BayesPharma: Bayesian methods for pharmacology models

Madeline J. Martin

Elayne Vieira Diaz

P. Walter German

Elyssa B. Margolis

Matthew J. O’Meara

In pharmacology, many experiments seek to measure how a reductive biological system responds to one or more treatments. Here, we present BayesPharma, a collection of Bayesian methods to increase the robustness of the analysis and interpretation of dose response data. BayesPharma is an R package built around the Stan ecosystem that facilitates a principled Bayesian workflow. It can be used to fit and analyze foundational pharmacology models, serve as a pedagogical framework for learning Bayesian methods, and help build and analyze sophisticated pharmacological models. We describe the BayesPharma framework and illustrate its application through several case studies.

## Introduction

As pharmacology experiments increase in complexity, it becomes increasingly challenging to analyze them. So, while there are various frameworks for fitting pharmacology models using maximum likelihood estimation, such as GraphPad Prism(GraphPad Software, LLC 2023) and the drc dose-response curves R package(Ritz et al. 2015), it can be difficult to use these robustly estimate model uncertainty and handle different different experimental designs such as sources of variation that are shared across different measurements. A principled alternative approach is to use Bayesian statistics to quantify the model uncertainty before and after data collection. The key steps for fitting a Bayesian model include (a) defining a functional form, (b) specifying a prior distribution over the model parameters and a likelihood function that describes the data generation process, (c) using Markov chain Monte Carlo (MCMC) or variational inference to combine the data, the prior, and the likelihood to generate samples from the posterior distribution over the parameters. These samples can be used directly to answer scientific questions like “what are the model parameters” and “how confident should we be?” Crucially, Bayesian modeling workflows enable incremental addition of complexity to allow for the capture of important signal in the data generation process.

While historically, building Bayesian models has been challenging, requiring specialized statistical and computational skills, there has been substantial progress in general computational frameworks to facilitate developing and applying Bayesian models. A key example is the Stan package and the ecosystem of supporting tools. Stan provides a domain-specific language to describe probabilistic models, inference engines, front-end interfaces through many common programming languages, and a suite of tools to analyze fit models(Carpenter et al. 2017; Bürkner 2017; Vehtari, Gelman, and Gabry 2017a; Gabry and Mahr 2017; Kay 2018; Wickham 2009; Wickham et al. 2019; Team and Others 2013). Among the front-end interfaces, the Bayesian Regression Modeling using Stan (BRMS) package in R facilitates rapid model development through formulas similar to other regression models in R like lm() and the mixed effects regression package lme4(R Core Team 2023; Bates et al. 2015). BRMS not only supports defining linear and nonlinear predictors, but also hierarchical models and a range of pre-specified or custom response functions. Once specified as formulas, brms transforms the model through a series of different representations to allow them to be run on computer hardware. Once the model has run, it returns a set of samples from the posterior distribution, which can be used to analyze the model fit and make inferences.

In pharmacology, dose-response modeling through the foundational sigmoidal Hill-equation, Michaelis Menten enzyme kinetics, and multi-drug synergy models are widely used to probe biological systems and develop therapeutics. A limitation of the current tools is that it is currently not straightforward for practitioners to implement and analyze these types of models using Bayesian statistics. To fill this gap, here we developed the BayesPharma R package, a Bayesian pharmacology modeling framework that implements foundational pharmacology models so that they can be easily used through brms, Stan, and other tools in the Stan ecosystem. After reviewing related work and basic Bayesian modeling concepts, we describe the package architecture and demonstrate the utility through several case studies.

