S6	*integrates into the broader Bayesian workflow* by enabling seamless exchange of information between the EPE and Bayesian workflows (e.g., through direct use of model definitions in EPE and transfer of learned priors into Bayesian inference)
D-S7	is *compatible with different probabilistic programming languages*
D-S8	is *facilitator-friendly* by providing an intuitive interface, comprehensive documentation, tutorials, and case studies.
D-S9	includes a *standard set of evaluation metrics, diagnostics, and visualization tools*

Table 2.3 List of desiderata for an ideal EPE software. Abbreviation: Desid., Desideratum.

CHAPTER 3

Simulation-based expert prior elicitation

The challenges identified in the previous chapter regarding EPE methods and their implementation highlight the need for further refinement of currently available approaches. In response, we introduce a hybrid, model-agnostic simulation-based EPE method composed of modular building blocks, allowing adaptation to a wide range of problem settings. Our method builds on recent advances in *predictive* elicitation, specifically those described by Manderson and Goudie (2024), Hartmann and Agiashvili (2020), and da Silva et al. (2023). In Bockting et al. (2024), we introduce a simulation-based EPE method with a focus on parametric prior distributions (Section 3.1). In Bockting et al. (2025), we extent this method to non-parametric joint prior distributions (Section 3.2). Furthermore, to improve reproducibility and accessibility, the EPE method has been implemented as a version-controlled, open-source Python package, *elicito* (Bockting and Bürkner, 2025), which is actively maintained and available via GitHub, PyPI, and conda-forge (Section 3.3). Table 3.1 summarizes our contributions to the EPE research field. A short summary of each contribution is provided in the subsequent sections.

Reference	*Description*	*Section*
(Bockting et al., 2024)	Introduction of a modular, simulation-based EPE method enabling flexible specification of data distributions, target quantities, and elicitation techniques, with a focus on learning parametric prior distributions.	Section 3.1
(Bockting et al., 2025)	Extension of the method proposed by Bockting et al. (2024) to accommodate non-parametric joint priors via normalizing flows.	Section 3.2
(Bockting and Bürkner, 2025)	Development of a modular, open-source, and version-controlled Python package implementing the simulation-based EPE method described in Bockting et al. (2024, 2025).	Section 3.3

Table 3.1 Overview of publications included in this thesis and their main contributions to the EPE research field

3.1 A simulation-based EPE method for parametric priors

3.1.1 Motivation

In Bockting et al. (2024), we propose an EPE method that builds upon recent advances in the field of *predictive* prior elicitation by Manderson (2023), da Silva et al. (2023), and Hartmann and Agiashvili (2020). Our approach extends this line of research by introducing a *hybrid* framework that allows target quantities to be specified in both the parameter space and the observable space. Furthermore, the method supports custom specifications of generative models, target quantities, and elicitation techniques, enabled by its modular design and simulation-based approach. Figure 3.1 illustrates the central idea of our simulation-based EPE method and its integration within the EPE process.

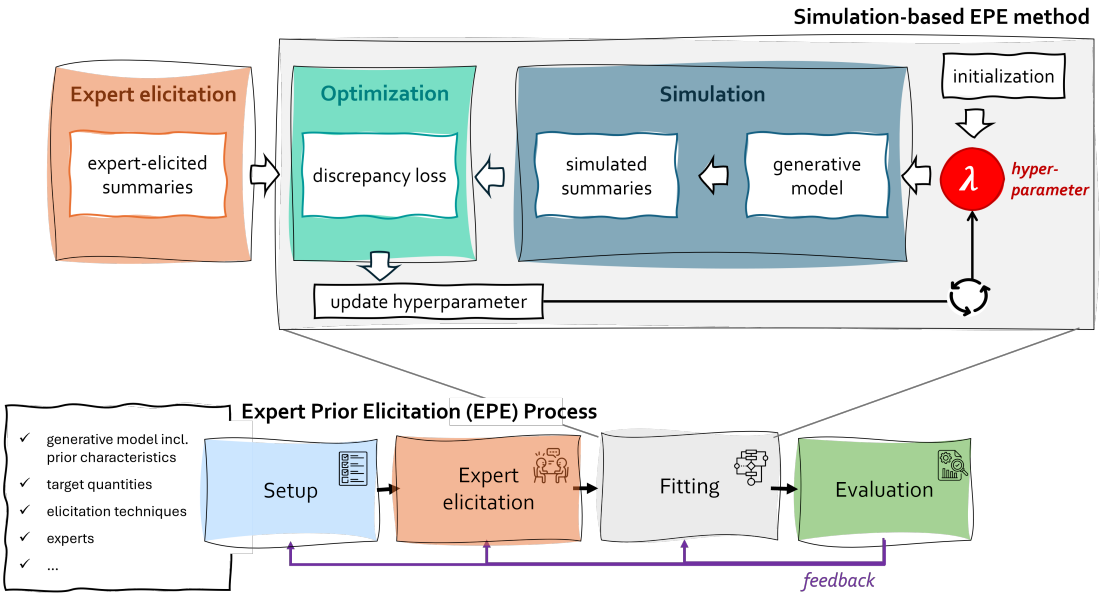


Fig. 3.1 Graphical representation of the conceptual workflow underlying the simulation-based EPE method in the context of the EPE process

Given a generative model and a set of initial hyperparameters defining the prior distributions, the model can be run in forward mode to simulate elicited summaries by computing the predefined target quantities and summary statistics. These simulated summaries are then compared with the expert-elicited summaries obtained during the expert-elicitation stage. An iterative optimization scheme is employed to update the hyperparameters of the parametric prior distributions so as to minimize the discrepancy between simulated and expert-elicited summaries. In other words, the objective is to identify the vector of hyperparameters that yields the closest alignment between simulated and expert-elicited summaries.

3.1.2 Method

The following section provides an overview of the methodology proposed in Bockting et al. (2024). To ensure consistency and clarity within this thesis, the notation has been adjusted from the original publication where necessary. The core logic of our EPE method is summarized in a five-step workflow.

1. *Define the generative model*: Specify the generative model, including the functional form of the data distribution and the parametric family of prior distributions.

2. *Define target quantities and elicitation techniques*: Select the set of target quantities and determine the elicitation techniques to query the expert (cf. elicited summaries).

3. *Simulate elicited summaries*: Draw samples from the generative model and compute the corresponding set of simulated elicited summaries.

4. *Evaluate discrepancy between simulated and expert-elicited summaries*: Assess the discrepancy between the simulated and expert-elicited summaries using a multi-objective loss function.

5. *Adjust prior hyperparameters to minimize discrepancy*: Apply an optimization scheme to update the prior hyperparameters such that the loss function is minimized.

In the following sections, we provide a formal introduction for each step.

Generative model The definition of the generative model $\mathcal{M}$ comprises the data distribution, $p(y \mid \theta)$, and the specification of parametric prior distribution families, $p(\theta \mid \lambda)$. In the formulation presented in Equation 3.1, we additionally introduce a vector of derived model parameters, η, and derived model outcomes, z, which are defined as transformations of the model parameters, $f_j^{\text{par}}(\theta_j)$, and the model outcomes, $f^{\text{dat}}(y)$, respectively:

$$\left.\begin{aligned} \theta &\sim p(\theta \mid \lambda) \\ \eta_j &= f_j^{\text{par}}(\theta_j) \end{aligned}\right\} \text{ parameter space}$$
$$\left.\begin{aligned} y &\sim p(y \mid b, \eta) \\ z &= f^{\text{dat}}(y) \end{aligned}\right\} \text{ observable space} \tag{3.1}$$

with $\eta = (\eta_1, \ldots, \eta_J)$ and b denotes a vector of potential covariates within a regression context. Consequently, given a set of hyperparameter values λ, the generative model

can be executed in forward mode to draw a total of S samples ($s = 1, \ldots, S$) for the parameters $(\theta^{(s)}, \eta^{(s)})$ and observables $(y^{(s)}, z^{(s)})$ specified by $\mathcal{M}$.

Target quantities Besides the generative model, it is necessary to formalize the selected set of *target quantities*. A target quantity is defined as a random variable $T \in \mathfrak{T}$ distributed according to p_T

$$T \sim p_T, \tag{3.2}$$

where T may refer to any quantity in the parameter space, the observable space, or any transformation thereof, i.e., $\mathfrak{T} = \phi_\Theta(\Theta) \cup \phi_\mathcal{Y}(\mathcal{Y})$ with $\phi_\Theta, \phi_\mathcal{Y}$ denoting arbitrary transformations of the respective spaces. Using the notation introduced in Equation 3.1, the different types of target quantities can be represented as

$T : \theta \sim p(\theta \mid \lambda)$ (parameter)	$T : y \sim p(y \mid b, \lambda)$ (outcome)
$T : \eta \sim p(\eta \mid \lambda)$ (deriv. parameter)	$T : z \sim p(z \mid b, \lambda)$ (deriv. outcome)

Instead of directly employing the density p_T, our simulation-based approach involves sampling from the generative model yielding realizations drawn from p_T

$$\{t^{(s)}\}_{s=1}^{S} \sim p_T(t \mid \lambda). \tag{3.3}$$

Typically, a *set* of P target quantities is defined, resulting in a vector of samples for each defined target quantity, $\{t_p^{(s)}\}_{s\in\mathbb{N},p\in\mathbb{N}}$.

Elicitation techniques Given the set of target quantities, the next step is to formalize *how* these quantities are queried from an expert, that is, to define the corresponding *elicitation techniques*. For each of the P target quantities Q_p elicitation techniques $f_{p,q}$ are specified with $q = 1, \ldots, Q_p$. An elicitation technique applied to a target quantity yields in an elicited summary

$$E_m = f_{p,q}\left(\{t_p^{(s)}\}_{s=1}^{S}\right) \quad \text{for } p = 1, \ldots, P. \tag{3.4}$$

The index $m = 1, \ldots, M$ arises from the combination of target quantities and the elicitation techniques specified for each target quantity. For instance, consider two target quantities, $\{t_1^{(s)}\}$ and $\{t_2^{(s)}\}$. Suppose the median, MED$(\{t_1^{(s)}\})$, is elicited for $\{t_1^{(s)}\}$, while the 25th, 50th, and 75th percentiles, $Q_{25}(\{t_2^{(s)}\})$, $Q_{50}(\{t_2^{(s)}\})$, and $Q_{75}(\{t_2^{(s)}\})$, are elicited for $\{t_2^{(s)}\}$. In this case, the total number of elicited summaries is $M = 4$. An elicited summary E_m may take the form of a scalar, for example, when the median or a quantile of a target quantity is elicited, or a vector. The final set of elicited summaries is denoted by $\{E_m\}_{m=1}^{M}$.

Discrepancy measure Once the generative model $\mathcal{M}$, target quantities $\{t_p^{(s)}\}$, and elicitation techniques $f_{p,q}$ are specified, the corresponding samples can be generated in forward mode for a given λ. This procedure results in the set of simulated summaries $\{E_m\}_{m=1}^M$. Subsequently, the discrepancy between each pair of simulated and expert-elicited summaries is computed via a discrepancy measure $\mathcal{D}_m$, which may vary across the elicited summaries $\{E_m\}_{m=1}^M$. For clarity, we introduced a slightly adjusted notation to differentiate between simulated data (i.e., the simulated elicited summaries), denoted by $\tilde{E}_m$, and observed data (i.e., the expert-elicited summaries), denoted by $\dot{E}_m$. We further use $\tilde{E}_m(\lambda)$ to explicitly show the functional dependence of the simulated summaries on the hyperparameters λ, which are subject to the optimization procedure. The discrepancy between the simulated and expert-elicited summaries can then be denoted by

$$\mathcal{D}_{m'}(\tilde{E}_{m'}(\lambda), \dot{E}_{m'}) \tag{3.5}$$

where $\mathcal{D}_{m'}$ denotes the chosen discrepancy measure. A new index, $m' = 1, \ldots, M'$, is introduced to account for the possibility of concatenating multiple elicited summaries into a single representation. For example, revisiting the case described above, the three elicited quantiles for $\{t_2^{(s)}\}$ may be concatenated into a single vector. In this case, the total number of loss components is reduced, yielding $m' = 1, 2$. However, the impact of concatenating elicited statistics on the optimization results is not yet fully understood and warrants further investigation.

Multi-objective loss function To obtain a single loss value, the individual discrepancy measures, $\mathcal{D}_{m'}$ (also referred to as *loss components*), are combined via a *multi-objective loss function* denoted by $\mathcal{L}(\lambda)$. As aggregation method, a weighted sum is employed, with weights $\gamma_{m'}$ assigned to each discrepancy measure $\mathcal{D}_{m'}$

$$\mathcal{L}(\lambda) = \sum_{m'=1}^{M'} \gamma_{m'} \cdot \mathcal{D}_{m'}(\dot{E}_{m'}(\lambda), \tilde{E}_{m'}). \tag{3.6}$$

Under this loss formulation, the optimization objective is to determine the hyperparameters λ^* that minimize the multi-objective loss, thereby reducing the overall discrepancy between the simulated and expert-elicited summaries

$$\lambda^* := \arg\min_{\lambda} \mathcal{L}(\lambda). \tag{3.7}$$

Gradient-based optimization The current procedure for solving the objective denoted in Equation 3.7 utilizes an iterative approach. In each iteration (or epoch), a set of simulated elicited summaries is derived based on the current hyperparameter vector λ^{epoch}. The discrepancy between these simulated and the expert-elicited summaries is

then computed and aggregated to yield a total loss value, which subsequently drives the update of the hyperparameters λ

$$\lambda^{\text{epoch}} = \lambda^{\text{epoch}-1} - \delta \frac{\partial \mathcal{L}}{\partial \lambda}, \tag{3.8}$$

where δ denotes the learning rate. The updated hyperparameters are then used to simulate a new set of summaries, which enters the next loss computation. This updating procedure continues until a convergence criterion is satisfied. To implement this optimization procedure, we employ mini-batch stochastic gradient descent with automatic differentiation, facilitated by the reparameterization trick (Kingma et al., 2015).

3.1.3 Simulation Studies

We implemented the EPE method described in Section 3.1.2 in Python 3.9, specifically utilizing TensorFlow (v2.14, Abadi et al., 2015) and TensorFlow Probability (v0.22, Dillon et al., 2017). The implemented method was tested across four simulation studies, encompassing normal linear regression, binomial regression, Poisson regression, and a hierarchical model.

To construct the set of elicited summaries, we focused on target quantities in the observable space, and employed elicitation techniques based on quantiles, moments, and histograms. Rather than eliciting the summaries from a "real" expert, the set of "expert"-elicited summaries, $\{\dot{E}_m\}_{m=1}^{M}$, was derived from a vector of *true* (i.e., known to the analyst) hyperparameters, λ^{true}. This was achieved by executing our simulation approach once in forward mode using λ^{true}. For quantifying the discrepancy between the simulated and "expert"-elicited summaries, we employed the Maximum-Mean Discrepancy (MMD, Gretton et al., 2012) for all loss components. To minimize the overall loss function, the optimization procedure was run for a predefined number of epochs (700–1,000), requiring 30–90 minutes to complete on CPUs. In each iteration, 128 mini-batches were processed to enhance learning efficiency, with 200–300 samples drawn from the prior distribution per batch. The variation in these numbers reflects differences across case studies; within a single case study, the parameters remained constant.

Convergence was achieved for all case studies and was assessed according to the following criteria: (i) the final loss value decreases over epochs and approaches a value near zero; (ii) individual discrepancy measures decrease over epochs and approach values near zero; (iii) gradients with respect to each hyperparameter converge toward zero across epochs; and (iv) hyperparameter values converge and stabilize at specific values across epochs. In addition, across all case studies we demonstrated successful learning of the prior distributions which was evaluated by assessing the correspondence between (i) simulated and expert-elicited summaries; and (ii) true and simulated prior distribu-

tions.

3.1.4 Discussion and Conclusion

The EPE method introduced in Bockting et al. (2024) extents the current line of research in *predictive* elicitation: (i) It allows for a flexible definition of the functional form of the data distribution (**D-M3.1**), (ii) target quantites (**D-M1**), and (iii) elicitation techniques (**D-M2**). (iv) It integrates naturally into the broader EPE workflow (**D-M5**) by accommodating various decisions in upstream task of the EPE process. (v) Moreover, the simulation-based approach ensures that potential inconsistencies in the expert-elicited summaries are always reconciled in the simulated elicited-summaries, that is, the learned priors will always imply a set of coherent elicited summaries (**D-M8**). In this way, our introduced EPE method represents a valuable contribution to the field and is currently among the most flexible approaches available for EPE.

However, although our EPE method provides a promising direction for future research in EPE, there are multiple limitations associated with both the EPE method and our approach to evaluating its performance. Starting with the EPE method itself, we note for example that is does not (i) accommodate non-parametric joint priors, and is therefore not agnostic to characteristics of the prior distribution (**D-M3.2**); (ii) propagate the full uncertainty from the elicitation process, as only point estimates of the hyperparameters λ are learned, and meta-uncertainty in expert judgment is neglected (**D-M6**); (iii) consistently return the same set of learned priors when fitted to the same set of expert-elicited summaries (**D-M9**); instead, the method returns only a sample from the *identification region* (see Section 2.2.3); and (iv) handle appropriately situations in which only very sparse data are available (**D-M7**); when the elicited summaries are uninformative, the EPE method simply returns the randomly initialized hyperparameters, $\lambda^{\text{epoch}=0}$.

Concerning the evaluation approach used in Bockting et al. (2024), at least two key limitations should be noted: (i) we focused primarily on the self-consistency aspect, demonstrating that our method can recover the "true" prior distributions when provided with a sufficiently informative set of elicited summaries. This resulted in a large number of elicited summaries, some of which were difficult to interpret. While such preliminary simulation studies are important for establishing the validity of the method, it is equally important to demonstrate that the method behaves reasonably in more realistic settings, i.e., when only a smaller set of easily interpretable elicited summaries is available. This aspect, however, was not addressed in Bockting et al. (2024). (ii) Another key limitation of this work is the absence of a sensitivity analysis. Specifically, results were presented and discussed for only a single seed in each case study. Given the challenges associated with identifiability, it is important to perform sensitivity analyses to assess

the variability of the learned prior distributions.

3.2 Extending simulation-based EPE to non-parameteric priors

3.2.1 Motivation

While the simulation-based EPE method introduced in Bockting et al. (2024) shows promising results, it also reveals several areas for improvement. One limitation of Bockting et al. (2024) is its restriction to parametric prior distributions. As highlighted in **D-M3.2**, however, an ideal EPE method should be agnostic to the characteristics of the prior. Furthermore, the evaluation methodology employed in Bockting et al. (2024) has notable shortcomings, particularly the absence of sensitivity analyses and the reliance on relatively large sets of elicited summaries. Consequently, we extended in Bockting et al. (2025), the previously introduced EPE method to also accommodate non-parametric joint priors. Moreover, we improved the evaluation approach by incorporating sensitivity analyses and introducing additional evaluation criteria and diagnostics. With these extensions, Bockting et al. (2025) presents one of the first EPE methods that, in addition to the flexibility outlined in the previous section, can accommodate both parametric and non-parametric prior distributions.

3.2.2 Method

The general methodology remains consistent with that outlined in Section 3.1.2, with the exception that the prior distribution is now defined by a non-parametric joint probability density function $p_\lambda(\theta)$ over the model parameters θ. To learn $p_\lambda(\theta)$, we employ a *flow-based* generative model, a class of deep generative models that combine generative modeling with *normalizing flows* (NFs, Papamakarios et al., 2021; Kobyzev et al., 2020). Although the hyperparameters λ in $p_\lambda(\theta)$ still represent the optimization target, as they define the prior distribution to be inferred, their interpretation differs from that in parametric distribution families. Specifically, λ corresponds now to the parameters of the normalizing flows, i.e., the weights of the underlying deep neural networks (see details below). For this reason, we adopt a slightly different notation and write the prior probability density with λ as an index, $p_\lambda(\theta)$.

Normalizing Flows NFs transform a simple probability distribution, referred to as the *base distribution* $p(u)$, into a more complex *target distribution*, which in our case corresponds to the joint prior $p_\lambda(\theta)$. This transformation is achieved through a series of invertible and differentiable mappings g_λ, parameterized by λ (Kobyzev et al., 2020). The composition of these invertible mappings yields an explicit expression for the den-

sity $p_\lambda(\theta)$ via the change-of-variables formula

$$p_\lambda(\theta) = p(u = g_\lambda(\theta)) \mid \det g'_\lambda(\theta) \mid,$$

where $g'_\lambda(\theta) = \frac{\partial}{\partial\theta} g_\lambda(\theta)$ denotes the Jacobian matrix of g_λ at θ and $\mid \det g'_\lambda(\theta) \mid$ is its absolute determinant. This formulation enables efficient evaluation of the prior density at arbitrary parameter values θ (Dinh et al., 2016). Samples from the joint prior can be obtained by first drawing samples from the base distribution, $p(u)$, and then applying the inverse transformation $g_\lambda^{-1}(u)$ to generate samples from $p_\lambda(\theta)$

$$\theta = g_\lambda^{-1}(u) \sim p_\lambda(\theta) \quad \text{for} \quad u \sim p(u).$$

Different types of NFs are typically distinguished based on the form of the mapping g_λ. In Bockting et al. (2025), *affine coupling flows* are employed, as they are easily invertible, computationally efficient, and sufficiently expressive (Kobyzev et al., 2020).

3.2.3 Simulation Studies

The extended EPE method has been implemented in the Python package *elicito* (v0.3.1 Bockting and Bürkner, 2025) using Python 3.11. Its performance was evaluated across four simulation studies involving Binomial and normal regression models. These case studies were selected to provide simple, well-controlled scenarios in which the behavior of the method could be systematically examined. The set of target quantities comprised (i) the outcome variable conditional on specific values of the predictor, (ii) the coefficient of determination (R^2) for the linear regression models, and (iii) the pairwise correlations between model parameters. The set of elicitation techniques included quantile elicitation and for the correlation a point estimate for the central tendency was used.

Following Bockting et al. (2024), the set of "expert"-elicited summaries was constructed by simulating from "true" prior distributions. Although the learning objective targets a non-parametric joint prior, we employed *parametric true* priors (independent or joint) due to practical reasons. As discrepancy measures, we used the L2 loss for the correlation coefficients and the MMD for all other elicited summaries. The architecture of the NFs was based on a standard multivariate Gaussian as the base distribution and comprised three affine coupling blocks. Each coupling block consisted of two dense layers with 128 units and ReLU activation functions. The optimization procedure was executed for a predefined number of epochs (500–800), requiring approximately 7–15 minutes to complete on GPUs. The variation in these numbers reflects differences across case studies. In each iteration, 128 mini-batches were processed, with 200 samples drawn from the prior distribution per batch.

We extended the evaluation framework of Bockting et al. (2024) with four supple-

mentary analyses to thoroughly investigate the method's performance: (i) an assessment of the *informativeness of elicited summaries*, testing their sensitivity to changes in the true prior; (ii) repeated execution of the optimization procedure with different seeds (30 times per case study) for *sampling from the identification region* to better characterize the range of possible priors; (iii) an analysis of *loss convergence* that included examining the slope of the final loss values alongside visual inspection; and (iv) the implementation of *prior averaging* as an alternative to selecting a single final distribution.

Moreover, two additional sets of simulations are reported in the supplementary material[1]. In the first set, each case study was conducted using the parametric prior method introduced in Bockting et al. (2024), enabling a preliminary comparison between the parametric and non-parametric approaches within the simulation-based framework. In the second set, the model parameters were treated as target quantities, thereby both evaluating the method's performance in the parameter space and facilitating comparison between the true and learned prior distributions, effectively serving as a validity check of the implementation. All results indicate satisfactory performance of the EPE method, demonstrating that it successfully learned prior distributions corresponding to the provided set of elicited summaries.

3.2.4 Discussion and Conclusion

Building on the discussion in Section 3.1.4, the extended EPE method now fulfills **D-M3.2**, demonstrating flexibility in handling parametric and non-parametric priors in both independent and joint settings. Furthermore, we refined the evaluation framework to systematically assess the validity, faithfulness, and replicability of the learned prior(s). Particular attention was given to the construction of elicited summaries, with a strong emphasis on interpretability. Nevertheless, an important limitation persists: the pairwise correlations between model parameters are difficult to interpret. To enhance the practical utility of the method, future research should aim to infer parameter correlations from more interpretable elicited summaries.

The results in Bockting et al. (2025) suggest that deep generative models offer an interesting opportunity for learning non-parametric joint priors within simulation-based EPE methods. Nonetheless, we found that achieving satisfactory learning performance required substantial tuning of algorithm hyperparameters, including the learning rate, the number of coupling layers, and the choice of activation functions. This requirement presents a practical limitation, as EPE methods are expected to provide rapid feedback to support active interaction with domain experts during the elicitation stage. Future research should investigate alternative generative approaches for learning non-parametric joint prior densities, such as flow matching (Lipman et al., 2022) or diffusion models

[1]The supplementary material for this publication is available at `https://github.com/florence-bockting/non-parametric-prior`.

(Cao et al., 2024), as well as strategies for determining robust default algorithm hyperparameter configurations that are broadly applicable across diverse data models.

3.3 Software implementation for simulation-based EPE

3.3.1 Motivation

To facilitate accessibility of the simulation-based EPE method introduced in Bockting et al. (2024, 2025) for a broad audience, including both method developers and applied practitioners, a sustainable research software implementation is required. A guideline for developing sustainable research software is provided by the FAIR principles (*F*indable, *A*ccessible, *I*nteroperable, *R*eusable; Lamprecht et al., 2020; Hasselbring et al., 2020). FAIR research software increases scientific value by promoting four key characteristics: *Findability* and *Accessibility* contribute to transparency by ensuring that code and associated results are easily retrievable and usable by the broader research community. *Interoperability* enables integration with other software tools and workflows, while *Reusability* supports the verification of methods and results, thereby facilitating both reproducibility and the extension of prior work. Guided by these principles, we developed the Python package *elicito*, which is described in detail in Bockting and Bürkner (2025).

3.3.2 Implementation

The Python package *elicito* adheres to the FAIR principles and is publicly released under the permissive Apache 2.0 open-source license. The package is available via PyPI (`https://pypi.org/project/elicito/`) and conda-forge (`https://anaconda.org/conda-forge/elicito`). Comprehensive documentation, including the API reference, installation instructions, detailed explanations, and tutorials, is maintained on ReadTheDocs (`https://elicito.readthedocs.io`). The source code, together with metadata such as a short description, author information, dependency specifications, and citation guidance via a Citation File Format (Smith et al., 2016), is hosted on GitHub (`https://github.com/florence-bockting/elicito`) to ensure version control and support collaborative development. In addition, a snapshot of the GitHub repository is archived on Zenodo (`https://zenodo.org/records/15671710`), where it is assigned a persistent identifier to support long-term accessibility and reproducibility of published research results. The package is compatible with Python versions 3.9 through 3.12 and has the following libraries as core dependencies: tensorflow ($>= 2.16$), tensorflow-probability ($>= 0.24$), tf-keras ($>= 2.16$), numpy ($>= 1.24$), joblib ($>= 1.4.2$), and tqdm ($>= 4.38$). Dependency management is handled through the *uv* Python pack-

age manager. The software integrates a testing framework for continuous integration to ensure the reliability of current and future adaptations. It adheres to established software and data standards to enable interoperability with other software components. The package has been tested on Linux, Windows, and macOS.

User Interface The primary user interface of *elicito* is the *Elicit* class, through which the user can configure an appropriate EPE method by defining the respective input parameters:

- *model*: Define the generative model used in the elicitation procedure.
- *parameters*: Specify assumptions regarding the prior distributions over model parameters, including (hyper)parameter constraints, dimensionality, and parametric form.
- *targets*: Define the elicited summaries in terms of target quantities and corresponding elicitation techniques. Specifies the discrepancy measure and weight.
- *expert*: Provide the expert-elicited summaries.
- *optimizer*: Specify the optimization algorithm to be used, along with its hyperparameters (e.g., learning rate).
- *trainer*: Configure the overall training procedure, including seed, number of epochs, sample size, and batch size.
- *initializer*: Define the initialization strategy for the hyperparameters used to instantiate the simulation-based optimization process; only required for parametric prior distributions.
- *network*: Specify the architecture of the deep generative model; only required for non-parametric prior distributions.

By configuring these core components, *elicito* can accommodate a wide range of different generative models, target quantities and elicitation techniques.

Computational algorithm Algorithm 1 outlines the computational steps underlying *elicito*. In this pseudo-code we assume a batch size of one. However, in the actual implementation we use a batch size greater one, resulting in all computational objects carrying an additional dimension of size B. Furthermore, we slightly deviate from the previously introduced notation and denote the prior in the following algorithm by p_λ, irrespective of whether it is parametric or non-parametric.

Algorithm 1 Computational algorithm underlying *elicito* (batch size of one)

Require: λ_0 ▷ specify initialization for hyperparameter
Require: either DNN or $p_\lambda(\cdot)$ ▷ specify (non-)parametric prior
Require: $p\left(y \mid \theta, \lambda\right)$ ▷ define generative model
Require: T_p with $p = 1, \dots P$ ▷ define set of target quantities
Require: $f_{p,q}$ with $q = 1, \dots, Q_p$ ▷ define elicitation technique(s) for each target quantity
Require: $\mathcal{D}_m, \gamma_m$ with $m = p \times q$ ▷ define discrepancy measure and weight
Require: $\dot{E}_m$ ▷ pass expert-elicited summaries
Require: epochs, S, δ ▷ define training settings: epochs, prior samples, learning rate

$\lambda \leftarrow \lambda_0$ ▷ initialize hyperparameter λ
for epoch in epochs **do**
  $p_\lambda \leftarrow \text{DNN}(\lambda)$ ▷ (optional) learn prior density from DNN
  for $m = p \times q$ **do**
    draw $\{t_p^{(s)}\}_{s=1}^S$ from $p_T(t \mid \lambda)$ ▷ simulate target quantity
    $\tilde{E}_m \leftarrow f_{p,q}\left(\{t_p^{(s)}\}_{s=1}^S\right)$ ▷ compute elicited summaries
  end for
  $\mathcal{L} = \sum_{m=1}^M \gamma_m \mathcal{D}_m(\tilde{E}_m(\lambda), \dot{E}_m)$ ▷ compute multi-objective loss
  $\lambda_{\text{epoch+1}} \leftarrow \lambda_{\text{epoch}} - \delta \nabla_{\lambda_{\text{epoch}}} \mathcal{L}$ ▷ update λ via backpropagation
end for

Case Studies In Bockting and Bürkner (2025), two case studies are presented that illustrate the specification of the *Elicit* class under different configurations: The first case study assumes independent parametric priors and the second case study a non-parametric joint prior. To support reproducibility, all code for the case studies is openly available on GitHub[2]. Within the context of these case studies, we illustrate the use of the *el.plots* submodule, which is included in the *elicito* package and provides functionality for the graphical evaluation of convergence diagnostics and the learned prior distributions. Additionally, we demonstrate how the package supports parallel execution of multiple replications of the fitting procedure, thereby facilitating efficient sensitivity analyses.

Example plots for Case Study 1 are presented in Figures 3.2 and 3.3, which illustrate the convergence of model hyperparameters across training epochs and the final learned prior distributions, respectively. We ran six replications of Case Study 1 in parallel, resulting in the multiple lines shown in both figures, each representing the outcome of an individual replication.

[2]Code for case studies in Bockting and Bürkner (2025) is available at `https://github.com/florence-bockting/elicito-software-paper`.

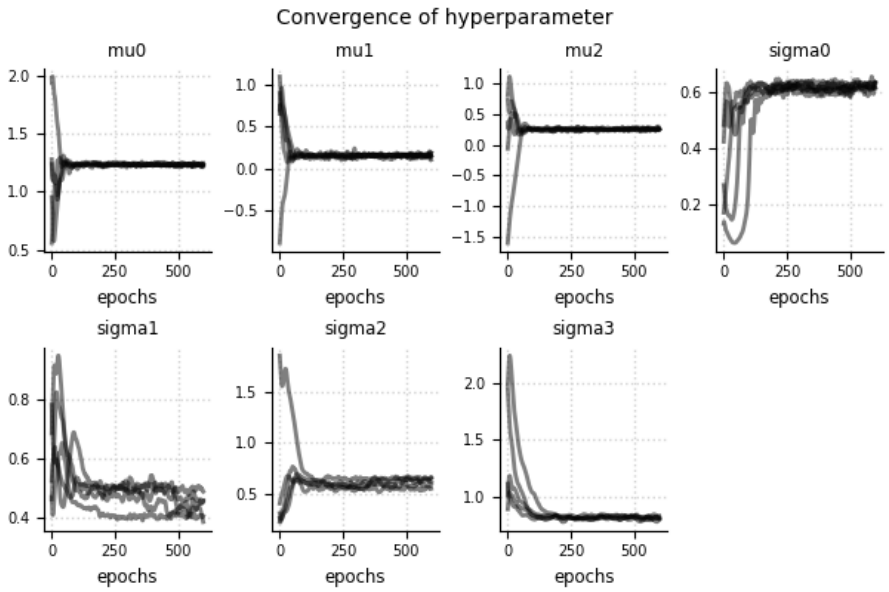


Fig. 3.2 Convergence of hyperparameters λ across epochs for six parallel runs. The plot is created using *el.plots.hyperparameter(eliobj)*.

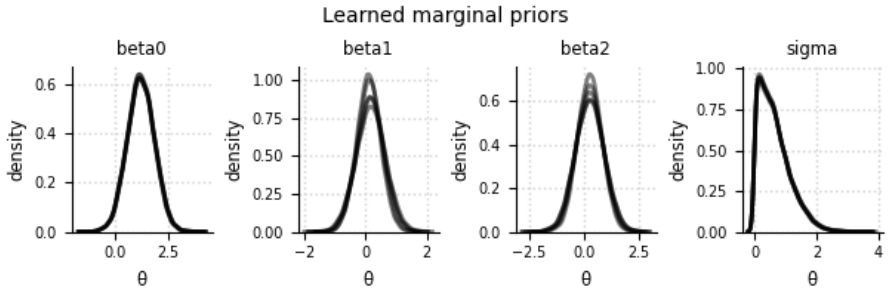


Fig. 3.3 Learned marginal prior distributions for all parallel runs. The plot is generated using *el.plots.prior_marginal(eliobj)*.

3.3.3 Discussion and Conclusion

With *elicito*, we provide a software implementation whose development has been guided by the FAIR principles (**D-S1**). To ensure interoperability with other software components, we adhere to established software and data standards. This enables, for example, future extensions of *elicito*, such as the integration of graphical user interfaces for the elicitation stage or the addition of domain-specific tools for post-processing of results (**D-S3**, **D-S4**). The architecture of *elicito* is highly modular, and the code is open source and version controlled (**D-S5**), enabling community-driven development. To ensure the reliability of the software implementation (**D-S2**), we provide a continuous testing framework to guarantee internal consistency, alongside the option to run the optimization procedure using true prior distributions instead of expert data. This enables verification of user-specific configurations of the *Elicit* class before observing any data (here the elicted expert information).

While the current implementation represents a promising first step, several enhance-

ments are necessary to make *elicito* truly accessible to a broad audience. The software currently includes documentation covering API references, tutorials, and additional explanations (**D-S8**); however, more comprehensive documentation is needed, along with further tutorials and case studies to demonstrate the use of *elicito* in practical applications. Although the implementation provides a basic set of visual tools for assessing convergence and evaluating the learned prior distributions, a more extensive suite of diagnostics, evaluation metrics, and visualizations is required to support thorough model assessment and interpretation (**D-S9**). Moreover, for seamless integration into the broader Bayesian workflow, the software should be implemented in, or at least compatible with, probabilistic programming languages commonly used in Bayesian inference, such as Stan (Stan Development Team, 2025) or PyMC (Salvatier et al., 2016). Future development efforts will therefore focus on restructuring or re-implementing *elicito* to ensure such compatibility (**D-S6**, **D-S7**).

CHAPTER 4

Discussion and future work

Using expert knowledge to specify prior distributions over parameters in Bayesian models enables the integration of existing domain knowledge into statistical inference. This approach is particularly important in settings where data are sparse. Typical examples include the development of new pharmaceuticals, the launch of a product in a novel market, the evaluation of educational interventions in unfamiliar contexts, or the assessment of risks associated with rare events (e.g., flooding, volcanic eruptions; O'Hagan, 2019; Knol et al., 2010; Choy et al., 2009; Azzolina et al., 2021; Lesaffre et al., 2020). Although the potential of EPE has long been acknowledged, its routine application in practice remains limited. Several barriers contribute to this gap, as discussed in a recent review by Mikkola et al. (2024b). Many existing EPE methods are highly model-specific, which hinders their adaptation to diverse applications (Manderson and Goudie, 2024; Kadane and Wolfson, 1998). Furthermore, clear guidelines for establishing standardized EPE workflows, particularly with respect to diagnostics and procedures for evaluating the elicited prior distributions, are lacking. The selection of an appropriate set of elicited summaries often remains ambiguous, and substantial challenges persist in the elicitation of non-parametric, joint prior distributions (Garthwaite et al., 2005; Clemen et al., 2000). Finally, while methodological developments have received considerable attention, comparatively little effort has been directed toward the development of open-source, user-friendly software to facilitate practical implementation (Mikkola et al., 2024b; Johnson et al., 2010a).

Against this background, the present thesis contributes to the field by developing a simulation-based EPE method (Bockting et al., 2024, 2025). The proposed method builds directly on recent developments in predictive EPE (e.g., da Silva et al., 2023; Hartmann and Agiashvili, 2020; Manderson and Goudie, 2024), while providing increased flexibility and modularity due to its simulation-based approach. The EPE method is both *model-agnostic* and *hybrid*: it accommodates parametric and non- parametric as well as joint and independent priors, different data distributions (continuous and discrete), target quantities, and elicitation techniques. Alongside this methodological contribution, the method has been implemented in a version-controlled, open-source Python package *elicito* (Bockting and Bürkner, 2025), developed in accordance with the

FAIR principles of research software engineering (Hasselbring et al., 2020).

The development of an EPE method involves several challenges, which can be broadly categorized into conceptual and implementation challenges, as well as the overarching challenge of embedding the method within the EPE workflow and, more generally, the Bayesian workflow. We begin our discussion by examining the main conceptual challenges encountered during the development of our simulation-based EPE method.

Conceptual challenges A distinctive feature of the proposed EPE method is its *flexibility*, which enables its application across a wide range of settings. However, this flexibility also introduces new challenges in providing adequate guidance for practitioner decision-making (Simpson et al., 2017). Our method enables users to specify target quantities related to model parameters, the outcome variable, or derivations of both. While the ability to select target quantities according to the needs of a given situation enhances interpretability, it simultaneously raises the question of whether such self-constructed sets of elicited summaries are sufficiently informative to learn the prior distributions, particularly when the target quantities refer to the observable space (i.e., predictive elicitation). Currently, the primary approach to addressing this question is conducting a sensitivity analysis, in which the fitting procedure is repeated with the same set of elicited summaries and subsequently the range of fitted prior distributions is examined (Manderson and Goudie, 2024; Mikkola et al., 2024b; Gelman et al., 2020; Depaoli et al., 2017). If the resulting prior distributions are inconsistent with the expectations of the domain expert or analyst, this indicates that the elicited information does not sufficiently constrain the problem and requires refinement. However, this approach presents its own challenges. For instance, how many repetitions are necessary to obtain a representative sample from the set of possible prior distributions? Can the sampling of this set be performed more efficiently? How should the elicited summaries be adjusted or extended to better constrain the implied set of priors? These questions highlight the need for methodological improvements.

One avenue for improvement concerns the *sensitivity analysis* itself. For example, implementing a Bayesian inference framework that provides the full distribution of potential priors, rather than random samples obtained through repeated fitting, would be advantageous. Furthermore, it would provide access to the full range of diagnostics for assessing the posterior distribution. For instance, one could evaluate posterior contraction across different sets of elicited summaries and select the set that yields the strongest contraction. However, a non-trivial challenge persists when formulating the problem as a Bayesian inference task, namely the specification of prior distributions. In this case, priors are required for the model hyperparameters, which effectively shifts the problem to a higher level of the hierarchy rather than resolving it.

Regardless of the approach adopted for sensitivity analysis, informative *diagnostics*

that quantify the information content of elicited summaries are essential for guiding the selection of target quantities and elicitation techniques. This consideration also extends to settings in which the outcome variable conditional on predictor values is used as target quantity, thereby raising the question of which predictor values should be selected for expert elicitation. While interpretability is an important criterion, the choice should additionally be guided by informativeness. Achieving this, in turn, requires diagnostics that facilitate the comparison of alternative configurations and the identification of the most informative option. However, a standardized set of diagnostics for predictive EPE methods is currently lacking (Mikkola et al., 2024b).

Another challenge concerns the specification of *non-parametric joint priors* (Garthwaite et al., 2005; Oakley et al., 2010; Oakley and O'Hagan, 2007). Such flexible prior distributions require highly specialized information, such as the specification of covariance structures, to adequately constrain the problem. Yet, this type of information is often difficult for domain experts to interpret (Clemen et al., 2000; O'Hagan, 2019). Although several approaches have been proposed, the current situation remains suboptimal for practical applications (Mikkola et al., 2024b; Gosling et al., 2007; Moala and O'Hagan, 2010; Clemen et al., 2000; Oakley and O'Hagan, 2007). Our proposed EPE method is capable of learning both parametric and non-parametric joint priors (Bockting et al., 2025). In the non-parametric case, we employ normalizing flows (Kobyzev et al., 2020) and can demonstrate that, when information regarding the outcome variable and pairwise correlations among model parameters is available, a non-parametric joint prior can be effectively learned. However, eliciting pairwise correlations among model parameters from domain experts is generally infeasible in practice (Clemen et al., 2000; O'Hagan, 2019). Consequently, we need to focus in future work on developing expert interfaces that query interpretable quantities and translate them into correlation information suitable for use in our EPE method. Notably, Mikkola et al. (2024a) recently proposed an approach that uses preferential judgments to infer the correlation structure of a non-parametric joint prior. Integrating their proposal with our EPE method represents a promising direction for future research.

We have already noted the absence of a standard set of diagnostics for evaluating the results of an EPE method. More broadly, a *general evaluation framework* is lacking, that additionally enables the comparison of EPE methods with respect to relevant criteria such as faithfulness, validity, replicability, and efficiency (Johnson et al., 2010a; Mikkola et al., 2024b; Johnson et al., 2010b). One reason for the absence of an overarching evaluation framework is that EPE methods have often been introduced as standalone, highly integrated, and model-specific solutions, creating the impression that they are largely incomparable (Gelman et al., 2020). This has resulted in a landscape in which numerous EPE methods coexist, complicating orientation for both method developers and practitioners. In this respect, we see substantial potential in developing

EPE methods within a modular workflow (Gelman et al., 2020). This approach allows the definition of main, interconnected modules that are common across EPE methods, thereby facilitating comparability, while the specification of submodules can remain highly case-specific, preserving flexibility between methods. We incorporated the principle of modular design in the development of our proposed EPE method, an approach strongly informed by a research software engineering perspective during its implementation (Johanson and Hasselbring, 2018; Hasselbring, 2018). We found that systematically identifying the main modules of an EPE method, as well as distinguishing which components should be easily extendable or modifiable, was highly valuable. It helped to impose structure on the complex task of developing an EPE method and facilitated clear communication of the primary tasks, conceptualized as modules. Moreover, it supports collaborative development, as the overall problem can be decomposed into smaller submodules that can be independently developed and readily integrated into the main workflow.

Implementation challenges In addition to the introduced conceptual challenges, we also encountered several practical difficulties related to the implementation of our proposed EPE method in the research software *elicito*. In the following, we outline the main implementation challenges, beginning with the formulation of the optimization problem. Specifically, the problem addressed by our EPE method can be expressed as a multi-objective optimization problem (see Section 3.1.2). In our implementation, this problem is expressed as a weighted sum of the individual loss components, which requires *specifying a weight* for each loss component. In the current implementation, we leave the specification of the weights to the user, although we acknowledge that this approach is suboptimal. Future developments should include options for automated weight computation, aiming to achieve an optimal balance among the loss components during training. This issue is precisely what automatic loss weighting strategies are designed to address, ensuring effective learning across all loss components. Several such strategies have been proposed, including Uncertainty Weighting (Kendall et al., 2018), Dynamic Weight Average (Liu et al., 2019), Gradient Normalization (Chen et al., 2018), and Projecting Conflicting Gradients (Yu et al., 2020). For an overview, see Song et al. (2022). Future extensions of *elicito* should incorporate a selection of these automatic loss weighting strategies as default option.

Directly related to the formulation of the loss components is the selection of an appropriate *discrepancy measure*, which quantifies the discrepancy between expert-elicited summaries and simulated summaries. In the current implementation, we provide two discrepancy measures: the L2 loss and the Maximum Mean Discrepancy (Gretton et al., 2012). Further research is required to systematically evaluate alternative discrepancy measures with respect to their efficiency and effectiveness. In particular,

different measures may be more appropriate for specific types of summaries. An advantage of our implementation is that the discrepancy measure is a readily extendable and modifiable submodule, thereby facilitating comparative analyses.

Another more general challenge with respect to the optimization scheme is the choice of the optimization method itself. Currently, *elicito* implements exclusively gradient-based optimization (Ruder, 2016). However, it comes with the prerequisite that all operations and functions in the computational graph must be differentiable or admit a reparameterization whose gradients can be approximated with sufficient accuracy. Consequently, for discrete random variables, specific techniques, such as the Gumbel-Softmax trick (Jang et al., 2016; Joo et al., 2025) are necessary to obtain approximate gradients. This technique has been applied in our case studies in Bockting et al. (2024, 2025). However, the current implementation of the Gumbel-Softmax trick becomes inefficient for complex discrete probability distributions, such as the Multinomial. The limitations associated with gradient-based methods highlight the potential of black-box optimization methods (Alarie et al., 2021). For example, Manderson and Goudie (2024) propose a two-stage optimization process based on Bayesian optimization (Frazier, 2018), which avoids the need for gradients altogether. A common limitation of black-box optimization methods, however, is their high computational cost (Wei et al., 2025), which can render them impractical for providing rapid feedback to domain experts during the elicitation process. Due to the modular design of *elicito*, different optimization schemes can be implemented while preserving all other components. This flexibility offers a promising avenue for systematically analyzing scenarios to identify conditions under which black-box optimization outperforms gradient-based methods, and vice versa.

Another key direction for future development of *elicito* is the improvement of the implemented *initialization method* used for learning parametric prior distributions. The simulation-based optimization method requires an initial set of hyperparameter values that is sufficiently close to those implied by the expert-elicited summaries. This requirement, however, introduces a circular problem, as the implied values are precisely the target of the optimization and therefore unknown. When the initial values deviate substantially from those implied by the expert-elicited summaries, the simulated summaries may diverge to infinity, thereby preventing gradient computation and causing the optimization to fail. This issue is particularly pronounced in high-dimensional problems or when non-trivial link functions are employed in the generative model. Currently, *elicito* offers two options for initialization: either manually specifying the initial value for each hyperparameter or sampling the initial values from a distribution. The latter approach is essentially equivalent to setting a prior on the hyperparameters. In both cases, the central question concerns which initial values lie within a sufficiently constrained space to avoid numerical instabilities. Future work should explore search

algorithms capable of efficiently and effectively navigating the space of possible values to identify a suitable set of initial hyperparameter values.

Integration within the EPE and Bayesian workflow The conceptual and implementation challenges discussed thus far refer specifically to the EPE method itself. However, it is also important to consider the broader integration of an EPE method within the overall EPE workflow, as the ultimate objective is its application in the elicitation process. This raises several follow-up questions. For instance, is there a *graphical user interface* capable of implementing different response formats and returning the input data in a format directly compatible with *elicito*? Existing tools such as MATCH (Morris et al., 2014) and SHELF (Oakley, 2025) address parts of this need, but they are implemented in R and therefore not directly compatible with our Python-based implementation. In this context, *elicito* may be regarded primarily as a backend, necessitating the development of practical user-facing frontends, which represent promising directions for future work.

Another important consideration when focusing on the elicitation stage, concerns the involvement of *multiple experts* and the challenge of combining individual expert prior distributions (Winkler, 1968; Rougier et al., 2013; Williams et al., 2021; Falconer et al., 2022). The combination of expert prior distributions is typically achieved through either mathematical aggregation, such as applying scoring rules to derive a combined prior, or behavioral aggregation, which entails group interaction to construct a consensus prior distribution (Colson and Cooke, 2018). Elicitation software can play a central role in supporting these tasks. For instance, appropriate visualization tools can facilitate group discussions in behavioral aggregation by displaying individual expert priors and highlighting variability, thereby helping experts to identify the sources of disagreement. Furthermore, elicitation software can support mathematical aggregation by providing implementations of different aggregation strategies, including axiomatic and Bayesian approaches (Colson and Cooke, 2018).

Overall, it would be advantageous to develop elicitation software that not only supports the translation of expert knowledge into prior distributions (i.e., the EPE method) but also facilitates interaction with experts during the elicitation stage (i.e., integration within the EPE workflow). Such software would enable a more integrated workflow, providing clearer *guidelines* for users on how to apply prior elicitation and associated tools in *real-world applications*. This, in turn, would address another limitation of current EPE workflows, namely the scarcity of documented case studies that demonstrate their application under the complexities of practical settings (Mikkola et al., 2024b).

Finally, the practical utility of EPE methods within the broader *Bayesian workflow* requires that the EPE software is compatible with widely used modeling ecosystems, ensuring consistency in data formats, interface conventions, and probabilistic program-

ming languages (Mikkola et al., 2024b). At present, our implementation in *elicito* is based on TensorFlow (Abadi et al., 2015) and TensorFlow Probability (Dillon et al., 2017). However, the main modeling ecosystems are currently implemented in Stan (Stan Development Team, 2025) or PyMC (Salvatier et al., 2016). Future work will focus on integrating *elicito* into one of these primary modeling ecosystems, with PyMC representing the most straightforward alternative due to its native Python implementation.

Conclusion

This thesis contributes to the field of Expert Prior Elicitation by proposing a simulation-based method capable of accommodating a broad range of statistical models, target quantities, and elicitation techniques. In addition, we provide an actively maintained, and highly modular software implementation of the proposed method. The modular structure of both the EPE framework and its software facilitates straightforward extension and modification, thereby supporting further method development and practical applications alike.