### Related work

Broadly, the increase in maturity in Bayesian modeling frameworks is making Bayesian modeling more accessible to practitioners, where it has seen a steady increase in popularity across the social sciences, econometrics, and biostatistics. For general Bayesian modeling theory, there are many excellent textbooks(Gelman et al. 2013; McElreath 2016; Gelman and Hill 2006; Johnson, Ott, and Dogucu 2022), online resources(Betancourt 2023; Posit Software, PBC 2023; Herbert Lee 2023), and prescriptive guidance(Depaoli and Schoot 2017; Kruschke 2021; Gelman et al. 2020). While in theory, Bayesian modeling relies on relatively straightforward statistical principles discussed in the following section, in practice, it is often impossible to fit models analytically. Instead, fitting Bayesian models typically requires using computational simulations or approximations. While bespoke inference methods can be implemented from scratch, for practitioners, it is useful to build on computational frameworks that support model specification, work with probability distributions and samples, and implement algorithms to conduct simulation and variational based inference. Frameworks differ by the language ecosystem they build on, how tightly the components are coupled, and the maturity of the framework, including support for diverse models and analyses and practitioner support such as documentation and usage guides. Beyond the Stan framework, which is written in C++ and has command line (cmdstanr), R (stanr) and, python (PyStan) interfaces, the JAGS (Just another Gibbs Sampler) framework is also implemented in C++ and has an R front end with a domain specific language and inference engine(Plummer 2003). Historically both Stan and JAGS grew out of the BUGS (Bayesian Inference using Gibbs Sampling) that was originally developed in the late 1980s(Spiegelhalter et al. 1996). In contrast, PyMC is implemented in C++ with a Python API and a range of modules for Bayesian inference(Salvatier, Wiecki, and Fonnesbeck 2016). Pyro is built on PyTorch(Paszke et al. 2019) and JAX(James Bradbury 2018) and wraps arbitrary Python code as a probabilistic model, and has an emphasis on deep-learning based variational inference. Turing is implemented in Julia, leveraging the expressive type system to support specifying probabilistic models and performing inference(Bezanson et al. 2017).

Over the past three decades, Bayesian methods have also become increasingly common in clinical pharmacology(Ashby 2006; Grieve 2007; Campbell 2017; Yang and Novick 2019; Lakshminarayanan and Natanegara 2019; Cooner 2019; Lesaffre, Baio, and Boulanger 2020; Faya et al. 2021; Ruberg et al. 2023). For example, the first approved COVID-19 Pfizer/BioNTech vaccine used a Bayesian clinical trial design(Polack et al. 2020; Senn 2022). While there has been some application of Bayesian modeling for the analysis of high-throughput screening(Wei et al. 2013; Lock and Dunson 2015; Shterev et al. 2021; Ma and Kummer 2021; Tansey et al. 2021), and dose response modeling(Smith and Marshall 2006; Johnstone et al. 2017; Labelle, Marinier, and Lemieux 2019; Gould 2019; Arezooji 2020; Semenova et al. 2021), the models tend to be bespoke and highly sophisticated. In contrast, we aimed to lower the barrier of entry by building simple models while maintaining the flexibility needed to incrementally add complexity by building on a mature Bayesian modeling framework.

### Bayesian Modeling Workflow

Bayesian statistics is a principled strategy to fit models to data. The key idea is that before seeing the data, the researcher defines a prior distribution over possible models indexed by model parameters, then the prior is combined with the data through Bayes theorem to produce the posterior distribution over the parameters. Bayes’ theorem can be derived from basic facts about probability distributions. While many have encountered examples of mathematical probability distributions, e.g., the normal distribution over all real numbers or binomial distribution for flips of a biased coin, it is worth thinking about what a probability distribution means from a computational perspective. Roughly, they are objects that support two types of operations, (1) it is possible to draw samples from them and (2) given a region of outcome space, we can ask what is the probability of drawing a sample in that region? For a one-dimensional probability distribution, when we draw many samples, we can form a histogram, and once we have generated this, we can then questions about the distribution, such as, what fraction have values larger than a given value. For two random variables, we can think of a sample as a scatter plot over the different dimensions.