The EPE research field offers an extensive body of literature addressing both methods for translating expert knowledge to prior distributions as well as guidelines for extracting knowledge from experts. However, in our view, what is currently lacking is a concerted effort to join the various contributions within this domain and to integrate them into a coherent "prior elicitation ecosystem". This observation applies equally to methodological developments and their software implementations. The absence of a unified framework is reflected, among other things, in the lack of a standardized set of diagnostics, systematic comparisons across EPE methods, and a software ecosystem that provides interoperable and compatible tools supporting the entire EPE workflow. Such an ecosystem would encompass expert-friendly front-end tools for eliciting expert knowledge, flexible EPE methods for learning prior distributions from elicited information, and tools for diagnosing, visualizing, and further processing of results (e.g., aggregation methods for multi-expert settings).

With this thesis, we aim to take an initial step toward addressing this gap by proposing a modular EPE method and a corresponding software implementation that allows for the integration of various EPE methods that have been proposed in the field. In future research, we plan to extend these initial developments to achieve stronger integration of both the method and software within the overall EPE workflow and the broader Bayesian workflow, thereby facilitating collaboration with the research community.

REFERENCES

Abadi, M., et al. (2015). TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems [Computer software]. https://doi.org/10.5281/zenodo.4724125.

Aguilar, J. E. and Bürkner, P.-C. (2023). Intuitive joint priors for Bayesian linear multilevel models: The R2D2M2 prior. *Electronic Journal of Statistics*, *17*(1), 1711–1767, https://doi.org/10.1214/23-EJS2136.

Aguilar, J. E. and Bürkner, P.-C. (2025). Generalized Decomposition Priors on R2. *Bayesian Analysis*, *1*(1), 1–34, https://doi.org/10.1214/25-BA1524.

Al-Awadhi, S. A. and Garthwaite, P. H. (1998). An elicitation method for multivariate normal distributions. *Communications in Statistics - Theory and Methods*, *27*(5), 1123–1142, https://doi.org/10.1080/03610929808832149.

Alarie, S., Audet, C., Gheribi, A. E., Kokkolaras, M., and Le Digabel, S. (2021). Two decades of blackbox optimization applications. *EURO Journal on Computational Optimization*, *9*, 100011, https://doi.org/10.1016/j.ejco.2021.100011.

Azzolina, D., Berchialla, P., Gregori, D., and Baldi, I. (2021). Prior Elicitation for Use in Clinical Trial Design and Analysis: A Literature Review. *International Journal of Environmental Research and Public Health*, *18*(4), 1833, https://doi.org/10.3390/ijerph18041833.

Bedrick, E. J., Christensen, R., and Johnson, W. (1996). A New Perspective on Priors for Generalized Linear Models. *Journal of the American Statistical Association*, *91*(436), 1450–1460, https://doi.org/10.1080/01621459.1996.10476713.

Bedrick, E. J., Christensen, R., and Johnson, W. (1997). Bayesian Binomial Regression: Predicting Survival at a Trauma Center. *The American Statistician*, *51*(3), 211–218, https://doi.org/10.1080/00031305.1997.10473965.

Berger, J. O. (1985). Prior Information and Subjective Probability. In *Springer Series in Statistics* (pp. 74–117). New York: Springer. https://doi.org/10.1007/978-1-4757-4286-2_3.

Berger, J. O. (2006). The case for objective Bayesian analysis. *Bayesian Analysis*, *1*(3), https://doi.org/10.1214/06-BA115.

Bernardo, J. M. (1979). Reference Posterior Distributions for Bayesian Inference. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, *41*(2), 113–128, https://doi.org/10.1111/j.2517-6161.1979.tb01066.x.

Bissiri, P. G., Holmes, C., and Walker, S. (2016). A General Framework for Updating Belief Distributions. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, *78*(5), 1103–1130, https://doi.org/10.1111/rssb.12158.

Bockting, F. and Bürkner, P.-C. (2025). elicito: A Python Package for Expert Prior Elicitation [Computer software]. *arXiv*, https://doi.org/10.48550/arXiv.2506.16830.

Bockting, F., Radev, S. T., and Bürkner, P.-C. (2024). Simulation-based prior knowledge elicitation for parametric Bayesian models. *Scientific Reports*, *14*(1), 17330, https://doi.org/10.1038/s41598-024-68090-7.

Bockting, F., Radev, S. T., and Bürkner, P.-C. (2025). Expert-elicitation method for non-parametric joint priors using normalizing flows. *Statistics and Computing*, *35*(5), 132, https://doi.org/10.1007/s11222-025-10665-z.

Bornkamp, B. and Ickstadt, K. (2009). A Note on B-Splines for Semiparametric Elicitation. *The American Statistician*, *63*(4), 373–377, https://doi.org/10.1198/tast.2009.08191.

Bürkner, P.-C. (2017). brms: An R package for Bayesian multilevel models using Stan [Computer software]. *Journal of Statistical Software*, *80*, 1–28, https://doi.org/10.18637/jss.v080.i01.

Cao, H., Tan, C., Gao, Z., Xu, Y., Chen, G., Heng, P.-A., and Li, S. Z. (2024). A survey on generative diffusion models. *IEEE Transactions on Knowledge and Data Engineering*, *36*(7), 2814–2830, https://doi.org/10.1109/TKDE.2024.3361474.

Chen, Z., Badrinarayanan, V., Lee, C.-Y., and Rabinovich, A. (2018). GradNorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In J. Dy and A. Krause (Eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research* (pp. 794–803).: PMLR. https://proceedings.mlr.press/v80/chen18a.html.

Choy, S. L., O'Leary, R., and Mengersen, K. (2009). Elicitation by design in ecology: using expert opinion to inform priors for Bayesian statistical models. *Ecology*, *90*(1), 265–277, https://doi.org/10.1890/07-1886.1.

Clemen, R. T., Fischer, G. W., and Winkler, R. L. (2000). Assessing Dependence: Some Experimental Results. *Management Science*, *46*(8), 1100–1115, https://doi.org/10.1287/mnsc.46.8.1100.12023.

Clemen, R. T. and Winkler, R. L. (1999). Combining Probability Distributions From Experts in Risk Analysis. *Risk Analysis*, *19*(2), 187–203, https://doi.org/10.1023/A:1006917509560.

Colson, A. R. and Cooke, R. M. (2018). Expert Elicitation: Using the Classical Model to Validate Experts' Judgments. *Review of Environmental Economics and Policy*, *12*(1), 113–132, https://doi.org/10.1093/reep/rex022.

Consonni, G., Fouskakis, D., Liseo, B., and Ntzoufras, I. (2018). Prior Distributions for Objective Bayesian Analysis. *Bayesian Analysis*, *13*(2), https://doi.org/10.1214/18-ba1103.

Cooke, R. M. (1991). *Experts in uncertainty: opinion and subjective probability in science*. Environmental ethics and science policy series. New York: Oxford University Press. https://doi.org/10.1093/oso/9780195064650.001.0001.

da Silva, E. d. S., Kúsmierczyk, T., Hartmann, M., and Klami, A. (2023). Prior Specification for Bayesian Matrix Factorization via Prior Predictive Matching. *Journal of Machine Learning Research*, *24*, 1–51, https://www.jmlr.org/papers/volume24/21-0623/21-0623.pdf.

De Finetti, B. (1974). *Theory of probability: a critical introductory treatment*. Wiley series in probability and mathematical statistics. London, New York: Wiley. https://doi.org/10.1002/9781119286387.

Denham, R. and Mengersen, K. (2007). Geographically assisted elicitation of expert opinion for regression models. *Bayesian Analysis*, *2*(1), 99–135, https://doi.org/10.1214/07-BA205.

Depaoli, S., Yang, Y., and Felt, J. (2017). Using Bayesian Statistics to Model Uncertainty in Mixture Models: A Sensitivity Analysis of Priors. *Structural Equation Modeling: A Multidisciplinary Journal*, *24*(2), 198–215, https://doi.org/10.1080/10705511.2016.1250640.

Dillon, J. V., et al. (2017). Tensorflow distributions [Computer software]. *arXiv*, https://doi.org/10.48550/arXiv.1711.10604.

Dinh, L., Sohl-Dickstein, J., and Bengio, S. (2016). Density estimation using Real NVP. *arXiv*, https://doi.org/10.48550/arXiv.1605.08803.

European Food Safety Authority (2014). Guidance on Expert Knowledge Elicitation in Food and Feed Safety Risk Assessment. *EFSA Journal*, *12*(6), https://doi.org/10.2903/j.efsa.2014.3734.

Falconer, J. R., Frank, E., Polaschek, D. L. L., and Joshi, C. (2022). Methods for Eliciting Informative Prior Distributions: A Critical Review. *Decision Analysis*, *19*(3), 189–204, https://doi.org/10.1287/deca.2022.0451.

Fazio, L., Scholz, M., Aguilar, J. E., and Bürkner, P.-C. (2024). Primed Priors for Simulation-Based Validation of Bayesian Models. *arXiv*, https://doi.org/10.48550/arXiv.2408.06504.

Frazier, P. I. (2018). Bayesian optimization. In *Recent advances in optimization and modeling of contemporary problems* (pp. 255–278). Informs. https://doi.org/10.1287/educ.2018.0188.

Garthwaite, P. H., Kadane, J. B., and O'Hagan, A. (2005). Statistical Methods for Eliciting Probability Distributions. *Journal of the American Statistical Association*, *100*(470), 680–701, https://doi.org/10.1198/016214505000000105.

Gelman, A. (2002). Prior distribution. In *Encyclopedia of Environmetrics*, volume 3 (pp. 1634–1637). Chichester: John Wiley and Sons, Ltd. https://doi.org/10.1002/9780470057339.vap039.

Gelman, A. (2014). *Bayesian data analysis*. Chapman and Hall/CRC texts in statistical science. Boca Raton: CRC Press, third edition. https://doi.org/10.1201/b16018.

Gelman, A. and Hennig, C. (2017). Beyond Subjective and Objective in Statistics. *Journal of the Royal Statistical Society Series A: Statistics in Society*, *180*(4), 967–1033, https://doi.org/10.1111/rssa.12276.

Gelman, A., Simpson, D., and Betancourt, M. (2017). The Prior Can Often Only Be Understood in the Context of the Likelihood. *Entropy*, *19*(10), 555, https://doi.org/10.3390/e19100555.

Gelman, A., et al. (2020). Bayesian Workflow. *arXiv*, https://doi.org/10.48550/arXiv.2011.01808.

Gosling, J. P. (2018). SHELF: The Sheffield Elicitation Framework. In L. C. Dias, A. Morton, and J. Quigley (Eds.), *Elicitation: The Science and Art of Structuring Judgement* (pp. 61–93). Cham: Springer International Publishing. https://doi.org/10.1007/978-3-319-65052-4_4.

Gosling, J. P., O'Hagan, A., and Oakley, J. E. (2007). Nonparametric elicitation for heavy-tailed prior distributions. *Bayesian Analysis*, *2*(4), https://doi.org/10.1214/07-BA228.

Gretton, A., Borgwardt, K. M., Rasch, M. J., Schölkopf, B., and Smola, A. (2012). A kernel two-sample test. *The Journal of Machine Learning Research*, *13*(1), 723–773, https://doi.org/10.5555/2188385.2188410.

Gribok, A. V., Urmanov, A. M., Wesley Hines, J., and Uhrig, R. E. (2004). Backward specification of prior in bayesian inference as an inverse problem. *Inverse Problems in Science and Engineering*, *12*(3), 263–278, https://doi.org/10.1080/10682760310001598689.

Gustafson, P. (2005). On Model Expansion, Model Contraction, Identifiability and Prior Information: Two Illustrative Scenarios Involving Mismeasured Variables. *Statistical Science*, *20*(2), https://doi.org/10.1214/088342305000000098.

Gustafson, P. (2010). Bayesian Inference for Partially Identified Models. *The International Journal of Biostatistics*, *6*(2), https://doi.org/10.2202/1557-4679.1206.

Hartmann, M. and Agiashvili, G. (2020). Flexible Prior Elicitation via the Prior Predictive Distribution. In *Proceedings of the 36th Conference on Uncertainty in Artificial Intelligence*, volume 124 (pp. 1129–1138).: PMLR. https://proceedings.mlr.press/v124/hartmann20a.html.

Hasselbring, W. (2018). Software Architecture: Past, Present, Future. In V. Gruhn and R. Striemer (Eds.), *The Essence of Software Engineering* (pp. 169–184). Cham: Springer International Publishing. https://doi.org/10.1007/978-3-319-73897-0_10.

Hasselbring, W., Carr, L., Hettrick, S., Packer, H., and Tiropanis, T. (2020). From FAIR research data toward FAIR and open research software. *it - Information Technology*, *62*(1), 39–47, https://doi.org/10.1515/itit-2019-0040.

Hem, I., Fuglstad, G.-A., and Riebler, A. (2024). makemyprior: Intuitive Construction of Joint Priors for Variance Parameters in R [Computer software]. *Journal of Statistical Software*, *110*(3), https://doi.org/10.18637/jss.v110.i03.

Huang, D., Stein, N., Rubin, D. B., and Kou, S. C. (2020). Catalytic prior distributions with application to generalized linear models. *Proceedings of the National Academy of Sciences*, *117*(22), 12004–12010, https://doi.org/10.1073/pnas.1920913117.

Ibrahim, J. G. (1997). On Properties of Predictive Priors in Linear Models. *The American Statistician*, *51*(4), 333–337, https://doi.org/10.1080/00031305.1997.10474408.

Ibrahim, J. G., Chen, M., Gwon, Y., and Chen, F. (2015). The power prior: theory and applications. *Statistics in Medicine*, *34*(28), 3724–3749, https://doi.org/10.1002/sim.6728.

Jang, E., Gu, S., and Poole, B. (2016). Categorical reparameterization with gumbel-softmax. *arXiv*, https://doi.org/10.48550/arXiv.1611.01144.

Jarociński, M. and Marcet, A. (2019). Priors about observables in vector autoregressions. *Journal of Econometrics*, *209*(2), 238–255, https://doi.org/10.1016/j.jeconom.2018.12.023.

Jaynes, E. T. (2003). *Probability theory: the logic of science*, volume 1. Cambridge, New York: Cambridge University Press. https://doi.org/10.1017/CBO9780511790423.

Jeffreys, H. (1961). *Theory of Probability*. Oxford: Oxford, third edition.

Johanson, A. and Hasselbring, W. (2018). Software Engineering for Computational Science: Past, Present, Future. *Computing in Science and Engineering*, *20*(2), 90–109, https://doi.org/10.1109/MCSE.2018.021651343.

Johnson, S. R., Tomlinson, G. A., Hawker, G. A., Granton, J. T., and Feldman, B. M. (2010a). Methods to elicit beliefs for Bayesian priors: a systematic review. *Journal of Clinical Epidemiology*, *63*(4), 355–369, https://doi.org/10.1016/j.jclinepi.2009.06.003.

Johnson, S. R., Tomlinson, G. A., Hawker, G. A., Granton, J. T., Grosbein, H. A., and Feldman, B. M. (2010b). A valid and reliable belief elicitation method for Bayesian priors. *Journal of Clinical Epidemiology*, *63*(4), 370–383, https://doi.org/10.1016/j.jclinepi.2009.08.005.

Joo, W., Kim, D., Shin, S., and Moon, I.-C. (2025). Generalized Gumbel-Softmax gradient estimator for generic discrete random variables. *Pattern Recognition Letters*, *196*, 148–155, https://doi.org/10.1016/j.patrec.2025.05.024.

Kadane, J. and Wolfson, L. J. (1998). Experiences in elicitation. *Journal of the Royal Statistical Society: Series D (The Statistician)*, *47*(1), 3–19, https://doi.org/10.1111/1467-9884.00113.

Kadane, J. B., Dickey, J. M., Winkler, R. L., Smith, W. S., and Peters, S. C. (1980). Interactive Elicitation of Opinion for a Normal Linear Model. *Journal of the American Statistical Association*, *75*(372), 845–854, https://doi.org/10.1080/01621459.1980.10477562.

Kass, R. E. and Wasserman, L. (1996). Formal Rules for Selecting Prior Distributions: A Review and Annotated Bibliography. *Journal of the American Statistical Association*, *91*(435), 1343–1370, https://doi.org/10.2307/2291752.

Kendall, A., Gal, Y., and Cipolla, R. (2018). Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. https://openaccess.thecvf.com/content_cvpr_2018/html/Kendall_Multi-Task_Learning_Using_CVPR_2018_paper.html.

Kingma, D. P., Salimans, T., and Welling, M. (2015). Variational dropout and the local reparameterization trick. *Advances in Neural Information Processing Systems*, *28*, https://doi.org/10.5555/2969442.2969527.

Knol, A. B., Slottje, P., Van Der Sluijs, J. P., and Lebret, E. (2010). The use of expert elicitation in environmental health impact assessment: a seven step procedure. *Environmental Health*, *9*(1), 19, https://doi.org/10.1186/1476-069X-9-19.

Kobyzev, I., Prince, S. J., and Brubaker, M. A. (2020). Normalizing flows: An introduction and review of current methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *43*(11), 3964–3979, https://doi.org/10.1109/TPAMI.2020.2992934.

Kuhnert, P. M., Martin, T. G., and Griffiths, S. P. (2010). A guide to eliciting and using expert knowledge in Bayesian ecological models. *Ecology Letters*, *13*(7), 900–914, https://doi.org/10.1111/j.1461-0248.2010.01477.x.

Lamprecht, A.-L., et al. (2020). Towards FAIR principles for research software. *Data Science*, *3*(1), 37–59, https://doi.org/10.3233/DS-190026.

Lesaffre, E., Baio, G., and Boulanger, B., Eds. (2020). *Bayesian methods in pharmaceutical research*. Chapman and Hall/CRC Biostatistics Series. Boca Raton: CRC Press. https://doi.org/10.1201/9781315180212.

LightTwist Software (2007). EXCALIBUR [Computer software]. http://www.lighttwist.net/wp/excalibur/.

Lipman, Y., Chen, R. T., Ben-Hamu, H., Nickel, M., and Le, M. (2022). Flow matching for generative modeling. *arXiv*, https://doi.org/10.48550/arXiv.2210.02747.

Liu, S., Johns, E., and Davison, A. J. (2019). End-to-end multi-task learning with attention. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. https://openaccess.thecvf.com/content_CVPR_2019/html/Liu_End-To-End_Multi-Task_Learning_With_Attention_CVPR_2019_paper.html.

Luo, Y., Stephens, D. A., Graham, D. J., and McCoy, E. J. (2023). Assessing the validity of Bayesian inference using loss functions. *arXiv*, https://doi.org/10.48550/arXiv.2103.04086.

Manderson, A. (2023). *Bayesian methodology for integrating multiple data sources and specifying priors from predictive information*. PhD thesis. https://www.repository.cam.ac.uk/handle/1810/350508.

Manderson, A. A. and Goudie, R. J. B. (2024). Translating predictive distributions into informative priors. *arXiv*, https://doi.org/10.48550/arXiv.2303.08528.

Mikkola, P., Acerbi, L., and Klami, A. (2024a). Preferential Normalizing Flows. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*. https://openreview.net/forum?id=sRSjr9SDKR.

Mikkola, P., et al. (2024b). Prior Knowledge Elicitation: The Past, Present, and Future. *Bayesian Analysis*, *19*(4), https://doi.org/10.1214/23-ba1381.

Moala, F. A. and O'Hagan, A. (2010). Elicitation of multivariate prior distributions: A nonparametric Bayesian approach. *Journal of Statistical Planning and Inference*, *140*(7), 1635–1655, https://doi.org/10.1016/j.jspi.2010.01.004.

Morris, D. E., Oakley, J. E., and Crowe, J. A. (2014). A web-based tool for eliciting probability distributions from experts [Computer software]. *Environmental Modelling and Software*, *52*, 1–4, https://doi.org/10.1016/j.envsoft.2013.10.010.

Oakley, J. E. (2025). SHELF: Tools to Support the Sheffield Elicitation Framework [Computer software]. https://github.com/OakleyJ/SHELF.

Oakley, J. E., Daneshkhah, A., and O'Hagan, A. (2010). *Nonparametric Prior Elicitation using the Roulette Method*. Technical report, School of Mathematics and Statistics, University of Sheffeld, UK.

Oakley, J. E. and O'Hagan, A. (2007). Uncertainty in prior elicitations: a nonparametric approach. *Biometrika*, *94*(2), 427–441, https://doi.org/10.1093/biomet/asm031.

O'Hagan, A. (1988). *Probability: Methods and measurement*. Dordrecht: Springer Netherlands. https://doi.org/10.1007/978-94-009-1211-3.

O'Hagan, A., et al. (2006). *Uncertain Judgements: Eliciting Experts' Probabilities*. Wiley, first edition. https://doi.org/10.1002/0470033312.

O'Hagan, A. and Oakley, J. E. (2004). Probability is perfect, but we can't elicit it perfectly. *Reliability Engineering and System Safety*, *85*(1), 239–248, https://doi.org/10.1016/j.ress.2004.03.014.

O'Hagan, T. (2005). Elicitation. *Significance*, *2*(2), 84–86, https://doi.org/10.1111/j.1740-9713.2005.00100.x.

O'Hagan, A. (2012). Probabilistic uncertainty specification: Overview, elaboration techniques and their application to a mechanistic model of carbon flux. *Environmental Modelling and Software*, *36*, 35–48, https://doi.org/10.1016/j.envsoft.2011.03.003.

O'Hagan, A. (2019). Expert Knowledge Elicitation: Subjective but Scientific. *The American Statistician*, *73*(sup1), 69–81, https://doi.org/10.1080/00031305.2018.1518265.

Papamakarios, G., Nalisnick, E., Rezende, D. J., Mohamed, S., and Lakshminarayanan, B. (2021). Normalizing Flows for Probabilistic Modeling and Inference. *Journal of Machine Learning Research*, *22*(57), 1–64, http://jmlr.org/papers/v22/19-1028.html.

Perepolkin, D., Goodrich, B., and Sahlin, U. (2024). Hybrid elicitation and quantile-parametrized likelihood. *Statistics and Computing*, *34*(1), 11, https://doi.org/10.1007/s11222-023-10325-0.

Piironen, J. and Vehtari, A. (2017). Sparsity information and regularization in the horseshoe and other shrinkage priors. *Electronic Journal of Statistics*, *11*(2), https://doi.org/10.1214/17-EJS1337SI.

Rietbergen, C., Klugkist, I., Janssen, K. J., Moons, K. G., and Hoijtink, H. J. (2011). Incorporation of historical data in the analysis of randomized therapeutic trials. *Contemporary Clinical Trials*, *32*(6), 848–855, https://doi.org/10.1016/j.cct.2011.06.002.

Robert, C. P. (2007). *The Bayesian choice: from decision-theoretic foundations to computational implementation*. Springer texts in statistics. New York: Springer, second edition. https://doi.org/10.1007/0-387-71599-1.

Rougier, J., Rougier, J., Hill, L. J., and Sparks, R. S. J. (2013). *Risk and uncertainty assessment for natural hazards*. New York: Cambridge University Press. https://doi.org/10.1017/CBO9781139047562.

Rowe, G. and Wright, G. (1999). The Delphi technique as a forecasting tool: issues and analysis. *International Journal of Forecasting*, *15*(4), 353–375, https://doi.org/10.1016/S0169-2070(99)00018-7.

Ruder, S. (2016). An overview of gradient descent optimization algorithms. *arXiv*, https://doi.org/10.48550/ARXIV.1609.04747.

Sahlin, U., et al. (2024). Online training courses on Expert Knowledge Elicitation (EKE). *EFSA Supporting Publications*, *21*(3), 8673E, https://doi.org/10.2903/sp.efsa.2024.EN-8673.

Salvatier, J., Wiecki, T. V., and Fonnesbeck, C. (2016). Probabilistic programming in Python using PyMC3. *PeerJ Computer Science*, *2*, e55, https://doi.org/10.7717/peerj-cs.55.

Seaman, J. W., Seaman III, J. W., and Stamey, J. D. (2012). Hidden Dangers of Specifying Noninformative Priors. *The American Statistician*, *66*(2), 77–84, https://doi.org/10.1080/00031305.2012.695938.

Simpson, D., Rue, H., Riebler, A., Martins, T. G., and Sørbye, S. H. (2017). Penalising Model Component Complexity: A Principled, Practical Approach to Constructing Priors. *Statistical Science*, *32*(1), https://doi.org/10.1214/16-STS576.

Smith, A. M., Katz, D. S., Niemeyer, K. E., and FORCE11 Software Citation Working Group (2016). Software citation principles. *PeerJ Computer Science*, *2*, e86, https://doi.org/10.7717/peerj-cs.86.

Song, M., et al. (2022). Dynamic Restrained Uncertainty Weighting Loss for Multitask Learning of Vocal Expression. *arXiv*, https://doi.org/10.48550/arXiv.2206.11049.

Stan Development Team (2025). RStan: the R interface to Stan [Computer software]. https://mc-stan.org.

Stefan, A. M., Evans, N. J., and Wagenmakers, E.-J. (2022). Practical challenges and methodological flexibility in prior elicitation. *Psychological Methods*, *27*(2), 177–197, https://doi.org/10.1037/met0000354.

Tversky, A. and Kahneman, D. (1974). Judgment under Uncertainty: Heuristics and Biases: Biases in judgments reveal some heuristics of thinking under uncertainty. *Science*, *185*(4157), 1124–1131, https://doi.org/10.1126/science.185.4157.1124.

Van De Schoot, R., et al. (2021). Bayesian statistics and modelling. *Nature Reviews Methods Primers*, *1*(3), https://doi.org/10.1038/s43586-020-00003-0.

Van De Schoot, R., Sijbrandij, M., Depaoli, S., Winter, S. D., Olff, M., and Van Loey, N. E. (2018). Bayesian PTSD-Trajectory Analysis with Informed Priors Based on a Systematic Literature Search and Expert Elicitation. *Multivariate Behavioral Research*, *53*(2), 267–291, https://doi.org/10.1080/00273171.2017.1412293.

Van Dongen, S. (2006). Prior specification in Bayesian statistics: Three cautionary tales. *Journal of Theoretical Biology*, *242*(1), 90–100, https://doi.org/10.1016/j.jtbi.2006.02.002.

Wei, Q., Wang, H., Cao, Z., Wang, S., Allmendinger, R., and Álvarez, M. A. (2025). Gradient-based Sample Selection for Faster Bayesian Optimization. *arXiv*, https://doi.org/10.48550/ARXIV.2504.07742.

Williams, C. J., Wilson, K. J., and Wilson, N. (2021). A Comparison of Prior Elicitation Aggregation Using the Classical Method and SHELF. *Journal of the Royal Statistical Society Series A: Statistics in Society*, *184*(3), 920–940, https://doi.org/10.1111/rssa.12691.

Winkler, R. L. (1967). The Assessment of Prior Distributions in Bayesian Analysis. *Journal of the American Statistical Association*, *62*(319), 776–800, https://doi.org/10.1080/01621459.1967.10500894.

Winkler, R. L. (1968). The Consensus of Subjective Probability Distributions. *Management Science*, *15*(2), https://doi.org/10.1287/mnsc.15.2.B61.

Yu, T., Kumar, S., Gupta, A., Levine, S., Hausman, K., and Finn, C. (2020). Gradient surgery for multi-task learning. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin (Eds.), *Advances in Neural Information Processing Systems*, volume 33 (pp.

5824–5836).: Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2020/file/3fe78a8acf5fda99de95303940a2420c-Paper.pdf.

Zhang, Y. D., Naughton, B. P., Bondell, H. D., and Reich, B. J. (2022). Bayesian regression using a prior on the model fit: The R2-D2 shrinkage prior. *Journal of the American Statistical Association*, *117*(538), 862–874, https://doi.org/10.1080/01621459.2020.1825449.

APPENDIX

scientific reports

Check for updates

OPEN

Simulation-based prior knowledge elicitation for parametric Bayesian models

Florence Bockting[1]✉, Stefan T. Radev[2] & Paul-Christian Bürkner[1]

A central characteristic of Bayesian statistics is the ability to consistently incorporate prior knowledge into various modeling processes. In this paper, we focus on translating domain expert knowledge into corresponding prior distributions over model parameters, a process known as prior elicitation. Expert knowledge can manifest itself in diverse formats, including information about raw data, summary statistics, or model parameters. A major challenge for existing elicitation methods is how to effectively utilize all of these different formats in order to formulate prior distributions that align with the expert's expectations, regardless of the model structure. To address these challenges, we develop a simulation-based elicitation method that can learn the hyperparameters of potentially any parametric prior distribution from a wide spectrum of expert knowledge using stochastic gradient descent. We validate the effectiveness and robustness of our elicitation method in four representative simulation studies covering linear models, generalized linear models, and hierarchical models. Our results support the claim that our method is largely independent of the underlying model structure and adaptable to various elicitation techniques, including quantile-based, moment-based, and histogram-based methods.

The essence of Bayesian statistics lies in the ability to consistently incorporate prior knowledge into the modeling process[1,2]. The specification of sensible prior distributions over the parameters of Bayesian models can have multiple advantages including improved convergence, sampling efficiency, parameter recoverability, and predictive performance[3–6].

Despite these apparent advantages, it is often unclear a priori what constitutes a "sensible" prior[7]. In this paper, we focus on the elicitation and translation of expert knowledge into prior distributions, also known as *prior elicitation*[5]. Against this background, *a sensible prior is one that accurately reflects domain knowledge as elicited from an expert or a group of experts*. However, meeting this criterion presents its own set of challenges: Model parameters for which priors are needed might lack intuitive meaning for the domain expert[8] and the relationship between priors and the data may not be apparent from the model, especially for complex models[9]. Moreover, constructing priors for every single model parameter in models with a large number of parameters might be inefficient or even infeasible.

To address these challenges, several tools for prior elicitation have been developed in the past[5,6,10–16]. Despite the widespread application of Bayesian statistics nowadays, the field of prior elicitation still lags behind in terms of its routine implementation by practitioners. One contributing factor is that many existing methods primarily aim to elicit information about the model parameters directly. This approach makes these methods inherently model-specific, limits their widespread applicability, and poses a challenge for experts in terms of interpretability[9,17,18].

In recent years, there has been an increasing focus on the development of model-agnostic approaches that center around the prior predictive distribution[6]. These methods allow for the integration of expert knowledge regarding observed data patterns (i.e., elicitation in the observable space). In contrast to interpreting model parameters, domain experts can usually effectively interpret the scale and magnitude of observable quantities[5,9,16,19,20]. Despite these recent developments, the general applicability as well as the actual application of elicitation methods remain limited[5]. This lack of popularity persists, at least in part, because existing methods are still relatively complex, do not easily generalize to different types of expert information, or necessitate substantial tuning or other manual adjustments. In light of the preceding considerations, we introduce an elicitation method that seeks to overcome these challenges. Specifically, this work makes a contribution to prior elicitation research by proposing a method that satisfies the following criteria:

[1]Department of Statistics, TU Dortmund University, Dortmund, Germany. [2]Cognitive Science Department, Rensselaer Polytechnic Institute, Troy, NY, USA. ✉email: florence.bockting@tu-dortmund.de

1. *Model independence* Our method is agnostic to the specific probabilistic model, as long as sampling from it is feasible and stochastic gradients can be computed.
2. *Effective utilization of expert knowledge* By incorporating diverse expert information on model parameters, observed data patterns, or other relevant statistics, our method maximizes the utility of expert knowledge.
3. *Flexibility in elicitation techniques* Our method can adapt to different elicitation techniques, ensuring that individual expert preferences are considered.
4. *Modular design* Due to its modular structure our method allows easy adaptation, improvement, or replacement of specific components, both during method development and application.

Related work

The process of prior elicitation involves the extraction of expert knowledge and its translation into corresponding prior distributions for the parameters in probabilistic models[5,10,11,14]. Knowledge extraction can incorporate asking an expert directly about the probability distribution of the model parameters or indirectly about other quantities that may be easier for the expert to understand[11,21,22]. These quantities include observable data patterns (i.e., variables in the data space such as expected mean responses) as well as familiar statistics derived from the predictive distribution of the outcome variable (e.g., the percentage of variance explained).

As interpretability is an essential requirement for elicited quantities, it has been argued that asking about parameters is only meaningful if they can be interpreted in terms of a limiting average of observables[5,23]. That said, experts may also have knowledge about parameter values through prior studies, meta-analyses, and similar sources, which do not necessarily require easy interpretability. A thorough discussion of the interpretability of various elicited quantities is beyond the scope of this paper but is discussed in detail elsewhere[11,14,21,22].

Based on their comprehensive review, Mikkola et al.[5] recently advocate that an elicitation method should include both a model's parameter and observable space, exhibit model-agnostic characteristics, and prioritize sample efficiency to minimize the human effort involved. Taking these desiderata into consideration, our method builds upon recent advancements in prior elicitation, specifically on the works of Hartmann et al.[16], da Silva et al.[9], and Manderson & Goudie[6]. All three methods are model-agnostic approaches that focus (mainly) on eliciting expert knowledge in the observable space but differ in their specification of target quantities, discrepancy measures, and the specific optimization procedure.

Manderson & Goudie[6] use multi-objective Bayesian optimization, while our approach employs stochastic gradient-based optimization in line with the methods proposed by da Silva et al.[9] and Hartmann et al.[16]. All three methods, including ours, support quantile-based elicitation. However, our method goes a step further by also allowing histogram or moment-based elicitation. While all of the considered methods allow for eliciting expert information about observable variables, da Silva et al.[9] additionally supports querying experts with respect to the parameter space. Our method follows this approach and enables the elicitation of expert knowledge about model parameters, observable quantities, and quantities derived from observable quantities (e.g. percentage of variance explained). As such, our method allows for the elicitation of model parameters and observable quantities, both directly and indirectly, thus extending beyond elicitation in the parameter and observable space. Finally, an essential feature of our method is the use of simulations to obtain prior hyperparameter inference, which classifies it as a variant of *simulation-based inference* (SBI)[24].

Simulation studies

In this section, we present four simulation studies demonstrating the performance of our elicitation method. We showcase our method using a normal linear regression model in simulation study 1 (Section "Simulation study 1: normal linear regression"), a binomial regression with logit link in simulation study 2 (Section "Simulation study 2: GLMs—binomial model"), a Poisson regression with log link in simulation study 3 (Section "Simulation study 3: GLMs—Poisson model"), and a multilevel model with normal likelihood in simulation study 4 (Section "Simulation study 4: hierarchical model"). All code and material can be found on GitHub https://github.com/florence-bockting/PriorLearning, our project website https://florence-bockting.github.io/PriorLearning/index.html, and our simulation results in https://osf.io/rxgv2.

General setup

Learning algorithm In each simulation study, we utilize mini-batch stochastic gradient descent to learn all model hyperparameters. Each optimization process is characterized by a set of *algorithm parameters* including the batch size (B) , the number of epochs (E) , the number of samples from the prior distributions (S) , and the initial learning rate (ϕ^0) of the cosine decay schedule with restarts used with the Adam optimizer. The specific settings of the optimization process are fully described in the respective sections. All simulation studies were implemented in Python, utilizing the *TensorFlow* library[25], and optimization was executed on the Linux HPC cluster at Technical University Dortmund (LiDO3) on high-end GPUs (NVIDIA Tesla K40). We note that our methods can also be easily run on the CPUs of common consumer laptops, where they rarely required more than 30 to 60 minutes (often less) until convergence; at least for the models investigated in our simulation studies. More details for each simulation study (e.g. computing time) can be found in the appendix.

Method verification An essential property of any method is its validity. In the following simulation studies, we aim to demonstrate the validity of our proposed method, which we define as the method's ability to recover a hypothetical ground truth. To achieve this, we use the following approach: First, we define a unique hyperparameter vector λ^* that represents the hypothetical ground truth. Conditional on λ^*, observations are simulated from the generative model, and predefined target quantities, along with the corresponding elicitation techniques, are computed. The resulting elicited statistics encode the ground truth. Consequently, a *valid* method should be able to learn λ^* when trained on these elicited statistics.

However, this approach has a caveat: learning unique prior distributions from elicited statistics becomes increasingly challenging as model complexity grows[21]. Since the outcome variable generally has lower dimensionality than the model parameters for which we aim to learn prior distributions[6], a specific set of elicited statistics may correspond to many equally valid priors and thus varying λ'. This makes it difficult to determine whether the method can recover λ^*. As a consequence, for each simulation study, we constructed a set of elicited statistics that, on the one hand, conveys sufficient information to approximately ensure model identification and, on the other hand, is as small as possible.

Selection of target quantities and elicited statistics To demonstrate the flexibility of our method in selecting target quantities and elicitation techniques, we utilized the following target quantities in the subsequent simulation studies: model parameters, prior predictions of the outcome variable, and statistics derived from these prior predictions (e.g., R^2). Regarding elicitation techniques, we employed quantile-based, histogram-based, and moment-based elicitation. For quantile-based elicitation, the quartiles Q_p with $p = (0.25, 0.5, 0.75)$; for moment-based elicitation, the mean and standard deviation of the target quantity; and for histogram-based elicitation, a histogram comprising S observations were used. Further specifications are provided in each simulation study.

Simulation study 1: Normal linear regression

Setup The first simulation study is presented along with an example inspired by a study from Unkelbach & Rom[26]. In this study, participants encounter general knowledge statements in two consecutive phases, during the second of which they must indicate whether each statement is true or false. The main objective is to investigate the influence of two factors on the proportion of true judgments (PTJs): (1) repetition (ReP), which involves presenting some statements from the first phase again in the second phase, and (2) encoding depth (EnC), whereby participants are randomly assigned to groups that differ in the level of elaboration required when processing the statements during the first phase. We consider a 2 (ReP: *repeated, new*) × 3 (EnC: *shallow, standard, deep*) between-subject factorial design with treatment contrasts for both factors. The baseline levels are *new* for ReP and *deep* for EnC. Following Unkelbach & Rom[26], we use a linear regression model to describe the data-generating process

$$\begin{aligned} y_i &\sim \text{Normal}(\theta_i, s) \\ \theta_i &= \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_4 + \beta_5 x_5 \\ \beta_k &\sim \text{Normal}(\mu_k, \sigma_k) \quad \text{for } k = 0, \ldots, 5 \\ s &\sim \text{Gamma}(\alpha, \beta). \end{aligned} \tag{1}$$

The responses y_i for each observation $i = 1, \ldots, N$ are normally distributed with mean θ_i and standard deviation s. The expected value θ_i is modeled as a linear function of ReP and EnC. The regression coefficients β_k for $k = 0, \ldots, 5$ are assigned normal prior distributions. The standard deviation s of the normal likelihood follows a Gamma prior with concentration parameter α and rate parameter β. The goal is to learn a total of 14 hyperparameters, $\lambda = (\mu_k, \sigma_k, \alpha, \beta)$.

Elicitation procedure The following four target quantities were selected: the expected PTJ for the marginal of both factors EnC (1) and ReP (2), the expected difference in PTJ (ΔPTJ) between repeated and new statements for each EnC level (3), and the expected R^2 defined as a variance ratio of the modeled predictive means and the predictive observations including the residual variance, $R^2 = \text{var}(\theta_i)/\text{var}(y_i)$ (4). For target quantities 1-3 quantile elicitation is used and for target quantity 4 histogram elicitation. As hypothetical ground truth, we specify the following hyperparameter vector $\lambda^* = (\mu_0 = 0.12, \sigma_0 = 0.02, \mu_1 = 0.15, \sigma_1 = 0.02, \mu_2 = -0.02, \sigma_2 = 0.06, \mu_3 = -0.03, \sigma_3 = 0.06, \mu_4 = -0.02, \sigma_4 = 0.03, \mu_5 = -0.04, \sigma_5 = 0.03, \alpha = 20., \beta = 200.)$. The elicited statistics conditional on λ^* are depicted in Fig. 1. The first column depicts the histogram for R^2 and the remaining columns the results of quantile-based elicitation.

Optimization To instantiate the optimization process the hyperparameters λ are randomly initialized as follows: $\mu_k \sim \text{Normal}(0, 0.1)$, $\log \sigma_k \sim \text{Uniform}(-2, -4)$, $\log \alpha \sim \text{Normal}(3, 0.1)$, and $\log \beta \sim \text{Normal}(5, 0.1)$,

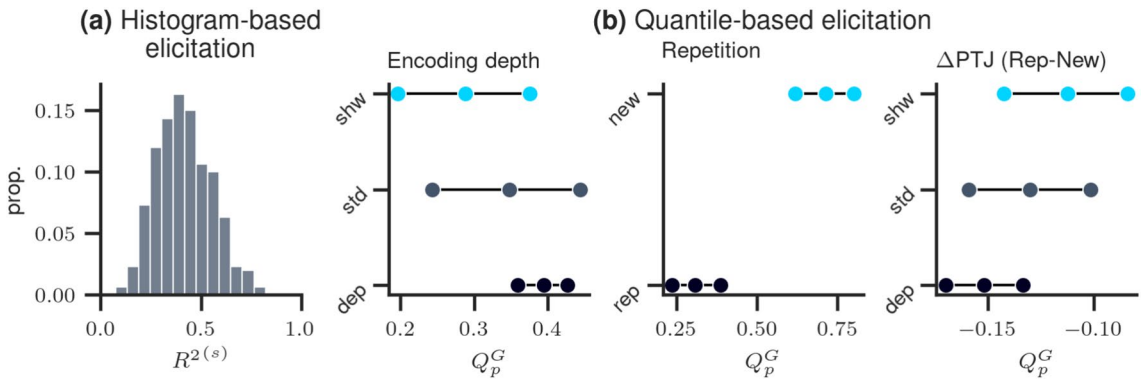


Figure 1. Elicited statistics conditional on λ^*. (**a**) elicited histogram of R^2; (**b**) three elicited quantiles for each remaining target quantity (see text for detailed information). Abbreviations: For the factor *Encoding depth*: dep-deep, std-standard, and shw-shallow and for the factor *Repetition*: rep-repeated and new.

whereby the scale, concentration, and rate parameter are initialized on the log scale. Subsequently, we simulate from the forward model and compute the corresponding model-implied target quantities, along with the elicited statistics. The discrepancy between the model-implied and true elicited statistics can then be computed and the hyperparameters updated. The learning process is considered completed once the maximum number of epochs has been reached. Details about the optimization algorithm can be found in the "Methods" section and the corresponding specification of the algorithm parameters can be found in "Appendix B.1".

To assess whether learning was successful, we first check the *convergence diagnostics* as summarized in Fig. 2. Examining the loss functions depicted in the leftmost column demonstrates the desired decreasing behavior for both the total loss as well as the individual loss components. The gradients of the hyperparameters λ are depicted in the upper, right row, indicating the expected decreasing behavior towards zero across time. Finally, convergence of hyperparameters λ during the learning process is illustrated in the lower, right row.

Results After having confirmed successful convergence, we shift our focus to the simulation results as depicted in Fig. 3. The final learned hyperparameter λ is computed as the average of the last 30 epochs. The resulting *learned* prior distributions are shown in the upper row of Fig. 3. Solid lines indicate the learned priors and dotted lines the *true* priors (according to λ^*).

The substantial overlap between these distributions indicates a successful learning process. This is further emphasized in the second row, where the error between the learned and true hyperparameter values gradually decreases towards zero.

Simulation study 2: GLMs—Binomial model

Setup In simulation study 2 we utilize a binomial response distribution with a logit-link function for the probability parameter. As accompanying example, we use the Haberman's survival dataset from the UCI machine learning repository[27]. The dataset contains cases from a study on the survival of patients who had undergone surgery for breast cancer. In the following, we use the detected number of axillary lymph nodes that contain cancer (i.e., (positive) axillary nodes) as numerical predictor X which consists in total of 31 observations ranging between 0 and 59 axillary nodes. The dependent variable y is the number of patients who died within five years out of $T = 100$ trials for each observation $i = 1, \ldots, N$. We consider a simple binomial regression model with one continuous predictor

$$\begin{aligned} y_i &\sim \text{Binomial}(T, \theta_i) \\ \text{logit}(\theta_i) &= \beta_0 + \beta_1 x_i \\ \beta_k &\sim \text{Normal}(\mu_k, \sigma_k) \quad \text{for} \quad k = 0, 1. \end{aligned} \tag{2}$$

We assume normal priors for the regression coefficients, with mean μ_k and standard deviation σ_k for $k = 0, 1$. Through the logit-link function, the probability θ_i is mapped to the scale of the linear predictor. The objective is to learn four hyperparameters $\lambda = (\mu_k, \sigma_k)$.

Elicitation procedure and optimization As target quantities we select the expected number of patients who died within five years for different numbers of axillary nodes x_i, with $i = 0, 5, 10, 15, 20, 25, 30$. For each selected design point, we consider quantile-based elicitation. The hypothetical ground truth is defined by the following hyperparameter vector $\lambda^* = (\mu_0 = -0.51, \sigma_0 = 0.06, \mu_1 = 0.26, \sigma_1 = 0.04)$. The specification of the algorithm parameters for the optimization procedure can be found in "Appendix B.2". The convergence diagnostics

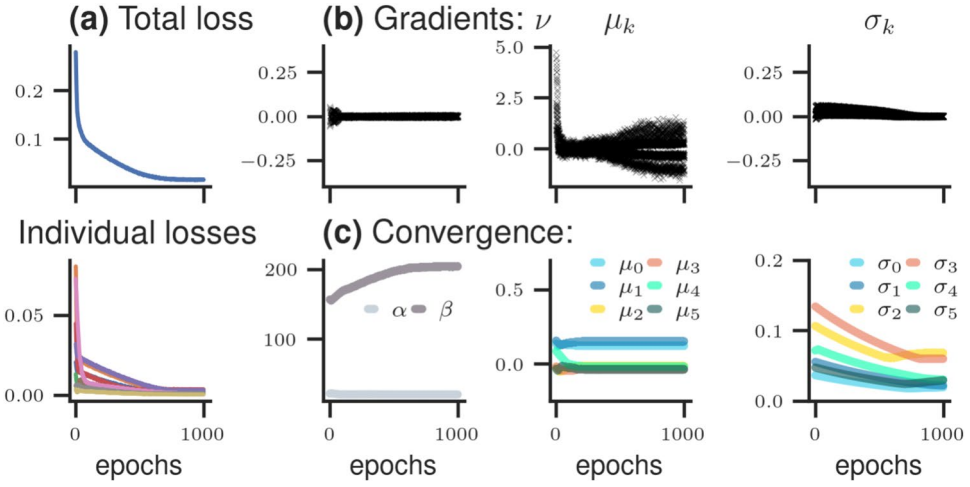


Figure 2. Convergence diagnostics for simulation study 1. (**a**) loss value across epochs, demonstrating the desired decreasing trend of all loss values (i.e., the total loss and the individual loss components); (**b**) expected decreasing trend towards zero of the gradients for each learned hyperparameter λ; (**c**) update values of each learned hyperparameter after each iteration step (epoch), stabilizing in the long run at a specific value.

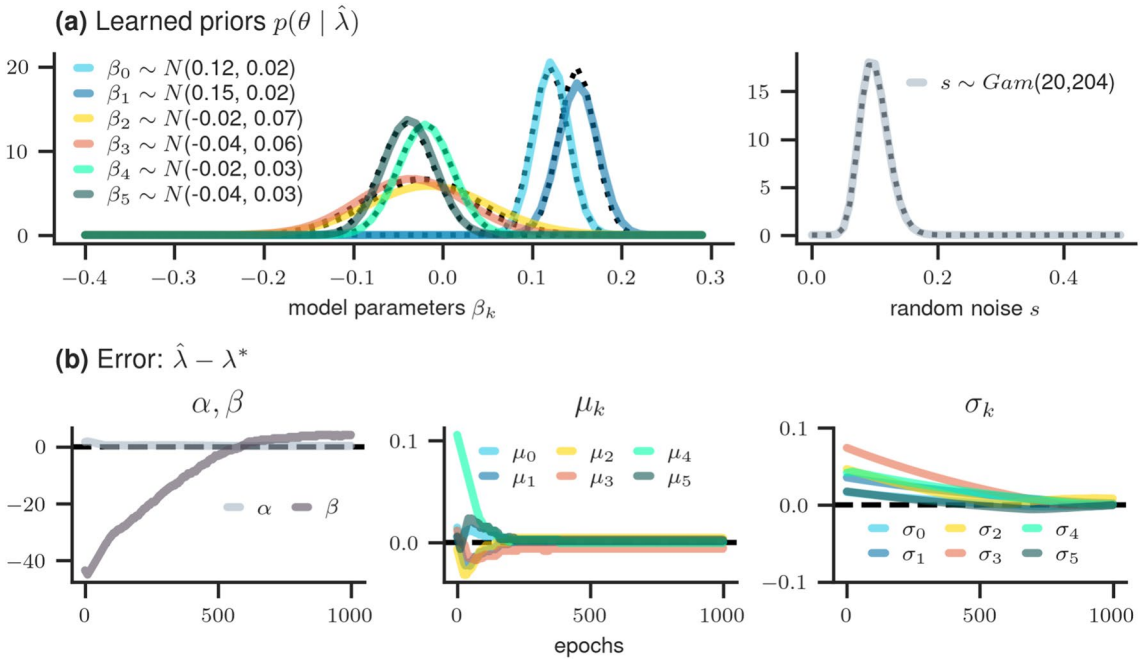


Figure 3. Results of simulation study 1. (**a**) true (dotted line) and learned (solid line) prior distributions per model parameter β_k and s; (**b**) error between learned and true hyperparameter values $(\alpha, \beta, \mu_k, \sigma_k)$ over time.

check follows the same procedure as discussed for simulation study 1, and showed successful convergence (see "Appendix B.2.1").

Results The simulation results, based on the final learned hyperparameters $\hat{\lambda}$, are presented in Fig. 4.

The upper row shows a comparison between the true and learned quantiles for each number of axillary nodes x_i, revealing an almost perfect match between both quantities. In the lower right panels, the error between the true and learned hyperparameters is depicted and indicates successful learning. Additionally, the lower right panel presents the true (dotted line) and learned (solid line) prior distributions which show a perfect match.

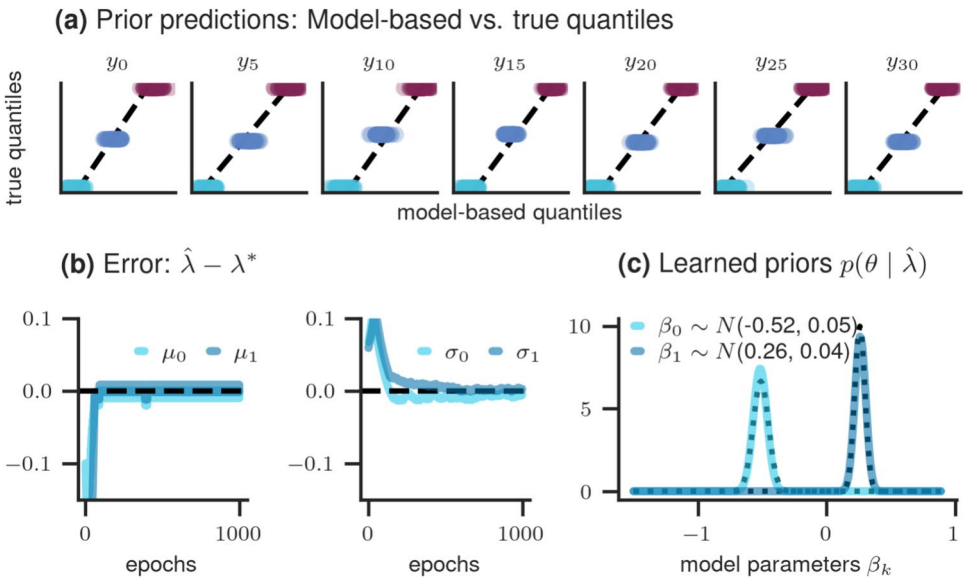


Figure 4. Results of simulation study 2: (**a**) comparison between learned and true quantiles for each selected x_i; (**b**) learning of hyperparameters across epochs, showcasing the difference between the true and learned values; (**c**) true (dotted line) and learned (solid line) prior distributions of each model parameter.

Simulation study 3: GLMs—Poisson model

Setup In simulation study 3, we expand our examination of count data likelihoods to include a Poisson distribution. For demonstration purposes, we adapt an example from Johnson et al.[28], which investigates the number of LGBTQ+ anti-discrimination laws in each US state. The distribution of these laws is assumed to follow a Poisson distribution, with the rate parameter being influenced by demographic and voting trend. The demographic trend is quantified by the percentage of a state's residents living in urban areas, ranging from 38.7% to 94.7%. Additionally, the voting trend is represented by historical voting patterns in presidential elections, categorizing each state as consistently voting for the Democratic or Republican candidate or being a Swing state. We employ a Poisson regression model including one treatment-coded categorical predictor: the *voting trend*. This predictor has three levels: Democrats, Republicans, and Swing, with Democrats serving as the reference category. Furthermore, the model incorporates one continuous predictor: the *demographic trend*, measured as a percentage. The Poisson regression model is represented as follows

$$\begin{aligned} y_i &\sim \text{Poisson}(\theta_i) \\ \log(\theta_i) &= \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 \\ \beta_k &\sim \text{Normal}(\mu_k, \sigma_k) \quad \text{for } k = 0, \ldots, 3. \end{aligned} \tag{3}$$

Here, y_i is the number of counts for observation $i = 1, \ldots, N$. The counts follow a Poisson distribution with rate θ_i and log-link function. The rate parameter is predicted by a linear combination of the two predictors demographic and voting trend. All regression coefficients are assumed to have normal prior distributions with mean μ_k and standard deviation σ_k for $k = 0, \ldots, 3$. Our goal is to learn eight hyperparameters $\lambda = (\mu_k, \sigma_k)$.

Elicitation procedure We consider two target quantities: the predictive distribution of the group means for states categorized as Democrats, Republicans, and Swing, and the expected number of LGBTQ+ anti-discrimination laws for selected US states x_i with $i = 0, 13, 14, 35, 37, 48$. Quantile-based elicitation is used for the distribution of group means and histogram elicitation for the observations per US state. Furthermore, the expected maximum number of LGBTQ+ anti-discrimination laws in one US state is required. This value is used as upper truncation threshold, t^u, of the Poisson distribution which is needed for applying the Softmax-Gumbel Trick that allows for computing gradients for discrete random variables (see Section "Gradient-based optimization" for details). For the current example, we assume $t^u = 80$ and define the following hyperparameter vector λ^* representing the ground truth: $\lambda^* = (\mu_0 = 2.91, \sigma_0 = 0.07, \mu_1 = 0.23, \sigma_1 = 0.05, \mu_2 = -1.51, \sigma_2 = 0.135, \mu_3 = -0.61, \sigma_3 = 0.105)$. The specification of algorithm parameters for the optimization procedure as well as a figure summarizing the convergence diagnostics can be found in "Appendix B.3".

Results The learned hyperparameters' results are presented in Fig. 5. In the upper panels, a comparison between model-based and true elicited statistics is presented and shows a high level of agreement: quantile-based elicitation for the voting groups is depicted in the first three panels and histogram elicitation for single states in the remaining upper panels.

The model-based histograms are depicted in blue and the ground truth in red. The lower left panels demonstrate that the error between learned and true hyperparameter values converges towards zero over time. Finally, the learned prior distributions are depicted in the lower right panel, with solid lines representing the learned and dotted lines the true priors.

Simulation study 4: Hierarchical model

Setup In this concluding simulation study, we investigate the performance of our elicitation method when applied to a hierarchical model. This specific model class poses a distinct challenge for analysts and domain experts alike due to the inherent complexity of the model and the non-intuitive nature of varying effects (i.e., varying intercepts and slopes). Our method allows for learning prior distributions within a hierarchical framework, while relying on expert knowledge that is articulated in terms of interpretable target quantities.

The accompanying example draws inspiration from the *sleepstudy* dataset[29]. This dataset contains information about the average reaction time (RT) in milliseconds for N individuals who undergo sleep deprivation for nine consecutive nights. In order to construct a model for this data, we consider a hierarchical model with days serving as a continuous predictor x,

$$\begin{aligned} y_{ij} &= \text{Normal}(\theta_{ij}, s) \\ \theta_{ij} &= \beta_0 + u_{0,j} + (\beta_1 + u_{1,j}) x_{ij} \\ (u_{0,j}, u_{1,j}) &\sim \text{MvNormal}(\mathbf{0}, \Sigma_u) \\ \Sigma_u &= \begin{pmatrix} \tau_0^2 & \rho_{01}\tau_0\tau_1 \\ \rho_{01}\tau_0\tau_1 & \tau_1^2 \end{pmatrix} \\ \beta_k &\sim \text{Normal}(\mu_k, \sigma_k) \qquad \text{for } k = 0, 1 \\ \tau_k &\sim \text{TruncatedNormal}(0, \omega_k) \qquad \text{for } k = 0, 1 \\ \rho_{01} &\sim \text{LKJ}(\alpha_{\text{LKJ}}) \\ s &\sim \text{Gamma}(\alpha, \beta). \end{aligned} \tag{4}$$

Here y_{ij} represents the average RT for the j^{th} participant at the i^{th} day with $j = 1, \ldots, 200$ and $i = 0, \ldots, 9$. The RT data is assumed to follow a normal distribution with local mean θ_{ij} and within-person standard deviation s. Here, θ_{ij} is predicted by a linear combination of the continuous predictor x with overall slope β_1 and intercept β_0. Given the potential variation in both baseline and change in RT across participants, the model incorporates

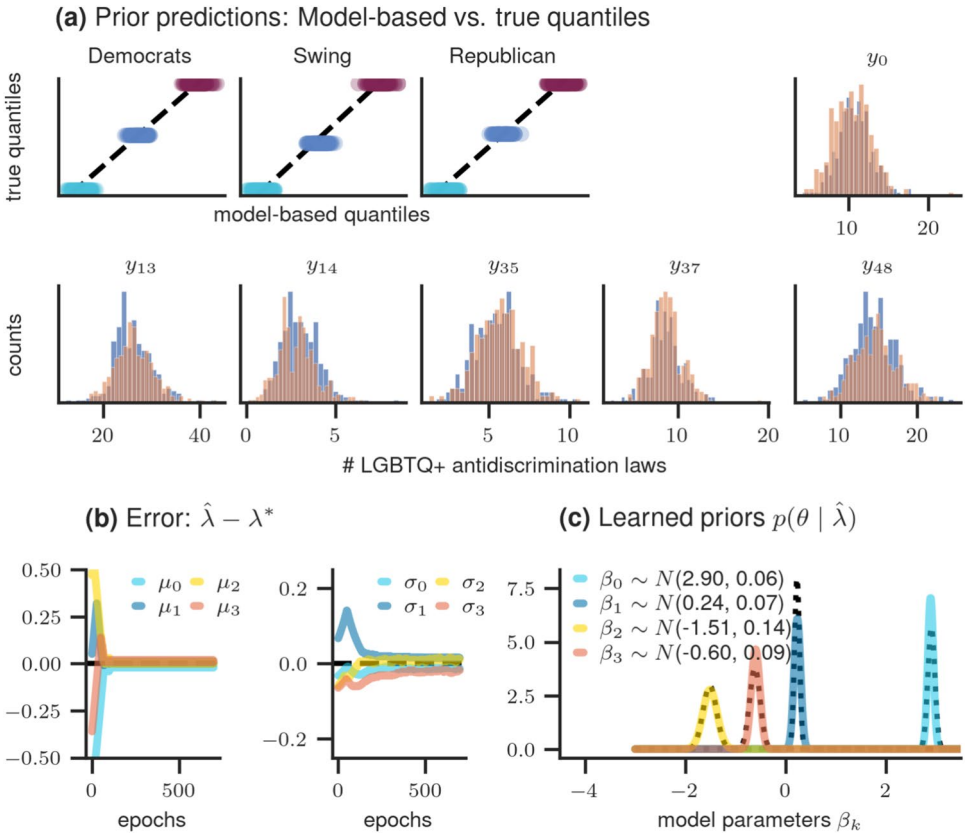


Figure 5. Results of simulation study 3: (**a**) comparison between model-based and true elicited statistics. First three panels depict quantile-based elicitation for the group means, while the remaining upper panels show histogram elicitation for each state x_i. The model-based histograms are depicted in blue and the ground truth in red. (**b**) learning of hyperparameters across epochs, showcasing the difference between the true and learned values; (**a**) true (dotted line) and learned (solid line) prior distributions of the model parameters.

varying (i.e., "random") intercepts $u_{0,j}$ and slopes $u_{1,j}$. These varying effects follow a multivariate normal distribution, centered at a mean vector of zero and with a covariance matrix Σ_u. This encodes the variability (τ_0, τ_1) and the correlation (ρ_{01}) between $u_{0,j}$ and $u_{1,j}$. For the resulting set of model parameters, the following prior distributions are assumed: A normal distribution for the overall (i.e., "fixed") effects β_k $(k = 0, 1)$ with mean μ_k and standard deviation σ_k. A truncated normal distribution centered at zero with a standard deviation of ω_k, is employed for the person-specific variation τ_k, which is constrained to be positive. The correlation parameter ρ_{01} follows a Lewandowski-Kurowicka-Joe [LKJ;[30]] distribution with scale parameter α_{LKJ}. In the subsequent context, we set α_{LKJ} to 1. Additionally, a Gamma prior distribution with concentration α and rate β is used for the within-person (error) standard deviation s. The goal is to learn eight hyperparameters $\lambda = (\mu_k, \sigma_k, \omega_k, \alpha, \beta)$.

Elicitation procedure and optimization We consider the following target quantities and elicitation techniques: quantile-based elicitation for the expected average RT for specific days x_i, where $i = 0, 2, 5, 6, 9$. Moment-based elicitation using mean and standard deviation for the within-person standard deviation s (elicitation in the parameter space), and histogram-elicitation for the expected distribution of R^2 for the initial and final day $(i = 0, 9)$. We define the expected ground truth by the following hyperparameter vector $\lambda^* = (\mu_0 = 250.40, \mu_1 = 30.26, \sigma_0 = 7.27, \sigma_1 = 4.82, \omega_0 = 33.00, \omega_1 = 23.00, \alpha = 200, \beta = 8)$. Please refer to "Appendix B.4" for detailed information about the algorithm parameters of the optimization procedure together with a figure summarizing the convergence diagnostics indicating successful convergence.

Results Figure 6 presents the results derived from the optimization process. The upper two rows depict the congruence between simulation-based and true elicited statistics, effectively highlighting successful learning. The first row illustrates the alignment between true and learned quantiles for the chosen days x_i. The first two plots in the lower row show the distributions of R^2 as predicted by the model and the ground truth for day 0 and 9. The model-based histograms are depicted in blue and the ground truth in red. Finally, moment-based elicitation (i.e., mean and standard deviation) for the model parameter s is depicted as remaining information in the second row.

The learned prior distributions for each model parameter are depicted in the lower, right column of Fig. 6. The high overlap between true (dashed lines) and learned (solid lines) prior distributions indicates an additional instance of successful learning. This is further supported by the assessment of the error between true and learned hyperparameters in the lower left column, revealing a progressive convergence towards zero across epochs.

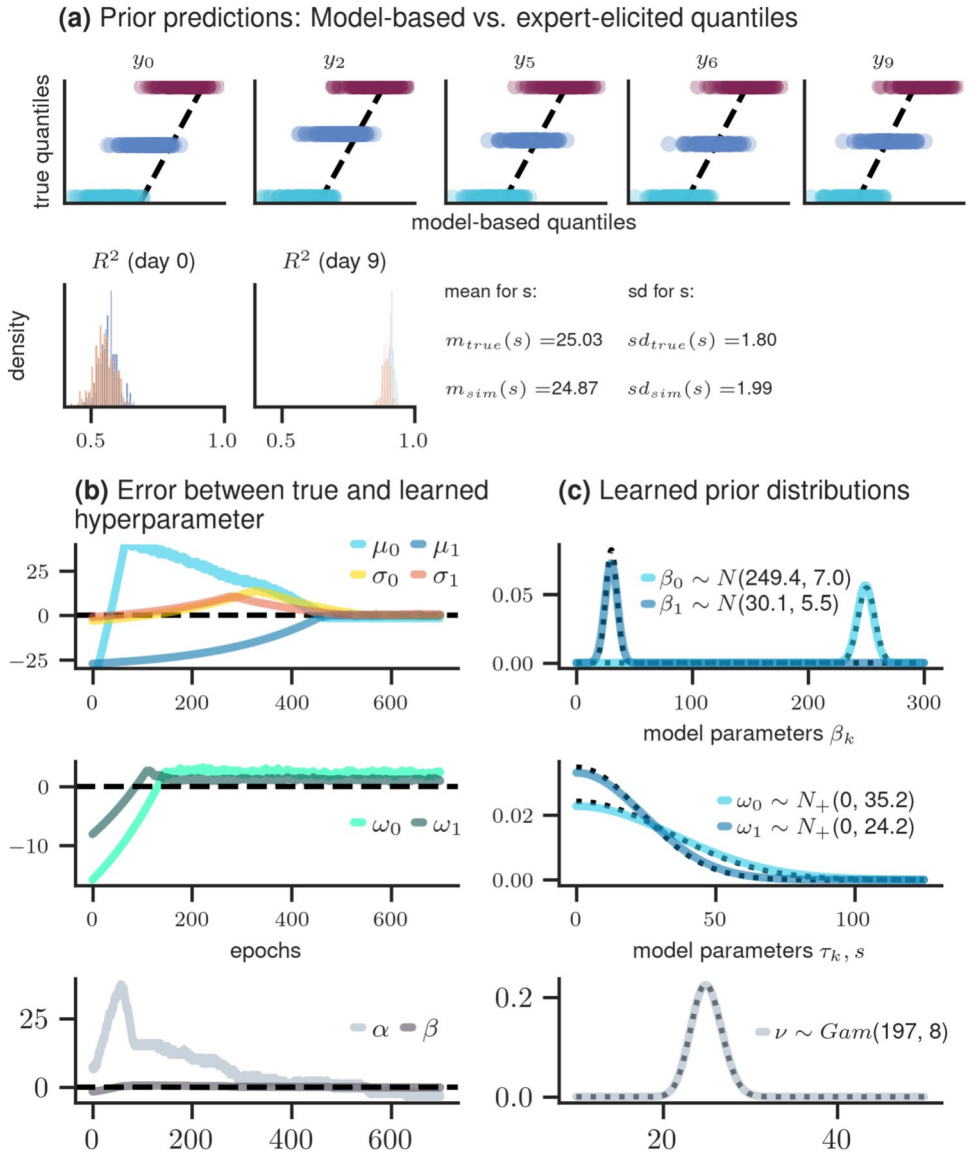


Figure 6. Results of simulation study 4: (**a**) comparison between model-based and true elicited statistics. First row depicts quantile-based elicitation for each day x_i. Second row shows histogram-based elicitation for R^2 (red true and blue model-implied) and moment-based elicitation for model parameter s (m_{true}, sd_{true} stands for true elicited mean and standard deviation, respectively). (**b**) learning of hyperparameters across epochs, showcasing the difference between the true and learned values; (**c**) true (dotted line) and learned (solid line) prior distributions of each model parameter.

Discussion

When developing Bayesian models, analysts face the challenge of specifying appropriate prior distributions for each model parameter, involving both the choice of the distributional family as well as the corresponding hyperparameter values. We proposed an elicitation method that assists analysts in identifying the hyperparameter values of given prior distribution families based on expert knowledge. Our method accommodates various types and formats of expert knowledge and is agnostic to the specific probabilistic model. In our simulation studies, we demonstrated the excellent performance of our method for various modeling tasks and kinds of expert knowledge. Despite these highly promising results, some relevant limitations remain, which are discussed below together with ideas for future research.

Our method employs gradient-based optimization to learn hyperparameter values which requires only the ability to sample from the generative model. However, it comes with the prerequisite that all operations and functions in the computational graph are differentiable or admit a reparameterization whose gradients can be approximated with sufficient accuracy. Consequently, for discrete random variables, specific techniques, such as the Softmax-Gumbel trick, are necessary. Alternatively, one could opt for optimization methods that entirely forego gradient computations such as Bayesian optimization[31] as used by Manderson & Goudie[6]. Nevertheless, this choice has its own limitations, notably in terms of scalability to higher-dimensional spaces[32].

Having a suitable optimization method is fundamental for learning hyperparameters based on expert knowledge. However, there are cases where hyperparameters cannot be uniquely determined from available expert data, leading to different learned hyperparameters upon multiple replications of the learning process. This situation raises the question of how to choose between prior distributions that represent the elicited expert knowledge equally well. Initial approaches, such as incorporating a regularization term in the loss function to favor priors with higher entropy, have been proposed to address this challenge[6]. Another avenue to achieve model identification involves the model architecture. For instance, statistical models that adopt joint priors for their parameters and thus keep the number of hyperparameters low, are expected to exhibit improved model identification [e.g.,[33]]. Nevertheless, further research is needed to develop informative metrics for assessing model identification as well as techniques that can efficiently handle unidentified models[34].

Finally, all gradient-based optimization methods share the objective of finding an optimal *point estimate* for the hyperparameters λ. By adopting this approach, any uncertainties surrounding the value of λ are neglected, despite the potential introduction of uncertainty during the prior elicitation process. To address this limitation, it would be advantageous to adopt a probabilistic approach that explicitly accounts for uncertainty in the hyperparameters [e.g.,[5,16]]. Given the flexibility of our method, it can readily accommodate this concept, offering a promising avenue for future development and next steps.

Methods

We propose a new elicitation method for translating knowledge from a domain expert into an appropriate parametric prior distribution. Building on recent contributions[6,9,16] we developed a model-agnostic method in which the search for appropriate prior distributions is formulated as an optimization problem. Thus, the objective is to determine the optimal hyperparameters that minimize the discrepancy between model-implied and expert-elicited statistics. Our elicitation method supports expert feedback in both the space of parameters and observable quantities (i.e., a *hybrid* approach) and minimizes human effort. The key ideas underlying our method are outlined as follows:

1. The analyst defines a generative model comprising a likelihood function $p(y \mid \theta)$ and a parametric prior distribution $p(\theta \mid \lambda)$ for the model parameters, where λ represents the prior hyperparameters to be inferred from expert knowledge.
2. The analyst selects a set of target quantities, which may involve queries related to observable quantities (data), model parameters, or anything else in between.
3. The domain expert is queried using a specific elicitation technique for each target quantity (*expert-elicited statistics*) .
4. From the generative model, parameters and (prior) predictive data are simulated, and the predefined set of target quantities is computed (*model-implied quantities*) .
5. The discrepancy between the model-implied and the expert-elicited statistics is evaluated via a specific loss function.
6. Stochastic gradient descent is employed to update the hyperparameters λ so as to minimize the loss function.
7. Steps 4 to 6 are repeated iteratively until an optimal set of hyperparameters λ is found that minimizes the discrepancy between the model-implied and the expert-elicited statistics.

In the upcoming sections, we will delve into the details of the outlined approach. To provide a visual representation of all steps involved in our proposed elicitation method, Fig. 7 presents a graphical overview. In addition, readers can find a symbol glossary in "Appendix A" for a quick reference. An illustrative example that details each step of the workflow using specific values can be found in our online supplement https://osf.io/rxgv2.

Elicited statistics from the expert

We assume that the analyst queries the domain expert regarding a predetermined set of I target quantities, represented as $\{z_i\} := \{z_i\}_{i=1}^{I}$. The set $\{z_i\}$ is selected by the analyst depending on the requirements of the statistical model and the knowledge of the expert[5,9,21]. Once this set is defined, the expert is queried regarding each individual target quantity z_i, assuming that the expert possesses an implicit representation, denoted as $\hat{z}_i$, which can be accessed using expert *elicitation techniques*[6,14,16,35]. While numerous elicitation techniques have been proposed in the literature[21], it can be argued that these techniques essentially represent different facets of the following three general method families: moment-based elicitation (e.g., mean and standard deviation), quantile-based elicitation (e.g., median, lower quartile, and upper quartile), and histogram elicitation (e.g., constructing a histogram by sampling from the distribution of z_i). Each target quantity z_i can be elicited through a distinct elicitation technique f_j. Within our notation, we represent the i^{th} target quantity elicited from the expert through the j^{th} elicitation technique as $\hat{t}_m = \hat{t}_{ij}$ and refer to it as *elicited statistics* $\hat{t}_m = f_j(\hat{z}_i)$. The index $m = 1, \ldots, M$ indicates the number of elicited statistics resulting from specific target-quantity $\times$ elicitation-technique combinations, as selected by the analyst.

Model-based quantities

Considering the set of elicited statistics queried from the expert $\{\hat{t}_m\}$, it is possible to assess the extent to which a generative model, as specified by the analyst, aligns with the expert's expectations. A Bayesian model comprises a likelihood $p(y \mid \theta)$ as well as parametric prior distributions $p(\theta \mid \lambda)$ for the model parameters θ. Here, λ represents the prior hyperparameters to be inferred by our method and y a vector of observations. The degree to which the model captures the expert's expectations relies on the specific values assigned to λ. Consequently, the objective

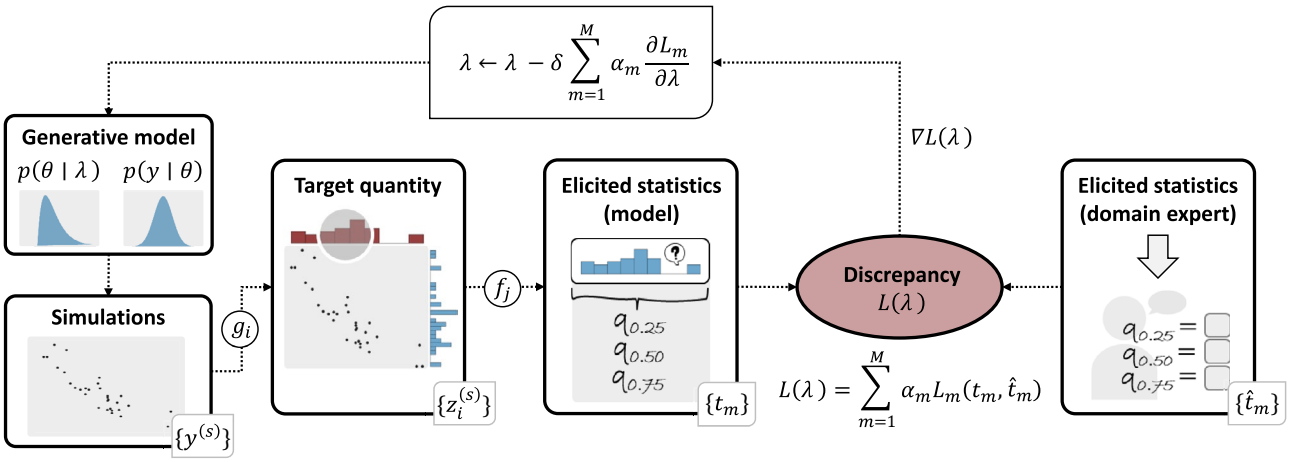


Figure 7. Graphical illustration of our simulation-based elicitation method. Step 1 involves employing elicitation techniques to extract target quantities from the domain expert. Subsequently, the objective is to minimize the discrepancy between model-implied and expert-elicited statistics by optimizing the hyperparameters λ. The optimization process iteratively simulates data using the current hyperparameters λ, computes model-implied elicited statistics, compares them with the expert-elicited statistics using a loss function (L_m), and updates λ to improve agreement between model-implied and expert-elicited statistics. Here, α_m is the weight of the m^{th} loss component and δ is the step size.

is to identify a specification of λ that minimizes the discrepancy between the set of expert-elicited statistics $\{\hat{t}_m\}$ and model-implied elicited statistics, $\{t_m\}$.

First, we need to derive the set of model-implied target quantities $\{z_i\}$. As a target quantity can represent an observable, a parameter, or anything else in between, we define it in the most general form as a function of the model parameters θ, denoted as $z_i = g_i(\theta)$, where the function g can take on various forms and be of deterministic or stochastic nature. In its simplest form, the target quantity directly corresponds to a parameter of interest in the data-generating model ($z_i = g_i(\theta) = \theta_i$; i.e., g would be a simple projection). Alternatively, g can be aligned with the generative model of the data, resulting in the target quantity being equivalent to the observations ($z_i = g_i(\theta) = y$). Moreover, the function g can take on more complex forms. Suppose the domain expert provides prior knowledge about the coefficient of determination R^2 commonly used to measure model fit in regression models[36]. To obtain the corresponding model-implied R^2, we first generate observations y using the specified generative model and then compute the R^2 value from the observations. Given the set of model-implied target quantities, we get the respective model-implied *elicited statistics*, denoted by $\{t_m\}$, by applying the elicitation technique f_j to the target quantity $z_i : t_m = f_j(z_i)$.

A challenge with this approach is that the distribution of $\{t_m\}$ may not be analytical or have a straightforward computational solution. For instance, consider the case where the target quantity is equivalent to the observations, $z_i = y$. In this case, the distribution of the predicted observations y gives rise to an integral equation known as the prior predictive distribution (PPD), denoted by $p(y \mid \lambda)$ and defined by averaging out the prior from the generative model: $p(y \mid \lambda) = \int_\Theta p(y \mid \theta) p(\theta \mid \lambda) d\theta$. Obtaining a closed-form expression for this integral is only feasible in certain special cases, such as when dealing with conjugate priors. This challenge extends to all situations where the target quantity is a function of the observations y. However, to ensure the broad applicability of our elicitation method to a wide range of models, we adopt a simulation-based approach that relies solely on the ability to generate samples from the relevant quantities. Bayesian models, by their very formulation, can simulate data from their prior and likelihood distributions, thereby enabling us to generate samples from the Bayesian probabilistic model[2,37]. For example, in the case where $z_i = y$, the simulation-based procedure involves two steps: Firstly, we sample the model parameters from the prior distribution conditioned on hyperparameters λ: $\theta^{(s)} \sim p(\theta \mid \lambda)$. Subsequently, we generate data by sampling from the likelihood distribution, resulting in $y^{(s)} \sim p(y \mid \theta^{(s)})$. The superscript (s) is used to denote the sth sample of the corresponding simulated quantity. By repeating these steps, we can generate a collection of S simulations $\{y^{(s)}\} := \{y^{(s)}\}_{s=1}^{S}$, where each element corresponds to a data point drawn from the PPD: $y^{(s)} \sim p(y \mid \lambda)$.

Multi-objective optimization problem

Once the elicited statistics $\{\hat{t}_m\}$ from the expert and a procedure to compute the corresponding model-implied elicited statistics $\{t_m\}$ are chosen, the focus can be shifted towards the main objective: Determine the hyperparameters λ that minimize some discrepancy measure (loss function) $L(\lambda)$ between the expert-elicited $\{\hat{t}_m\}$ and the model-implied statistics $\{t_m\} = \{t_m(\lambda)\}$. Since the evaluation of the discrepancy extends to all elicited statistics $\{t_m\}$, $L(\lambda)$ has to be formulated as a multi-objective loss function. This loss function encompasses a linear combination of discrepancy measures L_m, with corresponding weights α_m (see Section "Dynamic weight averaging").

In the following, we will also use the term *loss components* to refer to the individual components in the weighted sum. The selection of the discrepancy measure L_m is contingent upon the elicited statistic, therefore different choices may be appropriate depending on the specific quantity to be compared (see Section "Maximum mean discrepancy"). Independently of these specific choices, our main objective can be written as

$$\lambda^* = \arg\min_{\lambda} L(\lambda) = \arg\min_{\lambda} \sum_{m=1}^{M} \alpha_m L_m(t_m(\lambda), \hat{t}_m), \tag{5}$$

where λ^* denotes the optimal value of the hyperparameters λ given the provided expert knowledge.

Gradient-based optimization

The optimization procedure for solving Eq. (5) follows an iterative approach. In each iteration, we sample from the generative model, compute the model-implied elicited statistics, and update the hyperparameters λ. This update relies on calculating the gradient of the discrepancy loss with respect to the hyperparameters λ and adjusting them in the opposite direction of the gradient[38]. The procedure continues until a convergence criterion is met, usually when all elements of the gradient approach zero. We employ mini-batch stochastic gradient descent (SGD) with automatic differentiation, facilitated by the reparameterization trick [explicit or implicit;[39,40]. In our case, stochasticity in mini-batch SGD arises naturally as we simulate new model-implied quantities at each iteration step.

The reparameterization trick involves splitting the representation of a random variable into stochastic and deterministic parts. By differentiating through the deterministic part, we can compute gradients with respect to λ using automatic differentiation[41]. To leverage backpropagation, it is essential that all operations and functions in the computational graph are differentiable with respect to λ. This requirement extends to the loss function and all computational operations in the generative model[9,16].

However, dealing with discrete random variables poses a challenge due to the non-differentiable nature of discrete probability distributions, making gradient descent through such variables difficult. One approach to overcome this challenge is to use continuous relaxation of discrete random variables, which enables the estimation of gradients and thus the use of gradient-based optimization methods for models that involve discrete random variables[42–44]. For instance, both Maddison et al.[43] and Jang et al.[44] independently proposed the Gumbel-Softmax trick, which approximates a categorical distribution, with finite number of categories, with a continuous distribution. Joo et al.[45] proposed an extension of the Gumbel-Softmax trick to arbitrary discrete distributions by introducing truncation for those distributions that lack upper and/or lower boundaries. We used the Softmax-Gumbel trick in simulation study 2 and applied the truncation technique in simulation study 3 (Sections "Simulation study 2: GLMs—binomial model" and "Simulation study 3: GLMs—Poisson model").

Maximum mean discrepancy

A key aspect of the optimization problem, as expressed in Eq. (5), is the selection of an appropriate discrepancy measure, L_m. This measure depends on the characteristics of the elicited statistics $\{t_m\}$ and $\{\hat{t}_m\}$. Given that our method entails the generation of $\{t_m\}$ through repeated sampling from the generative model, a loss function is needed that can quantify the discrepancy between samples. The *maximum mean discrepancy* (MMD)[46,47] is a kernel-based method designed for comparing two probability distributions when only samples are available, making it suitable for our specific requirements. We utilize the MMD for all loss components in our applications. This decision is based on the robust simulation results and excellent performance as reported in the simulation studies section. That said, our method does not strictly require the MMD, but allows analysts to choose a different discrepancy measure for each loss component, if desired.

Let $x = \{x_1, \ldots, x_n\}$ and $y = \{y_1, \ldots, y_m\}$ be iid draws from the distributions p and q, respectively. The MMD measures the distance between two sets of samples by taking the maximum difference in sample averages over a function class $\mathcal{F}$ (Def. 2)[46]: $\text{MMD} = \sup_{f \in \mathcal{F}} \left(\mathbb{E}_{x \sim p}[f(x)] - \mathbb{E}_{y \sim q}[f(y)] \right)$. If $\mathcal{F}$ is a unit ball in the universal *reproducing kernel Hilbert space* $\mathcal{H}$ with associated reproducing kernel $k(\cdot, \cdot)$, the MMD is a strictly proper divergence, thus equals zero if and only if $p = q$[47]. The (biased) empirical estimate of the squared-MMD is defined as $\text{MMD}_b^2 = \frac{1}{n^2} \sum_{i,j=1}^{n} k(x_i, x_j) + \frac{1}{m^2} \sum_{i,j=1}^{m} k(y_i, y_j) - \frac{2}{mn} \sum_{i,j=1}^{m,n} k(x_i, y_j)$ where $k(\cdot, \cdot)$ is a continuous and characteristic kernel function. In our simulations, we used the *energy distance* kernel $k(x, y) = -||x - y||$, as proposed by Feydy[48] and Feydy et al.[49], which does not require an extra hyperparameter for tuning.

Dynamic weight averaging

In addition to selecting an appropriate discrepancy measure, another important consideration involves choosing the weights α_m in Eq. (5). One possibility is for the user to customize the choice of α_m, signifying the varying degrees of importance for each loss component in a particular application[50]. However, another consideration refers to the *task balancing problem*. When employing stochastic gradient descent to minimize the objective as outlined in Eq. (5), the hyperparameters λ are updated according to the following rule $\lambda \leftarrow \lambda - \delta \sum_{m=1}^{M} \alpha_m \frac{\partial L_m}{\partial \lambda}$, where δ is the step size (i.e., learning rate). The equation suggests that the hyperparameter update may not yield optimal results if one loss component significantly outweighs the others[50].

Consequently, a strategy is needed to dynamically modify the weights α_m to ensure effective learning of all loss components. For example, the *dynamic weight averaging* (DWA) method proposed by Liu et al.[51] determines the weights based on the learning speed of each component, aiming to achieve a more balanced learning process. Specifically, the weight of a component exhibiting a slower learning speed is increased, while it is decreased for faster learning components[52].

In our simulation studies, we consider an equal-weighting scheme ($\alpha_m = 1$), as this choice has demonstrated good learning outcomes without introducing additional free hyperparameters required by most task balancing approaches. However, we believe that investigating different task balancing approaches is a promising avenue for future research. Such exploration could have a beneficial impact on the method's performance, particularly in cases involving conflicting expert information.

Data and code availability

All code and data is openly available on OSF https://osf.io/rxgv2 and GitHub https://github.com/florence-bockting/PriorLearning.

Received: 18 September 2023; Accepted: 19 July 2024
Published online: 27 July 2024

References

1. Jaynes, E. T. *Probability Theory: The Logic of Science* (Cambridge University Press, 2003).
2. Gelman, A. et al. *Bayesian Data Analysis* (Chapmann and Hall/CRC, 2013).
3. Bürkner, P.-C., Scholz, M. & Radev, S. T. Some models are useful, but how do we know which ones? Towards a unified Bayesian model taxonomy. Preprint at https://doi.org/10.48550/arXiv.2209.02439 (2022).
4. Navarro, D. J. Between the devil and the deep blue sea: tensions between scientific judgement and statistical model selection. *Comput. Brain Behav.* **2**, 28–34 (2019).
5. Mikkola, P. *et al.* Prior knowledge elicitation: The past, present, and future. *Bayesian Anal.* **1**, 1–33 (2023).
6. Manderson, A. A. & Goudie, R. J. Translating predictive distributions into informative priors. Preprint at https://doi.org/10.48550/arXiv.2303.08528 (2023).
7. Gelman, A., Simpson, D. & Betancourt, M. The prior can often only be understood in the context of the likelihood. *Entropy* **19**, 555 (2017).
8. Albert, I. *et al.* Combining expert opinions in prior elicitation. *Bayesian Anal.* **7**, 503–532 (2012).
9. Da Silva, E. d. S., Kuśmierczyk, T., Hartmann, M. & Klami, A. Prior specification via prior predictive matching: Poisson matrix factorization and beyond. *J. Mach. Learn. Res.* **24**, 1–51 (2023).
10. Garthwaite, P. H., Kadane, J. B. & O'Hagan, A. Statistical methods for eliciting probability distributions. *J. Am. Stat. Asso.* **100**, 680–701 (2005).
11. Falconer, J. R., Frank, E., Polaschek, D. L. & Joshi, C. Methods for eliciting informative prior distributions: A critical review. *Decis. Anal.* **19**, 189–204 (2022).
12. Montague, T. H., Price, K. L. & Seaman, J. W. *Bayesian Applications in Pharmaceutical Development.* (Chapman and Hall/CRC, 2019).
13. Johnson, S. R., Tomlinson, G. A., Hawker, G. A., Granton, J. T. & Feldman, B. M. Methods to elicit beliefs for Bayesian priors: a systematic review. *J. Clin. Epidemiol.* **63**, 355–369 (2010).
14. O'Hagan, A. et al. *Uncertain judgements: Eliciting experts' probabilities* (John Wiley & Sons, 2006).
15. Perepolkin, D., Goodrich, B. & Sahlin, U. Hybrid elicitation and quantile-parametrized likelihood. *Stat. Comput.* **34**, 11 (2024).
16. Hartmann, M., Agiashvili, G., Bürkner, P. & Klami, A. Flexible prior elicitation via the prior predictive distribution. In *Proc. Conf. UAI* (eds Peters, J. & Sontag, D.) **124**, 1129–1138 (2020).
17. Bedrick, E. J., Christensen, R. & Johnson, W. A new perspective on priors for generalized linear models. *J. Am. Stat. Asso* **91**, 1450–1460 (1996).
18. Kadane, J., Dickey, J. M., Winkler, R. L., Smith, W. S. & Peters, S. C. Interactive elicitation of opinion for a normal linear model. *J. Am. Stat. Asso.* **75**, 845–854 (1980).
19. Muandet, K. *et al.* Kernel mean embedding of distributions: A review and beyond. *Found. Trends Mach. Learn.* **10**, 1–141 (2017).
20. Akbarov, A. *Probability elicitation: Predictive approach*. PhD thesis (2009).
21. Stefan, A. M., Evans, N. J. & Wagenmakers, E.-J. Practical challenges and methodological flexibility in prior elicitation. *Psychol. Methods* **27**, 177–197 (2022).
22. Falconer, J. R., Frank, E., Polaschek, D. L. & Joshi, C. Eliciting informative priors by modeling expert decision making. *Decis. Anal.* **21**, 77–90 (2024).
23. Bernardo, J. M. & Smith, A. F. *Bayesian theory.* (John Wiley & Sons, 1994).
24. Cranmer, K., Brehmer, J. & Louppe, G. The frontier of simulation-based inference. *Proc. Natl. A. Sci.* **117**, 30055–30062 (2020).
25. Abadi, M. et al. TensorFlow: Large-scale machine learning on heterogeneous systems. Tech. Rep. Software available from https://www.tensorflow.org/ (2015).
26. Unkelbach, C. & Rom, S. A referential theory of the repetition-induced truth effect. *Cognition* **160**, 110–126 (2017).
27. Dua, D. & Graff, C. Haberman's survival dataset. UCI Machine Learning Repository. http://archive.ics.uci.edu/ml (2017).
28. Johnson, A. A., Ott, M. Q. & Dogucu, M. *Bayes Rules!: An Introduction to Applied Bayesian Modeling* (Chapman and Hall/CRC Press, 2022).
29. Belenky, G. *et al.* Patterns of performance degradation and restoration during sleep restriction and subsequent recovery: a sleep dose-response study. *J. Sleep Res.* **12**, 1–12 (2003).
30. Lewandowski, D., Kurowicka, D. & Joe, H. Generating random correlation matrices based on vines and extended onion method. *J. Multivariate Anal.* **100**, 1989–2001 (2009).
31. Frazier, P. I. Bayesian optimization. In *Recent Advances in Optimization and Modeling of Contemporary Problems*, INFORMS TutORials in Operations Research, chap. 11, 255–278 (INFORMS, 2018).
32. Eriksson, D. & Jankowiak, M. High-dimensional Bayesian optimization with sparse axisaligned subspaces. In *Uncertainty in Artificial Intelligence, 2012. Proceedings* (eds. de Campos, C. & Maathuis, M. H.) 493–503 (2021).
33. Aguilar, J. E. & Bürkner, P.-C. Intuitive joint priors for Bayesian linear multilevel models: The R2D2M2 prior. *Electron. J. Stat.* **17**, 1711–1767 (2023).
34. Sameni, R. Beyond convergence: Identifiability of machine learning and deep learning models. Preprint at https://doi.org/10.48550/arXiv.2307.11332 (2023).
35. Kadane, J. & Wolfson, L. J. Experiences in elicitation. *J. Roy. Stat. Soc. D-Sta.* **47**, 3–19 (1998).
36. Gelman, A., Goodrich, B., Gabry, J. & Vehtari, A. R-squared for Bayesian regression models. *Am. Stat.* **73**, 307–309 (2019).
37. Aushev, A. *et al.* Online simulator-based experimental design for cognitive model selection. Preprint at https://doi.org/10.48550/arXiv.2303.02227 (2023).
38. Goodfellow, I., Bengio, Y. & Courville, *A. Deep Learning* (MIT Press, 2016).
39. Kingma, D. P. & Welling, M. Auto-encoding variational Bayes. *Stat* **1050**, 1 (2014).
40. Figurnov, M., Mohamed, S. & Mnih, A. In *Advances in Neural Information Processing Systems* (eds. Bengio, S. et al.) 31 (2018).
41. Sjölund, J. A tutorial on parametric variational inference. Preprint at https://doi.org/10.48550/arXiv.2301.01236 (2023).

42. Tokui, S. & Sato, I. Reparameterization trick for discrete variables. Preprint at https://doi.org/10.48550/arXiv.1611.01239 (2016).
43. Maddison, C. J., Mnih, A. & Teh, Y. W. The Concrete Distribution: A Continuous Relaxation of Discrete Random Variables. In *International Conference on Learning Representations. Proceedings* (2017).
44. Jang, E., Gu, S. & Poole, B. Categorical Reparameterization with Gumbel-Softmax. In *International Conference on Learning Representations. Proceedings* (2017).
45. Joo, W., Kim, D., Shin, S. & Moon, I.-C. Generalized gumbel-softmax gradient estimator for generic discrete random variables. Preprint at https://arxiv.org/abs/2003.01847 (2023).
46. Gretton, A., Borgwardt, K., Rasch, M., Schölkopf, B. & Smola, A. A kernel method for the two-sample problem. In *Advances in NIPS* (eds. Schölkopf, B., Platt, J. & Hoffman, T.) 19 (2006).
47. Gretton, A.Notes on the cramer gan. Medium. https://towardsdatascience.com/noteson-the-cramer-gan-752abd505c00 (2017).
48. Feydy, J. *Geometric Data Analysis, Beyond Convolutions.* PhD thesis (2020).
49. Feydy, J. et al. Interpolating between optimal transport and mmd using sinkhorn divergences. In *The 22nd International Conference on Artificial Intelligence and Statistics, 89* (eds. Chaudhuri, K. & Sugiyama, M.) 2681–2690 (2019).
50. Wang, L., Ng, A. H. C. & Deb, K. *Multi-objective Evolutionary Optimisation for Product Design and Manufacturing.* (Springer, 2011).
51. Liu, S., Johns, E. & Davison, A. J. End-to-end multi-task learning with attention. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 1871–1880 (2019).
52. Crawshaw, M. Multi-task learning with deep neural networks: A survey. Preprint at https://doi.org/10.48550/arXiv.2009.09796 (2020).

Acknowledgements

FB and PCB were supported by Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy – EXC-2075-390740016 (the Stuttgart Cluster of Excellence SimTech). STR war supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy—EXC-2181-390900948 (the Heidelberg Cluster of Excellence STRUCTURES). The authors gratefully acknowledge the computing time provided on the Linux HPC cluster at Technical University Dortmund (LiDO3), partially funded in the course of the Large-Scale Equipment Initiative by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) as Project 271512359.

Author contributions

FB and PCB drafted the manuscript and all authors contributed to writing the manuscript. All authors provided revisions to the manuscript, discussed the results and commented on the manuscript. PCB supervised the research.

Funding

Open Access funding enabled and organized by Projekt DEAL.

Competing interests

The authors declare no competing interests.

Additional information

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1038/s41598-024-68090-7.

Correspondence and requests for materials should be addressed to F.B.

Statistics and Computing (2025) 35:132
https://doi.org/10.1007/s11222-025-10665-z

ORIGINAL PAPER

Expert-elicitation method for non-parametric joint priors using normalizing flows

Florence Bockting[1] · Stefan T. Radev[2] · Paul-Christian Bürkner[1]

Received: 24 November 2024 / Accepted: 6 June 2025 / Published online: 19 June 2025

Abstract
We propose an expert-elicitation method for learning non-parametric joint prior distributions using normalizing flows. Normalizing flows are a class of generative models that enable exact, single-step density evaluation and can capture complex density functions through specialized deep neural networks. Building on our previously introduced simulation-based framework, we adapt and extend the methodology to accommodate non-parametric joint priors. Our framework thus supports the development of elicitation methods for learning both parametric and non-parametric priors, as well as independent or joint priors for model parameters. To evaluate the performance of the proposed method, we perform four simulation studies and present an evaluation pipeline that incorporates diagnostics and additional evaluation tools to support decision-making at each stage of the elicitation process.

Keywords Prior elicitation · Expert knowledge · Joint prior distribution · Normalizing flows · Non-parametric priors

1 Introduction

The Bayesian paradigm offers the possibility to incorporate *prior knowledge* into a *statistical model* through the specification of *prior distributions*. This possibility is a central advantage of the Bayesian paradigm (Mikkola et al. 2023), yet it also presents one of its most challenging aspects (Simpson et al. 2017; lgorzata et al. 2015; Van Dongen 2006). In the following, we define prior knowledge as the expertise provided by a *domain expert* — an individual with extensive knowledge of a specific subject matter (Falconer et al. 2022). This knowledge can be represented in various forms, but to integrate it into a Bayesian model, we need to translate it into a formal mathematical language that can be expressed as a prior distribution over the model parameters (Perepolkin et al. 2023; O'Hagan 2019; Martin et al. 2012; Garthwaite et al. 2005).

A distinct field of research, commonly known as *(expert) prior elicitation*, has developed around the question of how to gather expert knowledge and translate it into appropriate prior distributions. This area of study has a long history, dating back to the 1960s (Winkler 1967; Kadane et al. 1980; Kadane and Wolfson 1998), and continues to be an active area of research today (Stefan et al. 2022; Mikkola et al. 2023; Falconer et al. 2022). Garthwaite et al. (2005) identified four key stages in a *prior elicitation process*:

1. **Setup stage**: In this stage, the problem is defined, an expert is selected, and the quantities to be elicited from the expert (in this paper referred to as *target quantities*) are determined;
2. **Elicitation stage**: Here, the target quantities are queried from the expert using specific elicitation techniques, resulting in what we call *elicited statistics*;
3. **Fitting stage**: This involves fitting a (potentially joint) probability distribution based on the expert-elicited statistics;
4. **Evaluation stage**: Finally, the adequacy of the fitted probability distribution is assessed in collaboration with the expert.

✉ Florence Bockting
florence.bockting@tu-dortmund.de

Stefan T. Radev
stefan.radev93@gmail.com

Paul-Christian Bürkner
paul.buerkner@gmail.com

[1] Department of Statistics, TU Dortmund University, Vogelpothsweg 87, 44227 Dortmund, North Rhine-Westphalia, Germany

[2] Cognitive Science Department, Rensselaer Polytechnic Institute, 110 Eighth Street, 12180 Troy, NY, United States

In this context, *elicitation methods* aim to provide a systematic and formal procedure for deriving prior distributions based on expert-elicited statistics. Early elicitation methods primarily tackled the problem by seeking analytical solutions, such as conjugate models or problem-specific transformation functions resulting in highly model-specific prior elicitation methods (see Mikkola et al. 2023, for a recent review). This model dependence is partly due to the use of direct elicitation techniques, which involve asking experts directly about model parameters (Stefan et al. 2022; Falconer et al. 2022). This approach is problematic not only due to its inherent model dependence but also because model parameters are often challenging for domain experts to interpret meaningfully. Consequently, the quality of expert input can be questionable. In response to these limitations, substantial efforts have been made to develop methods that enable more flexible and interpretable prior specification for domain experts.

A recently proposed group of elicitation methods uses advances in machine learning to automate the process of translating expert knowledge into prior distributions (Hartmann et al. 2020; da Silva et al. 2019; Manderson and Goudie 2023; Bockting et al. 2024). These methods address both key challenges: they allow expert knowledge to be expressed in terms of observable quantities and ensure that the elicited information remains interpretable for domain experts. Additionally, since the quantities elicited from the expert are not tied to specific model parameters, these methods are highly versatile and model-independent. The underlying idea behind this approach is closely related to prior predictive checks, which are an integral part of the *Bayesian workflow* (Gelman et al. 2020; Gabry et al. 2019). The key concept is the *prior predictive distribution* (PPD), defined as $p(y) = \int p(y \mid \theta)p(\theta)d\theta$ with likelihood $p(y \mid \theta)$ and prior $p(\theta)$. The PPD establishes a formal relationship between the prior distributions of the model parameters and the model predictions, $p(y)$. In this way, the PPD provides a means to link expert knowledge, expressed in terms of observable quantities, to the latent model parameters. Since the PPD is often not analytically tractable, it is typically approximated using Monte Carlo integration (Mikkola et al. 2023) which involves first sampling $\theta' \sim p(\theta)$ and then $y' \sim p(y \mid \theta')$. The resulting model predictions, y', are then compared to expert knowledge on the outcome variable, y^*, using an appropriate discrepancy loss function. The goal is to learn prior distributions, $p(\theta)$, that produce model predictions consistent with expert knowledge by means of minimizing the discrepancy loss (Garthwaite et al. 2005; Gelman et al. 2017; Simpson et al. 2017; Betancourt 2020).