Here, we give an informal derivation of Bayes’ theorem: Consider a noisy data generation process with parameters that can generate data . Assume we have an initial guess of possible parameters represented as a prior distribution and a way to evaluate how likely a dataset is given a candidate set of parameters, which we call the likelihood conditional probability distribution . We would like to observe a data and use it to find a better estimate of the parameters, which we call the posterior conditional probability distribution, which can be represented symbolically as . Consider the joint distribution of and , . A basic fact of probability distributions is that it is possible to factorize joint distributions into conditional distributions in two different ways:

Dividing through by gives

Inspecting this equation, we see that the posterior distribution is proportional to the likelihood time of the prior. But what is the ? Since we want the posterior distribution to be a valid probability distribution that must integrate to , we can understand the , i.e. , to be the unique normalization constant needed to make the equation work. While in principle the evidence term can be computed explicitly, it is often intractable. Instead, we can borrow ideas from statistical mechanics and try to sample parameters according to the posterior distribution. If we interpret the negative log of the posterior probability as a potential energy function, we can simulate how a particle would move on the parameter space over time—like a ball rolling over a hilly landscape—and then take snapshots of the trajectory as our samples. Given enough time, if the particle can access all parts of the probability distribution, one can show that the snapshots will form an unbiased sample from the posterior distribution. The key insight that makes this strategy useful is that evaluating the local change in energy (which determines a particle’s trajectory) does not require computing the global absolute energy, and therefore sidesteps computing the intractable evidence normalization constant. Different algorithms have been implemented to do this type of sampling, including Markov chain based samplers such as Metropolis-Hastings Monte CarloHastings (1970), Gibbs Sampling(Geman and Geman 1984), and Hamiltonian Monte CarloNeal (1996), which considers the energy gradient. Stan uses a variant of Hamiltonian Monte Carlo called No-U-Turn Sampling (NUTs)(Hoffman, Gelman, and Others 2014), which aims to dampen unproductive oscillations to accelerate sampling. An alternative sampling strategy, called variational inference, aims to simplify the computation by defining a more tractable functional form for the posterior and optimizes it to estimate an evidence lower bound (ELBO)(Kucukelbir et al. 2015; Blei, Kucukelbir, and McAuliffe 2017). Depending on the complexity of the functional form, variational inference can be more effective than sampling-based strategies, though typically it is both faster and less accurate.

There are a few key takeaways from understanding the derivation of Bayes’ theorem. First, the posterior distribution–the thing we are trying to estimate in Bayesian statistics–is a distribution over the parameters, not a distribution over, e.g., sampled data. Second, for inference, we combine the prior and the likelihood in a sampling algorithm to generate samples from the posterior distribution. Third, sampling-based inference methods that are run for only a finite amount of time, such as NUTs, are biased by their the initial conditions, while variational inference based methods are biased by the mismatch between the low complexity of the variational family and the high complexity of the true posterior, therefore inference is typically not guaranteed to work, and requires checks that the modelling makes sense.

Since Bayesian modeling requires multiple steps, it can be non-trivial for beginners to get started. So, towards using Bayesian modeling effectively in practices, there have been efforts in the Bayesian modeling community to describe a principled Bayesian workflow, such as that described by (Gelman et al. 2020; Van de Schoot et al. 2020), for building robust Bayesian models and using them to draw inferences. The main steps involve:

1. Define and fit a probabilistic model, which combines a *prior* distribution over a set of parameters with the data to draw samples from the *posterior* distribution over the parameters using Hamiltonian Monte Carlo.
2. Check for sampling convergence.
3. Use prior and posterior predictive checks to evaluate the model specification and fit.
4. Use cross validation to evaluate the generalizability of the model.
5. Assess inferences that can be made from the model.

### BayesPharma package design

We designed the BayesPharma package to support modeling of pharmacology data using the principled Bayesian workflow. As described above, the workflow involves four phases: model specification, model fitting, model evaluation, and interpretation. Here we will describe the BayesPharma interface and how we recommend using it.

To provide data to the BayesPharma package, the user provides an R data.frame with columns for the response, treatments, and optionally additional covariates such as drug\_id or batch\_id. The data is passed to a model function that optionally includes arguments to customize the formula, prior, initial values, and other arguments to control the model fitting. The BayesPharma package then passes the user input to brms::brm() along with custom stan code specific for the selected model. Once the model is fit, the resulting BayesPharma::bpfit object is returned to the user and can be used for analysis.