The methods introduced so far that use this *simulation-based* approach focus on learning independent, parametric priors $p(\theta \mid \lambda)$, parameterized by a set of prior hyperparameters λ (Hartmann et al. 2020; da Silva et al. 2019; Manderson and Goudie 2023). While assuming *independent* priors for model parameters can be reasonable in some cases due to theoretical considerations or model constraints, this approach may not be sufficient for more complex or high-dimensional problems (Gelman et al. 2017, 2020; Simpson et al. 2017). In such cases, it becomes important to account for the *joint distribution* of model parameters to capture dependencies and interactions that reflect the underlying structure more accurately. Similarly, the use of *parametric* prior distribution families can be well-justified in some cases, but may lead to model misspecification in others, prompting the need for *non-parametric* approaches. Elicitation methods that focus on non-parametric priors are less studied, but examples include the use of Gaussian processes (Oakley and O'Hagan 2007) and quantile-parameterized distributions (Perepolkin et al. 2023, 2024).

Building in particular on the work of Hartmann et al. (2020); da Silva et al. (2019); and Manderson and Goudie (2023), we previously introduced a simulation-based framework with a highly modular structure that closely aligns with the elicitation process described by Garthwaite et al. (2005). The advantage of this modular structure is its flexibility, allowing for easy adaptation to different types of methods. In our previous work (Bockting et al. 2024), we demonstrated how to use this framework to learn parametric prior distributions for the model parameters. In this paper, we show how the same simulation-based framework can be used to learn flexible (i.e., non-parametric) joint priors for the model parameters with only minor adjustments to the workflow. In both approaches, we employ mini-batch stochastic gradient descent as the optimization method.

Main Contributions Our motivation is two-fold: First, we aim to promote the development of a unifying, highly modular framework that supports a wide range of prior elicitation methods. We believe that such a framework would benefit both users and developers of prior elicitation methods by providing structure and clarity in a field currently populated with numerous coexisting methods, which can make comparisons challenging. Second, we want to demonstrate how our simulation-based framework, introduced in Bockting et al. (2024), can serve as a flexible approach for learning either parametric or non-parametric, as well as independent or joint, priors with only minor adjustments. While our previous work focused on learning parametric, independent priors, the present work emphasizes learning non-parametric, joint priors.

2 Methodology

In this paper, we propose an elicitation method that extends our recently introduced simulation-based framework (Bockt-

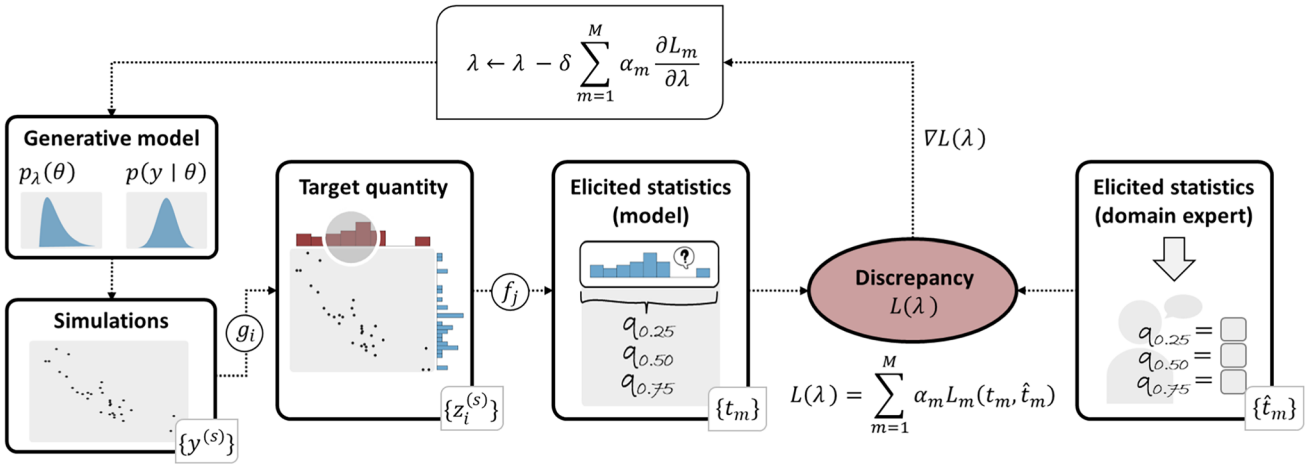


Fig. 1 *Graphical illustration of our simulation-based prior-elicitation framework.* The process begins by identifying target quantities to be elicited from the domain expert and selecting appropriate elicitation techniques, which result in (expert-)elicited statistics. Next, predictions are simulated from the generative model by sampling from the prior $p_\lambda(\theta)$ and computing the corresponding model-implied target quantities and elicited statistics. The consistency between model-and expert-elicited statistics is assessed using a loss function L_m, where α_m is the weight of the m^{th} loss component. Hyperparameters λ that define the learned prior are adjusted based on this evaluation to reduce the loss and align model predictions more closely with expert knowledge. This iterative process continues until a prior is found that minimizes the discrepancy between model and expert-elicited statistics. In the updating rule, δ refers to the step size

ing et al. 2024) to support the learning of non-parametric joint priors. The overall workflow remains unchanged; the modifications primarily concern how the prior distributions are specified and learned. We will first provide a brief overview of the workflow to establish the background, followed by a more detailed discussion of the modification introduced to the original framework, which is the focus of the current paper.

The general workflow of the framework closely resembles the approach of *prior predictive checks* (Gelman et al. 2020): We simulate from the joint model $p(\theta, y)$ and assess how well the resulting prior predictions align with the expert's expectations. If there is a discrepancy between the expert's expectations and the model simulations, the prior specification needs to be adjusted accordingly. Figure 1 provides a graphical representation of the framework and Table 1 offers a summary of the symbols and notation used in the following sections. The general workflow of our framework can be summarized as follows:

1. *Define the generative model*: Define the generative model including dimensionality and parameterization of prior distribution(s). (Setup stage; Section 2.1)
2. *Identify variables and elicitation techniques for querying expert knowledge*: Select the set of variables to be elicited from the domain expert (target quantities) and determine which elicitation techniques to use for querying the selected variables from the expert (elicited statistics). (Setup stage; Section 2.2)
3. *Elicit statistics from expert and simulate corresponding predictions from the generative model*: Sample from the generative model and perform all necessary computational steps to generate model predictions (model-elicited statistics) corresponding to the set of expert-elicited statistics. (Elicitation stage; Section 2.3)
4. *Evaluate consistency between expert knowledge and model predictions*: Evaluate the discrepancy between the model- and expert-elicited statistics via a multi-objective loss function. (Fitting stage; Section 2.4)
5. *Adjust prior to align model predictions more closely with expert knowledge*: Use mini-batch stochastic gradient descent to adjust the prior so as to reduce the loss. (Fitting stage; Section 2.4)
6. *Find prior that minimizes the discrepancy between expert knowledge and model predictions*: Repeat steps 2 to 5 iteratively until a prior is found that minimizes the discrepancy between the model and expert-elicited statistics. (Fitting stage; Section 2.4)
7. *Evaluate the learned prior distributions*: Run the learning algorithm (steps 2 to 6) multiple times to obtain a set of prior distributions that can equally well represent the expert data. Select a plausible prior distribution in consultation with the domain expert or apply model averaging techniques. (Evaluation stage; Section 2.5)

The following sections discuss each step in greater detail.

Table 1 Notation and symbols used in this paper

Symbol	*Description*
Simulation-based prior elicitation	
y_n	data $n = 1, \ldots, N$
θ_k	model parameter $k = 1, \ldots, K$
λ	model hyperparameter
$p(y \mid \theta)$	likelihood
$p(\theta \mid \lambda)$	prior parameterized by λ
$p(y)$	prior predictive distribution
$z_i = c_i(\lambda)$	simulated model target quantity defined by function c_i
$\hat{z}_i$	expert representation of the target quantity
$t_m = f_j(z_i)$	model-elicited statistic defined by elicitation technique f_j
$\hat{t}_m$	expert-elicited statistic
$L_m(t_m(\lambda), \hat{t}_m)$	loss component
$L(\lambda) = \sum_{m=1}^{M} \alpha_m L_m(t_m(\lambda), \hat{t}_m)$	total loss as weighted sum of L_m with weights α_m
$p_{\lambda_1,\ldots,\lambda_R}(\theta) = \sum_{r=1}^{R} w_r \cdot p_{\lambda_r}(\theta)$	averaged prior across R replications and weighted by w_r with $r = 1, \ldots, R$
$\Delta_r(L) = L(\lambda_r) - L(\lambda_{\min})$	difference between total loss of replication r and replication with minimum total loss
$w_r = \frac{\exp\{-\gamma \Delta_r(L)\}}{\sum_{v=1}^{V} \exp\{-\gamma \Delta_v(L)\}}$	weights used for model averaging with γ being a scaling factor; in all simulation studies $\gamma = 1$.
Normalizing flow	
$u \sim p_U(u)$	samples from base distribution
$g_\lambda(\cdot)$	generator function
$u = g_\lambda(\theta)$	normalizing direction
$\theta = g_\lambda^{-1}(u)$	generative direction
$g_\lambda = g_{\lambda_H} \circ \cdots \circ g_{\lambda_1}$	composition of H coupling layers

2.1 Prior Specification & Learning

In our previous work (Bockting et al. 2024), we introduced the simulation-based framework along with a method for learning a set of *independent parametric* prior distributions, $p(\theta_k \mid \lambda_k)$, for the model parameters θ_k with $k = 1, \ldots, K$. In this paper, we extend our framework by introducing a method to learn a *joint non-parametric* prior distribution $p_\lambda(\theta) = p_\lambda(\theta_1, \ldots, \theta_K)$ over all model parameters θ. In both the parametric and non-parametric approaches, the objective of the optimization process is to learn the hyperparameters λ that minimizes the discrepancy between the model simulations and the expert expectations. However, the interpretation of these hyperparameters differs between approaches, which we emphasize by using a different notation for the prior distributions. In the non-parameteric approach, we employ *normalizing flows* (NFs) to induce a flexible family of prior distributions which entail specialized deep neural networks with trainable parameters λ (Kobyzev et al. 2020). Thus, in the non-parametric approach, λ represents the weights of the deep neural networks within the NFs, whereas in the parametric approach, λ denotes the parameters of the associated prior distribution families (i.e., the prior hyperparameters). A more detailed introduction to NFs will be provided in the upcoming section.

In summary, the main difference between the parametric approach (proposed by Bockting et al. (2024) and the non-parametric approach (focus of the current paper) is that, in the former, the user must specify a parametric prior distribution family for each model parameter, and the corresponding prior hyperparameters are learned via stochastic optimization. In the new extension, the user no longer needs to specify a parametric prior for the model parameters. Instead, a complex non-parametric prior distribution is assumed. This non-parametric joint prior is represented and learned using deep neural networks (specifically, NFs), with the weights learned via stochastic optimization. Table 2 summarizes the main steps in the elicitation process and highlights where the extension modifies the existing workflow of our simulation-based framework. Furthermore, we provide in this paper additional diagnostics to support users in interpreting the simulation results.

Normalizing Flows NFs transform a simple probability distribution, known as the *base distribution* $p(u)$, into a more complex *target distribution*, which, in our case, is the joint

Table 2 Comparison between the deep-prior and parametric-prior approach on a conceptual level. Note: The symbol '%' refers to the same content as provided in parametric-prior approach. NFs is an abbreviation for Normalizing Flows (see text for details)

Elicitation process	*parametric-prior approach*	*deep-prior approach*
Setup		
Model	specify generative model	%
Priors	specify **parametric prior** distribution families	assume **non-parametric joint prior** (specified via NFs)
Data	specify target quantities and elicited statistics	%
	(**less expert input** needed due to stronger regularized priors)	(**more expert input** required (e.g., dependency information) due to flexible prior)
Elicitation	specify elicitation techniques used to query the domain expert	%
Fitting	apply mini-batch stochastic gradient descent	%
Evaluation	provide diagnostics, visualizations, etc. to evaluate simulation results	%

prior $p_\lambda(\theta)$. This transformation is achieved through a series of invertible and differentiable mappings g_λ, parameterized by λ (Kobyzev et al. 2020). The composition of invertible functions g_λ yields an explicit form of the density $p_\lambda(\theta)$ via the change of variables formula

$$p_\lambda(\theta) = p(u = g_\lambda(\theta)) \mid \det g'_\lambda(\theta) \mid,$$

where $g'_\lambda(\theta) = \frac{\partial}{\partial\theta} g_\lambda(\theta)$ is the Jacobian matrix of g_λ at θ and $\mid \det g'_\lambda(\theta) \mid$ is its absolute determinant. This formula allows us to easily evaluate the prior density at arbitrary points of interest θ (Dinh et al. 2016). Obtaining samples from the joint prior can be achieved by first sampling from the base distribution, $p(u)$, and then applying the inverse $g_\lambda^{-1}(u)$ to obtain samples from $p_\lambda(\theta)$:

$$\theta = g_\lambda^{-1}(u) \sim p_\lambda(\theta) \quad \text{for} \quad u \sim p(u).$$

Typically, $p(u)$ is chosen to be a simple density, such as a standard Gaussian or a uniform distribution (Kobyzev et al. 2020).

For NFs to be practical, the mapping g_λ should be easily invertible, computationally efficient, and sufficiently expressive to model any target distribution of interest (Kobyzev et al. 2020). Several types of NFs that meet these requirements have been proposed, including affine (Dinh et al. 2014) and spline coupling flows (Durkan et al. 2019). A detailed discussion and introduction to coupling flows can be found in various sources (Draxler et al. 2024; Dinh et al. 2014, 2016; Radev et al. 2020; Kobyzev et al. 2020; Papamakarios et al. 2021; Bond-Taylor et al. 2021). Due to their simple architecture, affine coupling flows are easy to invert and computationally efficient, although they have restricted expressiveness (Bond-Taylor et al. 2021). However, the expressiveness of a coupling flow can be increased by stacking multiple affine coupling layers, $g_\lambda = g_{\lambda_H} \circ \cdots \circ g_{\lambda_1}$ with $h = 1, \ldots, H$ coupling blocks (Dinh et al. 2016). In fact, Draxler et al. (2024) prove that affine coupling flows are distributional universal approximators, capable of representing any target distribution despite their seemingly restrictive architecture. In line with these considerations and the strong empirical performance of affine coupling flows, we focused on this type of normalizing flows in our simulation studies (see Section 4), using the implementation provided by BayesFlow (Radev et al. 2023).

Finally, it is worth noting that, since our focus is solely on the mapping from the base distribution to the target distribution (i.e., the generative direction of NFs), virtually any other generative model family could be used to learn how to generate samples from a joint prior. Examples of common generative architecture alternatives include diffusion models (Cao et al. 2024) and flow matching (Lipman et al. 2022). However, we used NFs here because they perform well in practice and are computationally efficient. In particular, they do not require solving differential equations during sampling (Draxler et al. 2024).

2.2 Target Quantities & Elicitation Techniques

A key task in the setup stage of the elicitation process is the selection of target quantities and elicitation techniques to effectively gather the expert information. Two main criteria should guide this selection: *interpretability* and *informativeness* (Mikkola et al. 2023; Crowder 1992; da Silva et al. 2019; Garthwaite et al. 2005).

Selecting Target Quantities *Interpretability* refers to the selection of target quantities that allows domain experts to meaningfully express their knowledge based on their experience and expertise (Garthwaite et al. 2013). Target quantities on the same scale as the outcome variable (cf., *observable* quantities) are considered to be highly interpretable (da Silva et al. 2019; Manderson and Goudie 2023). In contrast, target quantities that refer to model parameters are more challenging to interpret, especially when the outcome variable is not

on the same scale as the model parameters (Kadane and Wolfson 1998; Kadane et al. 1980). Therefore, it is sometimes advocated asking experts exclusively about observable quantities (Kadane and Wolfson 1998). We share the view of Denham and Mengersen (2007) that the appropriateness of either approach depends on the specific problem being addressed and the type of expert providing the information.

Guided by this idea, we define a target quantity within our framework in the most general sense as a function of the trainable hyperparameters of the prior: $z_i = c_i(\lambda)$ for $i = 1, \ldots, I$. Typically, multiple target quantities (I in total) are elicited from the user to inform different aspects of the data-generating process. For instance, if we want to query the expert regarding the outcome variable (i.e., the observable space), one of the target quantities can be simply defined as $z_i = y$. This is obtained by defining the function c_i as a query to the prior predictive distribution $p(y) = \int p(y \mid \theta) p_\lambda(\theta) d\theta$, where the likelihood is marginalized over the prior. Note that this expression is not just a constant as it refers to the prior predictive distribution and should not be confused with the marginal likelihood of observed data (see for this point also Hartmann et al. (2020)). As another example, expert knowledge may be elicited about the parameter space. In this case, the target quantity is defined as $z_i = \theta_k$, which is obtained by defining c_i as a simple projection onto θ_k. In this way, we can also accommodate any other target quantity that can be derived from the model parameters or the data (e.g., R^2; the proportion of variance explained by the model; (Gelman et al. 2019)).

Selecting Elicitation Techniques Once the set of target quantities has been selected, we need to determine how to elicit this information from the expert, which requires choosing an appropriate *elicitation technique*. While experts can be asked to provide the full distribution of a target quantity (e.g., in the form of a histogram (Johnson et al. 2010)), another common technique is to focus on specific summary statistics (Morris et al. 2014). Research on prior elicitation suggests that experts can reasonably be asked to describe a target quantity using proportions, the mode, the median, and generally quantiles (Garthwaite et al. 2005; Stefan et al. 2022; O'Hagan et al. 2006), with quantile-based elicitation techniques being particularly recommended (Kadane and Wolfson 1998). In contrast, asking about the mean is less advisable when the distribution is skewed (Peterson and Miller 1964), and similarly, querying variances is in general discouraged (Garthwaite et al. 2005; Kadane and Wolfson 1998). Furthermore, assessing dependencies between variables is challenging, making the elicitation of joint priors especially difficult (Garthwaite et al. 2005). Elicitation of correlations in the context of joint priors has been studied among others by Gokhale and Press (1982); Clemen et al. (2000), and Dickey et al. (1985) as reviewed by Mikkola et al. (2023).

In our elicitation method, we represent an elicitation technique as a function f_j for $j = 1, \ldots, J$, of a target quantity z_i. The resulting set of *elicited* target quantities is referred to as *elicited statistics*, $\{t_m(\lambda)\}$, where $t_m(\lambda) = f_j(z_i)$ and m refers to the corresponding $i \times j$ combination. In the following, we sometimes omit the explicit dependence on λ and simply write t_m instead of $t_m(\lambda)$. Note also that, depending on the elicitation technique employed, t_m may accept either a single value or a set of values as input. To illustrate this point more clearly, consider the case in which we elicit expert knowledge about the predictive distribution conditional on a category of a categorical predictor, i.e., $z = y \mid \text{cat}$. To elicit this information, we query the expert regarding the 25%, 50%, and 75% quantiles of this predictive distribution. Thus, we have $t = \{Q_{25\%}(y \mid \text{cat}), Q_{50\%}(y \mid \text{cat}), Q_{75\%}(y \mid \text{cat})\}$.

Sensitivity Analysis In addition to interpretability, the second criterion for selecting target quantities and elicitation techniques is *informativeness*, which refers to the relevance of the elicited statistics for learning the prior distributions. Evaluating this criterion is nontrivial because the relationship between the elicited statistics and the model (hyper-)parameters is complex and often analytically intractable. One computational approach to asses informativeness is to conduct a *sensitivity analysis* (Depaoli et al. 2020). In this approach, we systematically vary one aspect of the prior distribution at a time while keeping all other aspects constant. For each specific change in the prior, we run the generative model in forward mode by sampling from the prior, then from the likelihood, and subsequently computing the target quantities and elicited statistics. Sensitivity analysis enables us to assess how changes in one aspect of the prior distribution impact each elicited statistic. If we observe no variation in a particular elicited statistic for a given prior variation, this statistic does not provide any useful information for the learning algorithm to determine the corresponding aspect of the prior. This approach can help inform the selection of elicited statistics and identify which aspects of the prior are most likely to remain unidentifiable given the current set of expert information.

2.3 Model-implied & Expert-elicited Statistics

After determining the set of target quantities and corresponding elicitation techniques, we can proceed to gather information from the expert through specific *prior elicitation protocols* (Gosling 2018; Cooke 1991; European Food Safety 2014). This process may be preceded by a training phase to familiarize the expert with the elicitation tasks (Stefan et al. 2022). In prior elicitation research, an entire subfield has focused on determining how to conduct optimal expert interviews, taking into account factors such as cognitive biases

and heuristics. While it is beyond the scope of this paper to review this literature, several comprehensive reviews are available (e.g., Mikkola et al. 2023; Stefan et al. 2022; Falconer et al. 2022). At this point, we will skip the actual interrogation process and assume that the expert information has been successfully gathered, providing us with a set of expert-elicited statistics, individually denoted by $\hat{t}_m$ and together as $\{\hat{t}_m\}$.

Given the set of expert-elicited statistics, $\{\hat{t}_m\}$, we can evaluate the discrepancy between the expert expectations and the model-implied statistics, $\{t_m\}$. The latter are computed by simulating from the model. This proceeds as follows: We start with an arbitrary initial prior distribution from which we sample model parameters $\theta^{(s)} \sim p_{\lambda_0}(\theta)$, where λ_0 represent the initial hyperparameter values that define the prior distribution. The superscript s denotes the s^{th} sample out of a total of S samples. Subsequently, we perform all necessary computational steps to derive samples from the target quantities $z_i^{(s)}$. Finally, we apply the corresponding elicitation technique to obtain the model-implied statistics $t_m = f_j(\{z_i^{(s)}\})$ from the simulations.

2.4 Discrepancy Evaluation & Training

To evaluate the discrepancy between the model- and expert-elicited statistics, an appropriate loss function is used for each statistic, $L_m(t_m(\lambda), \hat{t}_m)$. Different loss functions may be preferable depending on the type of the respective elicited statistic. Since the discrepancy is computed for each statistic within the set of elicited statistics, the total loss comprises multiple *loss components*, which are combined into a single total loss using a weighted sum:

$$L(\lambda) = \sum_{m=1}^{M} \alpha_m L_m(t_m(\lambda), \hat{t}_m), \tag{1}$$

where α_m denote the weights. The choice of weights α_m can be guided by different considerations: One approach involves manually adjusting the weights, guided either by the expert's assessment of the importance of individual loss components or by ensuring that the effects of different magnitudes, arising from varying scales, are balanced, such that no single loss component dominates the others (Wang et al. 2011). Alternatively, automated weighting methods, known as loss-balancing methods, can be used to optimally balance the contribution of each individual term to the total gradient (Liu et al. 2019; Crawshaw 2020; Bischof and Kraus 2021).

After computing the total loss, $L(\lambda)$, the gradients with respect to the hyperparameters λ are calculated. These gradients are then used to adjust the hyperparameters in the opposite direction:

$$\lambda' \leftarrow \lambda - \delta \sum_{m}^{M} \alpha_m \frac{\partial L_m}{\partial \lambda}, \tag{2}$$

with δ being the step size (Goodfellow et al. 2016). We employ mini-batch stochastic gradient descent (SGD) with automatic differentiation, facilitated by the (explicit or implicit) reparameterization trick (Kingma and Welling 2014; Figurnov et al. 2018). With the adjusted hyperparameters λ', new samples from the prior $p_{\lambda'}(\theta)$ can be generated and the updated model-implied statistics $t_m(\lambda')$ computed. We then reassess the discrepancy between the model-implied and expert-elicited statistics and continue to adjust the hyperparameters as needed. This iterative process continues until a convergence criterion is met or a stopping rule is applied.

2.5 Evaluation of Training Results

Convergence Checks Training is considered successful if the loss in minimized and no further learning occurs. With a proper discrepancy measure as loss function, it is guaranteed that the total loss (and the individual loss components) approach zero as learning progresses. The learning progress can be tracked by examining the loss behavior across epochs, where a decreasing trend is expected until the loss stabilizes at a value close to zero. Further insights into the learning progress can be obtained by tracking the convergence of additional quantities of interest beside the loss, for instance the mean and standard deviation of the marginal priors.

A further method to assist in the examination of convergence is to compute the slope of a linear regression fitted to the loss values over the last m epochs, whereby m should not be taken too large such that the linearity assumption holds. Typically, the overall loss function exhibits exponential decay, but once the loss stabilizes, a linear trend is a good approximation. Ideally, a slope of zero would indicate perfect convergence, though some variation is expected in practice. To evaluate whether the slope remains acceptable, the training algorithm can be run repeatedly with random seeds and the absolute slopes across all replications can be compared. For training runs with the highest final slopes, we can perform a visual inspection of the trajectories of key quantities across epochs, such as the total loss, loss components, and prior means and standard deviations. If these runs show successful convergence, we can reasonably conclude that other runs (with smaller final slopes) have also converged.

Non-uniqueness of Learned Priors Once the elicited statistics have been learned accurately, we can examine the corresponding learned prior distribution. As there are often multiple optimal values λ^* that align closely with the same set of elicited statistics, multiple replications will yield different prior distributions (da Silva et al. 2019; Manderson and Goudie 2023; Stefan et al. 2022). This lack of uniqueness

is anticipated, given the limited information provided by the set of elicited statistics compared to the complexity of the generative model (Manderson and Goudie 2023). Furthermore, when target quantities refer solely to the observable space, we only gain access to the model's overall uncertainty with limited ability to differentiate between aleatoric and epistemic uncertainty (Nemani et al. 2023; Perepolkin et al. 2024). Aleatoric uncertainty, by definition, is irreducible; however, several methods exist to reduce epistemic uncertainty. These include (i) selecting more informative target quantities, (ii) eliciting more detailed expert information (e.g., increasing the number of quantiles in quantile-based elicitation), (iii) adding theory-informed regularization terms to constrain the search space (as illustrated in Manderson and Goudie (2023)), (iv) improving the initialization method to achieve better starting points, and (v) employing more advanced optimization algorithms when dealing with a large number of model parameters (Nemani et al. 2023).

Model Averaging Given the (potential) non-uniqueness of the learned prior distributions, it is advisable to run the training multiple times on the same expert data but with different random seeds. This approach helps provide insight into the variability among the learned prior distributions. From the resulting set of candidate priors $p_{\lambda_r}(\theta)$ for $r = 1, \ldots, R$, we can, in consultation with the domain expert, exclude prior distributions that do not meet the expert's expectations. For the remaining set of prior distributions, which may be considered practically equivalent from the expert's perspective, either a single prior distribution can be chosen, or alternatively, model averaging techniques can be applied. In the case of model averaging, the combined prior would take the form:

$$p_{\lambda_1,\ldots,\lambda_R}(\theta) = \sum_{r=1}^{R} w_r \cdot p_{\lambda_r}(\theta) \tag{3}$$

where the weights w_r sum to one. The weights w_r can either be assigned by the user, based on theoretical assumptions, or selected according to one of several proposed weighting schemes (Claeskens and Hjort 2008). Here, we apply an approach suggested by Buckland et al. (1997) for the general purpose of model averaging and selection, where the weights w_r are defined as

$$w_r = \frac{\exp\{-\gamma I_r\}}{\sum_{v=1}^{V} \exp\{-\gamma I_v\}}.$$

In the above expression, γ is a scaling factor, and I represents the performance criterion by which the models are weighted. This definition ensures that models with identical performance receive equal weight (Buckland et al. 1997). In our setting, it is sensible to use the total loss $L(\lambda_r)$ of each replication r as the performance criterion. Consequently, the "best" model is the one with the smallest total loss, $L(\lambda_{\min})$ (Bissiri et al. 2016). This yields the following definition of weights

$$w_r = \frac{\exp\{-\gamma \Delta_r(L)\}}{\sum_{v=1}^{V} \exp\{-\gamma \Delta_v(L)\}}. \tag{4}$$

where $\Delta_r(L) = L(\lambda_r) - L(\lambda_{\min})$. The difference is used primarily for computational reasons to ensure numerical stability (Claeskens and Hjort 2008). The formulation in Eq. (4) can be interpreted as the probability of replication r representing the best prior, given the collection of learned prior distributions (Wagenmakers and Farrell 2004). The scaling factor γ controls the influence of each replication's relative performance $\Delta_r(L)$. When $\gamma > 1$, discrepancies in model performance are amplified, with better models receiving higher weights and worse models receiving lower weights. As $\gamma \to \infty$, this weighting would approach a one-hot vector, effectively performing model selection. Conversely, when $\gamma < 1$, the performance differences between models are smoothed out, reducing the influence of performance discrepancies.

Simulating From an "Oracle" The simulation-based framework enables us to evaluate the degree of non-identifiability and determine if further regularization is required, even before querying any data from the expert. An effective approach involves using an *oracle*, where a pre-defined prior distribution serves as the *ground truth*. From this "true" prior, we perform forward simulations: sampling first from the prior, then from the likelihood, and carrying out all necessary computational steps to generate a set of model-implied statistics corresponding to the ground truth. This set of simulated statistics can then serve as a proxy for the "expert-elicited" statistics.

This *oracle* approach allows us to assess the self-consistency of the algorithm, the degree of non-identifiability (i.e., how informative are the elicited statistics), and the suitability of the chosen algorithm parameters used for training NFs with mini-batch SGD (such as the learning rate or the number of epochs). However, there is one important caveat when using an *oracle*: the user might be tempted to compare the learned prior distributions against the true prior distributions (which are now available). Specifically, if there is an unsatisfactory match between the true and learned prior distributions, the user might interpret this mismatch as a failure in learning. It is important to remember, however, that the method only knows about the elicited statistics, which are incorporated into the loss function. Thus, when the elicited statistics do not incorporate information about the model parameters but instead pertain to observable quantities, there is, from the method's perspective, no concept of a true prior. In particular, if the true and learned elicited statistics match

but there is a mismatch in the learned prior, this is likely a result of an underidentified problem, and the objective function requires additional information to properly constrain the problem.

3 Implementation

In all simulation studies, we use a batch size of 128. To simulate from an oracle (discussed in Section 4.1), we draw once $S = 10,000$ samples from the true joint prior and then compute the corresponding set of "expert"-elicited statistics. During training with mini-batch SGD we draw $S = 200$ samples from the joint prior. To evaluate the discrepancy between the model-implied and expert-elicited statistics, we use a squared, biased *Maximum Mean Discrepancy* (MMD) loss with an energy kernel (Gretton et al. 2006; Feydy et al. 2019), following our previous success with this metric (Bockting et al. 2024). For the correlation between parameters, however, we apply a squared-error loss (L2 loss; for details see Section 4.1). The total loss is computed as a weighted sum of the individual loss components. The values of the respective weights are discussed together with the selection of the elicited statistics whereby each elicited statistic represents one loss component (see Section 4.1).

The architecture of the NFs is rather simple compared to typical applications in deep learning (e.g., Kingma and Dhariwal 2018,) and consists of a standard multivariate Gaussian as a base distribution and three affine coupling blocks. Each coupling block consists of two dense layers with 128 units and ReLU activation functions. In Simulation Study 1, the number of learnable hyperparameters is $|\lambda| = 202,776$, while in the case studies corresponding to Simulation Study 2, it is $|\lambda| = 205,872$.

In Simulation Study 1, we assume a discrete likelihood and use the gumble-softmax trick to compute the gradients of a discrete random variable (for more information, see Bockting et al. 2024). The gradients of the total loss with respect to the hyperparameters are used to update the hyperparameters λ. Optimization was performed with the Adam optimizer for 500 epochs in Simulation Study 1 and 800 epochs in Simulation Study 2. We used a fixed learning rate of 0.00025 for Simulation Study 2, Scenario 1 & 2, and 0.0001 for Simulation Study 1 and Stimulation Study 2, Scenario 3. These algorithm (hyper-)parameter values were obtained by manual tuning until good performance across all of our case studies was achieved.

All simulations and plots were performed in Python 3.11 using our Python package *elicito* 0.3.1 (Bockting and Bürkner 2025), which relies, among others, on the following libraries: TensorFlow 2.14 (Abadi et al. 2016), TensorFlow Probability 0.23 (Dillon et al. 2017), NumPy 1.26 (Harris et al. 2020), pandas 2.2.3 (McKinney et al. 2010), and BayesFlow 1.1.6 (Radev et al. 2023). To ensure the reproducibility of our results, the specific version of *elicito* used in this paper has been archived on Zenodo and is accessible via the following DOI https://doi.org/10.5281/zenodo.15241853. Simulations were executed on a YOGA Pro 9i laptop equipped with GPU acceleration.

4 Simulation Studies

4.1 General Setup

In the following, we present four case studies organized across two simulation studies to evaluate the performance of our elicitation method. The selection of case studies is motivated by the desire to choose scenarios that are simple and allow us to study the behavior of the method in a well-controlled setting. Since this paper presents the extension with NFs for the first time, it is important for us to first develop an understanding of the method's behavior. In future work, we will examine the behavior of our method in more complex models that are closer to real-world applications.

In Simulation Study 1, we introduce a simple Binomial regression model with one continuous predictor. In Simulation Study 2, we present three scenarios based on a normal regression model with a three-level categorical predictor. The three normal scenarios differ only in the pre-defined joint prior representing the ground truth. Table 3 provides an overview of the generative models and the joint priors representing the ground truth used in the simulation studies. Before introducing the simulation studies, we will discuss the decisions made in the simulation setup in greater detail. All code and results can be found on GitHub (https://github.com/florence-bockting/non-parametric-prior) and OSF (https://osf.io/xrzh6/), respectively. Information about the training time for each simulation study is provided in Appendix A.1.

Simulated Expert as Ground Truth For the simulation studies in this paper, we simulate expert input using synthetic data generated from a predefined joint prior representing the ground truth (i.e., oracle, see Section 2.5 for details). For simplicity, we use a parametric prior with fixed hyperparameter values as ground truth for each case study. Further details on the exact specification of the "true" joint prior is provided in the respective section and in Table 3.

Selection of Target Quantities and Elicitation Techniques Following recommendations in the field, we tried to select mainly observable quantities as target quantities for our simulation studies (see Section 2.2 for details). For the Binomial regression model, we used the expected observations of the outcome variable conditional on two observations of the continuous predictor, $y \mid x_0$ and $y \mid x_1$. For the normal regression model, we considered the prior predictive observations con-

Table 3 Overview of generative models and ground truth used in each simulation study. Symbols: **R** denotes the correlation matrix and s the vector of standard deviations

Model	*Generative Model*	*Ground Truth*
Binomial — M1	$y_i \sim \text{Binomial}(p_i, 30)$	$\beta_0 \sim \text{Normal}(0.1, 0.1)$
(Section 4.2)	$p_i = \text{sigmoid}(\beta_0 + \beta_1 x_i)$	$\beta_1 \sim \text{Normal}(-0.1, 0.3)$
Normal models		
Scenario 1 — M2	$y_i \sim \text{Normal}(\mu_i, \sigma)$	$\beta_0 \sim \text{Normal}(10, 2.5)$
(Section 4.3.1)	$\mu_i = \beta_0 + \beta_1 x_{1,i} + \beta_2 x_{2,i}$	$\beta_1 \sim \text{Normal}(7, 1.3)$
		$\beta_2 \sim \text{Normal}(2.5, 0.8)$
		$\sigma \sim \text{Gamma}(5, 2)$
Scenario 2 — M3	see M2	$\beta_0 \sim \text{Normal}(10, 2.5)$
(Section 4.3.2)		$\beta_1 \sim \text{SkewNormal}(7, 1.3, 4)$
		$\beta_2 \sim \text{SkewNormal}(2.5, 0.8, 4)$
		$\sigma \sim \text{Gamma}(5, 2)$
Scenario 3 — M4 (Section 4.3.3)	see M2	$\beta \sim \text{Mv-Normal}\left(\begin{bmatrix} 10 \\ 7 \\ 2.5 \end{bmatrix}, \text{D}(s)\,\mathbf{R}\,\text{D}(s)\right)$
		$\mathbf{R} = \begin{bmatrix} 1. & 0.3 & -0.3 \\ 0.3 & 1. & -0.2 \\ -0.3 & -0.2 & 1. \end{bmatrix}$
		$s = (2.5, 1.3, 0.8)$
		$\sigma \sim \text{Gamma}(5, 2)$

Table 4 Overview of the analysis workflow used in the simulation studies

Task	*Approach*	*Insights*
Setup & Elicitation Stage		
Select the set of elicited statistics and evaluate its informativeness.	(1) Simulate elicited statistics by defining a true joint prior (i.e., the oracle). (2) Conduct a sensitivity analysis.	Can the set of elicited statistics provide sufficient information to learn the key aspects of the joint prior?
Fitting Stage		
Evaluate variability in learned priors to assess non-identifiability.	Run the training algorithm several times with different random seeds.	Do different training runs yield different results?
Verify convergence of the training algorithm.	(1) Examine the slope of final loss values across replications. (2) Visually check the convergence of key quantities for selected seeds.	Will training the model for additional epochs lead to a practically relevant improvement in performance or a reduction in total loss?
Verify accurate learning of expert data.	Visually compare model-implied and expert-elicited statistics.	Was the method successful in accurately capturing the expert data?
Evaluation Stage		
Examine the learned priors across the different seeds	Visually inspect the marginal priors and correlations between model parameters.	How much do the learned priors vary between different seeds? Were any degenerate prior distributions learned? Do we observe any form of multi-modality? Can any of the learned priors be ruled out as unreasonable post hoc?
Obtain the final learned prior for subsequent analysis.	Perform model averaging using weights based on relative loss performance.	Instead of selecting a single prior, compute a model average over practically equivalent priors to account for additional variability due to non-identifiability.

ditional on three groups of the categorical predictor, $y \mid \text{gr}_i$ for $i = 1, 2, 3$.

However, we had to deviate from the recommendation to select observable target quantities in two respects. First, we included R^2 as an additional target quantity for the normal model (Simulation Study 2), motivated by the following reasoning: if we only have information about the outcome variable for each of the three groups, we provide the learning algorithm with no means to differentiate between parameter uncertainty and data uncertainty. The coefficient of determination, R^2, represents the ratio of parameter uncertainty to total uncertainty (which encompasses both data and parameter uncertainty). Consequently, it offers more information for the learning algorithm to differentiate between these two types of uncertainty, even though it may be somewhat more challenging for a domain expert to interpret. However, as R^2 is commonly reported as a metric in a regression context, we would argue that this quantity is still well understood by domain experts.

Second, we included the pairwise correlation between model parameters in the set of target quantities for each case study. If independence between model parameters is assumed, all correlations are set to zero; otherwise, we use the exact correlation structure indicated by the ground truth. We acknowledge that this information cannot be reasonably demanded from a domain expert in most cases. However, we currently lack suitable elicitation methods for asking domain experts about interpretable quantities that provide sufficient information regarding the correlation structure between the model parameters. A future task will be to develop such elicitation methods and to then incorporate these methods into our computational framework.

Regarding the selection of *elicitation techniques*, we follow the recommendation to query quantiles from a domain expert (Kadane and Wolfson 1998). For each target quantity, except for the correlation information, we elicit five quantiles: 5%, 25%, 50%, 75%, 95%. For the correlation values, we assume that the expert provides a single point estimate corresponding to a moment-based elicitation approach.

In summary, the elicited statistics for Simulation Study 1 consist of five quantiles each for $y \mid x_0$ (t_1) and $y \mid x_1$ (t_2), as well as the correlation information $\rho(\beta_0, \beta_1)$ (t_3). Each of these three elicited statistics corresponds to a distinct loss component in the objective function. The elicited statistics for Simulation Study 2 consist of five quantiles each for R^2 (t_1) and $y \mid \text{gr}_i$ with $i = 1, 2, 3$ (t_2, t_3, t_4), as well as the correlation information between the model parameters (t_5). The discrepancy in the correlation values is measured using an L2 loss, whereas the discrepancies for the remaining components are computed using the MMD^2 loss. To account for differences in scale-arising both from the use of different discrepancy measures and from the varying domains of the elicited statistics-we weight the individual loss components as follows: the correlation loss component is scaled by a factor of 0.1, the R^2 loss component by a factor of 10.0, and all other loss components are uniformly weighted with a factor of 1.0.

To assess whether the selected set of elicited statistics is sufficiently informative to determine a prior distribution for the model parameters, we conduct a *sensitivity analysis* for each case study.

Evaluation of Simulation Results We run each simulation study 30 times with different random seeds. As preliminary convergence checks, we calculate the slope of the loss trajectory over the last 100 epochs and perform a visual convergence check for the five replications with the highest absolute slopes (i.e., worst cases). Additionally, we examine the trajectories of the total loss, the individual loss components related to each elicited statistic, as well as the mean and standard deviation for each marginal prior across epochs. Furthermore, to visually assess whether the learning criterion has been accurately captured, we compare the final model-implied statistics with the expert-elicited statistics. Finally, we examine the learned prior distributions for each replication and compute the model average across the 30 learned prior distributions. Table 4 provides a summary of the analysis workflow followed in each of the subsequent simulation studies.

Additional simulation results For each case study, we conduct two additional sets of simulations, which are not part of the main analysis but are included in the supplementary material for completeness and further comparison. The supplementary material can be found on GitHub at https://github.com/florence-bockting/non-parametric-prior. As these simulations are complementary to the main focus of the paper, we do not report the results in the main text.

In the first additional set of simulations, we re-ran all case studies using the *model parameters* as the "elicited statistics" (instead of an indirect quantity derived from the model parameters). In this case, the method trains directly on the model parameters, enabling us to assess whether it is, in principle, capable of recovering the true joint prior distribution when complete information is available. Thus, this set of simulations serves as a validation set for our method.

In the second additional set of simulations, we applied the parametric-prior method proposed in Bockting et al. (2024) to all case studies. This allows for a direct comparison between the performance of the proposed non-parametric approach and the previously introduced parametric variant within the same simulation-based framework. The only exception is Scenario 3 of Simulation Study 2, where we introduce a multivariate normal prior. In this case, a comparison is currently not possible, as the optimization of a full covariance matrix is not yet implemented in *elicito* (Bockt-

ing and Bürkner 2025), but it is planned for a (near) future release.

4.2 Simulation Study 1: Binomial Likelihood with Independent Normal Priors

We introduce the general idea of our method using a Binomial model with a logit link (sigmoid response function) and a single continuous predictor, x. The predictor was generated by first constructing a sequence from 1 to 50, which was then scaled by its standard deviation. In the following we refer to this model as M1:

$$\begin{aligned} y_i &\sim \text{Binomial}(p_i, 30) \\ p_i &= \text{sigmoid}(\beta_0 + \beta_1 x_i) \\ \beta_0, \beta_1 &\sim p_\lambda(\beta_0, \beta_1) \\ \theta &\equiv (\beta_0, \beta_1) \end{aligned} \tag{M1}$$

The goal is to learn a joint prior for the model parameters β_0 (intercept) and β_1 (slope), assuming independence between these parameters.

Setup & Elicitation Stage As elicited statistics, we select prior predictions for two values of the continuous predictor, $y \mid x_0$ and $y \mid x_1$. The two values of x correspond to the 25% and 75% quantiles of the predictor. We select specific quantiles rather than randomly sampling two observations from the predictor to avoid the observations being too close together, which would reduce informativeness. From the distribution of the two selected observations, we compute five quantiles which represent the elicited statistics (see Section 4.1 for details). Note that $y \mid x_0$ and $y \mid x_1$ each have an associated distribution, resulting from simulating $y^{(s)}$ for S samples drawn from the joint prior and conditioning on the respective x value. The distribution over $y \mid x$ encodes the parameter uncertainty.

To obtain the "expert"-elicited statistics, we define a true prior that represents the ground truth and simulate from the generative model in forward mode, computing the corresponding true-elicited statistics. The *true* joint prior is defined by independent normal distributions for each model parameter: $\beta_0 \sim \text{Normal}(0.1, 0.1)$ and $\beta_1 \sim \text{Normal}(-0.1, 0.3)$.

To assess the *informativeness* of the selected set of elicited statistics, we conduct a sensitivity analysis, with the results shown in Figure 2. In each row, a single hyperparameter is varied across the range shown on the x-axis, while all other hyperparameters are held constant at their true values, indicated by the red vertical line. The columns present the two elicited statistics, specifically the five quantiles (in shades of blue and green) for $y \mid x_0$ and $y \mid x_1$. The upper two rows show results for the intercept parameter β_0 and the lower two rows for the slope parameter β_1. The results of the sensitivity analysis indicate that the selected elicited statistics provide sufficient informativeness for the model hyperparameters.

Fitting Stage We ran the training algorithm 30 times with different random seeds and conducted an initial convergence check by inspecting the slope of the last 100 epochs for each replication. In Figure 3, the upper row shows the absolute slope (scaled by a factor of 100) for each replication. In the lower row, the learning trajectory of the total loss is shown for the best-performing (leftmost) and the four worst-performing replications. We observe a near-zero slope for the best model, indicating convergence, while the negative slopes for the worst-performing seeds suggest that the learning algorithm continues to make progress.

However, a visual inspection of the learning trajectories of the total loss and individual loss components (Figure 4) and the means and standard deviations of the marginal priors (Figure 5), indicates satisfactory convergence. Therefore, although the slope analysis indicates minor ongoing learning progress, no substantial changes in the results are expected if the model is trained for additional epochs. Overall, we conclude that the training process was successful.

The corresponding final learned elicited statistics for each replication are shown in Figure 6. In the first two plots, each true quantile (x-axis) is plotted against its corresponding learned quantile (y-axis) for each replication. Points that align perfectly with the diagonal dashed line represent a perfect match between true and learned quantiles. Points above the diagonal indicate that the learned quantiles are higher than the true ones (and vice versa for lower points). Accuracy of the learned correlation information is presented in Figure 6. The true (red) correlation matches perfectly with the learned (black) correlations.

Evaluation Stage Finally, we examine the learned prior distributions. Figure 7 shows the learned marginal priors for each replication, alongside the results from the model averaging approach. The weights used for model averaging are shown in the upper plot of Figure 7 and reflect the relative performance of each replication based on their total loss (see Section 2.5 for details). In this case, all replications perform similarly well, resulting in an almost uniform weighting scheme. The resulting averaged prior is represented by the solid red line in the lower plot, shown together with the learned priors from each replication.

4.3 Simulation Studies 2: Normal Likelihood

In the remaining simulation studies, we aim to evaluate the method's performance using a model with slightly increased complexity, specifically a normal regression model with standard deviation σ and mean μ_i, where μ_i depends on a three-level grouping variable encoded via the dummy variables x_1 and x_2:

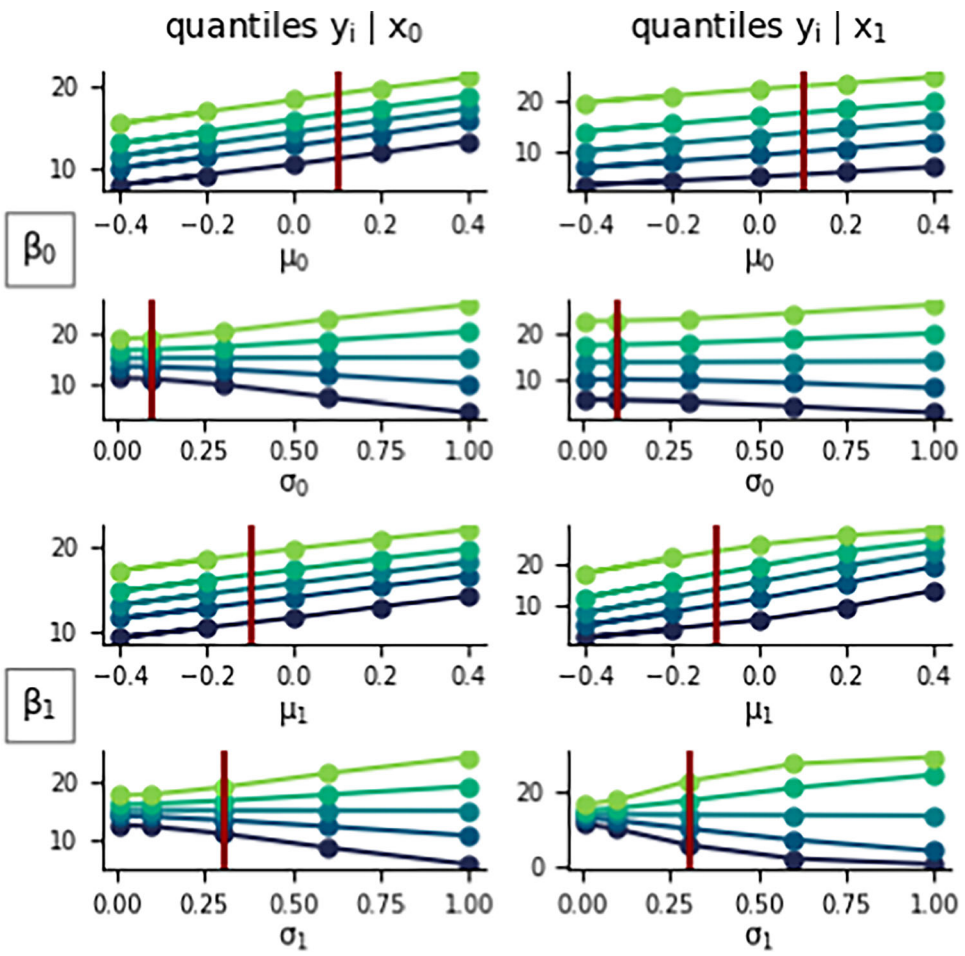


Fig. 2 *Binomial model—Sensitivity Analysis*: Rows represent hyperparameters of each model parameter. Columns represent elicited statistics: five quantiles for $y \mid x_0$ and $y \mid x_1$. Quantiles are depicted in different colors. In each row, the corresponding hyperparameter, is varied across the range shown on the x-axis, while all other hyperparameters are held constant at their true values, indicated by the red vertical line. The upper two rows indicate results for the intercept parameter β_0 and the lower two rows for the slope parameter β_1

$$
\begin{aligned}
y_i &\sim \text{Normal}(\mu_i, \sigma) \\
\mu_i &= \beta_0 + \beta_1 x_{1,i} + \beta_2 x_{2,i} \\
\beta_0, \beta_1, \beta_2, \sigma &\sim p_\lambda(\beta_0, \beta_1, \beta_2, \sigma) \\
\theta &\equiv (\beta_0, \beta_1, \beta_2, \sigma)
\end{aligned}
$$

with β_0, β_1, and β_2 being the regression coefficients representing the intercept and the two group contrasts, respectively. The goal is to learn a joint prior for the model parameters θ. To systematically assess our method, we use the same model across all upcoming simulation studies, varying only the *true* joint prior in each scenario:

- **Scenario 1 (M2)**: Independent normal priors for the regression coefficients are defined as $\beta_0 \sim$ Normal $(10, 2.5)$, $\beta_1 \sim \text{Normal}(7, 1.3)$, $\beta_2 \sim \text{Normal}(2.5, 0.8)$, with a Gamma prior for the random noise, $\sigma \sim \text{Gamma}(5, 2)$
- **Scenario 2 (M3)**: Identical to M2 but introduces skewed normal priors for β_1 and β_2, where $\beta_1 \sim \text{SkewNormal}(7, 1.3, 4)$[1] and $\beta_2 \sim \text{SkewNormal}(2.5, 0.8, 4)$.
- **Scenario 3 (M4)**: Identical to M2 but introduces a dependency structure among the model coefficients β using a correlated multivariate normal distribution, with marginal distributions identical to those in M2. The correlated prior is specified as:

$$
\beta \sim \text{Mv-Normal}\left(\begin{bmatrix} 10 \\ 7 \\ 2.5 \end{bmatrix}, \ \text{D}(s)\,\mathbf{R}\,\text{D}(s)\right) \quad \text{with}
$$

$$
\mathbf{R} = \begin{bmatrix} 1. & 0.3 & -0.3 \\ 0.3 & 1. & -0.2 \\ -0.3 & -0.2 & 1. \end{bmatrix}, \quad s = (2.5, 1.3, 0.8)
$$

where $\mathbf{R}$ denotes the correlation matrix and s the vector of standard deviations.

[1] For the skewed normal distribution, we use the implementation from TensorFlow Probability (Abadi et al. 2016), which features the two-piece normal distribution (Fernández and Steel 1998). This distribution is parameterized by location, scale, and shape parameters. The shape parameter controls the skewness, with values above one resulting in a right-skewed distribution.

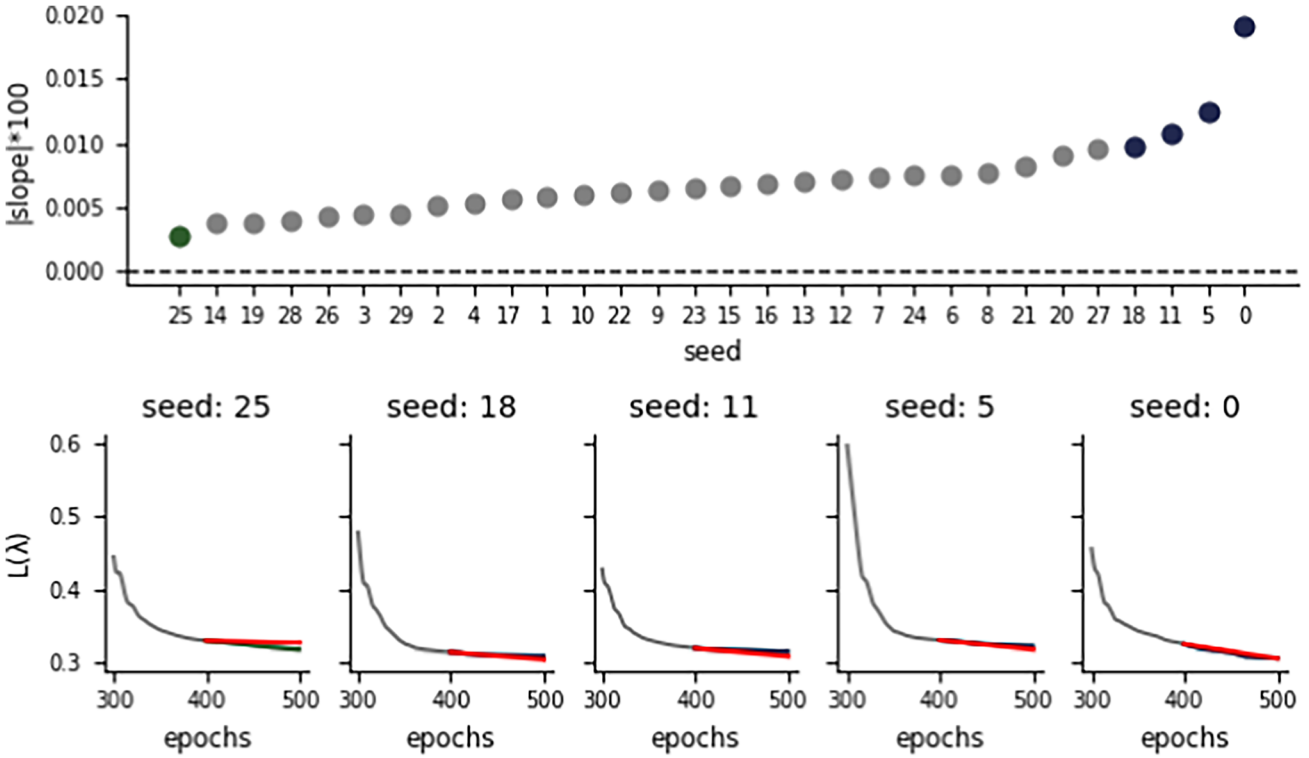


Fig. 3 *Binomial model—Preliminary convergence check for all 30 replications*: Upper plot: Absolute slope of the total loss trajectory (scaled by a factor of 100) over the last 100 epochs on the y-axis, with each replication shown along the x-axis. Replications (i.e., seeds) are arranged in ascending order based on the magnitude of the absolute slope. Best-performing model is seed 25, while the worst-performing models are seeds 18, 11, 5, and 0. Lower plot: Loss trajectories of the best- and worst-performing seeds over the last 200 epochs

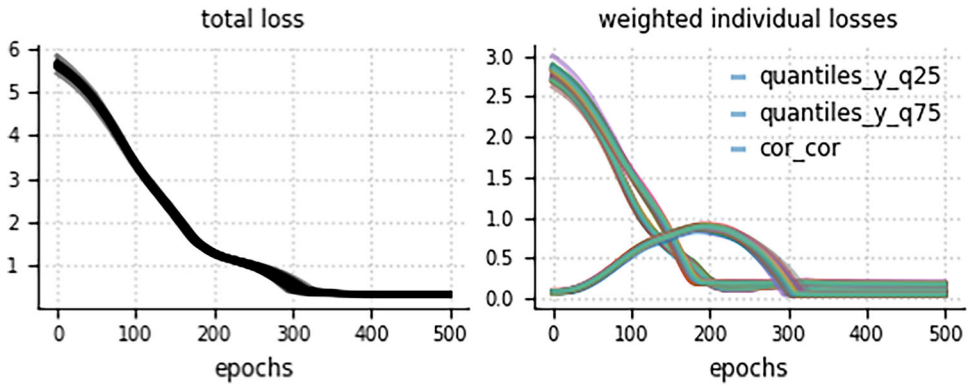


Fig. 4 *Binomial model—Visual convergence check 1*: Updating trajectory across epochs for all 30 replications. On the left, trajectory of the total loss and on the right, trajectory of the weighted individual loss components

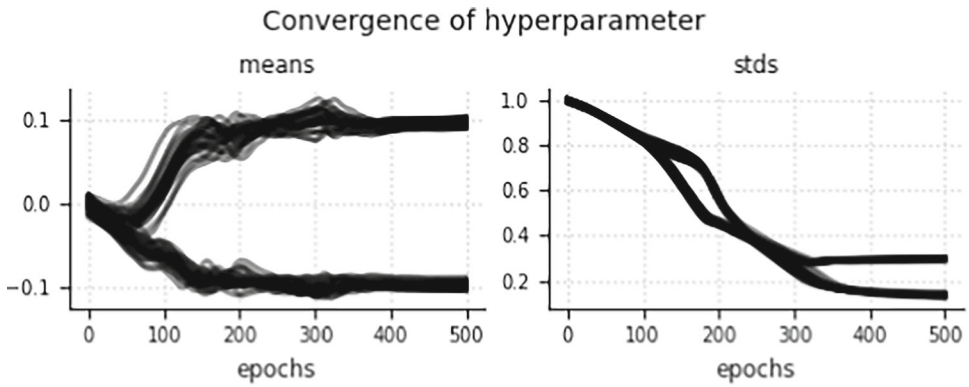


Fig. 5 *Binomial model—Visual convergence check 2*: Updating trajectory across epochs for for all 30 replications. Trajectory of the mean (left plot) and standard deviation (right plot) of the marginal priors

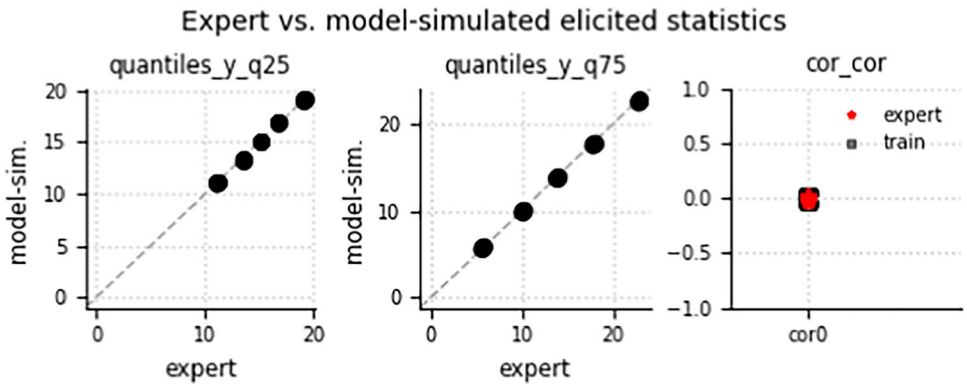


Fig. 6 *Binomial model—Learned elicited statistics for all 30 replications*: The two leftmost plots show the learned quantiles (y-axis) vs. the true quantiles (x-axis) of each replication for the two target quantities, $y \mid x_0$ and $y \mid x_1$. The diagonal line serves as a reference, where points lying on the line indicate a perfect match between learned and true quantiles. The rightmost plot displays the learned (black) vs. true (red) correlations. Given that we assume independence between model parameters in this scenario, the true correlation is zero

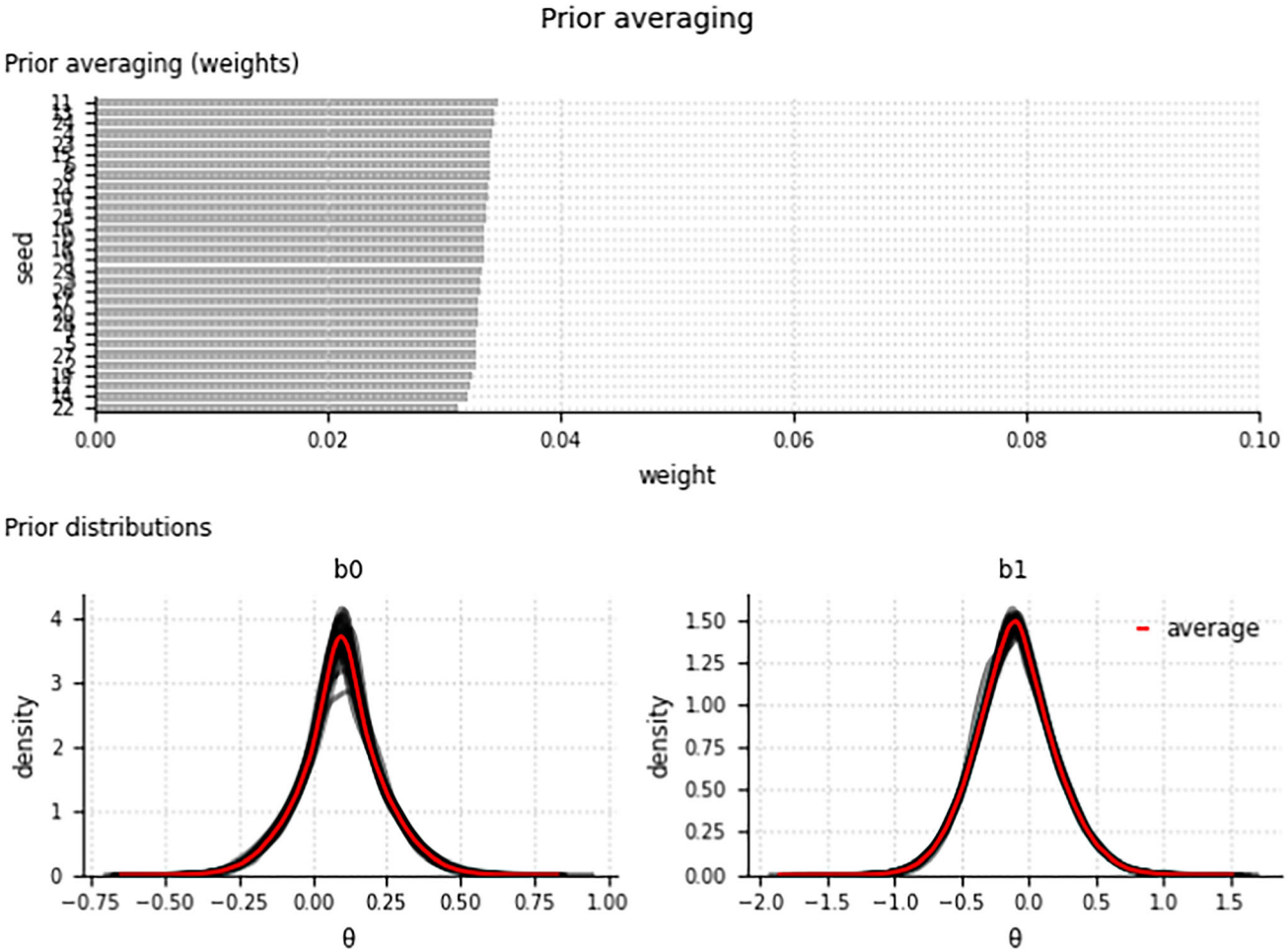


Fig. 7 *Binomial model—Learned marginal priors for all 30 replications*: Upper plot: Computed weights per replication used for model averaging, reflecting the relative performance of each model based on the final total loss. In this scenario, all replications perform equally well, resulting in almost uniform weights. Lower plot: The learned marginal priors for each replication and in red the prior average. In this case, the elicited statistics are sufficiently informative to effectively identify the generative model, resulting in only minor variations in the learned priors across replications

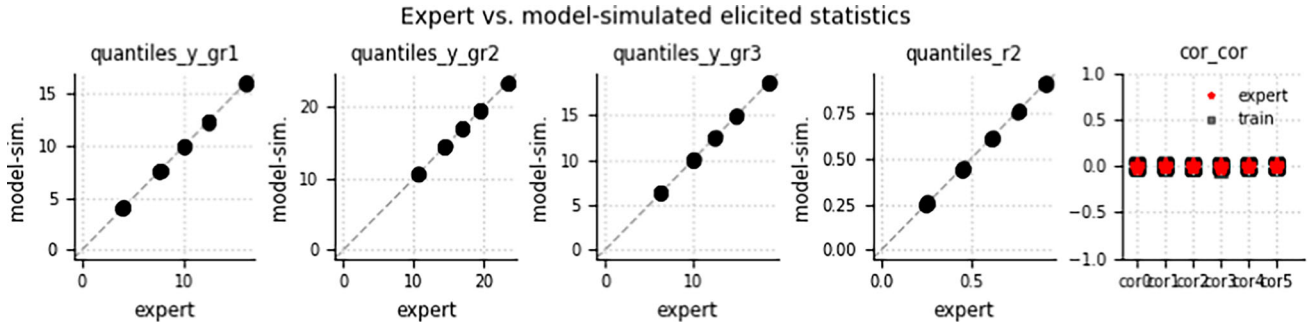


Fig. 8 *Normal model / Scenario 1 (M2) — Learned elicited statistics for all 30 replications*: First three plots (from the left) show the learned quantiles (y-axis) vs. the true quantiles (x-axis) of each replication for the four target quantities $y \mid \text{gr}_i$ with $i = 1, 2, 3$ and R^2. The diagonal line serves as a reference, where points lying on the line indicate a perfect match between learned and true quantiles. The rightmost plot displays the learned (black) vs. true (red) correlations between the model parameters. Given the independence assumption, the true correlation is zero

4.3.1 Scenario 1: Independent Normal Priors

Setup & Elicitation Stage We begin with a brief assessment of the selected elicited statistics, with sensitivity analysis results detailed in Appendix A.2. While the intercept coefficient, β_0, is well-informed by all three target quantities related to the predictive distributions of the groups, the slope coefficients, β_1 and β_2, are each informed by only one target quantity, specifically those corresponding to group 2 and group 3, respectively. Consequently, we anticipate greater difficulty in accurately learning these coefficients. Additionally, we included R^2 among the target quantities to aid the learning algorithm in distinguishing parameter uncertainty from data uncertainty.

Fitting Stage To verify successful convergence, we first examined the absolute slopes across all 30 replications, followed by a detailed visual inspection of the learning trajectories for the loss components and additional quantities of interest (see Appendix A.2 for results). The analysis indicates that the learning process was successful. The model-implied statistics are shown in Figure 8 and demonstrate a strong match between the learned and true elicited statistics, supporting the assumption of successful convergence.

Evaluation Stage The corresponding learned marginal priors of each replication alongside with the model averaging results are depicted in Figure 9. In this scenario, we observe noticeably greater variation in the learned priors across different replications, particularly for β_1 and β_2. This variation is also reflected in the model averaging weights, which now deviate clearly from the uniform distribution observed in Simulation Study 1. These results suggest that while the elicited statistics provide sufficient information to learn the marginal priors of β_0 and σ, they are insufficient to uniquely identify the priors for β_1 and β_2. It is important to emphasize that the only information available to the method for learning the prior distributions is based on the elicited statistics shown in Figure 8. Given this limited set of information, learning a non-parametric joint prior becomes a highly underconstrained problem, where multiple prior distributions can be consistent with the same set of elicited statistics. However, this should not be interpreted as a limitation of the method itself, but rather as a consequence of the limited information available to the method for learning the joint prior.

4.3.2 Scenario 2: Skewed Normal Priors

Setup & Elicitation Stage Scenario 2 differs from the previous one only in the ground truth, where we introduce a skew-normal marginal prior for the two slope coefficients, β_1 and β_2. In this scenario, we examine whether the method can learn this additional skewness property using the same "elicited statistics" as those introduced in the previous case study. The sensitivity analysis (see Appendix A.3) already suggests that there is likely insufficient information to disentangle the variance and skewness components for the two slope coefficients; therefore, we except to observe more variation in the learned prior distributions.

Fitting & Evaluation Stage The convergence diagnostics are depicted in Appendix A.3 and indicate successful convergence. The corresponding match between the model-implied and true "elicited" statistics is shown in Figure 10, demonstrating overall good performance.

Figure 11 shows the corresponding learned marginal priors together with the results from prior averaging. The results indicate relatively stable learning of the prior distributions across all replications. As before, the priors for the slope coefficients exhibit higher variation. Most importantly, however, the method is able to learn the introduced skewness.

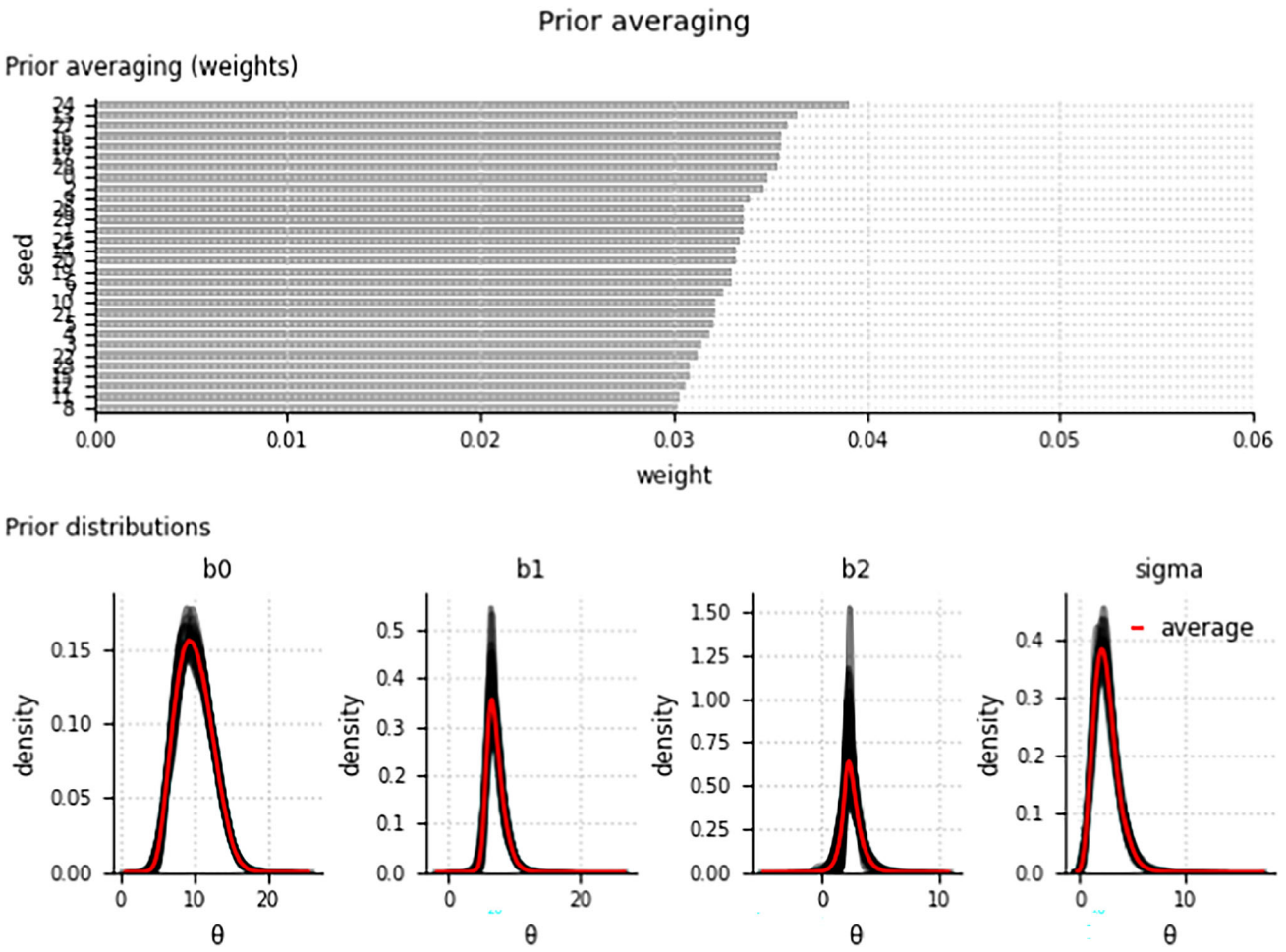


Fig. 9 *Normal model / Scenario 1 (M2) — Learned marginal priors across replications*: Upper plot: Computed weights per replication used for model averaging, reflecting the relative performance of each model based on the final total loss. Lower plot: The learned marginal priors for each replication (in black) and the prior average (in red)

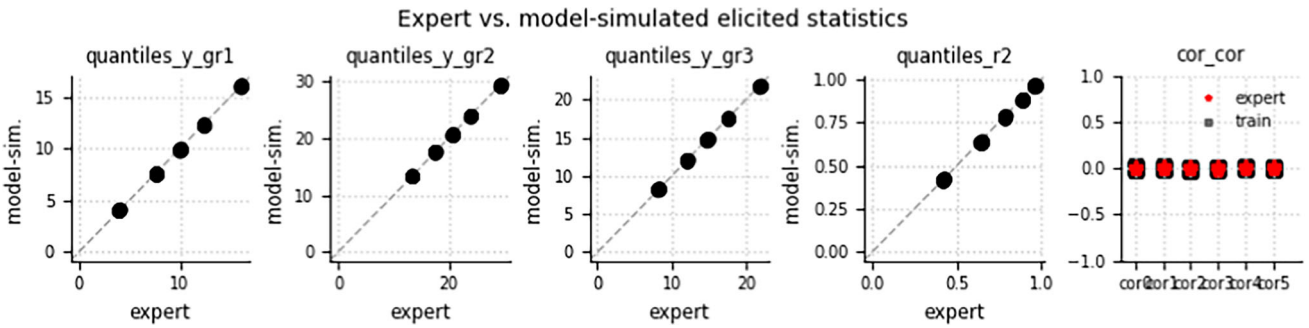


Fig. 10 *Normal model / Scenario 2 (M3) — Learned elicited statistics for all 30 replications*: First four plots from the left depict the learned quantiles (y-axis) vs. the true quantiles (x-axis) of each replication for the four target quantities $y \mid \text{gr}_i$ with $i = 1, 2, 3$ and R^2. The diagonal line serves as a reference, where points lying on the line indicate a perfect match between learned and true quantiles. The rightmost plot displays the learned (black) vs. true (red) correlations between the model parameters. Given the independence assumption, the true correlation is zero

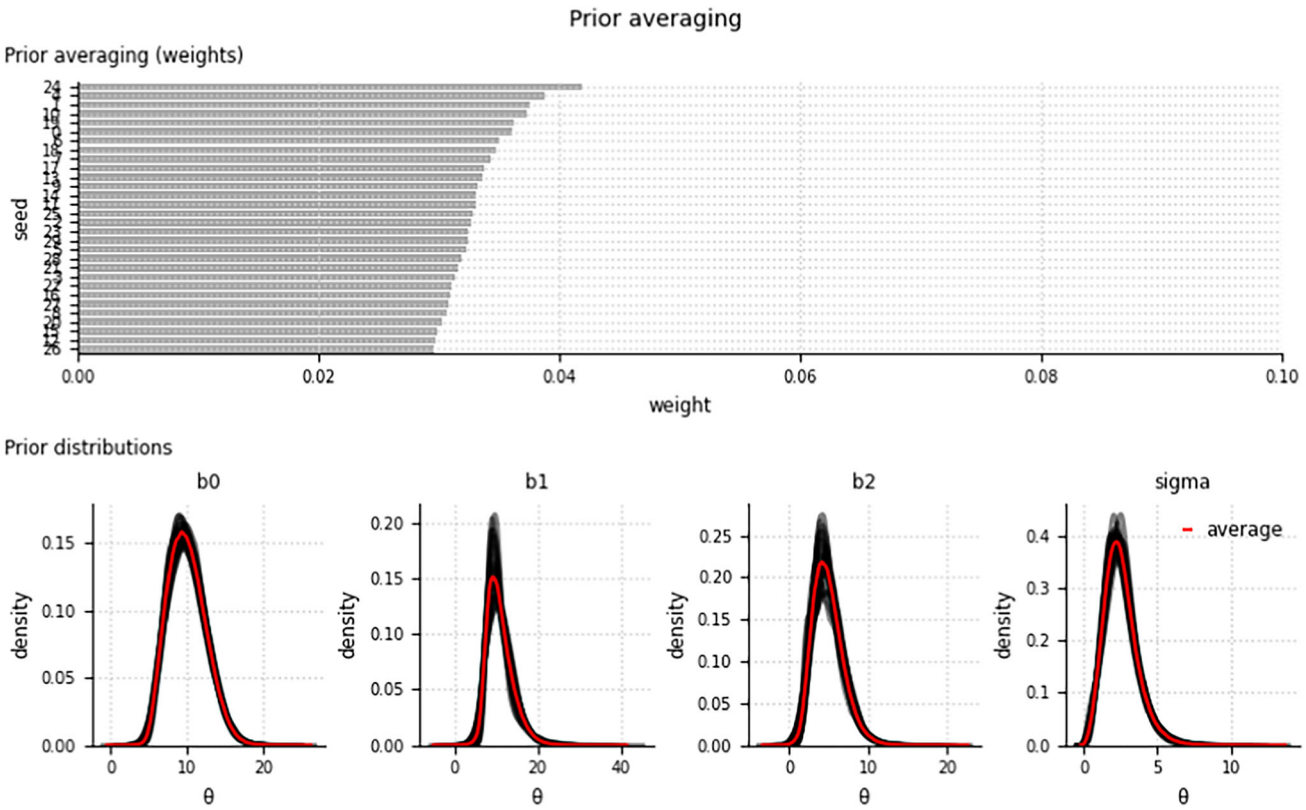


Fig. 11 *Normal model / Scenario 2 (M3) — Learned marginal priors for all 30 replications*: Upper plot: Computed weights per replication used for model averaging, reflecting the relative performance of each model based on the final total loss. Lower plot: The learned marginal priors for each replication (in black) and the prior average (in red)

4.3.3 Scenario 3: Correlated Normal Priors

In this final scenario, we introduced a dependency structure among the regression coefficients, while all other aspects remained identical to Scenario 1 (M1). The results of the sensitivity analysis and the convergence checks are provided in Appendix A.4 and show successful convergence. The model-implied and true "elicited" statistics are shown in Figure 12 and indicate a close match for all elicited statistics.

In Figure 13, the final learned marginal priors for each replication are shown alongside the results from prior averaging. Results show substantial variation in the learned priors for β_2, whereas the priors for all other parameters show relatively high consistency across replications. Since the focus of this case study is specifically on the correlation structure of the regression coefficients, it is informative to also examine the joint prior distribution, which is shown in Figure 14. The joint priors from different replications are shown in different colors. Although there is clear variation in the learned joint priors, the method is able to capture the underlying correlational pattern. Although this result is not particularly surprising, as we provided the method with the exact correlation information between the model parameters, it nonetheless indicates that the method is capable of learning the correlational structure of a joint prior when provided with the relevant information. Therefore, future work is needed to develop elicitation methods that enable the method to obtain relevant information by querying domain experts indirectly.

5 Discussion & Outlook

In this paper, we introduce an extension of our simulation-based framework (Bockting et al. 2024) for learning *non-parametric joint priors*. This extension builds on using normalizing flows (Kobyzev et al. 2020) to learn flexible joint priors for the model parameters. Although we employ normalizing flows in this work, other generative models, such as flow matching (Lipman et al. 2022) or diffusion models (Cao et al. 2024) are also compatible with our framework and can be readily used as a drop-in replacement.

Across four simulation studies, we demonstrated successful learning of joint priors based primarily on interpretable quantities that can be meaningfully provided by domain experts. One exception is the use of correlation information, which the current method requires to reasonably constrain the

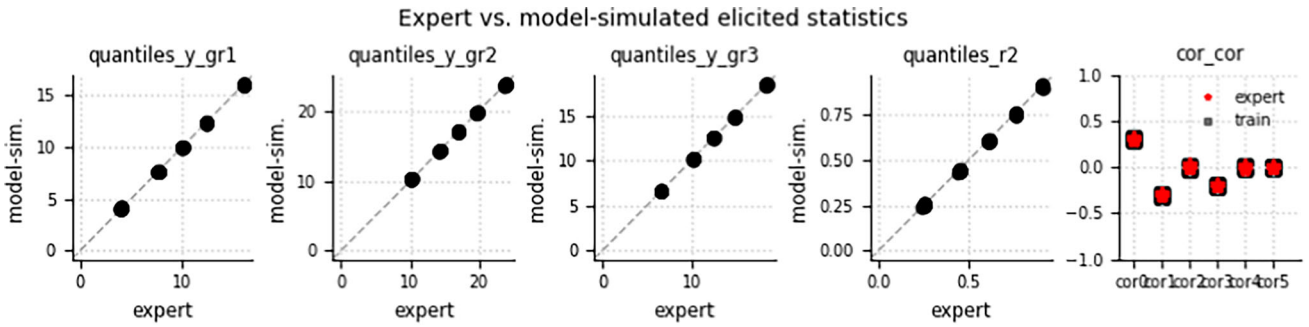


Fig. 12 *Normal model / Scenario 3 (M4) — Learned elicited statistics for all 30 replications*: First four plots show the learned quantiles (y-axis) vs. the true quantiles (x-axis) of each replication for the four target quantities $y \mid \text{gr}_i$ with $i = 1, 2, 3$ and R^2. The diagonal line serves as a reference, where points lying on the line indicate a perfect match between learned and true quantiles. The rightmost plot displays the learned (black) vs. true (red) correlations between the model parameters

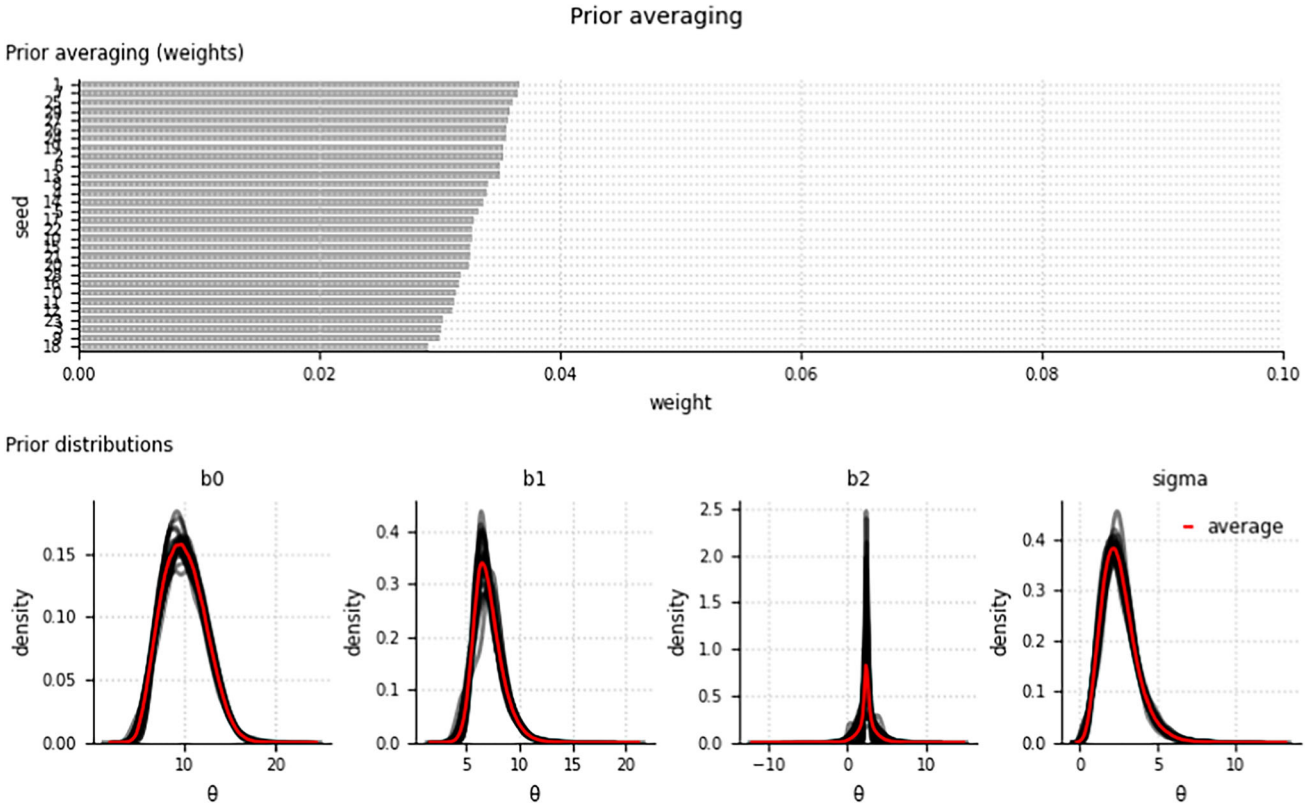


Fig. 13 *Normal model / Scenario 3 (M4) — Learned marginal priors for all 30 replications*: Upper plot: Computed weights per replication used for model averaging, reflecting the relative performance of each model based on the final total loss. Lower plot: The learned marginal priors for each replication (black) and the prior average (red)

optimization problem. Future work is required to integrate in our method an elicitation technique that infers the correlation indirectly from the expert by asking about interpretable quantities. Recently, Mikkola et al. (2024) proposed a method that uses pairwise comparisons or ranking techniques to learn a multivariate prior density, which could serve as an interesting avenue for future research.

With the extension of our originally proposed simulation-based framework to support non-parametric prior distributions, we offer an elicitation framework, implemented in *elicito* (Bockting and Bürkner 2025), which can accommodate various types of elicitation methods within a single, unified structure. This facilitates the development of a stan-

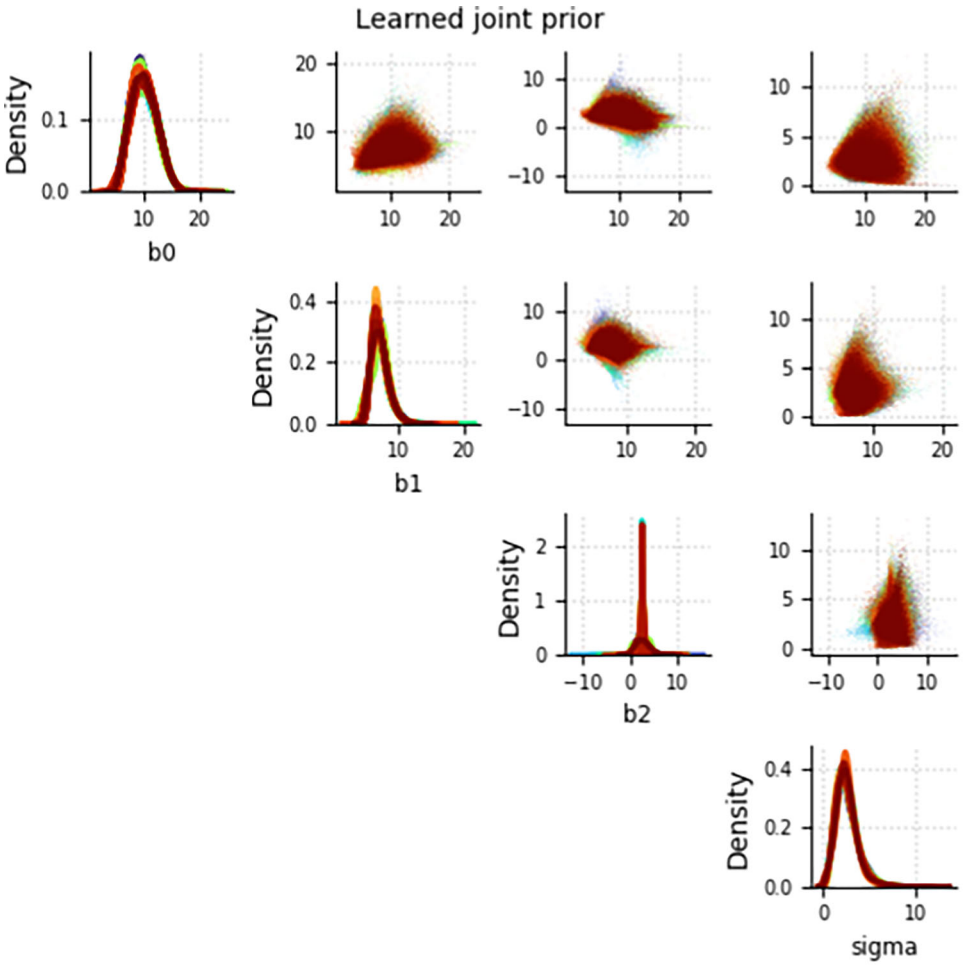


Fig. 14 *Normal model / Scenario 3 (M4) — Learned joint prior for all 30 replications* Each replication is shown in a different color. The correlation structure of the regression coefficients used for learning the prior is $\rho(\beta_0, \beta_1) = 0.3$, $\rho(\beta_1, \beta_2) = -0.2$, and $\rho(\beta_0, \beta_2) = -0.3$. Independence between σ and correlation coefficients is assumed

dardized set of diagnostics and analysis tools, addressing a current gap in the literature (Mikkola et al. 2023).

In addition to accommodating different types of prior distributions, another strength of the framework is that *expert knowledge* can be specified in various ways. This ranges from querying model parameters to observable quantities or any derived statistics, allowing the queried information to be tailored to the specific problem and the expertise of the domain expert. This design choice provides users with maximum flexibility in specifying the set of quantities to be elicited from the domain expert. However, it also places the responsibility on the user to define a valid and appropriate set of quantities to query from the expert (Simpson et al. 2017). We would argue that this flexibility is in general a desirable property of a method, as it avoids imposing specific modeling choices on the user. Thus, instead of requiring that the problem is framed in a predefined way, the method allows users to adapt it to their specific needs (Regenwetter and Cavagnaro 2019; Simpson et al. 2017; Scott 2017). However, in practice, this flexibility is only effective if good recommendations, guidelines, and diagnostics are available to help users define an appropriate set of quantities to query from the expert (Garthwaite et al. 2013).

A significant challenge in this context is assessing whether the chosen set of statistics is *sufficiently informative* for the corresponding model. The goal is not necessarily to identify a unique prior that fully defines the generative model. Instead, the aim is to determine a set of equivalently plausible priors, supporting the assumption that a domain expert is unlikely to have a single "true" prior in mind (Winkler 1967). This introduces at least two additional requirements for an elicitation method. First, it should include an efficient implementation that allows the training algorithm to be executed multiple times with different seeds, enabling the assessment of variability in the learned prior distributions. Second, the elicitation method should ideally offer additional diagnostics or advanced techniques, such as model selection or model averaging, to assist the user (Falconer et al. 2022). In this work, we propose a model averaging method in which the weights calculated for averaging can also serve as diagnostics, helping to identify potential outliers within the set of learned priors. Identifying these edge cases could provide a

valuable starting point for evaluating the plausibility of the learned priors in collaboration with a domain expert.

If the set of learned prior distributions includes degenerate cases, this signals the need for additional constraints. These constraints could involve eliciting further or alternative quantities (i.e., elicited statistics) from the expert and/or adding a regularization term to the total loss function to better guide the learning process. To evaluate the informativeness of the elicited statistics, we use a sensitivity analysis as a preliminary evaluation tool (O'Hagan and Oakley 2004). Additionally, we advocate for employing an oracle (i.e., simulating data from a known ground truth) and conducting multiple training replications with varying random seeds. These strategies provide a solid initial check to determine if the learned priors require further constraints. Overall, this discussion highlights the importance of developing comprehensive tutorials and resources to guide users effectively through the elicitation workflow as a key task for future work.

Other important design choices within our framework that require appropriate default settings include the specification of the discrepancy measure and the selection of the optimization approach, among other aspects. So far, in all simulation studies, we have used the maximum mean discrepancy (MMD, Gretton et al. 2006) to evaluate the discrepancy between the model-implied and expert-elicited statistics. The only exception is the correlation information, for which we employed an L2 loss. One key advantage of the MMD is its flexibility in handling a wide range of statistics, from simple point estimates to complex histogram data. However, a notable limitation of the MMD is its high (i.e., quadratic) computational cost. In contrast, the L2 loss is computationally efficient but lacks the capability to process histogram data. Despite the higher computational cost, the MMD consistently showed very good performance and robustness across all simulation studies so far. Therefore, it appears to be a solid choice as a default discrepancy measure. Future work should study the performance of different loss functions depending on different elicited statistics, with the goal to develop user guidelines for selecting an appropriate loss function.

Furthermore, in all simulation studies conducted so far, we used mini-batch SGD to optimize the prior network parameters. Given the generally good performance observed, we believe that this optimization method is a solid default choice. However, we aim to explore additional optimization approaches within our framework, such as Bayesian optimization (as employed by Manderson and Goudie (2023)). Comparing these approaches in terms of performance and the level of hyperparameter tuning required could offer valuable insights into the conditions under which optimization method might be more advantageous than the other.

Finally, while our current methodology learns prior distributions based on expert data, it does currently not account for additional 'meta' uncertainty that arises during the elicitation process. This includes uncertainties introduced by the elicitation technique itself or the expert's uncertainty regarding the 'true' quantity (see e.g., Falconer et al. 2022, 2023; Stefan et al. 2022). We acknowledge that these are important sources of information that should be incorporated in future developments of our method. One promising approach is to view this problem from a Bayesian perspective and treat the proposed workflow as a Bayesian inference problem. This idea was already proposed by Mikkola et al. (2023) and referred to as the *supra-Bayesian approach*. We believe this is an important and valuable direction for the next steps in our elicitation method.

Appendix A Simulation Studies

A.1 Training time

The median training time across the 30 replications (± sd of training time):

- **Binomial model**: 6.87 minutes (± 1.45).
- **Independent normal scenario**: 10.18 minutes (± 1.68).
- **Skewed normal scenario**: 14.45 minutes (± 2.80).
- **Correlated normal scenario**: 10.10 minutes (± 0.17).

A.2 Independent normal scenario

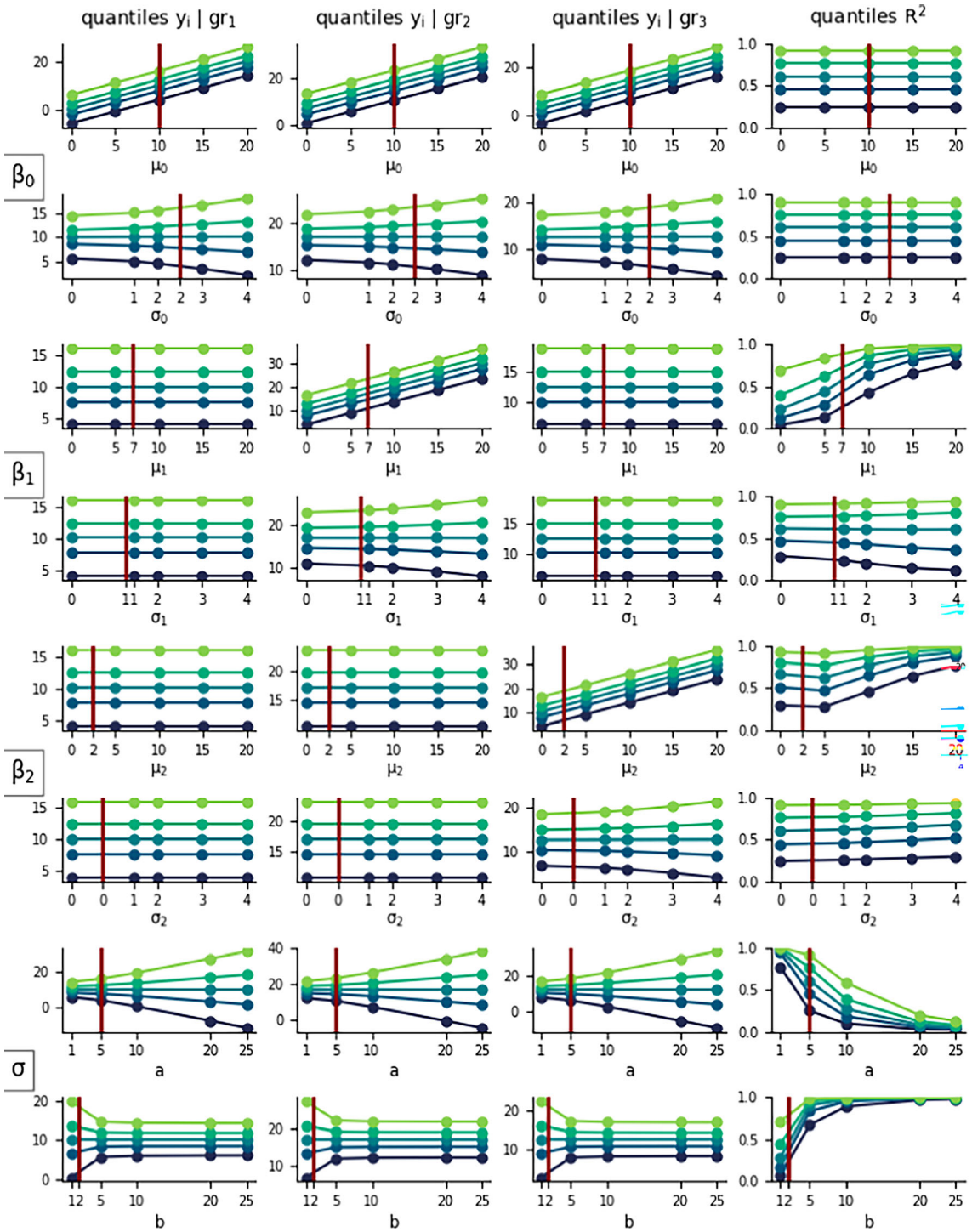


Fig. 15 Normal model / Scenario 1 (M2) — Sensitivity Analysis: Rows represent hyperparameters of each model parameter. Columns represent elicited statistics: five quantiles for $y \mid \text{gr}_i$ with $i = 1, 2, 3$ and for R^2. Quantiles are depicted in different colors. In each row, the corresponding hyperparameter, is varied across the range shown on the x-axis, while all other hyperparameters are held constant at their true values, indicated by the red vertical line

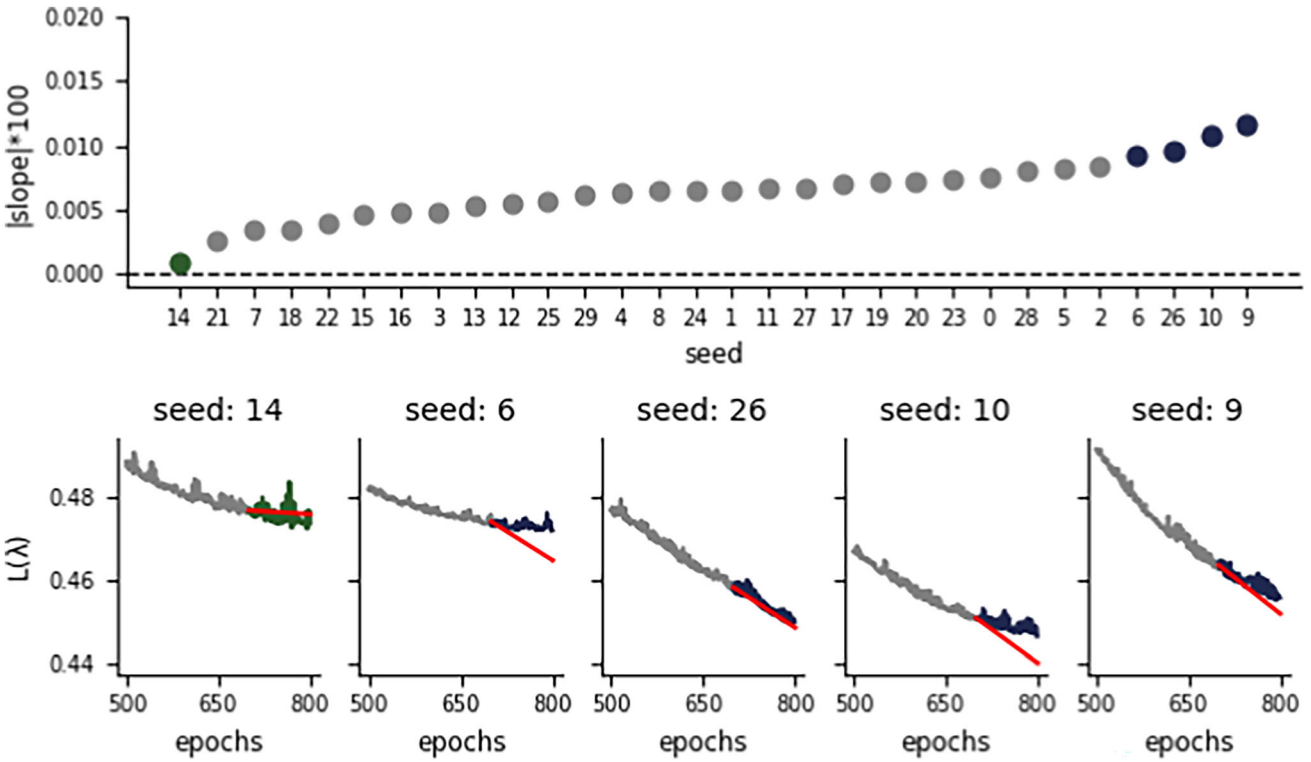
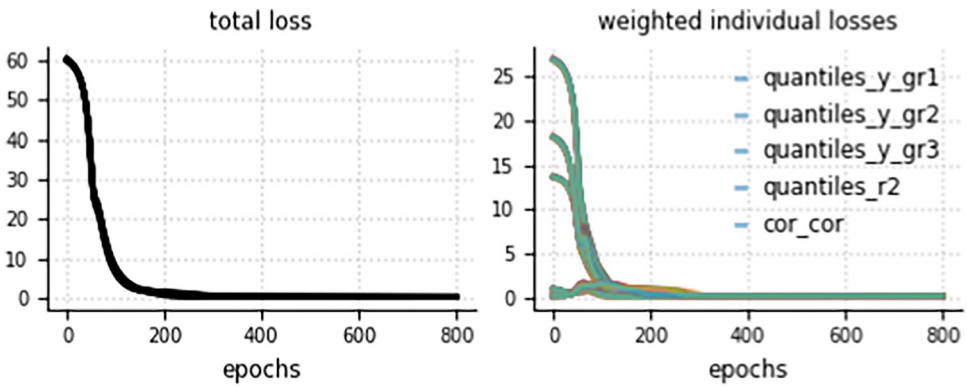


Fig. 16 *Normal model / Scenario 1 (M2) — Preliminary Convergence Check*: Upper plot displays the absolute slope of the total loss trajectory (scaled by a factor of 100) over the last 100 epochs on the y-axis, with each replication shown along the x-axis. The replications (i.e., seeds) are arranged in ascending order based on the magnitude of the absolute slope. The best-performing model is seed 14, while the worst-performing models are seeds 6, 26, 10, and 9. The lower plot illustrates the loss trajectories of the best- and worst-performing seeds over the last 300 epochs. The loss updates for the final 100 epochs are highlighted in green (best model) and blue (worst models), with a red line segment indicating the respective slope

Fig. 17 *Normal model / Scenario 1 (M2) — Visual convergence check 1*: Updating trajectory across epochs for the total loss (left) and the individual (weighted) loss components (right)



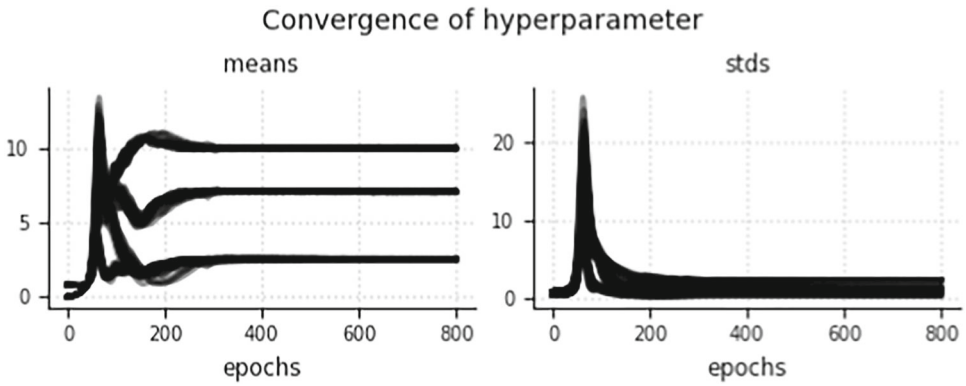


Fig. 18 *Normal model / Scenario 1 (M2) — Visual convergence check 2*: Updating trajectory across epochs for the mean (left) and standard deviation (right) of the final learned marginal priors for all 30 replications

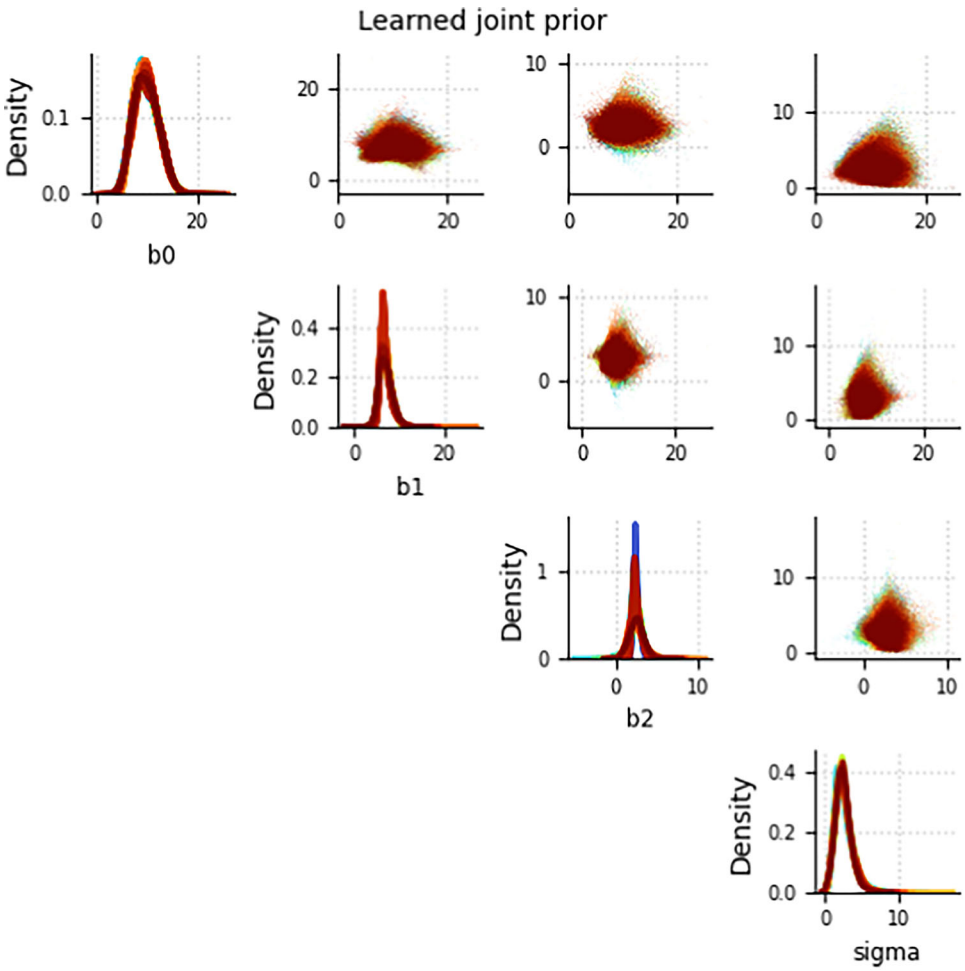


Fig. 19 *Normal model / Scenario 1 (M2) — Joint prior*: Each replication is visualized in a different color

A.3 Skewed normal scenario

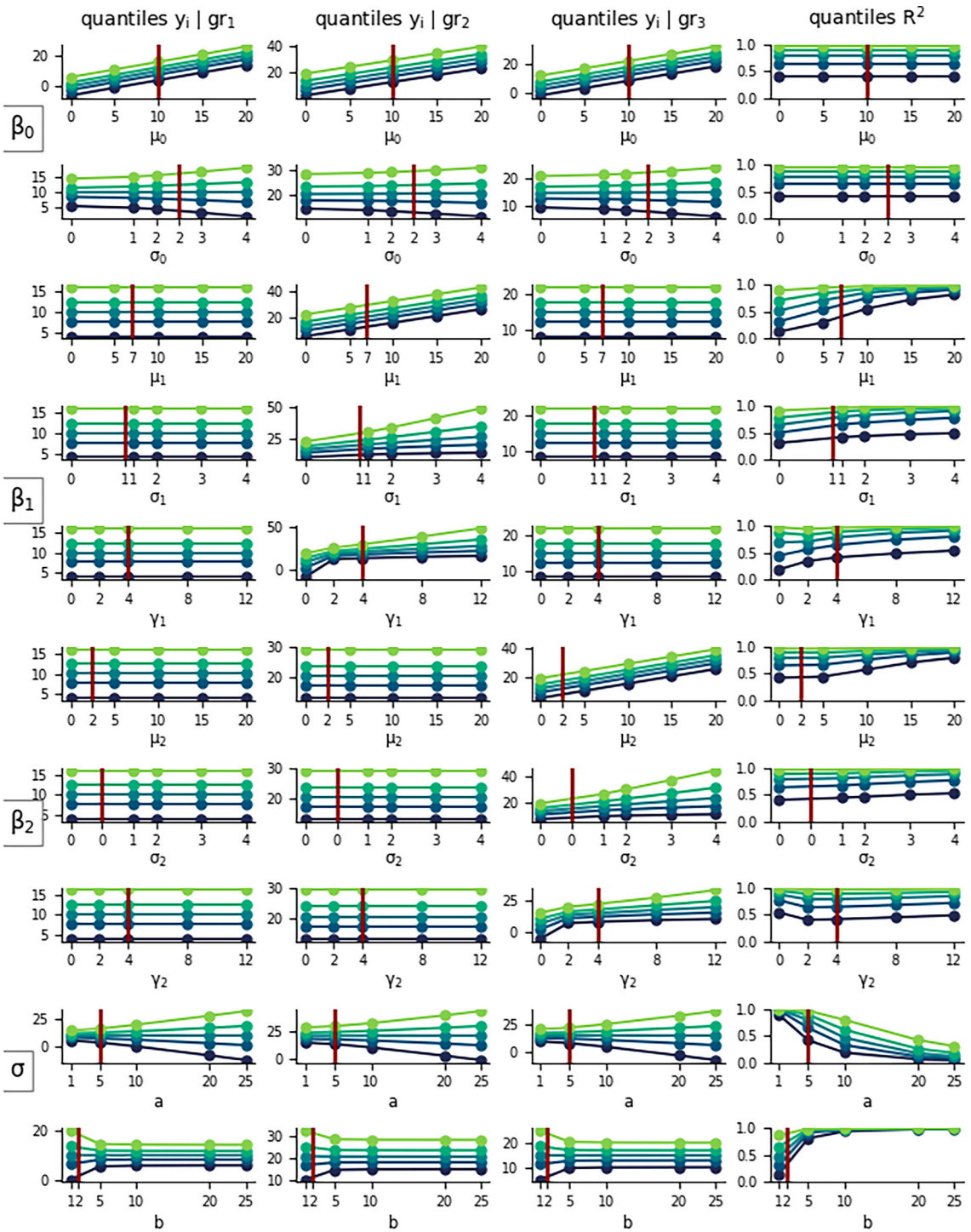


Fig. 20 *Normal model / Scenario 2 (M3) — Sensitivity Analysis*: Rows represent hyperparameters of each model parameter. Columns represent elicited statistics: five quantiles for $y \mid \text{gr}_i$ with $i = 1, 2, 3$ and for R^2. Quantiles are depicted in different colors. In each row, the corresponding hyperparameter, is varied across the range shown on the x-axis, while all other hyperparameters are held constant at their true values, indicated by the red vertical line

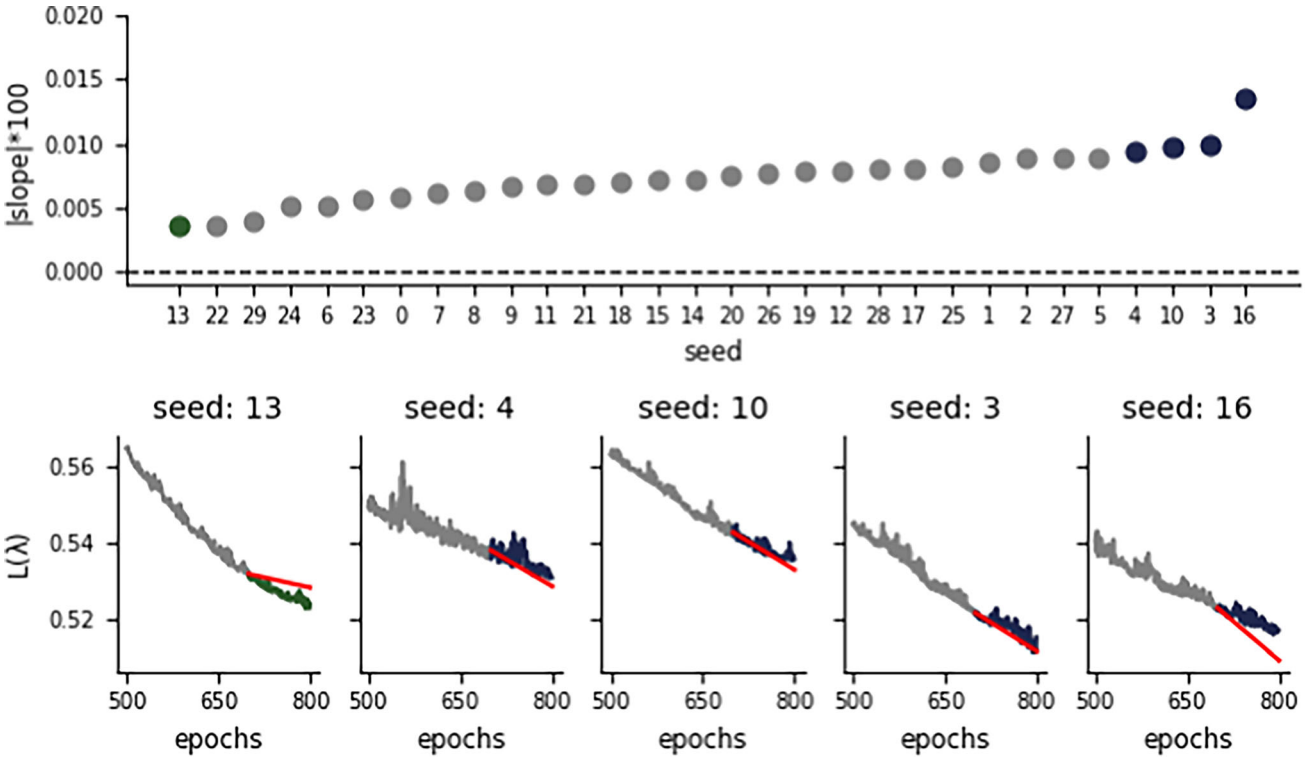


Fig. 21 *Normal model / Scenario 2 (M3) — Preliminary Convergence Check for all 30 replications*: The upper plot displays the absolute slope of the total loss trajectory (scaled by a factor of 100) over the last 100 epochs on the y-axis, with each replication shown along the x-axis. The replications (i.e., seeds) are arranged in ascending order based on the magnitude of the absolute slope. The best-performing model is seed 13, while the four worst-performing models are seeds 4, 10, 3, and 16. The lower plot illustrates the loss trajectories of the best- and worst-performing seeds over the last 300 epochs. The loss updates for the final 100 epochs are highlighted in green (best seed) or blue (worst seeds). The red line segment indicates the respective slope

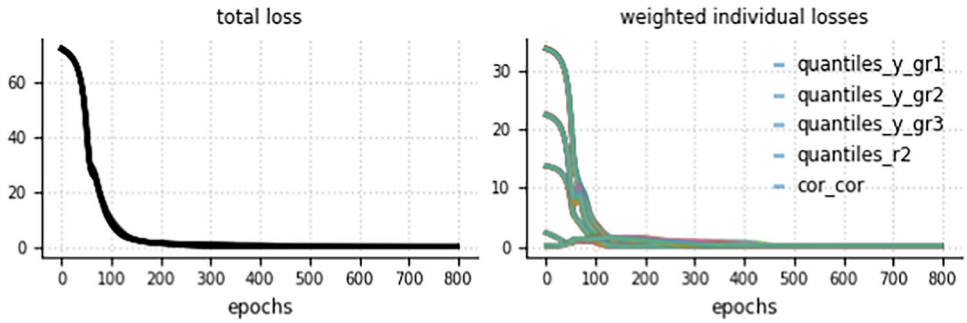


Fig. 22 *Normal model / Scenario 2 (M3) — Visual convergence check 1*: Updating trajectory across epochs for the total loss (left) and the individual (weighted) loss components (right)

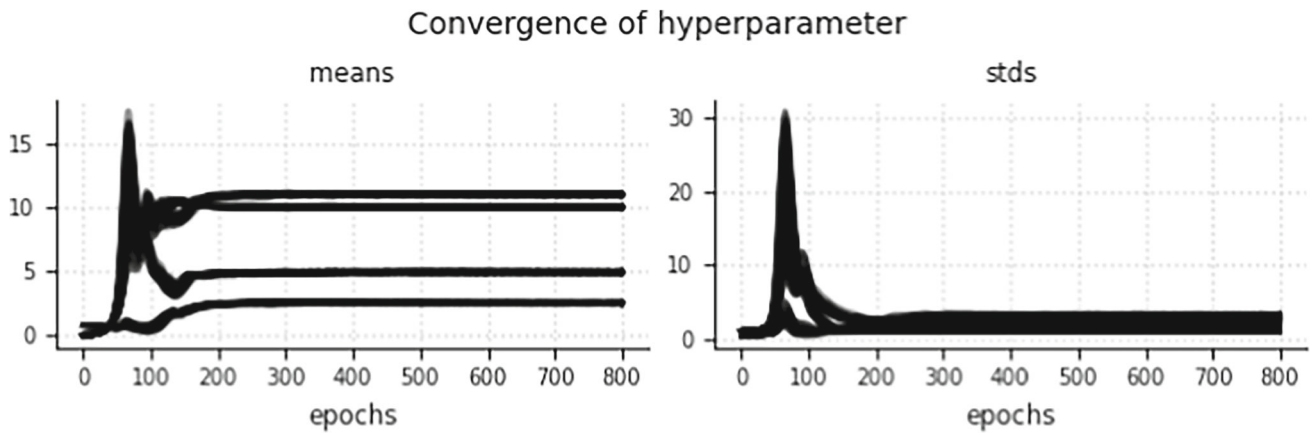
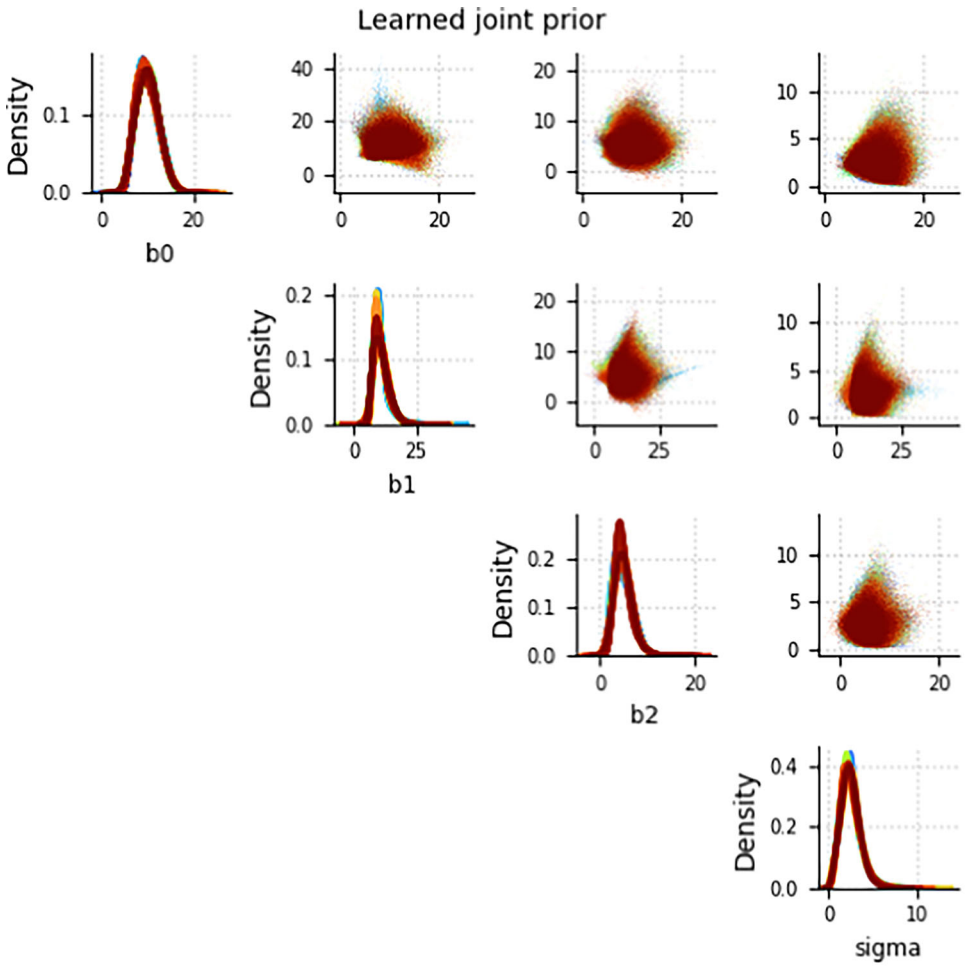


Fig. 23 *Normal model / Scenario 2 (M3) — Visual convergence check 2*: Updating trajectory across epochs for the mean (left) and standard deviation (right) of the marginal priors

Fig. 24 *Normal model / Scenario 2 (M3) — Visual convergence check 2*: Joint prior distribution. Each replication is visualized in a different color



A.4 Correlated normal scenario

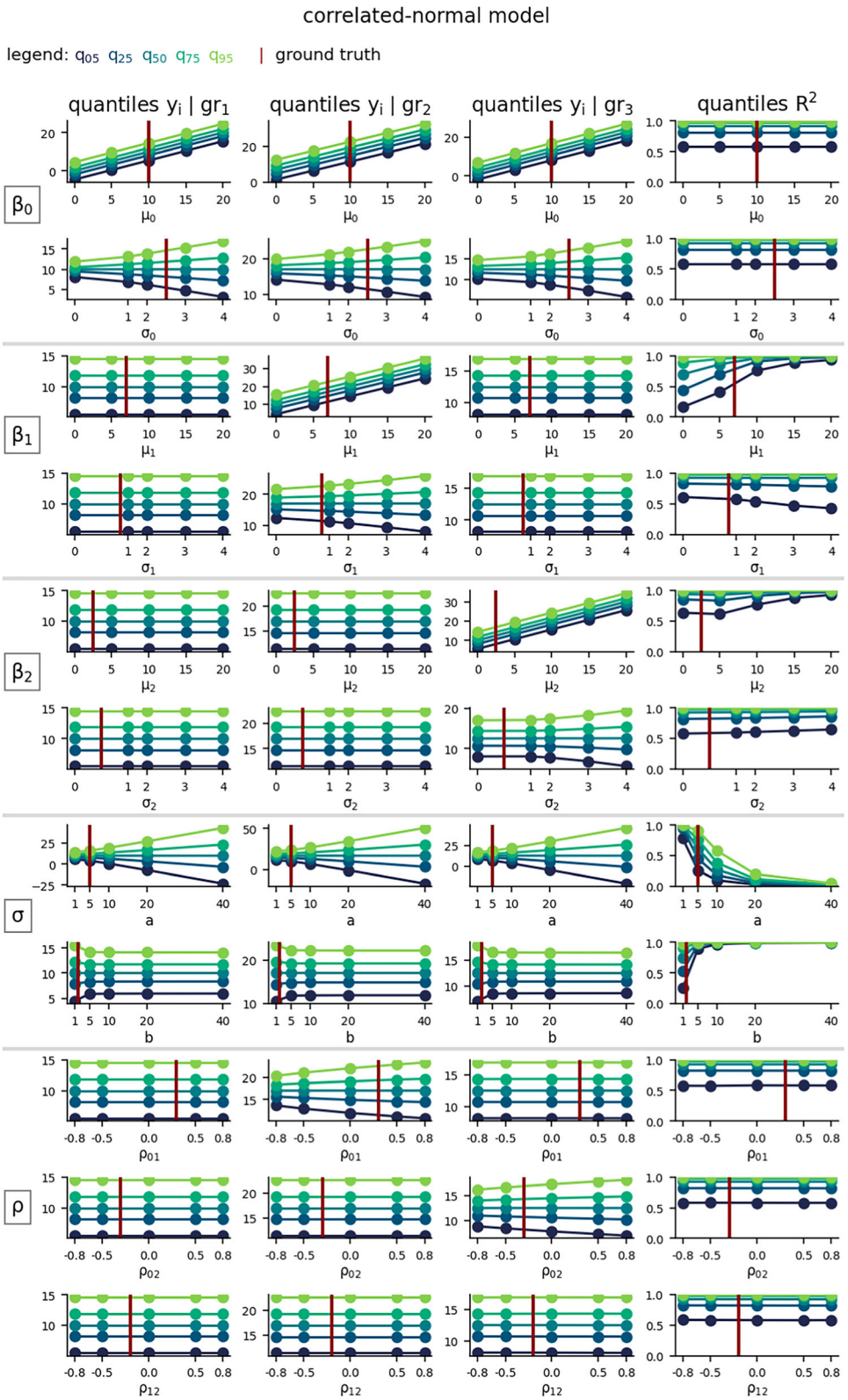


Fig. 25 *Normal model / Scenario 3 (M4) — Sensitivity Analysis*: Rows represent hyperparameters of each model parameter. Columns represent elicited statistics: five quantiles for $y \mid \text{gr}_i$ with $i = 1, 2, 3$ and for R^2. Quantiles are depicted in different colors. In each row, the corresponding hyperparameter is varied across the range shown on the x-axis, while all other hyperparameters are held constant at their true values, indicated by the red vertical line

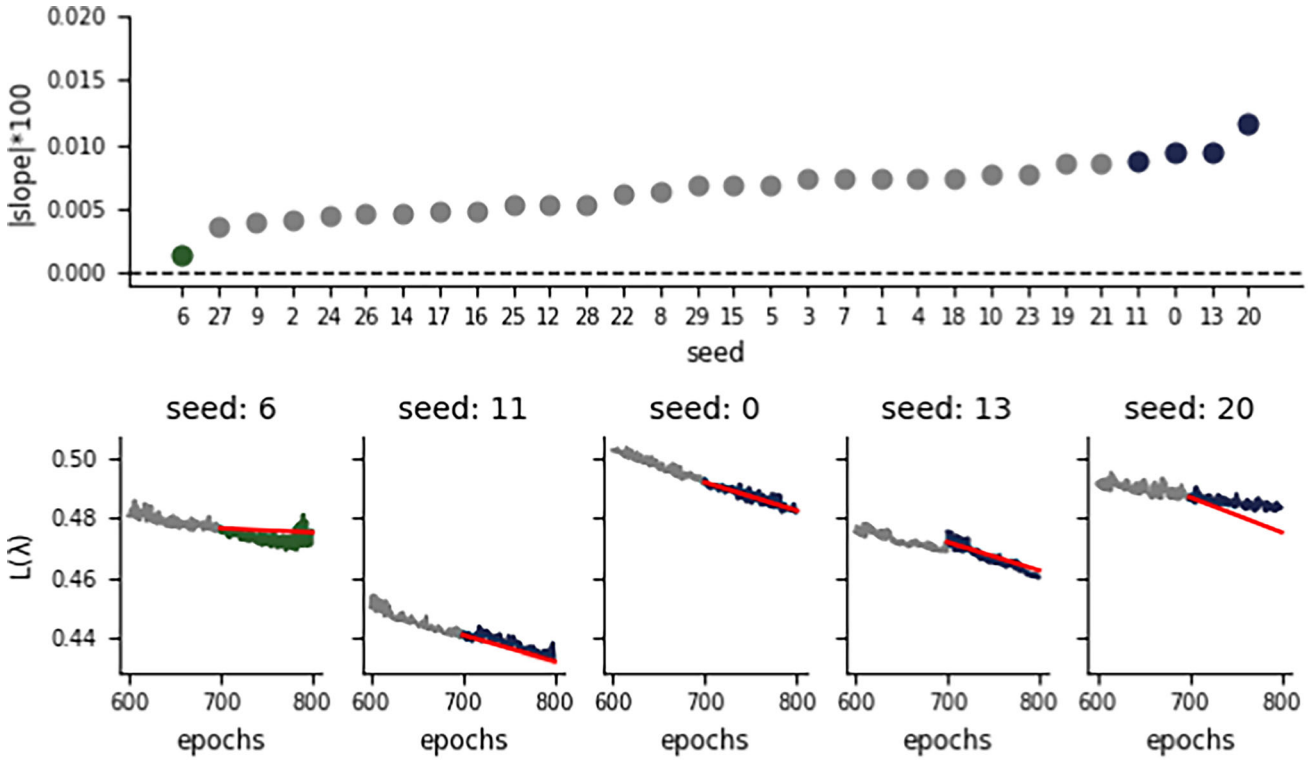


Fig. 26 *Normal model / Scenario 3 (M4) — Preliminary Convergence Check for all 30 replications*: The upper plot displays the absolute slope of the total loss trajectory (scaled by a factor of 100) over the last 100 epochs on the y-axis, with each replication shown along the x-axis. The replications (i.e., seeds) are arranged in ascending order based on the magnitude of the absolute slope. The best-performing model is seed 6, while the worst-performing models are seeds 11, 0, 13, and 20. The lower plot illustrates the loss trajectories of the best- and worst-performing seeds over the last 200 epochs. The loss updates for the final 100 epochs are highlighted in green for the best model and in blue for the worst models, with a red line segment indicating the respective slope

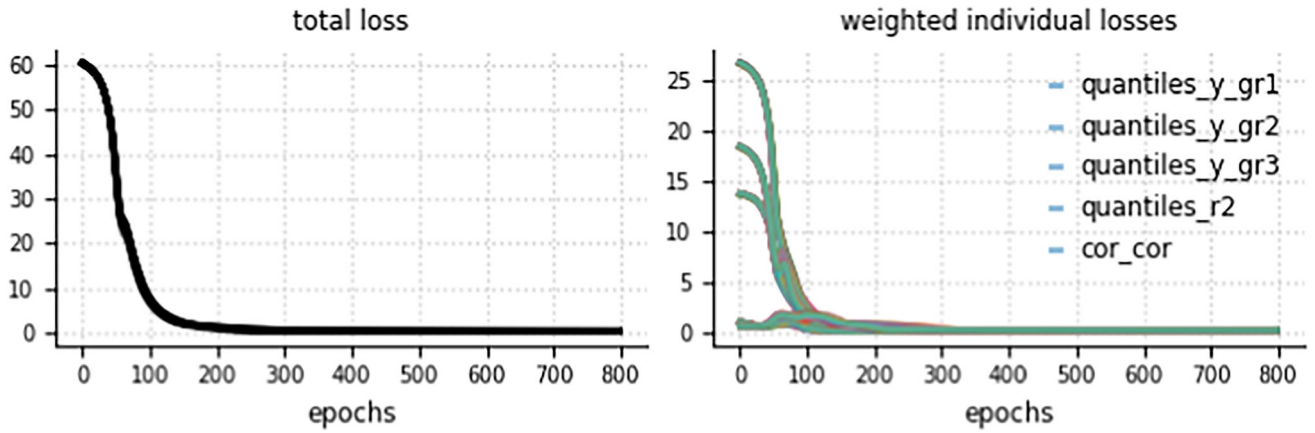


Fig. 27 *Normal model / Scenario 3 (M4) — Visual convergence check 1*: Updating trajectory across epochs for the total loss (left) and the individual (weighted) loss components (right)

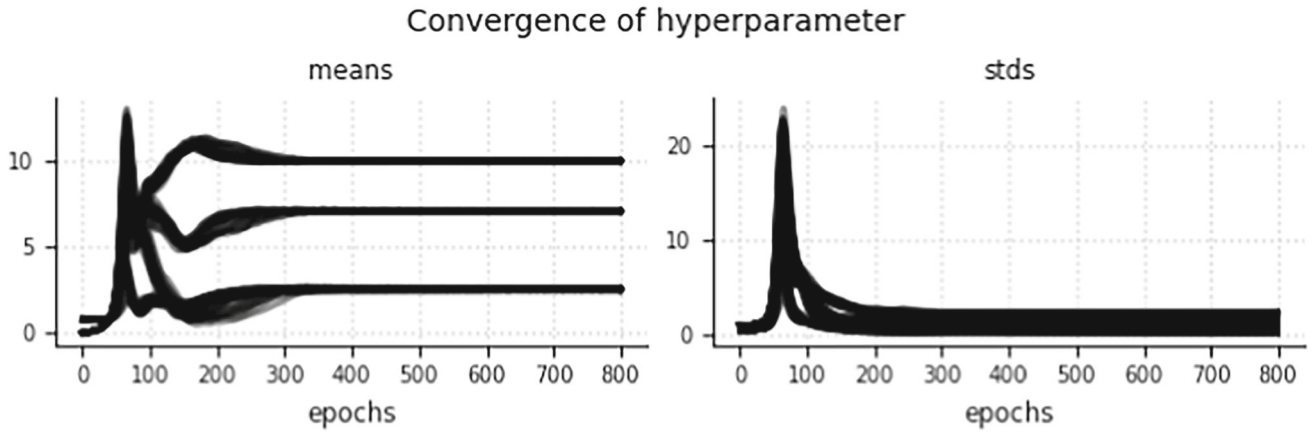


Fig. 28 *Normal model / Scenario 3 (M4) — Visual convergence check 1*: Updating trajectory across epochs for the mean (left) and standard deviation (right) of the marginal priors

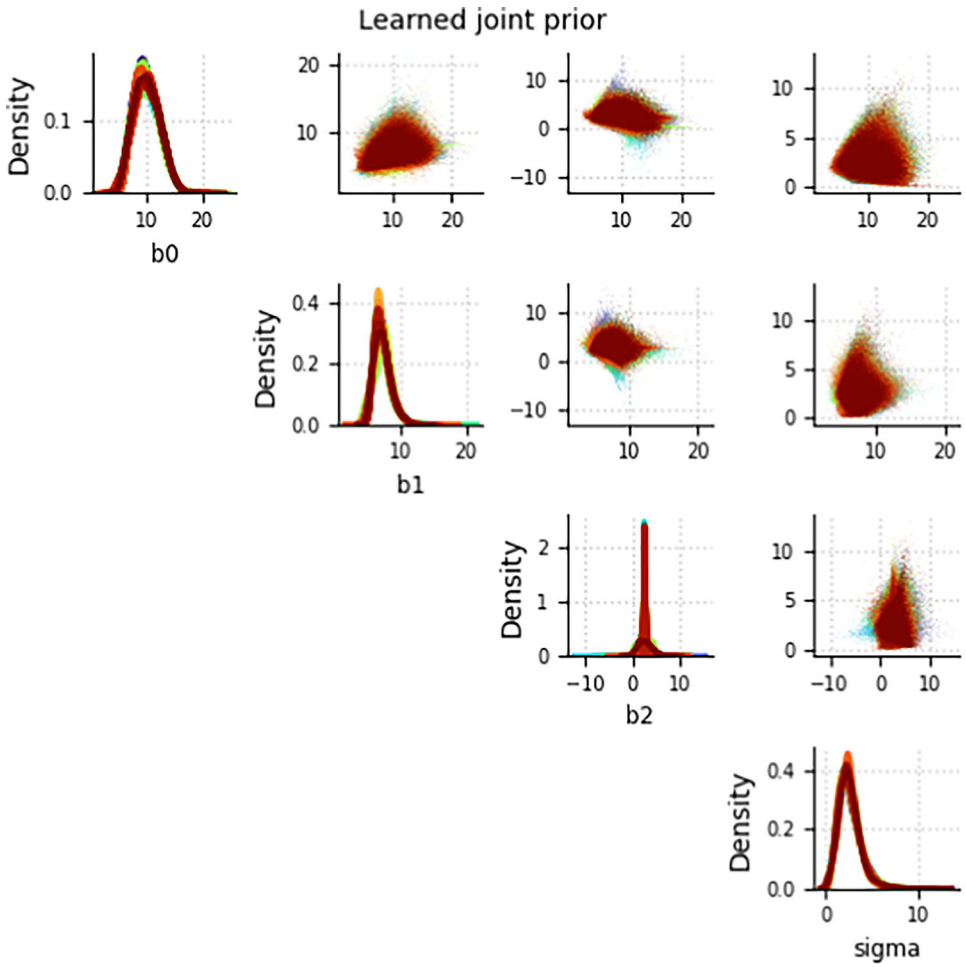


Fig. 29 Normal model / Scenario 3 (M4) — Visual convergence check 2: Joint prior distribution. Each replication is visualized in a different color

Acknowledgements We thank Dmytro Perepolkin and Luna Fazio for their valuable comments and suggestions that greatly improved the manuscript.

Author Contributions Conceptualization: PB, SR, FB, Methodology: PB, FB, Software: FB, Writing (Original Draft): FB, Writing (Editing): FB, SR, PB, Writing (Review): FB, PB, Visualization: FB, Supervision: PB, Funding acquisition: PB

Funding Not applicable.

Data Availability The simulation results are available on OSF at https://osf.io/xrzh6/.

Declarations

Competing interests The authors have no competing interests to declare that are relevant to the content of this article.

Consent to participate Not applicable.

Consent for publication Not applicable.

Code availability Code for running the simulations and plotting the results is available on GitHub at https://github.com/florence-bockting/non-parametric-prior. Simulations and most plots are based on/provided by our Python package *elicito* (v0.3.1 Bockting and Bürkner 2025). To ensure the reproducibility of our results, the specific version of *elicito* used in this study has been archived on Zenodo and is accessible via the following DOI https://doi.org/10.5281/zenodo.15241853.

Materials availability Supplementary material is available on GitHub at https://github.com/florence-bockting/non-parametric-prior.

References

Abadi, M., Barham, P., Chen, J., et al.: TensorFlow: A system for large-scale machine learning. In: 12th USENIX symposium on operating systems design and implementation (OSDI 16), pp 265–283, (2016) https://doi.org/10.5555/3026877.3026899

Betancourt, M.: Towards a principled bayesian workflow. Blog Post, (2020) https://betanalpha.github.io/assets/case_studies/principled_bayesian_workflow.html

Bischof, R., Kraus, M.: Multi-objective loss balancing for physics-informed deep learning. (2021) arXiv preprint https://doi.org/10.48550/arXiv.2110.09813

Bissiri, P.G., Holmes, C.C., Walker, S.G.: A general framework for updating belief distributions. J. R. Stat. Soc. Series. B Stat. Methodol **78**(5), 1103–1130 (2016). https://doi.org/10.1111/rssb.12158

Bockting, F., Bürkner, PC.: elicito - a tool for expert prior elicitation. (2025) https://github.com/florence-bockting/elicito

Bockting, F., Radev, S.T., Bürkner, P.C.: Simulation-based prior knowledge elicitation for parametric bayesian models. Sci. Rep. **14**(1), 17330 (2024). https://doi.org/10.1038/s41598-024-68090-7

Bond-Taylor, S., Leach, A., Long, Y., et al.: Deep generative modelling: A comparative review of vaes, gans, normalizing flows, energy-based and autoregressive models. IEEE Trans. Pattern Anal. Mach. Intell. **44**(11), 7327–7347 (2021). https://doi.org/10.1109/TPAMI.2021.3116668

Buckland, ST., Burnham, KP., Augustin, NH.: Model selection: an integral part of inference. Biometrics pp 603–618. (1997) https://doi.org/10.2307/2533961

Cao, H., Tan, C., Gao, Z., et al.: A survey on generative diffusion models. IEEE Trans. Knowl. Data Eng. **36**(7), 2814–2830 (2024). https://doi.org/10.1109/TKDE.2024.3361474

Claeskens, G., Hjort, NL.: Model selection and model averaging. Cambridge books (2008)

Clemen, R.T., Fischer, G.W., Winkler, R.L.: Assessing dependence: Some experimental results. Manage. Sci. **46**(8), 1100–1115 (2000). https://doi.org/10.1287/mnsc.46.8.1100.12023

Cooke, R.M.: Experts in uncertainty: opinion and subjective probability in science. Oxford University Press, USA (1991)

Crawshaw, M.: Multi-task learning with deep neural networks: A survey. (2020) arXiv preprint https://doi.org/10.48550/arXiv.2009.09796

Crowder, M.: Bayesian priors based on a parameter transformation using the distribution function. Ann. Inst. Stat. Math. **44**, 405–416 (1992). https://doi.org/10.1007/BF00050695

Denham, R., Mengersen, K.: Geographically assisted elicitation of expert opinion for regression models. Bayesian Anal. **2**(1), 99–136 (2007). https://doi.org/10.1214/07-BA205

Depaoli, S., Winter, S.D., Visser, M.: The importance of prior sensitivity analysis in bayesian statistics: demonstrations using an interactive shiny app. Front. Psychol. **11**, 608045 (2020). https://doi.org/10.3389/fpsyg.2020.608045

Dickey, J., Lindley, D.V., Press, S.J.: Bayesian estimation of the dispersion matrix of a multivariate normal distribution. Commun. Stat. Theory Methods **14**(5), 1019–1034 (1985). https://doi.org/10.1080/03610928508828960

Dillon, JV., Langmore, I., Tran, D., et al.: Tensorflow distributions. arXiv preprint (2017) https://doi.org/10.48550/arXiv.1711.10604

Dinh, L., Krueger, D., Bengio, Y.: Nice: Non-linear independent components estimation. In: ICLR Workshop, (2014) https://doi.org/10.48550/arXiv.1410.8516

Dinh, L., Sohl-Dickstein, J., Bengio, S.: Density estimation using real nvp. (2016) arXiv preprint https://doi.org/10.48550/arXiv.1605.08803

Draxler, F., Wahl, S., Schnörr, C., et al.: On the universality of coupling-based normalizing flows. (2024) arXiv preprint https://doi.org/10.48550/arXiv.2402.06578

Durkan, C., Bekasov, A., Murray, I., et al.: Neural spline flows. Adv. Neural. Inf. Process. Syst. **32**,(2019). https://doi.org/10.5555/3454287.3454962

European Food Safety Authority: Guidance on expert knowledge elicitation in food and feed safety risk assessment. EFSA J. **12**(6), 3734 (2014). https://doi.org/10.2903/j.efsa.2014.3734

Falconer, J.R., Frank, E., Polaschek, D.L., et al.: Methods for eliciting informative prior distributions: A critical review. Decis. Anal. **19**(3), 189–204 (2022). https://doi.org/10.1287/deca.2022.0451

Falconer, J.R., Frank, E., Polaschek, D.L., et al.: Eliciting informative priors by modeling expert decision making. Decis. Anal. **21**(2), 77–90 (2023). https://doi.org/10.1287/deca.2023.0046

Fernández, C., Steel, M.F.: On bayesian modeling of fat tails and skewness. J. Am. Stat. Assoc. **93**(441), 359–371 (1998). https://doi.org/10.2307/2669632

Feydy, J., Séjourné, T., Vialard, FX., et al.: Interpolating between optimal transport and mmd using sinkhorn divergences. In: The 22nd International Conference on Artificial Intelligence and Statistics, PMLR, pp 2681–2690, (2019) https://proceedings.mlr.press/v89/feydy19a.html

Figurnov, M., Mohamed, S., Mnih, A.: Implicit reparameterization gradients. Adv. Neural. Inf. Process. Syst. **31**,(2018). https://doi.org/10.5555/3326943.3326984

Gabry, J., Simpson, D., Vehtari, A., et al.: Visualization in bayesian workflow. J. R. Stat. Soc. Ser. A Stat. Soc **182**(2), 389–402 (2019). https://doi.org/10.1111/rssa.12378

Garthwaite, P.H., Kadane, J.B., O'Hagan, A.: Statistical methods for eliciting probability distributions. J. Am. Stat. Assoc. **100**(470), 680–701 (2005). https://doi.org/10.1198/016214505000000105

Garthwaite, P.H., Al-Awadhi, S.A., Elfadaly, F.G., et al.: Prior distribution elicitation for generalized linear and piecewise-linear models. J. Appl. Stat. **40**(1), 59–75 (2013). https://doi.org/10.1080/02664763.2012.734794

Gelman, A., Simpson, D., Betancourt, M.: The prior can often only be understood in the context of the likelihood. Entropy **19**(10), 555 (2017). https://doi.org/10.3390/e19100555

Gelman, A., Goodrich, B., Gabry, J., et al.: R-squared for bayesian regression models. Am. Stat. pp 307–309. (2019) https://doi.org/10.1080/00031305.2018.1549100

Gelman, A., Vehtari, A., Simpson, D., et al.: Bayesian workflow. (2020) arXiv preprint https://doi.org/10.48550/arXiv.2011.01808

Gokhale, D., Press, S.J.: Assessment of a prior distribution for the correlation coefficient in a bivariate normal distribution. J. R. Stat. Soc. Ser. A Stat. Soc. **145**(2), 237–249 (1982). https://doi.org/10.2307/2981537

Goodfellow, I., Bengio, Y., Courville, A.: Deep learning. MIT press (2016)

Gosling, J.P.: Shelf: the sheffield elicitation framework, pp. 61–93. The science and art of structuring judgement, Elicitation (2018)

Gretton, A., Borgwardt, K., Rasch, M., et al.: A kernel method for the two-sample-problem. Adv. Neural. Inf. Process. Syst. **19**,(2006). https://doi.org/10.5555/2976456.2976521

Harris, C.R., Millman, K.J., van der Walt, S.J., et al.: Array programming with NumPy. Nature **585**(7825), 357–362 (2020). https://doi.org/10.1038/s41586-020-2649-2

Hartmann, M., Agiashvili, G., Bürkner, P., et al.: Flexible prior elicitation via the prior predictive distribution. In: Peters J, Sontag D (eds) Proc.Conf.UAI, PMLR, pp 1129–1138, (2020) https://proceedings.mlr.press/v124/hartmann20a.html

Johnson, S.R., Tomlinson, G.A., Hawker, G.A., et al.: Methods to elicit beliefs for Bayesian priors: a systematic review. J. Clin. Epidemiol. **63**(4), 355–369 (2010). https://doi.org/10.1016/j.jclinepi.2009.06.003

Kadane, J., Wolfson, L.J.: Experiences in elicitation. Statistician **47**(1), 3–19 (1998). https://doi.org/10.1111/1467-9884.00113

Kadane, J.B., Dickey, J.M., Winkler, R.L., et al.: Interactive elicitation of opinion for a normal linear model. J. Am. Stat. Assoc. **75**(372), 845–854 (1980). https://doi.org/10.1080/01621459.1980.10477562

Kingma, DP., Dhariwal, P.: Glow: Generative flow with invertible 1x1 convolutions. Advances in neural information processing systems 31 (2018)

Kingma, DP., Welling, M.: Auto-encoding variational bayes. stat 1050:1. (2014) https://doi.org/10.48550/arXiv.1312.6114

Kobyzev, I., Prince, S.J., Brubaker, M.A.: Normalizing flows: An introduction and review of current methods. IEEE Trans. Pattern Anal. Mach. Intell. **43**(11), 3964–3979 (2020). https://doi.org/10.1109/TPAMI.2020.2992934

Lipman, Y., Chen, RT., Ben-Hamu, H., et al.: Flow matching for generative modeling. (2022) arXiv preprint https://doi.org/10.48550/arXiv.2210.02747

Liu, S., Johns, E., Davison, AJ.: End-to-end multi-task learning with attention. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 1871–1880, (2019) https://doi.org/10.1109/CVPR.2019.00197

Manderson, AA., Goudie, RJ.: Translating predictive distributions into informative priors. (2023) arXiv preprint https://doi.org/10.48550/arXiv.2303.08528

Martin, T.G., Burgman, M.A., Fidler, F., et al.: Eliciting expert knowledge in conservation science. Conserv. Biol. **26**(1), 29–38 (2012). https://doi.org/10.1111/j.1523-1739.2011.01806.x

McKinney, W., et al.: Data structures for statistical computing in python. In: Proceedings of the 9th Python in Science Conference, Austin, TX, pp 51–56, (2010) https://doi.org/10.25080/Majora-92bf1922-00a

Mikkola, P., Martin, O.A., Chandramouli, S., et al.: Prior knowledge elicitation: The past, present, and future. Bayesian Anal. **1**(1), 1–33 (2023). https://doi.org/10.1214/23-BA1381

Mikkola, P., Acerbi, L., Klami, A.: Preferential normalizing flows. (2024) arXiv preprint https://doi.org/10.48550/arXiv.2410.08710

Morris, D.E., Oakley, J.E., Crowe, J.A.: A web-based tool for eliciting probability distributions from experts. Environ. Model. Softw. **52**, 1–4 (2014). https://doi.org/10.1016/j.envsoft.2013.10.010

Nemani, V., Biggio, L., Huan, X., et al.: Uncertainty quantification in machine learning for engineering design and health prognostics: A tutorial. Mech. Syst. Signal Process. **205**, 110796 (2023). https://doi.org/10.1016/j.ymssp.2023.110796

Oakley, J.E., O'Hagan, A.: Uncertainty in prior elicitations: a nonparametric approach. Biometrika **94**(2), 427–441 (2007). https://doi.org/10.1093/biomet/asm031

O'Hagan, A., Oakley, J.E.: Probability is perfect, but we can't elicit it perfectly. Reliab. Eng. Syst. Saf. **85**(1–3), 239–248 (2004). https://doi.org/10.1016/j.ress.2004.03.014

O'Hagan, A., Buck, C.E., Daneshkhah, A., et al.: Uncertain judgements: eliciting experts' probabilities. John Wiley & Sons (2006)

O'Hagan, A.: Expert knowledge elicitation: subjective but scientific. Am. Stat. **73**(sup1), 69–81 (2019). https://doi.org/10.1080/00031305.2018.1518265

Papamakarios, G., Nalisnick, E., Rezende, D.J., et al.: Normalizing flows for probabilistic modeling and inference. J. Mach. Learn. **22**(57), 1–64 (2021). (http://jmlr.org/papers/v22/19-1028.html)

Perepolkin, D., Lindsröm, E., Sahlin, U.: Quantile-parameterized distributions for expert knowledge elicitation. (2023) OSF preprint https://doi.org/10.31219/osf.io/tq3an

Perepolkin, D., Goodrich, B., Sahlin, U.: Hybrid elicitation and quantile-parametrized likelihood. Stat. Comput. **34**(1), 11 (2024). https://doi.org/10.1007/s11222-023-10325-0

Peterson, C., Miller, A.: Mode, median, and mean as optimal strategies. J. Exp. Psychol. **68**(4), 363 (1964). https://doi.org/10.1037/h0040387

Radev, S.T., Mertens, U.K., Voss, A., et al.: Bayesflow: Learning complex stochastic models with invertible neural networks. IEEE Trans. Neural. Netw. Learn. Syst. **33**(4), 1452–1466 (2020). https://doi.org/10.1109/TNNLS.2020.3042395

Radev, S.T., Schmitt, M., Schumacher, L., et al.: Bayesflow: Amortized bayesian workflows with neural networks. J. Open Source Softw **8**(89), 5702 (2023). https://doi.org/10.21105/joss.05702

Regenwetter, M., Cavagnaro, D.R.: Tutorial on removing the shackles of regression analysis: How to stay true to your theory of binary response probabilities. Psychol. methods **24**(2), 135 (2019). https://doi.org/10.1037/met0000196

lgorzata, Roos, M., Martins, TG., Held, L., et al.: Sensitivity analysis for bayesian hierarchical models. Bayesian Anal. **10**(2), 321–349 (2015). https://doi.org/10.1214/14-BA909

Scott, JG.: Prior specification is engineering, not mathematics. Stat. Sci. 32(1):29–32. (2017) DOI: https://doi.org/1214/16-STS5761214/16-STS576

da Silva, EdS, Kuśmierczyk, T., Hartmann, M., et al.: Prior specification via prior predictive matching: Poisson matrix factorization and beyond. (2019) Preprint at https://doi.org/10.48550/arXiv.1910.12263

Simpson, D., Rue, H., Riebler, A., et al.: Penalising model component complexity: A principled, practical approach to constructing

priors. Stat. Sci. **32**(1), 1–28 (2017). https://doi.org/10.1214/16-STS576

Stefan, A.M., Evans, N.J., Wagenmakers, E.J.: Practical challenges and methodological flexibility in prior elicitation. Psychol. Methods **27**(2), 177–197 (2022). https://doi.org/10.1037/met0000354

Van Dongen, S.: Prior specification in bayesian statistics: three cautionary tales. J. Theor. Biol. **242**(1), 90–100 (2006). https://doi.org/10.1016/j.jtbi.2006.02.002

Wagenmakers, E.J., Farrell, S.: AIC model selection using akaike weights. Psychon. Bull. Rev. **11**, 192–196 (2004). https://doi.org/10.3758/BF03206482

Wang, L., Ng, A.H., Deb, K.: Multi-objective evolutionary optimisation for product design and manufacturing. Springer (2011)

Winkler, R.L.: The assessment of prior distributions in bayesian analysis. J. Am. Stat. Assoc. **62**(319), 776–800 (1967). https://doi.org/10.1080/01621459.1967.10500894

elicito: A Python Package for Expert Prior Elicitation

Florence Bockting
TU Dortmund University

Paul-Christian Bürkner
TU Dortmund University

Abstract

Expert prior elicitation plays a critical role in Bayesian analysis by enabling the specification of prior distributions that reflect domain knowledge. However, expert knowledge often refers to observable quantities rather than directly to model parameters, posing a challenge for translating this information into usable priors. We present **elicito**, a Python package that implements a modular, simulation-based framework for expert prior elicitation. The framework supports both structural and predictive elicitation methods and allows for flexible customization of key components, including the generative model, the form of expert input, prior assumptions (parametric or nonparametric), and loss functions. By structuring the elicitation process into configurable modules, **elicito** offers transparency, reproducibility, and comparability across elicitation methods. We describe the methodological foundations of the package, its software architecture, and demonstrate its functionality through a case study.

Keywords: expert prior elicitation, elicitation method, prior distributions, Bayesian analysis, simulation-based framework, Python, **elicito**.

1. Introduction

The goal of *expert prior elicitation* is to specify prior distributions for model parameters or events (Falconer, Frank, Polaschek, and Joshi 2023; Mikkola, Acerbi, and Klami 2024) in Bayesian models that accurately reflect the expectations of a domain expert (for a recent review, see Mikkola, Martin, Chandramouli, Hartmann, Pla, Thomas, Pesonen, Corander, Vehtari, Kaski *et al.* 2023). A central difficulty of this objective is that expert knowledge often relates indirectly, or in a non-transparent way, to the model quantities for which priors are required. To address this challenge, the research field *prior elicitation* has emerged, originating in the 1960s (Winkler 1967; Kadane, Dickey, Winkler, Smith, and Peters 1980; Kadane and Wolfson 1998), and continues to evolve as an active area of study (Stefan, Evans, and Wagenmakers 2022; Mikkola *et al.* 2023; Falconer, Frank, Polaschek, and Joshi 2022).

Two major strands can be distinguished within this field, depending on whether the focus is on the extraction or the translation process of expert knowledge (Manderson and Goudie 2023). The *extraction process* focuses on the interaction with the expert, including identifying

which types of questions can reasonably be posed, understanding cognitive heuristics that may influence expert judgments, training experts to effectively quantify their knowledge, and recommending a set of quantities that can be meaningfully elicited (Dias, Morton, and Quigley 2018; Stefan *et al.* 2022; European Food Safety Authority 2014). In contrast, the *translation process* focuses on understanding the statistical and mathematical relationships between the model parameters, for which a prior distribution is required, and any other model-derived quantities about which the expert might provide information, such as the outcome variable or summary statistics derived from them (Hartmann, Agiashvili, Bürkner, and Klami 2020; da Silva, Kuśmierczyk, Hartmann, and Klami 2019; Manderson and Goudie 2023; Perepolkin, Lindström, and Sahlin 2025).

Insights from both domains have informed the development of prior *elicitation methods* (Mikkola *et al.* 2023). These methods can be broadly classified into structural and predictive elicitation approaches (Kadane and Wolfson 1998; Falconer *et al.* 2022). In *structural* elicitation methods, expert knowledge is directly expressed in terms of the parameters of a Bayesian model. As a result, these methods primarily focus on the fitting task, namely, fitting a prior distribution given the expert data (Kadane *et al.* 1980; Bedrick, Christensen, and Johnson 1996). In contrast, *predictive* elicitation methods elicit expert beliefs about the outcome variable rather than the model parameters. In this case, the methods must address not only the fitting task but also a translation task: converting beliefs about observable outcomes into corresponding priors over model parameters (Akbarov 2009; Muandet, Fukumizu, Sriperumbudur, Schölkopf *et al.* 2017; Mikkola *et al.* 2023; Hartmann *et al.* 2020; Manderson and Goudie 2023; da Silva *et al.* 2019; Perepolkin, Goodrich, and Sahlin 2024).

The entire *elicitation process*, which encompasses the workflow from extracting knowledge from a domain expert to learning prior distributions for the model quantities of interest, can be summarized in four stages (Garthwaite, Kadane, and O'Hagan 2005): The *setup stage* involves preparing for the main elicitation, including the identification of suitable expert(s) and the selection of relevant information to be elicited. This is followed by the *elicitation stage*, during which the selected information is extracted from the expert(s) using structured elicitation protocols. In the subsequent *fitting stage*, the elicited information is used to derive corresponding prior distribution(s). The final *evaluation stage*, entails assessing the plausibility of the learned priors in collaboration with the expert(s). If discrepancies or inconsistencies are identified, this stage may prompt revisions of decisions made in earlier stages and repetition of the process to obtain more accurate or reliable priors.

In our previous work, we focused on the process of translating expert knowledge into prior distributions. Specifically, building on the work of Hartmann *et al.* (2020); Manderson and Goudie (2023); da Silva *et al.* (2019), we introduced a simulation-based approach for learning prior distributions of model parameters from different types of expert input ranging from the parameter to the observable space (Bockting, Radev, and Bürkner 2024). In a recent extension, we increased the flexibility of our simulation-based approach by also supporting different types of prior distributions from joint non-parametric to independent, parametric priors (Bockting, Radev, and Bürkner 2025).

Building on this work, we developed a modular framework that incorporates all core components of a prior elicitation method. Each module in the framework can be customized which allows for the support of a wide range of elicitation methods that may differ in their choice of generative models, the type and structure of expert information, the nature of prior assumptions (including both parametric and nonparametric priors), and the specification of

loss functions. We translated this conceptual idea into an algorithmic design implemented via the Python package **elicito** which is introduced in this paper. We think that the possibility to develop different elicitation methods within the same framework offers several advantages. It increases transparency regarding the decisions required to implement a given elicitation method, facilitates the comparison of different methods, and enables the use of common diagnostics to evaluate both method performance and the resulting prior distributions.

The remainder of this paper is organized as follows: Section 2 provides an overview of the methodological and statistical foundations underpinning our software implementation, **elicito**. Section 3 presents the software architecture, detailing each module in relation to the conceptual stages of the prior elicitation process. In Section 4, we demonstrate the application of **elicito** through a simple case study that illustrates its functionality in a concrete context. Section 5 reviews related software packages currently available in the field and discusses their relevance to the proposed implementation. Finally, Section 6 concludes the paper with a general discussion and an outline of future developments of **elicito**.

2. Method

In the following section, we present a conceptual overview of our simulation-based prior elicitation framework, as introduced in Bockting *et al.* (2024) and Bockting *et al.* (2025). Readers already familiar with this workflow may wish to skip directly to Section 3, which focuses on the implementation details of the **elicito** software.

The overall workflow of our framework follows the five stages of the prior elicitation process as outlined by Garthwaite *et al.* (2005), and is operationalized through the following six steps:

1. *Define the generative (statistical) model*: Specify the generative model, including dimensionality and parameterization of the prior distribution(s). (Setup stage, Section 2.1)

2. *Identify the information to be elicited and the corresponding elicitation techniques*: Determine the relevant expert knowledge to be elicited and select appropriate elicitation techniques. (Setup stage, Section 2.2)

3. *Extract information from the expert and simulate corresponding predictions from the generative model*: Collect expert input and simulate predictions from the generative model. Perform all necessary computational steps to generate model predictions corresponding to the expert-elicited information. (Elicitation stage, Section 2.3)

4. *Evaluate consistency between expert information and model predictions*: Quantify the discrepancy between expert-provided information and model predictions using a multi-objective loss function. (Fitting stage, Section 2.4)

5. *Learn prior distributions consistent with expert information*: Use an optimization algorithm to minimize the loss function, thereby identifying prior distributions that align with the elicited expert information. (Fitting stage, Section 2.4)

6. *Evaluate the learned prior distributions*: Repeat the optimization with different random seeds to assess the variability of priors that fit the expert data. Select a plausible prior distribution in collaboration with the domain expert or apply model averaging techniques. (Evaluation stage, Section 2.5)

The following sections provide a detailed explanation of each step. An overview of the symbols used throughout the text, along with their descriptions, is provided in Appendix A.

2.1. Generative Model and Prior Assumptions

The foundation of our framework is the generative model $\mathcal{M}$, which defines a statistical model over the joint probability distribution $p(y, \theta)$, where y denotes the observed data and θ represents the unobserved parameters (Haines, Sullivan-Toole, and Olino 2023). Within the Bayesian setting considered here, specifying the generative model further requires assumptions regarding the dimensionality, dependence structure, and parametric form of the prior distribution $p_\lambda(\theta)$ over the model parameters, where λ denotes the hyperparameters of the prior.

The interpretation of λ depends on the nature of the prior. Under a *parametric prior*, λ corresponds to the hyperparameters of the chosen parametric distribution family. In contrast, when a *non-parametric prior* is employed, our framework utilizes deep generative models to learn the functional form of the prior distribution. In this case, λ denotes the parameters of the deep generative model, typically the weights of a (deep) neural network.

The generative model $\mathcal{M}$ is specified by the user with the aim of learning prior distributions over the model parameters. Since the priors are parameterized by the hyperparameter vector λ, the objective of the elicitation framework is to identify a suitable value for λ. The guiding criterion for this learning task can be formualted as follows: When the generative model $\mathcal{M} \mid \lambda$, conditioned on a specific hyperparameter vector, is executed in forward mode, the resulting simulated model-derived quantities should align with the expectations specified by a domain expert, which are detailed in the following sections.

2.2. Target Quantities and Elicited Statistics

To enable the learning task, it is necessary to define a set of expert-informed criteria, or in other words, the expectations of the domain expert quantified through specific summary statistics. Ideally, this set should be both, *interpretable* to domain experts and *informative* for the generative model (Mikkola *et al.* 2023). In practice, achieving both objectives simultaneously is challenging, as they are often not aligned. For example, information directly referring to model parameters can be highly informative for the learning process, yet frequently lacks interpretability from the perspective of a domain expert (Kadane *et al.* 1980). By contrast, information related to the outcome variable is typically highly interpretable for domain experts but offers limited utility in constraining the individual hyperparameters λ, often resulting in poorly informed or weakly constrained prior distributions (Stefan *et al.* 2022). The suitability of a particular set of expert information is highly dependent on both the structure of the generative model and the specific expertise of the individual being queried (Denham and Mengersen 2007). Consequently, an effective elicitation method should be capable of accommodating a wide range of different forms of expert knowledge.

Within our framework, expert information is specified through a two-stage process. First, we define *what* type of information is to be elicited from the expert. This information is formalized as a function of the hyperparameters, denoted by $z_i = c_i(\lambda)$ for $i = 1, \ldots, I$, and referred to as *target quantities*. Typically, multiple target quantities (I in total) are elicited from the user to inform different aspects of the data-generating process. For instance, if we want to query the expert regarding the outcome variable (i.e., the observable space), one of

the target quantities can be simply defined as $z_i = y$. This is obtained by defining the function c_i as a query to the prior predictive distribution $p(y) = \int p(y \mid \theta) p_\lambda(\theta) d\theta$, where the likelihood is marginalized over the prior. Note that this expression is not just a constant as it refers to the prior predictive distribution and should not be confused with the marginal likelihood of observed data (see for this point also Hartmann *et al.* (2020)). As another example, expert knowledge may be elicited about the parameter space. In this case, the target quantity is defined as $z_i = \theta_k$, which is obtained by defining c_i as a simple projection onto θ_k. In this way, we can also accommodate any other target quantity that can be derived from the model parameters or the data (e.g., R^2; the proportion of variance explained by the model; Gelman, Goodrich, Gabry, and Vehtari 2019).

Second, we define *how* each target quantity is to be queried by selecting an appropriate elicitation technique. In our elicitation method, we represent an elicitation technique as a function f_j for $j = 1, \ldots, J$, of a target quantity z_i. The resulting set of *elicited* target quantities is referred to as *elicited statistics*, $\{t_m(\lambda)\}$, where $t_m(\lambda) = f_j(z_i)$ and m refers to the corresponding $i \times j$ combination. In the following, we sometimes omit the explicit dependence on λ and simply write t_m instead of $t_m(\lambda)$. Note also that, depending on the elicitation technique employed, t_m may accept either a single value or a set of values as input. To illustrate this point more clearly, consider the case in which we elicit expert knowledge about the predictive distribution conditional on a category of a categorical predictor, i.e., $z = y \mid \text{cat}$. To elicit this information, we query the expert regarding the $25\%, 50\%$, and 75% quantiles of this predictive distribution. Thus, we have $t = \{Q_{25\%}(y \mid \text{cat}), Q_{50\%}(y \mid \text{cat}), Q_{75\%}(y \mid \text{cat})\}$.

2.3. Expert-elicited and model-implied statistics

Once the target quantities and elicitation techniques have been specified, the corresponding quantities of interest must be obtained from the domain expert through an appropriate interrogation. This yields a set of expert-elicited summary statistics, denoted by $\{\hat{t}_m\}_{m=1}^M$. Similarly, a corresponding set of model-implied statistics, denoted by $\{t_m\}_{m=1}^M$, can be generated by simulating the generative model in forward mode and applying the introduced transformation $t_m = f_j(c_i(\lambda))$.

Given the expert-elicited statistics $\{\hat{t}_m\}$ and the model-implied statistics $\{t_m\}$, their agreement can be assessed using a discrepancy measure $\mathcal{D}_m(t_m(\lambda), \hat{t}_m)$ for $m = 1, \ldots, M$. The choice of discrepancy measure may vary across statistics, depending on the nature and scale of each elicited statistic. Each weighted discrepancy term is referred to as a *loss component* L_m, where w_m represents the corresponding weight. The total loss is defined as the sum over all individual loss components, resulting in a *multi-objective loss function* that quantifies the overall mismatch between expert knowledge and model simulations. Formally, this is given by

$$\mathcal{L}(\lambda) = \sum_{m=1}^{M} \underbrace{w_m \cdot \mathcal{D}_m\left(t_m(\lambda), \hat{t}_m\right)}_{L_m}. \tag{1}$$

The learning task of identifying a hyperparameter vector λ such that the model-derived quantities align with expert expectations can thus be reformulated as an optimization problem

$$\lambda^* = \min_{\lambda} \mathcal{L}(\lambda) \tag{2}$$

which seeks the hyperparameter values λ^* that minimize the multi-objective loss function.

2.4. Prior hyperparameter optimization

To minimize the total loss $\mathcal{L}(\lambda)$, we formulate the problem as an optimization task and solve it using mini-batch stochastic gradient descent (SGD), as defined in Equation 2. This approach requires computing the gradient of the loss function with respect to the hyperparameter vector λ, followed by iterative updates of λ in the direction opposite to the gradient (Goodfellow, Bengio, and Courville 2016). Gradient computation is enabled via automatic differentiation using the explicit or implicit reparameterization trick (Kingma and Welling 2014; Figurnov, Mohamed, and Mnih 2018).

The updated hyperparameters λ' are then used to generate new samples from the prior distribution $p_{\lambda'}(\theta)$, from which an updated set of model-implied statistics $\{t_m(\lambda')\}$ is computed. The discrepancy between these updated model-implied statistics and the expert-elicited statistics is then reassessed. Based on the resulting loss, the hyperparameters are further adjusted using the mini-batch SGD procedure. This iterative process continues until a predefined convergence criterion is met or a specified stopping condition is reached.

2.5. Evaluation of learned prior distribution

As an initial step, the learning process is assessed for unexpected behavior that may indicate issues during optimization. If no such irregularities are observed, the learned prior distribution can be evaluated.

Monitoring the learning process A first assessment of the learning progress involves examining the trajectory of the total loss, along with the individual loss components, throughout the optimization process. Ideally, the total loss should exhibit a generally decreasing trend, often resembling exponential decay, and eventually stabilize near zero (Goodfellow *et al.* 2016). However, it is important to note that the *individual* loss components may temporarily increase before decreasing, particularly in the presence of conflicting objectives within the multi-objective loss function. The shape and slope of the loss trajectory can offer insights into the effectiveness of the optimization procedure. For example, an excessively steep slope may indicate an overly large learning rate, which can cause the optimizer to overshoot local or global minima. Conversely, a learning rate that is too small may result in negligible parameter updates per iteration, leading to slow convergence and prolonged training times (Buduma and Locascio 2017).

In addition to monitoring the loss, the update trajectory of the hyperparameters λ can offer valuable insights into the optimization process. Ideally, the trajectory should stabilize over successive iterations, converging to a specific value. However, in some cases, inspecting individual hyperparameters is impractical due to their high dimensionality. This is particularly true for non-parametric priors, where λ correspond to the weights of deep neural networks. In such settings, it more feasible to monitor the evolution of specific summary characteristics of the prior distribution across optimization iterations. These may include location (e.g., mean), scale (e.g., variance), skewness, or correlations between marginal priors. Visualizing the trajectories of these components can provide insight into the stability and convergence of the learning process.

Regardless of whether individual hyperparameters or summary characteristics of the prior are examined, a failure of the trajectory to converge or persistent high variability may signal

problems in the learning process. Such behavior can stem from an inappropriate learning rate or from insufficient information in the elicited statistics to meaningfully constrain the prior. For example, a collapse of the prior variance toward zero during training may indicate that the available expert information is inadequate to guide the learning process, resulting in overly narrow or degenerate prior distributions.

Another useful diagnostic for assessing the success of the learning process is to directly compare the expert-elicited statistics with the final model-implied statistics. While the individual loss components already quantify the discrepancies between these two sets of quantities, this additional diagnostic offers a complementary perspective by evaluating the match in the original scale of the elicited statistics. This can make it easier to identify meaningful mismatches. In contrast, interpreting the loss trajectory alone can be challenging, as it is not always evident whether a near-zero loss is small enough in practical terms.

Inspecting the learned prior Once the inspection of the optimization process is complete and no unexpected behavior is observed, the learned prior distributions can be assessed in detail. This assessment may include visual inspection of the marginal prior distributions, as well as examination of the joint prior through the analysis of the correlation structure between marginal components.

An important consideration is that the elicited statistics used to guide the learning of the prior are typically insufficient to fully identify the generative model. As a result, there is not a unique hyperparameter vector λ^* that matches the set of elicited statistics $\{\hat{t}_m\}$; instead, there exists a set of equally valid solutions, $\{\lambda^*\}_{k=1}^K$. Consequently, this gives rise to a corresponding set of prior distributions $\{p_{\lambda^*}(\theta)\}_{k=1}^K$, all of which are equally consistent with the elicited expert knowledge.

To understand the variability among supported prior distributions, it is instructive to sample from the set of learned priors. This can be achieved by running the optimization process multiple times, each with different initializations, and subsequently inspecting the range of resulting priors. Together with the domain expert, these priors can be evaluated for plausibility. If some of the learned priors are deemed implausible, this suggests that the current elicited information is insufficient to adequately constrain the solution space, and additional constraints or elicited statistics should be incorporated into the loss function. If, on the other hand, the resulting set consists of multiple priors that are all considered plausible, one may either select a single prior from this set or adopt a prior averaging strategy. In the latter case, the final prior is formed by averaging over the set of plausible priors, thereby reflecting the uncertainty inherent in the prior specification process.

2.6. Implementation

The introduced methodological framework is implemented via the Python package **elicito**, which is released under the open-source Apache 2.0 license. **elicito** is available on PyPI (https://pypi.org/project/elicito/) and conda-forge (https://anaconda.org/conda-forge/elicito). The documentation is maintained on ReadTheDocs (https://elicito.readthedocs.io) and the codebase is hosted on GitHub (https://github.com/florence-bockting/elicito) to facilitate version control and collaborative development. The code adheres to the Black style guide, a widely adopted standard for Python formatting and linting. The package supports Python versions 3.9 through 3.12 and depends on `tensorflow >= 2.16`,

`tensorflow-probability >= 0.24`, `tf-keras >= 2.16`, `numpy >= 1.24`, `joblib >= 1.4.2`, and `tqdm >= 4.38`. All dependencies are managed via the `uv` Python package manager.

The primary user interface of **elicito** is the `Elicit` class, through which the user can specify the entire elicitation procedure. The arguments of the `Elicit` class are designed to capture all necessary information required to implement an elicitation method. A brief overview of these arguments is provided below:

- `model`: Defines the generative model used in the elicitation procedure.
- `parameters`: Specifies assumptions regarding the prior distributions over model parameters, including (hyper)parameter constraints, dimensionality, and parametric form.
- `targets`: Defines the elicited statistics in terms of target quantities and corresponding elicitation techniques. Specifies the discrepancy measure and weight used for the associated loss component.
- `expert`: Provides the expert information that serves as the basis for the learning criterion.
- `optimizer`: Specifies the optimization algorithm to be used, along with its hyperparameters (e.g., learning rate).
- `trainer`: Configures the overall training procedure, including settings such as the random seed, number of epochs, sample size, and batch size.
- `initializer`: Defines the initialization strategy for the hyperparameters used to instantiate the simulation-based optimization process; required only when using parametric prior distributions.
- `networks`: Specifies the architecture of the deep generative model; required only when using non-parametric prior distributions.

By configuring these core components, the **elicito** package supports a wide range of elicitation methods, including both structural and predictive approaches (Kadane and Wolfson 1998; Falconer *et al.* 2022). It accommodates univariate and multivariate as well as parametric and nonparametric prior distributions. In the following section, we detail the specification of each argument and demonstrate the full range of functionality offered by **elicito**.

3. Software

The Python package **elicito** implements the simulation-based prior elicitation framework proposed by Bockting *et al.* (2024, 2025). An overview of the underlying computational algorithm is provided in Section 3.1. The subsequent sections introduce and explain the arguments of the `Elicit` class, which constitutes the primary user interface of **elicito**. In the following, we assume that the **elicito** package has been imported using the alias `el`.

3.1. Computational algorithm

Algorithm 1 outlines the computational steps underlying the **elicito** framework. In this pseudo-code we assume a single batch while we use a batch size greater one in the actual implementation, resulting in all computational objects carrying an additional dimension of size B.

Algorithm 1 Computational algorithm underlying the **elicito** framework (single batch)

Require: λ_0 ▷ specify initialization for hyperparameter
Require: either DNN or $p_\lambda(\cdot)$ ▷ specify non-parametric prior via deep generative model or parametric prior distribution families
Require: $p\left(y \mid \theta\right)$ ▷ define generative model
Require: c_i with $i = 1, \dots I$ ▷ define set of target quantities
Require: f_j with $j = 1, \dots, J$ ▷ define elicitation techniques
Require: D_m, w_m for $m = 1, \dots, M$ ▷ define discrepancy measure and weight of individual loss component
Require: epochs, S, η ▷ define training settings: learning rate for SGD optimizer, number of epochs and prior samples

$\lambda \leftarrow \lambda_0$ ▷ initialize hyperparameter λ
for epoch in epochs **do**
  $p_\lambda \leftarrow \text{DNN}(\lambda)$ ▷ (optional) learn prior density from DNN
  for $s = 1, \dots, S$ **do**
    draw $\theta^{(s)}$ from $p_\lambda(\cdot)$ ▷ sample model parameter from prior
    for $n = 1, \dots, N$ **do**
      draw $y_n^{(s)}$ from $p\left(y \mid \theta^{(s)}\right) \cdot p_\lambda\left(\theta^{(s)}\right)$ ▷ sample from generative model
    end for
    for $i = 1, \dots, I$ **do**
      $\{z_i^{(s)}\} \leftarrow c_i(\lambda)$ ▷ compute target quantity
    end for
    for $m = 1, \dots, M$ **do**
      $\{t_m^{(s)}\} \leftarrow f_j(z_i^{(s)})$ ▷ compute elicited statistics
      $L_m \leftarrow w_m D_m(t_m^{(s)}(\lambda), \hat{t}_m)$ ▷ compute individual loss component
    end for
  end for
  $\mathcal{L} = \sum_{m=1}^{M} L_m$ ▷ Compute total loss
  $\lambda_{\text{epoch}+1} \leftarrow \lambda_{\text{epoch}} - \eta \nabla_{\lambda_{\text{epoch}}} \mathcal{L}$ ▷ Update λ via backpropagation
end for

3.2. Setup Stage

In the *setup stage*, the generative model is defined, assumptions about the prior distributions are specified, and the set of target quantities to be elicited from the expert is identified, along with the corresponding elicitation techniques used to gather expert input. These configurations are specified through the arguments `model`, `parameters`, and `targets` of the `Elicit` class, which are detailed in the following subsections.

Generative model

The generative model is specified using the `model` argument of the `Elicit` class, via the `el.model()` function. This function follows the general structure

```
model = el.model(obj=<class callable>, **kwargs).
```

The generative model must be implemented as a Python class that defines a `call` method. This method has to accept `prior_samples` as a mandatory input argument, with additional optional arguments as needed. It must return a dictionary containing the desired model-derived target quantities, where each entry corresponds to a name-value pair representing a specific target quantity.

The class implementing the generative model is passed to the `el.model()` function via the `obj` argument. Any additional parameters required by the model can be specified using keyword arguments (`**kwargs`).

The following example illustrates an implementation for a simple normal regression model with design matrix $\mathbf{X}$, formally defined as

$$\begin{aligned}\mu &= (\beta_0, \beta_1)^\top \mathbf{X} \\ \mathbf{y} &\sim \text{Normal}(\mu, \sigma).\end{aligned}$$

The implementation of the Python class is

```
class GenerativeModel:
    def __call__(self, prior_samples, X):
        # extract samples per model parameters
        beta0, beta1, sigma = [prior_samples[:,:,i] for i in range(3)]

        # compute linear predictor term
        mu = beta0 + beta1*X

        # sample prior predictions from likelihood
        y = tfd.Normal(mu, sigma[:,:,None]).sample()

        return dict(y=y)
```

The corresponding model specification is `el.model(obj=GenerativeModel, X=X)`.

Special case: Discrete random variables When using mini-batch SGD as optimization procedure, it is essential that gradients can be computed at each step of the computational graph.

However, this is generally infeasible for discrete random variables (for a detailed discussion, see Bockting *et al.* 2024). One strategy to overcome this limitation involves approximating a discrete random variable with a continuous distribution, thereby enabling the use of gradient-based optimization methods. One such technique is the Gumbel-Softmax trick (Maddison, Mnih, and Teh 2017; Jang, Gu, and Poole 2017; Joo, Kim, Shin, and Moon 2020), which is implemented in **elicito** via the `el.utils.gumbel_softmax_trick()` function. This function facilitates the definition of discrete likelihoods within the `GenerativeModel`, with current support limited to lower- and double-bounded random variables.

The function `el.utils.gumbel_softmax_trick()` requires two arguments: `likelihood` which expects a member of `TensorFlow Probability distributions (tfd)` representing the discrete likelihood and `upper_thres` which expects an integer defining the upper truncation threshold for lower-bounded distributions. For double-bounded distributions, this value corresponds to the upper bound. An additional optional argument is the softmax temperature, `temp`, with default `temp=1.6`, a value that has shown good empirical performance.

The following example shows the implementation of a generative model with Binomial likelihood. To highlight the differences from the previous example using a continuous likelihood, each modified line is marked with `>>`:

```
   class GenerativeModel:
>>      def __call__(self, prior_samples, X, total_count):
            # extract samples per model parameters
            beta0, beta1, sigma = [prior_samples[:,:,i] for i in range(3)]

            # compute linear predictor term
            mu = beta0 + beta1*X

            # define discrete likelihood
>>          likelihood = tfd.Binomial(total_count, tf.math.sigmoid(mu))

            # approximate samples from Binomial distribution
>>          y = el.utils.gumbel_softmax_trick(
                    likelihood=likelihood,
                    upper_thres=total_count
                )

            return dict(y=y)
```

The corresponding model specification is then provided via `el.model(obj=GenerativeModel, X=X, total_count=total_count)`.

Model parameters

Once the generative model has been defined, the assumptions regarding the prior distributions over the model parameters must be specified. This is accomplished via the `parameters` argument of the `Elicit` class. This argument expects a list of model parameter specifications, with each parameter defined using the `el.parameter()` function. The required input format for `el.parameter()` depends on whether a parametric or non-parametric prior is assumed.

If a *parametric prior* is assumed, both the distribution family and its associated hyperparameters must be provided. In this case, the `el.parameter()` function requires three arguments: `name`, `family`, and `hyperparams`. For instance, specifying a half-normal prior distribution for a scale parameter, $\sigma \sim \text{HalfNormal}(\sigma_0)$, is implemented as follows:

```
el.parameter(
    name="sigma",
    family=tfd.HalfNormal,
    hyperparams=dict(scale=el.hyper("sigma0", lower=0.))
).
```

The `name` argument assigns a custom identifier to the model parameter. The `family` argument specifies the prior distribution family, currently limited to members of the `tfd` module. Hyperparameters for the selected prior distribution family are provided via the `hyperparams` argument, which accepts a dictionary. Each key corresponds to a required input parameter of the specified `tfd` member, while the associated values are defined using the `el.hyper()` function. This function enables users to assign custom names to hyperparameters and impose constraints where applicable.

The `el.hyper()` function supports multiple configurations, with the default configuration:

```
el.hyper(name=<custom name>, lower=float("-inf"), upper=float("inf"),
    vtype="real", dim=1, shared=False)
```

By default, a hyperparameter is treated as an unconstrained scalar. Constraints can be imposed via the `lower` and `upper` arguments to enforce lower bounds, upper bounds, or double-bounded intervals. The hyperparameter's dimensionality is controlled using the `vtype` (variable type) and `dim` (dimension) arguments. Additionally, hyperparameters can be *shared* across multiple prior distributions by setting the argument `shared=True`. The following code snippet illustrates three examples: (1) an unconstrained hyperparameter, (2) a non-negative shared hyperparameter, and (3) a multidimensional hyperparameter:

```
# unconstrained scalar hyperparameter
el.hyper(name="mu0")

# non-negative scalar hyperparameter shared across priors
el.hyper(name="sigma0", lower=0., shared=True)

# unconstrained 2D hyperparameter
el.hyper(name="tau", vtype="array", dim=2)
```

When a *non-parametric prior* is assumed for a model parameter, the `el.parameter()` function requires only the `name` argument. Additional constraints on the parameter's domain can be specified using the `lower` and `upper` arguments. For example, `el.parameter(name="sigma", lower=0)` specifies a non-parametric prior for the scale parameter `sigma`, constrained to be non-negative.

Note that constraints on model parameters are only required when *non-parametric* priors are specified. In the case of parametric priors, constraints on the *hyper*parameters inherently

impose the corresponding constraints on the model parameters. All implemented constraints follow a standardized form closely following the parameter constraint implementation in **Stan** (Carpenter, Gelman, Hoffman, Lee, Goodrich, Betancourt, Brubaker, Guo, Li, and Riddell 2017). Specifically, for double-bounded variables, the inverse of the log-odds transformation is applied and for variables with either a lower or upper boundary, a softplus transformation is used. In this latter case, we deviate from the conventional exponential transform, as empirical results suggest that the softplus transformation provides greater numerical stability.

The complete specification of the `parameters` argument in the `Elicit` class for a generative model with two parameters, assuming a non-parametric joint prior, is illustrated in the following example:

```
parameters = [
    el.parameter(name="beta"),
    el.parameter(name="sigma", lower=0)
    ]
```

Target quantities and Elicitation techniques

The specification of the target quantities and their associated elicitation techniques is provided via the `targets` argument of the `Elicit` class. This argument accepts a list of elements, each defined using the `el.target()` function. The function follows the default configuration

```
el.target(name=<custom name>, query=<el.queries>, loss=<function callable>,
    target_method=None, weight=1.0).
```

Target quantities As introduced in Section 2.2, a target quantity is defined as a function of the hyperparameters λ, $z_i = c_i(\lambda)$. The transformation function c_i can be implemented directly within the generative model, with the resulting target quantity z_i returned as a key-value pair in the output dictionary. In this case, the corresponding target is specified using `el.target(name="z_i", target_method=None, ...)`. Here, `"z_i"` must match the key under which the generative model returns the computed quantity.

Alternatively, the transformation can be decoupled from the generative model by implementing a user-defined Python function. For example, consider the target quantity of interest is the coefficient of determination, defined as $R^2 = \text{Var}(\mu)/\text{Var}(y)$ (Gelman *et al.* 2019). This transformation can be specified via a custom function, as illustrated below:

```
def r2(mu, y):
    return tf.divide(tf.math.reduce_variance(mu, -1),
                     tf.math.reduce_variance(y, -1)
                     )
```

In this example, the input arguments `mu` and `y` correspond to keys in the output dictionary returned by the generative model. The user-defined function is then passed to the `el.target()` function via the `target_method` argument: `el.target(name="r2_custom", target_method=r2, ...)`. Here, the `name` argument can be chosen freely.

Elicitation techniques Once the target quantity is defined, the corresponding *elicitation technique* can be specified via the `query` argument of the `el.target()` function. To support this process, several pre-implemented elicitation techniques are available through the `el.queries` class. Alternatively, users may define and supply their own elicitation technique via a custom function. The following examples illustrate the different approaches:

```
# computes the quantiles of the provided target quantity
el.queries.quantiles(quantiles=(0.25, 0.50, 0.75))

# returns the target quantity as is
el.queries.identity()

# computes the correlation between model parameters
el.queries.correlation()

# defines a custom elicitation technique
def custom_func(target_name):
    return f(target_name)

el.queries.custom(custom_func)
```

The `el.queries.quantiles()` function computes specified quantiles of the target quantity and requires as input a list or tuple of probabilities representing the desired quantile levels. The `el.queries.identity()` function returns the target quantity without applying any transformation, effectively serving as a pass-through operator. The `el.queries.correlation()` function computes the correlation between model parameters, which is particularly useful for learning joint priors in non-parametric settings. Lastly, the `el.queries.custom()` function allows users to define and incorporate custom elicitation techniques.

Note that although both target quantities and elicited statistics are defined within the `el.target()` function, their computation is carried out sequentially (see Algorithm 1). First, the target quantities are either extracted directly from the output dictionary of the generative model or computed via a user-defined transformation function. This results in a dictionary in which each entry corresponds to a distinct target quantity. In the second step, the elicitation technique, specified via the `query` argument, is applied to the corresponding target quantity. The output is a dictionary containing the simulated elicited statistics.

Discrepancy measure and weight Each simulated elicited statistic is compared to its corresponding expert-elicited counterpart using an appropriate discrepancy measure. Together with a weight w_m, this defines an individual loss component (see Section 2.3). The discrepancy measure and its associated weight are specified via the `loss` and `weight` arguments in the `el.target()` function, respectively. The **elicito** package currently includes two built-in discrepancy measures: a squared-error loss (`el.losses.L2`) and a squared, biased Maximum Mean Discrepancy loss (MMD, Gretton, Borgwardt, Rasch, Schölkopf, and Smola 2012) (`el.losses.MMD2`), which supports both energy and Gaussian kernels. For greater flexibility, users may also provide custom discrepancy functions tailored to their specific application.

In summary, a complete specification of the `targets` argument within the `el.Elicit` class is illustrated in the following example:

```
targets = [
    el.target(name="y", query=el.queries.quantiles((0.25, 0.50, 0.75)),
        loss=MMD2(kernel="energy"), weight=1.),
    el.target(name="R2", query=el.queries.custom(sd), target_method=r2,
        loss=L2, weight=10.)
    el.target(name="mu", query=el.queries.identity(),
        loss=MMD2(kernel="gaussian", sigma=1.), weight=1.)
 ]
```

3.3. Elicitation stage

Once the set of elicited statistics has been defined, the corresponding expert information must be collected and provided as input to the elicitation procedure. This is achieved through the `expert` argument of the `el.Elicit` class.

Expert-provided data is supplied to the elicitation method using the `el.expert.data()` function. This function takes a dictionary containing the expert information via its `dat` argument. The keys in this dictionary must exactly match a predefined structure that depends on the `targets` configuration. To facilitate this step, **elicito** offers the helper function `el.utils.get_expert_datformat()`, which generates a template dictionary illustrating the expected key-value structure for a given `targets` specification.

The following example illustrates a scenario in which an expert provides information about the expected outcome distribution, conditioned on each level of a three-level categorical predictor. A quantile-based elicitation approach is employed.

```
targets = [
    el.target(
        name=f"gr{i}",
        query=el.queries.quantiles((.05, .25, .50, .75, .95)),
        loss=el.losses.MMD2(kernel="energy"),
        weight=1.0) for i in [1,2,3]
        ]

expert = el.expert.data(dat={
            "quantiles_gr1": [-12.55, -0.57, 3.29, 7.14, 19.15],
            "quantiles_gr2": [-11.18, 1.45, 5.06, 8.83, 20.42],
            "quantiles_gr3": [-9.28, 3.09, 6.83, 10.55, 23.29]
        })
```

3.4. Fitting stage

During the *fitting* stage, the primary objective is to minimize the total loss function to learn the hyperparameters λ that define the prior distributions aligned with the expert expectations. The configuration of the optimization procedure and the learning process is controlled through the `optimizer` and `trainer` arguments of the `Elicit` class. Additionally, when parametric priors are to be learned, the initial values required to instantiate the learning process must be specified using the `initializer` argument. In contrast, if non-parametric priors are used,

the deep generative model needs to be configured through the `networks` argument. In the following sections, each of these arguments is described in detail.

Optimization procedure

The current optimization method supported in **elicito** is mini-batch SGD (see Section 2.4). This requires the specification of an optimization algorithm, which can be provided via the `el.optimizer()` function and the corresponding argument of the `Elicit` class. The optimizer must be an instance of the `tf.keras.optimizers` module. By default, the Adam optimizer is used (Kingma and Welling 2014).

Optimizer-specific parameters, such as the learning rate, can be specified as additional keyword arguments to the `el.optimizer()` function. The learning rate may be provided either as a fixed scalar value or as a learning rate schedule, implemented as a member of the `tf.keras.optimizers.schedules` module.

The following example demonstrates the configuration of an Adam optimizer using a cosine decay learning rate schedule, along with an additional `clipnorm` argument, which constrains the gradient norm of each hyperparameter value λ.

```
optimizer = el.optimizer(
    optimizer=tf.keras.optimizers.Adam,
    learning_rate=tf.keras.optimizers.schedules.CosineDecay(
        initial_learning_rate=0.1, decay_steps=600),
    clipnorm=1.0
    )
```

Learning of prior distributions

The overall learning process is configured via the `trainer` argument of the `Elicit` class, which is defined using the `el.trainer()` function. This function requires three arguments: `method`, `seed`, and `epochs`. The `method` argument specifies whether parametric (`"parametric_prior"`) or non-parametric priors (`"deep_prior"`) are assumed for the model parameters. The `seed` argument is applied internally to all sampling steps to ensure reproducibility. Finally, the `epochs` argument determines the number of iterations used to update the hyperparameters λ. In addition to the required arguments, `el.trainer` also accepts several optional arguments that further refine the training process. These include the batch size (`B=128`), the number of samples drawn from the prior distributions per epoch (`num_samples=200`), and the verbosity level (`progress=1`), which controls whether a progress bar is shown during training.

In the following example, we demonstrate the learning of parametric prior distributions over 400 epochs. In each epoch, 200 samples are drawn from the prior distributions to compute the model-implied elicited statistics. The batch size is set to 256, the random seed is 4, and progress output is disabled.

```
trainer = el.trainer(
    method="parametric_prior",
    seed=4,
    epochs=400,
    B=256,
```

```
    num_samples=200,
    progress=0
)
```

Initialization and deep generative model

To initiate simulation-based training, initial values for the hyperparameters λ must be specified. For parametric priors, these initial values are provided via the `initializer` argument of the `Elicit` class. This is achieved using the `el.initializer()` function, which currently supports two initialization strategies: (1) explicitly specifying initial values for each hyperparameter λ, or (2) employing a method that explores the loss landscape to identify an initial configuration of λ that yields a low loss value.

For non-parametric priors, deep generative models are employed to represent the joint prior distribution. These models must be specified via the `network` argument of the `Elicit` class. Currently, **elicito** supports normalizing flows (NFs) for learning non-parametric joint priors. In this setting, the `initializer` argument should be set to `None`, as the deep generative model relies on default initialization schemes. Specifically, the kernel weight matrices in the dense layers are initialized using the Glorot uniform initializer, while the bias vectors are initialized to zero (Glorot and Bengio 2010).

3.5. Evaluation stage

Instantiation and methods of the Elicit class

Once all arguments of the `Elicit` class have been specified, an instance of the class can be created. We refer to this instance as `eliobj` in the following.

```
eliobj = el.Elicit(
    model=model,
    parameters=parameters,
    targets=targets,
    expert=expert,
    optimizer=optimizer,
    trainer=trainer,
    initializer=initializer,
    networks=None
)
```

Each argument passed to the `Elicit` class is stored as a corresponding attribute within the `eliobj` instance, enabling retrieval and inspection of the configuration at a later stage. In addition, two attributes, `history` and `results`, are initialized as empty lists upon instantiation. These serve as containers for recording results from the fitting process during optimization. The fitting procedure is initiated using the `eliobj.fit()` method.

The `fit()` method supports parallel execution via its `parallel` argument, which enables running multiple fitting processes with different random seeds simultaneously. This parallelization is configured through the `el.parallel()` function.

An `Elicit` instance can be saved to disk using the `eliobj.save()` method and subsequently restored using the `el.utils.load()` helper function. Furthermore, the `eliobj.update()` method facilitates targeted modifications to one or more arguments of the original `Elicit` configuration. This allows users to selectively update components without the need to re-specify the entire configuration.

Postprocessing and evaluation of optimisation results

The outcomes of the fitting procedure are stored in the `history` and `results` attributes of the `eliobj` instance. When multiple replications are performed in parallel, the results corresponding to each replication are stored in separate sublists, resulting in both `history` and `results` being represented as nested lists. Results from a specific replication can be accessed using standard Python indexing; for instance, `eliobj["results"][2]` retrieves the results from the third replication.

The `history` attribute records detailed information at each iteration across all epochs, including quantities such as loss values, hyperparameter values, and iteration durations. These data are particularly useful for monitoring the training progress and conducting convergence diagnostics.

In contrast, the `results` attribute stores results only once, upon completion of the fitting process. These include, for example, simulated prior samples, samples drawn from the generative model, and simulated elicited statistics. Such outputs are valuable for evaluating the learned prior distributions and for analyzing the model-implied quantities that result from these priors.

To facilitate result inspection, **elicito** provides a set of built-in graphical evaluation tools implemented in the `el.plots` module. Among others, `el.plots.loss()` and `el.plots.hyperparameter()` allow for visualization of the learning progress over time, `el.plots.elicits()` enables comparison between model-implied and expert-elicited statistics, and `el.plots.prior_joint()`, `el.plots.prior_marginals()`, and `el.plots.prior_averaging()` allow for examining the learned prior distributions.

4. Case Study

In this section, we illustrate the use of the **elicito** package through two toy examples. We begin with a scenario in which independent parametric prior distributions are assumed, and subsequently discuss the modifications required to accommodate a non-parametric joint prior distribution. The complete implementation of each toy example is provided in the supplementary material hosted on GitHub at https://github.com/florence-bockting/elicito-software-paper.

4.1. Example 1: Independent parametric priors

For the following case study, we consider a linear regression model with a normally distributed likelihood and a three-level, dummy-coded categorical predictor. The corresponding statistical model ($\mathcal{M}$) comprises four parameters, $\boldsymbol{\theta}$: the intercept (β_0), two contrast coefficients associated with the categorical predictor (β_1 and β_2), and the residual standard deviation (σ).

Formally, the model $\mathcal{M}$ is defined as

$$\begin{aligned} \mu_i &= \beta_0 + \beta_1 X_1 + \beta_2 X_2 \\ y_i &\sim \text{Normal}(\mu_i, \sigma) \end{aligned} \tag{$\mathcal{M}$}$$

where the observations y_i are drawn from a normal distribution centered at the predicted mean μ_i. In this first example, we assume independent parametric prior distributions for the model parameters $\boldsymbol{\theta}$. Specifically, the regression coefficients are assigned normal priors, while the residual standard deviation σ is assigned a half-normal prior:

$$\begin{aligned} \beta_j &\sim \text{Normal}(\mu_j, \sigma_j) \quad \text{for } j = 0, 1, 2 \\ \sigma &\sim \text{HalfNormal}(\sigma_3). \end{aligned} \tag{$\mathcal{P}_{\text{parametric}}$}$$

Under this specification, the objective is to learn a set of seven hyperparameters, denoted by $\lambda = (\mu_0, \sigma_0, \mu_1, \sigma_1, \mu_2, \sigma_2, \sigma_3)$.

Next, we define the set of quantities to be elicited from the domain expert. We assume that the expert is able to express expectations regarding the dependent variable's outcome within each of the three groups, denoted as $y_i \mid \text{gr}_k$ for $k = 1, 2, 3$. In addition, we include the expected coefficient of determination (R^2), representing the proportion of variance explained by the predictor. Together, these four quantities form the set of *target quantities*, denoted as $z_i = (z_1, z_2, z_3, z_4)$. To elicit this information from a domain expert, we employ a quantile-based elicitation approach for all four target quantities. Specifically, the expert is asked to provide the 25th, 50th, and 75th percentiles, denoted as Q_{25}, Q_{50}, and Q_{75}, for each target quantity. This procedure yields a set of four *elicited statistics*, denoted as $\{t_m\} = (t_1, t_2, t_3, t_4)$ where

$$\begin{aligned} t_1 &= (Q_{25}(z_1), Q_{50}(z_1), Q_{75}(z_1)) \quad \text{with } z_1 = y_i \mid \text{gr}_1, \\ &\dots \\ t_4 &= (Q_{25}(z_4), Q_{50}(z_4), Q_{75}(z_4)) \quad \text{with } z_4 = R^2. \end{aligned}$$

To estimate the seven hyperparameters λ, the optimization procedure searches for values that minimize the discrepancy between the expert-elicited and model-simulated sets of elicited statistics, t_m. Based on this setup, we now describe how to specify the corresponding `Elicit` class using the **elicito** package, assuming it has been imported in Python via the alias `el`.

Generative model

Information about the generative model is specified via the `model` argument of the `Elicit` class. In the current example, the data-generating process is implemented through the `GenerativeModel` class, which returns information about the specified target quantities z_i.

```
class GenerativeModel:
    def __call__(self, prior_samples, design_matrix, n_gr):
        # extract prior samples per parameter type
        betas=prior_samples[:,:,:-1]
        sigma=prior_samples[:,:,-1][:,:,None]

        # linear predictor
```

```
        mu = tf.matmul(betas, design_matrix, transpose_b=True)

        # data-generating model
        y = tfd.Normal(loc=mu, scale=sigma).sample()

        # selected observations per group
        (y_gr0, y_gr1, y_gr2) = (y[:, :, i::] for i,j in zip(
                                 [0,n_gr,2*n_gr],[n_gr,2*n_gr,-1]))

        return dict(y_gr0=y_gr0, y_gr1=y_gr1, y_gr2=y_gr2,
                    mu=mu, y=y)
```

The corresponding `model` argument of the `Elicit` class is then specified as follows:

```
model=el.model(
    obj=GenerativeModel,
    n_gr=30,
    design_matrix=design_categorical(n_gr=30)
  )
```

The design matrix is constructed using the user-defined function `design_categorical()`, whose implementation is provided in Appendix B.

Prior specifications

Next, the prior assumptions $\mathcal{P}_{\text{parametric}}$ are incorporated by specifying the `parameters` argument of the `Elicit` class. In this example, we assume independent parametric priors for each of the four model parameters θ.[1]

```
parameters=[
        el.parameter(
          name=f"beta{i}",
          family=tfd.Normal,
          hyperparams=dict(loc=el.hyper(f"mu{i}"),
                           scale=el.hyper(f"sigma{i}", lower=0))
        ) for i in range(3)
      ]+[
        el.parameter(
          name="sigma",
          family=tfd.HalfNormal,
          hyperparams=dict(scale=el.hyper("sigma3", lower=0))
        ),
      ]
```

In this implementation, constraints are applied to each scale hyperparameter σ_i by setting `lower=0`, thereby ensuring non-negativity.

[1] For readers unfamiliar with Python syntax, the expression `[x for x in range(3)]` represents a *list comprehension*. This approach offers a more compact alternative to traditional for-loops. Furthermore, the addition of two lists using the `+` operator (i.e., `list() + list()`) results in their concatenation.

Target quantities and elicitation techniques

Information about each of the four target quantities z_i with respective elicitation technique is provided through the `targets` argument of the `Elicit` class.

```
targets=[
    el.target(
      name=f"y_gr{i}",
      query=el.queries.quantiles(quantiles=(.25, .50, .75)),
      loss=el.losses.MMD2(kernel="energy"),
    ) for i in range(3)
    ]+[
    el.target(
      name="r2",
      query=el.queries.quantiles(quantiles=(.25, .50, .75)),
      loss=el.losses.MMD2(kernel="energy"),
      target_method=r2,
      weights=10.
    )
  ]
```

The target quantities $y_i \mid \mathrm{gr}_k$ for $k = 1, 2, 3$ are extracted directly from the output dictionary of the `GenerativeModel`, while the target quantity R^2 is computed using a custom function that takes the returned values `y` and `mu` from the `GenerativeModel` as input parameters.

```
def r2(y, mu):
  return tf.divide(
    tf.math.reduce_variance(mu, axis=-1),
    tf.math.reduce_variance(y, axis=-1)
  )
```

The discrepancy between each model-implied statistic t_m and its corresponding expert-elicited counterpart $\hat{t}_m$ is quantified using the MMD^2 loss with an energy kernel. The loss weights are set to unity ($w_1 = w_2 = w_3 = 1$) for the first three statistics, while a higher weight is assigned to R^2 ($w_4 = 10$).

Expert data

Having specified the generative model and the set of target quantities as well as elicited statistics, the corresponding quantities can be elicited from a domain expert. For the current example, we consider the following expert data as input:

```
expert = el.expert.data(
    dat={'quantiles_y_gr0': [0.64, 1.22, 1.89],
         'quantiles_y_gr1': [0.72, 1.39, 2.07],
         'quantiles_y_gr2': [0.76, 1.48, 2.22],
         'quantiles_r2':    [0.07, 0.23, 0.61]}
    )
```

Optimization procedure

The remaining arguments of the `Elicit` class (i.e., `optimizer`, `trainer`, and `initializer`) are used to configure the optimization procedure itself. For the current example, we employ mini-batch SGD with the Adam optimizer and a constant learning rate of 0.05, specified via `el.optimizer()` in the `optimizer` argument of `Elicit`:

```
optimizer=el.optimizer(
    optimizer=tf.keras.optimizers.Adam,
    learning_rate=0.05
)
```

Additional configurations are passed via the `trainer` argument of the `Elicit` class. These include the number of training epochs, the seed for reproducibility, and the specification of which approach is used to learn the prior distributions. In the present example, we employ the `"parametric_prior"` approach, which does not utilize a deep generative model to learn the priors:

```
trainer=el.trainer(
    method="parametric_prior",
    seed=1234,
    epochs=600
)
```

Finally, the initial hyperparameters $\boldsymbol{\lambda}^0$ must be specified to initiate the simulation-based training procedure. In the following example, 32 hyperparameter vectors are sampled from a Mv-Uniform$(0, 2)$ distribution using a quasi-random sampling approach (i.e., Sobol sampling). For each sampled hyperparameter vector, the corresponding loss value is computed by running the elicitation method in forward mode. The hyperparameter vector $\boldsymbol{\lambda}^*$ that yields the minimum loss is selected as the set of initial values $\boldsymbol{\lambda}^0$:

```
initializer=el.initializer(
    method="sobol",
    iterations=32,
    distribution=el.initialization.uniform(radius=2., mean=0.)
)
```

With these settings, the `Elicit` class is fully specified and can be used to instantiate an `eliobj` object:

```
eliobj = el.Elicit(
    model=model,
    parameters=parameters,
    targets=targets,
    expert=expert,
    optimizer=optimizer,
    trainer=trainer,
    initializer=initializer
)
```

Fitting and inspection of results

In the current example, we run five instances of the optimization procedure in parallel to evaluate the sensitivity of the results to different initializations.

```
eliobj.fit(parallel=el.utils.parallel(runs=5))
```

Inspecting results The results of the optimization process are stored in `eliobj.results` and `eliobj.history`, both of which are nested lists, where each sublist corresponds to a single replication. The following example demonstrates how to inspect the results from the first optimization run.

```
> eliobj.results[0].keys()

dict_keys(['target_quantities', 'elicited_statistics', 'prior_samples',
    'model_samples', 'loss_tensor_expert', 'loss_tensor_model',
    'expert_elicited_statistics', 'expert_prior_samples',
    'init_loss_list', 'init_prior', 'init_matrix', 'seed'])

> eliobj.history[0].keys()

dict_keys(['loss', 'loss_component', 'time', 'hyperparameter',
    'hyperparameter_gradient'])
```

Convergence analyses The training progress can be examined using the graphical tools provided by the `el.plots` module. First, we visualize the trajectory of the total loss and the individual loss components, each corresponding to one of the elicited statistics, across epochs. Since multiple optimization runs were conducted in parallel, the results from all runs are displayed simultaneously in Figure 1.

```
el.plots.loss(eliobj)
```

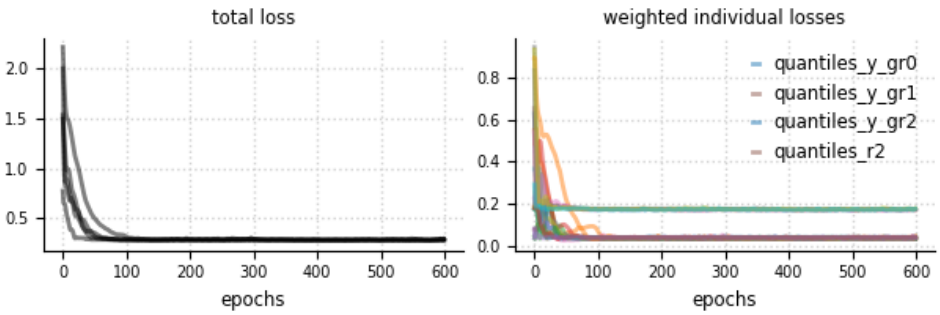


Figure 1: Convergence of total loss (left panel) and loss per component (right panel) across epochs for all parallel runs. Plot is created using `el.plots.loss(eliobj)`. The plots demonstrate successful convergence, with loss trajectories stabilizing at values near zero.

In addition to evaluating the convergence of the loss components, it is informative to examine the convergence behavior of the individual hyperparameters λ which is shown in Figure 2.

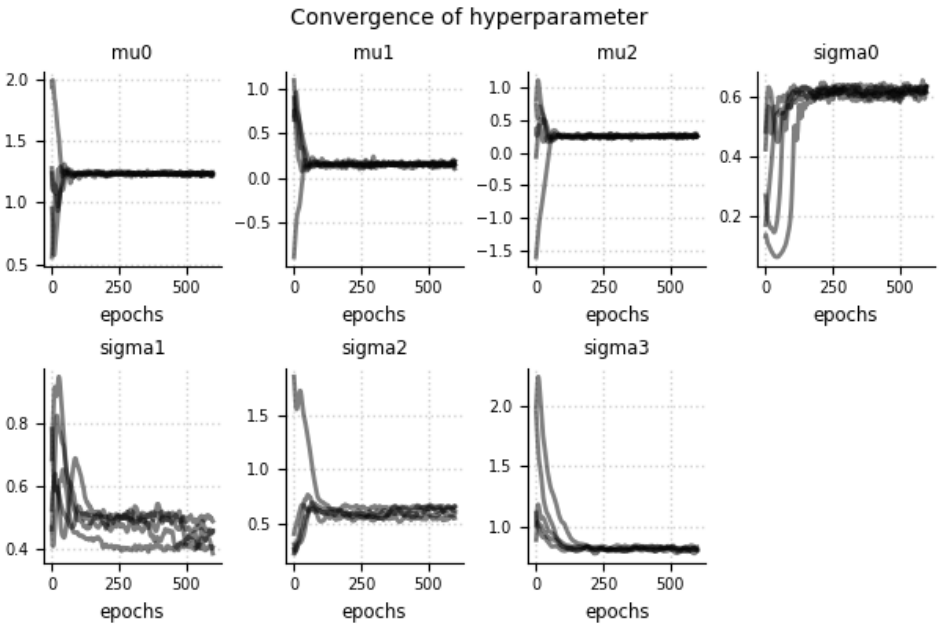


Figure 2: Convergence of hyperparameters λ across epochs for all parallel runs. Plot is created using `el.plots.hyperparameter(eliobj)`. The learning trajectory of almost all hyperparameters stabilizes over time, with parallel runs converging towards the same value. Some difficulties of the optimization algorithm in learning σ_1 can be observed.

```
el.plots.hyperparameter(eliobj)
```

Finally, we can also inspect the agreement between the expert-elicited statistics, $\hat{t}_m$, and the model-generated statistics, t_m as visualized in Figure 3.

```
el.plots.elicits(eliobj)
```

The gray dashed diagonal line in Figure 3 represents perfect agreement between model-implied and expert-elicited statistics. Deviations of the scatter points from this diagonal indicate discrepancies between the expert-elicited values and the statistics derived from the learned priors.

Learned prior distributions Upon confirming the successful learning of the optimization algorithm across all parallel runs, we proceed with examining the learned prior distributions corresponding to the elicited statistics. With `el.plots.prior_marginals(eliobj)` we can plot the learned marginal prior distributions for each model parameter as depicted in Figure 4.

```
el.plots.prior_marginals(eliobj)
```

Finally, a prior average across replications can be computed using `el.plots.prior_averaging(eliobj)` with results depicted in Figure 5. However, in the current example, this analysis offers limited additional insight, as the replications exhibit minimal variation.

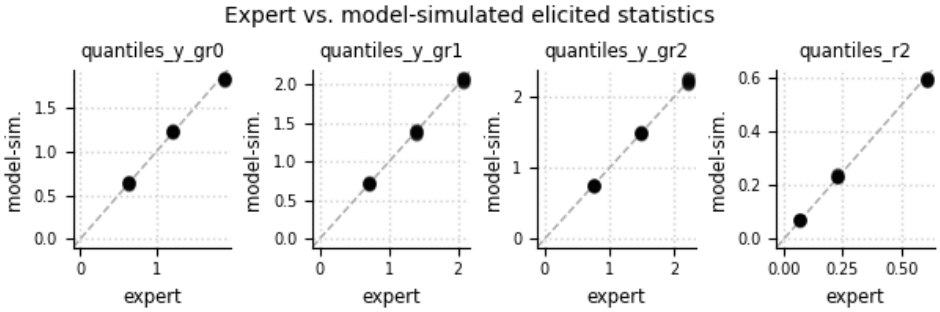


Figure 3: Comparison of expert-elicited and model-simulated statistics for all parallel runs. Plot is created using `el.plots.elicits(eliobj)`. For the current example, a perfect match is observed between the model-simulated and expert-elicited statistics for all four elicited statistics, as indicated by the scatter points lying along the diagonal line.

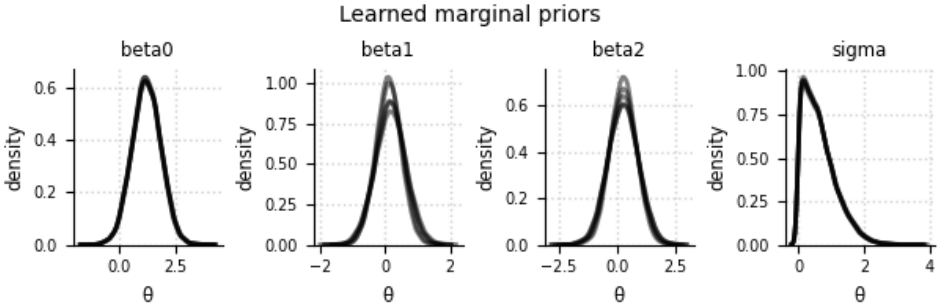


Figure 4: Learned marginal prior distribution for all parallel runs. The plot is generated using `el.plots.prior_marginal(eliobj)`.

```
el.plots.prior_averaging(eliobj)
```

The upper panel in Figure 5 displays the weights assigned to each replication during the averaging procedure, while the lower panel shows the corresponding marginal prior distributions. The averaged prior distribution is highlighted in red.

4.2. Example 2: Joint non-parametric prior

In the remainder of this section, we retain the previously introduced regression model $\mathcal{M}$ but change the assumptions regarding the prior distributions of the model parameters. Specifically, we assume a joint, non-parametric prior distribution of the form

$$(\beta_0, \beta_1, \beta_2, \sigma) \sim p_\lambda(\cdot). \qquad (\mathcal{P}_{\text{non-parametric}})$$

The hyperparameters λ represent now the weights of the deep generative model used to learn the complex joint prior density function. In the following, we revisit the specification of the `Elicit` class, focusing exclusively on aspects that differ from the previously discussed parametric case. For clarity, certain code-snippets from the previous example are repeated. In such instances, the symbol `>>` is used to indicate where exactly modifications relative to the previous example were made.

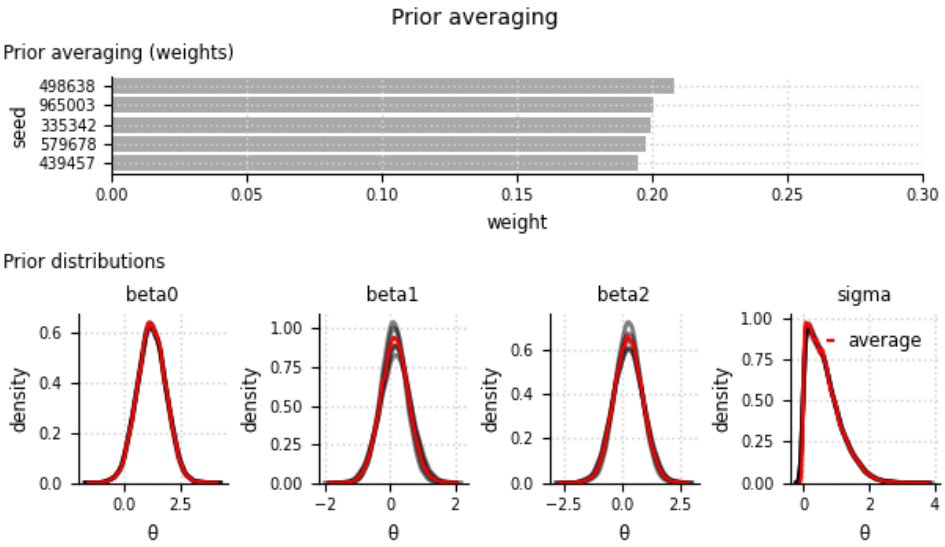


Figure 5: Prior averaging across multiple parallel runs with average prior highlighted in red. Upper panel shows weight associated with each replication used in model averaging. Lower plots shows the individual marginal prior distributions as well as the averaged prior distribution. Plot is created using `el.plots.prior_averaging(eliobj)`.

Prior specifications The modified assumptions about the model parameters, $\mathcal{P}_{\text{non-parametric}}$, are incorporated through the `parameters` argument of the `Elicit` class. In this case, only the specification of the parameter names and their constraints is required.

```
parameters_deep=[
        el.parameter(name=f"beta{i}") for i in range(3)
    ]+[
        el.parameter(name="sigma", lower=0),
    ]
```

Elicited statistics and expert data The same expert information as described in the previous example is used in this case. Additionally, the independence assumption among the model parameters is explicitly incorporated as a loss component by specifying an additional `el.target()` function. The `targets` and `expert` arguments of the `Elicit` class are specified as follows:

```
targets_deep = targets + [
>>    el.target(
        name="cor",
        query=el.queries.correlation(),
        loss=el.losses.L2,
        weight=0.1
    )
]
```

```
expert_deep = el.expert.data(
    dat={'quantiles_y_gr0': [0.64, 1.22, 1.89],
         'quantiles_y_gr1': [0.72, 1.39, 2.07],
         'quantiles_y_gr2': [0.76, 1.48, 2.22],
         'quantiles_r2': [0.07, 0.23, 0.61],
>>       'cor_cor': [0.]*6
     }
)
```

To quantify the deviation of the simulation-based correlations from zero (model parameters are assumed to be independent), the L_2 loss is used with an associated weight of 0.1.

Optimization procedure The other adjustments refer to the optimization procedure itself (i.e., `optimizer`, `trainer`, and `network`). We use again mini-batch SGD as optimization algorithm, but for the deep generative model approach typically smaller learning rates are required

```
  optimizer_deep=el.optimizer(
      optimizer=tf.keras.optimizers.Adam,
>>    learning_rate=0.001
  )
```

For training, it is now necessary to specify that a deep generative model should be used to learn the prior, which is achieved by setting the approach to `"deep_prior"`. Additionally, the number of training epochs is increased, as deep generative models typically require more iterations to converge.

```
  trainer_deep=el.trainer(
>>    method="deep_prior",
      seed=2025,
>>    epochs=800
  )
```

Finally, we have to setup the deep-generative model via the `network` argument of the `Elicit` class. For the current example we use NFs implement via `el.networks.NF()`. We employ a relatively simple NF architecture, consisting of a standard multivariate Gaussian as base distribution and three affine coupling blocks. Each coupling block comprises two dense layers with 128 units and ReLU activation functions:

```
network=el.networks.NF(
    inference_network=el.networks.InvertibleNetwork,
    network_specs=dict(
        num_params=4,
        num_coupling_layers=3,
        coupling_design="affine",
        coupling_settings={
            "dense_args": {
```

```
                "units": 128,
                "activation": "relu"
            },
            "num_dense": 2,
        },
    ),
    base_distribution=el.networks.base_normal
)
```

With these modifications in place, we are ready to execute the elicitation procedure under the non-parametric prior assumption. Rather than reinitializing the `Elicit` class and creating a new `eliobj` instance, we update the existing `eliobj` instance created in the previous example and proceed to fit the updated object accordingly.

```
import copy

# copy existing eliobj
eliobj_deep = copy.deepcopy(eliobj)

# update eliobj
eliobj_deep.update(
    parameters=parameters_deep,
    targets=targets_deep,
    optimizer=optimizer_deep,
    trainer=trainer_deep,
    initializer=None,
    network=network
    )

# fit updated eliobj
eliobj_deep.fit(parallel=el.utils.parallel(runs=5))
```

Fitting and inspection of results

The fitting procedure and evaluation tools remain consistent with those described previously, with only minor modifications. In the following, we focus exclusively on the differences; the remaining plots are provided in Appendix C.

The function `el.plots.hyperparameter(eliobj)` visualizes the convergence of the mean and standard deviation for each marginal prior distribution instead of the hyperparameter values λ, as depicted in Figure 6.

```
el.plots.hyperparameter(eliobj_deep)
```

While the model reliably learns the locations of the prior distributions between the parallel runs, substantially greater variability is observed in the learned scales. This suggests that the information provided as input is insufficient to accurately inform the scale of the prior distributions, resulting in a lack of identifiability.

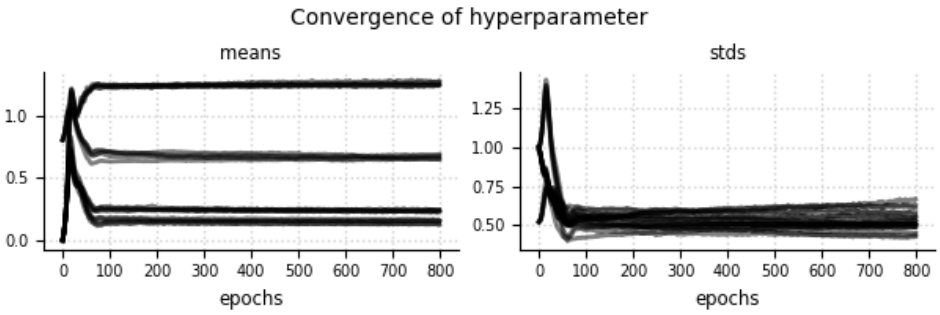


Figure 6: Convergence of mean and standard deviation of marginal prior distributions for all parallel runs. Plot is created using `el.plots.hyperparameter(eliobj_deep)`.

The comparison between the simulated and expert-elicited statistics is shown in Figure 7 and includes now also the correlation information.

```
el.plots.elicits(eliobj_deep)
```

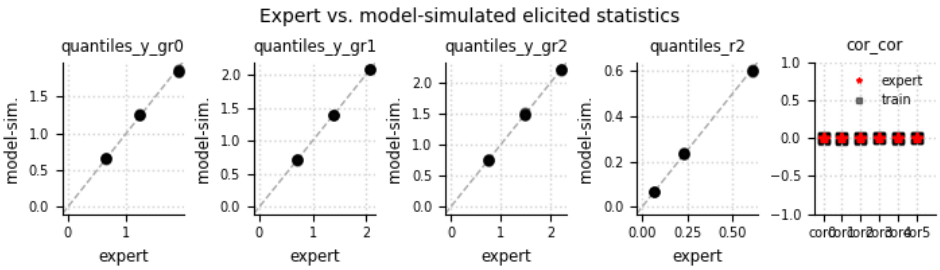


Figure 7: Comparison between simulated and expert-elicited statistics. Plot is created using `el.plots.elicits(eliobj_deep)`. The first four plots (from the left) compare the model-implied and expert-elicited quantiles, while the right-most plot illustrates the independence assumptions among the model parameters, quantified via pairwise correlations. For all elicited statistics the expected and learned values exhibit a perfect match.

Finally, we can inspect the learned joint prior distribution using `el.plots.prior_joint(eliobj_deep, idx=list(range(5)))` depicted in Figure 8.

```
el.plots.prior_joint(eliobj_deep, idx=list(range(5)))
```

5. Related software

Over the past decades, numerous prior elicitation methods have been proposed, many accompanied by specialized software implementations. Early examples of such tools, though no longer actively maintained, include **ElicitN** for expert knowledge elicitation on species richness and related count data (Fisher, O'Leary, Low-Choy, Mengersen, and Caley 2012) or **Elicitator** for regression analysis in ecological applications (James, Choy, and Mengersen 2010).

More recent examples, include among others the R package **DTEAssurance** (Salsbury, Oakley, Julious, and Hampson 2024), which implements the assurance method for delayed treatment

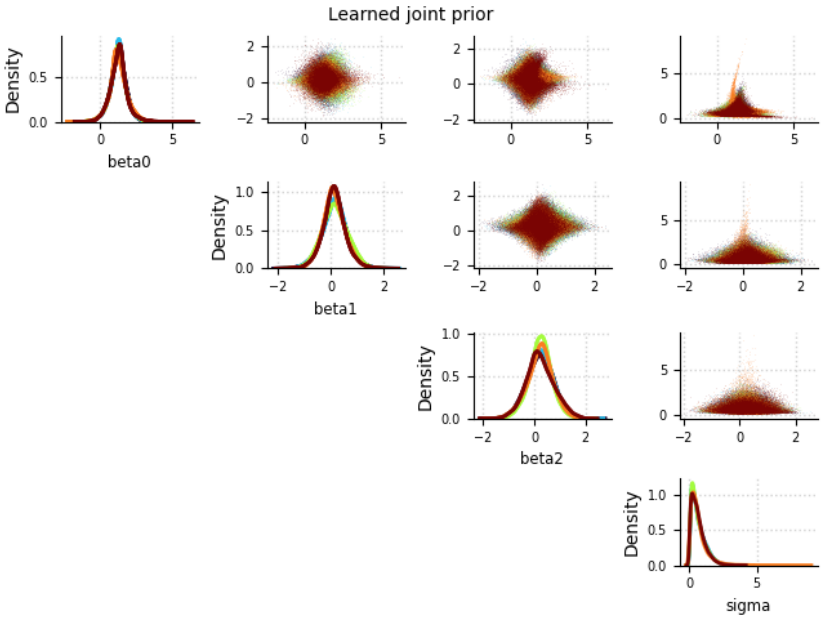


Figure 8: Learned prior distributions including information about correlation structure. The plot is created using `el.plots.prior_joint(eliobj_deep, idx=list(range(5)))`.

.

effects by eliciting prior distributions for both the delay duration and post-delay hazard ratio. Similarly, the **assurance** package (Alhussain and Oakley 2020) provides an implementation for normally distributed data, incorporating elicited prior distributions for treatment effects along with population variances in both treatment and control arms. More general approaches are offered for example via the **eglm** package (Hosack 2024), which enables the elicitation of prior distributions for Bayesian generalized linear models and the **makemyprior** package (Hem, Fuglstad, and Riebler 2024) for constructing joint priors for variance parameters in latent Gaussian models.

Despite these developments, general-purpose packages for expert prior elicitation remain relatively scarce (Mikkola *et al.* 2023). Below, we present a selection of actively maintained tools that we consider most relevant to our current application.

The R package **pbbo** (Manderson and Goudie 2023) implements a simulation-based prior elicitation method using multi-objective Bayesian optimization. It enables elicitation based on expert-specified prior predictive distributions and model-derived quantities such as R^2. The Python package **PreliZ** (Icazatti, Abril-Pla, Klami, and Martin 2023) provides a toolbox for elicitation and exploration of prior distributions. The package supports both structural and predictive elicitation through four core components: prior distribution specification and manipulation, visualization of priors and their predictive distributions, interactive user interfaces, and algorithms for prior inference from user-provided information. The R package **SHELF** (Oakley 2025) provides a comprehensive toolbox for prior elicitation within the Sheffield Elicitation Framework. It facilitates structured elicitation protocols involving single or multiple experts by offering multiple univariate elicitation methods (including roulette, bisection, and range techniques), support for multivariate parameter elicitation (such as vectors of proportions), and distribution fitting via least-squares minimization. It incorporates interactive

feedback tools during elicitation sessions and is designed primarily as a facilitator's toolkit to support the Sheffield elicitation protocol. Finally, the R package **qdp** (Perepolkin *et al.* 2025) introduces quantile and quantile-parameterized distributions as a flexible and interpretable approach to prior specification. These distributions allow expert knowledge to be expressed directly in the observable space, enhancing transparency and interpretability. The associated R package provides tools for defining and inverting quantile functions.

Each of these packages adopts a distinct perspective and can be viewed as complementary to **elicito**. The aim of our package is to propose a unified workflow for prior elicitation, structured around a fixed set of interconnected modules, corresponding to the arguments of the `Elicit` class. The functionality of each module is designed to be extensible, allowing for integration with other packages where appropriate. For example, future versions of **elicito** could incorporate Bayesian optimization techniques, such as those implemented in **pbbo**. Similarly, the loss regularization methods proposed by Manderson and Goudie (2023) represent a promising avenue for enhancing the loss module within **elicito**. The **PreliZ** package also complements **elicito** by offering exploratory tools that can enrich both the setup and evaluation stages of the elicitation process. Additionally, future extensions may include native support for quantile and quantile-parameterized priors, as implemented in **qdp**.

6. Discussion and Conclusion

The Python package **elicito** implements a modular workflow that encompasses the essential steps of the expert prior elicitation process. It provides a structured approach to the challenging task of formulating prior distributions that accurately reflect expert knowledge. To address this task, we use a simulation-based approach: (1) knowledge about desired quantities (target quantities) is extracted from a domain expert, (2) a forward model comprising of the generative model and the computation of target quantities, simulates the expert knowledge based on a set of parameters, (3) an optimization algorithm adjusts parameters to minimize the discrepancy between the simulated and observed expert knowledge; and (4) the resulting set of parameters defines the model parameter priors corresponding to the expert knowledge. Currently, **elicito** addresses this optimization task using SGD. Future developments should explore alternative optimization strategies, such as Bayesian optimization (see for example, Manderson and Goudie 2023), to enable comparative performance assessment of the resulting solutions.

Due to its modular structure and the resulting flexibility, **elicito** facilitates the implementation of a wide variety of prior elicitation methods. This modular structure enables users to easily and transparently adjust assumptions throughout the elicitation process. This includes modifying expert input, altering loss components, or incorporating post-hoc evaluation procedures. This possibility supports a systematic exploration of the elicitation problem under varying assumptions and contexts. Consequently, **elicito** may also be of interest from a methodological research perspective, as it enables the study of relationships between different input–output structures for a given problem.

Accordingly, the target audience of **elicito** includes both researchers developing prior elicitation methodologies and applied practitioners aiming to construct prior distributions for specific use cases. While the current implementation of **elicito** remains relatively technical, future development will focus on creating higher-level interfaces (i.e., facades) tailored to common

practical applications. These interfaces aim to simplify user interaction with the core `Elicit` object by incorporating default assumptions at appropriate stages of the workflow. Although this design choice may reduce transparency to some extent, it is expected to improve usability and enhance accessibility for end users (Simpson, Rue, Riebler, Martins, and Sørbye 2017). In this context, future work should also include the development of user guidelines for specific sub-tasks or problem types encountered in the prior elicitation process. One example is the handling of non-uniqueness which arises primarily from limitations in expert information, either in quantity or quality (López C., Barz, Körkel, and Wozny 2015). Several strategies can be employed to address this challenge. These include modifying the loss function, for example, by incorporating additional expert input or introducing structural priors via regularization, reparameterizing the statistical model to reduce the dimensionality of the parameter space or to increase the influence of the parameters on the outcome variable, and applying post-hoc methods that aggregate across the solution space to derive a representative solution that appropriately reflects the underlying uncertainty. This knowledge about solution strategies should be made available in form of user-guidelines.

A central challenge in the development of prior elicitation methods lies in balancing the use of default assumptions with the incorporation of individualized expert feedback. On one hand, elicitation methods should minimize the cognitive, temporal, and financial demands placed on experts (Mikkola *et al.* 2023). On the other hand, maintaining a "human-in-the-loop" is essential to ensure that the resulting prior distributions align with the expert's expectations (Hartmann *et al.* 2020). Expressing prior beliefs in quantitative terms is inherently non-trivial. An iterative feedback mechanism is therefore essential, enabling users to refine their input through a process of trial and error. While **elicito** represents the key steps of the prior elicitation workflow as modular, algorithmic components, it is not intended to fully automate the elicitation process. Instead, its primary objective is to make explicit the input information required to construct prior distributions, thereby facilitating a systematic and transparent approach to this inherently complex task.

Finally, **elicito** offers substantial scope for future enhancements and continued development. Two key directions for extension are particularly noteworthy. First, expanding compatibility with multiple probabilistic programming languages is a central objective. At present, user-defined functions must rely on the TensorFlow backend to ensure integration within the computational graph and enable correct backpropagation. However, the conceptual foundation of our simulation-based framework should ideally remain independent of any specific implementation, so as not to constrain the potential user base or limit interoperability. The second major extension involves transforming the current workflow into a fully Bayesian framework, thereby enabling the inference of posterior distributions over hyperparameters. This would enable a more accurate representation of uncertainties introduced throughout the elicitation process, from initial knowledge extraction to the fitting of prior distributions.

Declarations

Acknowledgments We thank Luna Fazio for the long discussions with valuable comments and suggestions that greatly improved this work.

Funding Not applicable.

Competing interests The authors have no competing interests to declare that are relevant to the content of this article.

Consent to participate Not applicable.

Consent for publication Not applicable.

Data & Materials availability The simulation results and notebooks with an implementation of the case study are available in the supplementary material on GitHub https://github.com/florence-bockting/elicito-software-paper.

Code availability Code underlying the case stduy is based on/provided by our Python package *elicito* (Bockting and Bürkner 2025, v0.6.0). To ensure the reproducibility of our results, the specific version of *elicito* used in this study has been archived on Zenodo and is accessible via the following DOI https://doi.org/10.5281/zenodo.15671710.

Author contribution Conceptualization: FB, Methodology: FB, Software: FB, Writing (Original Draft): FB, Writing (Editing): FB, Writing (Review): FB, Visualization: FB, Supervision: PB

References

Akbarov A (2009). *Probability elicitation: Predictive approach*. Ph.D. thesis, University of Salford.

Alhussain ZA, Oakley JE (2020). "Assurance for clinical trial design with normally distributed outcomes: Eliciting uncertainty about variances." *Pharmaceutical Statistics*, **19**(6), 827–839. doi:https://doi.org/10.1002/pst.2040.

Bedrick EJ, Christensen R, Johnson W (1996). "A new perspective on priors for generalized linear models." *Journal of the American Statistical Association*, **91**(436), 1450–1460. doi:10.1080/01621459.1996.10476713.

Bockting F, Bürkner PC (2025). "elicito: A Python package for expert-prior elicitation." doi:https://doi.org/10.5281/zenodo.15671710. URL https://github.com/florence-bockting/elicito.

Bockting F, Radev ST, Bürkner PC (2024). "Simulation-based prior knowledge elicitation for parametric Bayesian models." *Scientific Report*, **14**(1), 17330. doi:10.1038/s41598-024-68090-7.

Bockting F, Radev ST, Bürkner PC (2025). "Expert-elicitation method for non-parametric joint priors using normalizing flows." *Statistics and Computing*, **35**(132). doi:10.1007/s11222-025-10665-z.

Buduma N, Locascio N (2017). *Fundamentals of deep learning: designing next-generation machine intelligence algorithms*. 1 edition. O'Reilly Media, Sebastopol, CA. ISBN 978-1-4919-2558-4.

Carpenter B, Gelman A, Hoffman MD, Lee D, Goodrich B, Betancourt M, Brubaker M, Guo J, Li P, Riddell A (2017). "Stan: A probabilistic programming language." *Journal of Statistical Software*, **76**, 1–32. doi:10.18637/jss.v076.i01.

da Silva EdS, Kuśmierczyk T, Hartmann M, Klami A (2019). "Prior specification via prior predictive matching: Poisson matrix factorization and beyond." *Journal of Machine Learning Research*. URL http://arxiv.org/abs/1910.12263v1.

Denham R, Mengersen K (2007). "Geographically Assisted Elicitation of Expert Opinion for Regression Models." *Bayesian Analysis*, **2**(1), 99–136. doi:10.1214/07-BA205.

Dias LC, Morton A, Quigley J (eds.) (2018). *Elicitation: The Science and Art of Structuring Judgement*, volume 261 of *International Series in Operations Research & Management Science*. Springer. doi:10.1007/978-3-319-65052-4.

European Food Safety Authority (2014). "Guidance on expert knowledge elicitation in food and feed safety risk assessment." *EFSA Journal*, **12**(6), 3734. doi:10.2903/j.efsa.2014.3734.

Falconer JR, Frank E, Polaschek DL, Joshi C (2022). "Methods for eliciting informative prior distributions: A critical review." *Decision Analysis*, **19**(3), 189–204. doi:10.1287/deca.2022.0451.

Falconer JR, Frank E, Polaschek DL, Joshi C (2023). "Eliciting informative priors by modeling expert decision making." *Decision Analysis*, **21**(2), 77–90. doi:https://doi.org/10.1287/deca.2023.0046.

Figurnov M, Mohamed S, Mnih A (2018). "Implicit reparameterization gradients." *Advances in neural information processing systems*, **31**. doi:10.5555/3326943.3326984.

Fisher R, O'Leary RA, Low-Choy S, Mengersen K, Caley MJ (2012). "A software tool for elicitation of expert knowledge about species richness or similar counts." *Environmental Modelling & Software*, **30**, 1–14. doi:https://doi.org/10.1016/j.envsoft.2011.11.011.

Garthwaite PH, Kadane JB, O'Hagan A (2005). "Statistical Methods for Eliciting Probability Distributions." *Journal of the American Statistical Association*, **100**(470), 680–701. doi:10.1198/016214505000000105.

Gelman A, Goodrich B, Gabry J, Vehtari A (2019). "R-squared for Bayesian regression models." *The American Statistician*, pp. 307–309. doi:10.1080/00031305.2018.1549100.

Glorot X, Bengio Y (2010). "Understanding the difficulty of training deep feedforward neural networks." In YW Teh, M Titterington (eds.), *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pp. 249–256. PMLR. URL http://proceedings.mlr.press/v9/glorot10a.

Goodfellow I, Bengio Y, Courville A (2016). *Deep learning*. MIT press. ISBN 9780262035613.

Gretton A, Borgwardt KM, Rasch MJ, Schölkopf B, Smola A (2012). "A kernel two-sample test." *The Journal of Machine Learning Research*, **13**(1), 723–773. doi:10.5555/2188385.2188410.

Haines N, Sullivan-Toole H, Olino T (2023). "From Classical Methods to Generative Models: Tackling the Unreliability of Neuroscientific Measures in Mental Health Research." *Biological Psychiatry: Cognitive Neuroscience and Neuroimaging*, **8**(8), 822–831. doi:10.1016/j.bpsc.2023.01.001.

Hartmann M, Agiashvili G, Bürkner P, Klami A (2020). "Flexible prior elicitation via the prior predictive distribution." In *Conference on Uncertainty in Artificial Intelligence*, pp. 1129–1138. PMLR. URL http://proceedings.mlr.press/v124/hartmann20a.html.

Hem I, Fuglstad GA, Riebler A (2024). "makemyprior: Intuitive Construction of Joint Priors for Variance Parameters in R." *Journal of Statistical Software*, **110**(3), 1–39. doi:10.18637/jss.v110.i03.

Hosack GR (2024). "eglm: Elicitation for Generalised Linear Models (GLMs) and Extensions." URL https://doi.org/10.5281/zenodo.13858701.

Icazatti A, Abril-Pla O, Klami A, Martin OA (2023). "PreliZ: A tool-box for prior elicitation." *Journal of Open Source Software*, **8**(89), 5499. doi:10.21105/joss.05499.

James A, Choy SL, Mengersen K (2010). "Elicitator: An expert elicitation tool for regression in ecology." *Environmental Modelling & Software*, **25**(1), 129–145. doi:https://doi.org/10.1016/j.envsoft.2009.07.003.

Jang E, Gu S, Poole B (2017). "Categorical Reparameterization with Gumbel-Softmax." In *International Conference on Learning Representations*. URL https://openreview.net/forum?id=rkE3y85ee.

Joo W, Kim D, Shin S, Moon IC (2020). "Generalized gumbel-softmax gradient estimator for generic discrete random variables." *Preprint at https://doi.org/10.48550/arXiv.2003.01847*.

Kadane J, Wolfson LJ (1998). "Experiences in elicitation." *Statistician*, **47**(1), 3–19. doi:10.1111/1467-9884.00113.

Kadane JB, Dickey JM, Winkler RL, Smith WS, Peters SC (1980). "Interactive elicitation of opinion for a normal linear model." *Journal of the American Statistical Association*, **75**(372), 845–854. doi:10.1080/01621459.1980.10477562.

Kingma DP, Welling M (2014). "Auto-Encoding Variational Bayes." *stat*, **1050**, 1. doi:10.48550/arXiv.1312.6114.

López C DC, Barz T, Körkel S, Wozny G (2015). "Nonlinear ill-posed problem analysis in model-based parameter estimation and experimental design." *Computers & Chemical Engineering*, **77**, 24–42. doi:https://doi.org/10.1016/j.compchemeng.2015.03.002.

Maddison C, Mnih A, Teh Y (2017). "The concrete distribution: A continuous relaxation of discrete random variables." In *Proceedings of the international conference on learning Representations*. International Conference on Learning Representations. doi:https://doi.org/10.48550/arXiv.1611.00712.

Manderson AA, Goudie RJ (2023). "Translating predictive distributions into informative priors." *Preprint at 10.48550/arXiv.2303.08528*.

Mikkola P, Acerbi L, Klami A (2024). "Preferential Normalizing Flows." In *Advances in Neural Information Processing Systems*. doi:10.48550/arXiv.2410.08710.

Mikkola P, Martin OA, Chandramouli S, Hartmann M, Pla OA, Thomas O, Pesonen H, Corander J, Vehtari A, Kaski S, *et al.* (2023). "Prior knowledge elicitation: The past, present, and future." *Bayesian Analysis*, **1**(1), 1–33. doi:10.1214/23-BA1381.

Muandet K, Fukumizu K, Sriperumbudur B, Schölkopf B, *et al.* (2017). "Kernel mean embedding of distributions: A review and beyond." *Foundations and Trends® in Machine Learning*, **10**(1-2), 1–141. doi:http://dx.doi.org/10.1561/2200000060.

Oakley J (2025). "SHELF: Tools to Support the Sheffield Elicitation Framework." doi:10.32614/CRAN.package.SHELF.

Perepolkin D, Goodrich B, Sahlin U (2024). "Hybrid elicitation and quantile-parametrized likelihood." *Stat. Comput.*, **34**(1), 11. doi:10.1007/s11222-023-10325-0.

Perepolkin D, Lindström E, Sahlin U (2025). "Quantile-parameterized distributions for expert knowledge elicitation." *Decision Analysis*. doi:https://doi.org/10.1287/deca.2024.0219.

Salsbury JA, Oakley JE, Julious SA, Hampson LV (2024). "Assurance methods for designing a clinical trial with a delayed treatment effect." *Statistics in Medicine*, **43**(19), 3595–3612. doi:https://doi.org/10.1002/sim.10136.

Simpson D, Rue H, Riebler A, Martins TG, Sørbye SH (2017). "Penalising Model Component Complexity: A Principled, Practical Approach to Constructing Priors." *Statistical Science*, **32**(1), 1–28. doi:10.1214/16-STS576.

Stefan AM, Evans NJ, Wagenmakers EJ (2022). "Practical challenges and methodological flexibility in prior elicitation." *Psychological Methods*, **27**(2), 177–197. doi:10.1037/met0000354.

Winkler RL (1967). "The assessment of prior distributions in Bayesian Analysis." *Journal of the American Statistical association*, **62**(319), 776–800. doi:10.1080/01621459.1967.10500894.

Appendix

A. Symbols and their definition

Symbol	*Description*	*Comment*
θ_p	model parameter	$p = 1, \ldots, P$
λ_k	model hyperparameter	$k = 1, \ldots, K$
$p_\lambda(\theta)$	priors parameterized by λ	in case of non-parameteric priors λ refer to the parameters of the deep generative model
$\mathcal{M}(\lambda)$	generative model incl. priors	for short $\mathcal{M}$
$\mathcal{P}_{pr}$	prior specification	$pr = (\text{parametric}, \text{non-parametric})$
z_i	target quantity	defined as $z_i = c_i(\lambda)$ with $i = 1, \ldots, I$
t_m	simulated elicited statistics	defined as $t_m = f_j(z_i)$ with $m = (i, j)$
f_j	elicitation technique	$j = 1, \ldots, J$
$\hat{t}_m$	expert-elicited statistic	
$\mathcal{D}_m$	discrepancy measure	
L_m	loss component	$L_m = w_m D_m(t_m(\lambda), \hat{t}_m)$
w_m	weight of loss component	
$\mathcal{L}(\lambda)$	total loss wrt λ	$\mathcal{L}(\lambda) = \sum_{m=1}^{M} L_m$
E	epochs	
B	batch size	default $B = 128$
M	number of prior samples	default $M = 200$

Table 1: List of symbols and their definitions used throughout the paper

B. Implementation of parametric-prior example

B.1. Implementation of design matrix

```
# design-matrix for three-level dummy-coded predictor
def design_categorical(n_gr):
    incpt = [1]*3
    contrast_01 = [0, 1, 0]
    contrast_02 = [0, 0, 1]

    # contrast matrix
    c = tf.stack([incpt, contrast_01, contrast_02], axis=-1)

    # design matrix
    x = tf.concat([tf.broadcast_to(c[i, :], (n_gr, c.shape[1])) for
                  i in range(c.shape[0])], axis=0)
    return tf.cast(x, tf.float32)

# example output of design matrix for n_gr=1
> design_categorical(n_gr=1)

<tf.Tensor: shape=(3, 3), dtype=float32, numpy=
array([[1., 0., 0.],
       [1., 1., 0.],
       [1., 0., 1.]], dtype=float32)>
```

C. Additional results of toy example 2

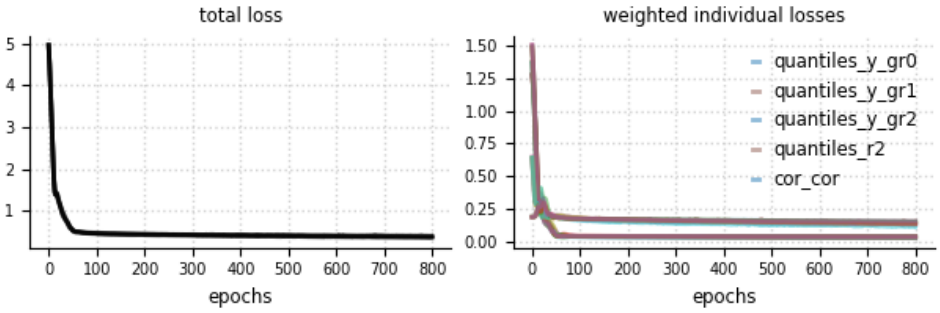


Figure 9: Convergence of the total loss (left) and the single loss components (right). Plot is created using `el.plots.loss(eliobj_deep)`.

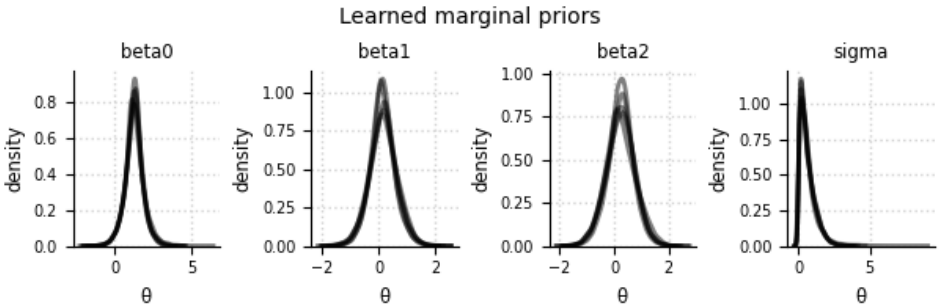


Figure 10: Learned marginal prior distribution for all parallel run. The plot is generated using `el.plots.prior_marginals(eliobj_deep)`.

Affiliation:

Florence Bockting
Department of Statistics
Chair of Computational Statistics
TU Dortmund University
44227 Dortmund, Germany
E-mail: florence.bockting@tu-dortmund.de
URL: https://florence-bockting.github.io/

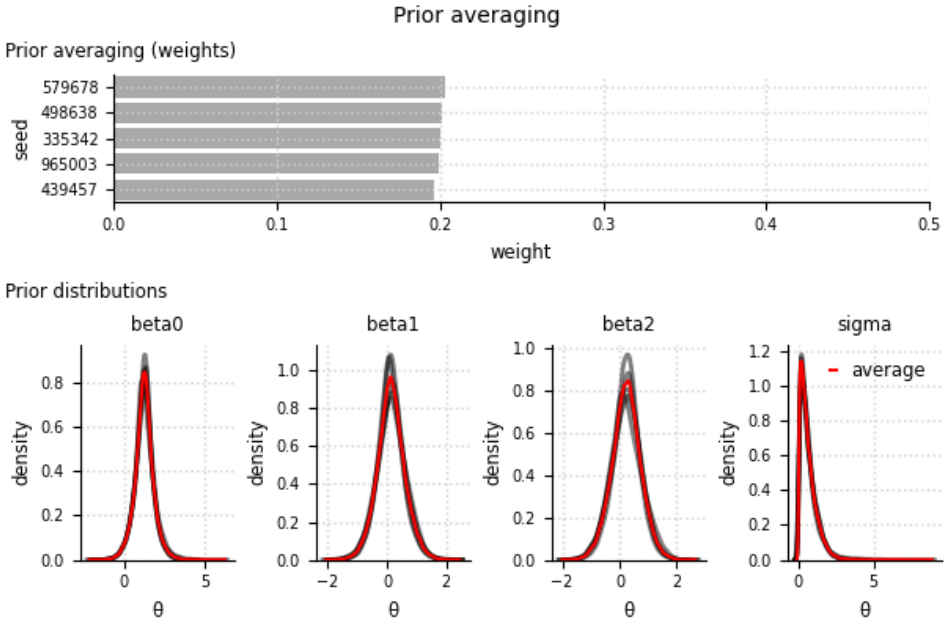


Figure 11: Prior averaging across multiple parallel runs. Upper panel demonstrates weights associated with each replication used for computing the prior average. Lower panel shows the individual marginal priors as well as the averaged prior (in red). Plot is created using `el.plots.prior_averaging(eliobj_deep)`.