#### Model specification

**Formula**: The goal of the formula is to describe how the data is generated conditional on the model parameters. Syntactically, BayesPharma model formulas build a brms::brmsformula, which is similar to the formula specification syntax in base R and other R regression modeling packages. A brms::brmsformula consists of an equation that declares how the response on the left side is sampled from a parameterized distribution on the right side. For the linear formulas in brms, use brms::lf(), the right side specifies mean response with a linear combination of covariates added together with implicitly defined model parameters. For example, the formula

says that response is sampled from a distribution with mean where are scalar parameters and is an indicator variable for drug . By default, the sampling distribution is a Gaussian, but other distributions can be specified from the distribution family with a link function using the family argument. For example, to model count data, which is strictly positive, set family=brms::poisson(). To model more general sampling equations, brms::formula can be specified as non-linear by setting nl=TRUE, and all model parameters must be explicitly defined. Building on this framework, the brms package supports a wide range of types of regression models including hierarchical models or random effects models. Moreover, it can support observational models that handle, for example, missing data or measurement error, which are described in detail in(Bürkner 2017). The BayesPharma package extends the brms formula syntax by defining Stan functions for foundational model types, such as the sigmoid function to model Hill-equation dose response models. For each model, functions are provided to help build the formula, for example

# This formula...  
demo\_formula <- BayesPharma::sigmoid\_agonist\_formula(  
 predictors = 1 + drug\_id)  
  
# will generate the equivalent formula as this  
demo\_formula\_alt <- brms::brmsformula(  
 response ~ sigmoid(ec50, hill, top, bottom, log\_dose),  
 nl = TRUE,  
 family = brms::student()) +  
 brms::lf(ec50 ~ 1 + drug\_id) +   
 brms::lf(hill ~ 1 + drug\_id) +  
 brms::lf(top ~ 1 + drug\_id) +  
 brms::lf(bottom ~ 1 + drug\_id)

**Data**: The data to be passed to the model should be in an R data.frame and organized into a “tidy” format. This means that each observation is in a single row and there there are columns for the response, and model specific covariates such as log\_dose for the sigmoid model, and additional experimental covariates that can be used as predictors. See table **?@tbl-model-types** for the required columns (treatments and response) for each of the implemented model types

To get data in to a tidy form, it is possible to use the range of tools from the R tidyverse libraries. For example, if data is organized in plate-layout with multiple observations per row, the tidyr::pivot\_longer function can be used to transform the shape of data.frame into one observation per row. Once the data is in a tidy format, the dplyr package can be used to manipulate the data, including using filter to subset the rows, select to subset the columns, mutate to compute new columns row-wise, group\_by with summarize and reframe to perform split-apply-combine patterns over groups of rows, or <inner, left, right, full, cross, or semi>\_join to merge using SQL-like semantics. The stringr package can manipulate string data, and for exploratory visualization, the ggplot2 package implements the grammar-of-graphics workflow for mapping data in a tidy format to aesthetic attributes of geometric objects in the plot, such as the coordinates of points or lines. For a good introduction to data manipulation using the tidyverse see the R for Data Science (2e) book and website (https://r4ds.hadley.nz/). Through each of the vignettes below, we will make use of the tidyverse data manipulation.

To illustrate, we will create demonstration data for three drugs C1, C2, C3 with different values of , , and $1\,\mbox{{\mu}M}$ measured and sigmoidal responses.

demo\_data <- tibble::tibble(  
 drug\_id = c("C1", "C2", "C3"),  
 ec50 = c(-8, -7, -6)) |>  
 dplyr::cross\_join(  
 tidyr::expand\_grid(  
 log\_dose = seq(-7, -5, length.out = 10),  
 replica = c(1,2,3))) |>  
 dplyr::mutate(  
 mean\_response = BayesPharma::sigmoid(  
 ac50 = ec50,  
 hill = 1,  
 top = 1,  
 bottom = 0,  
 log\_dose = log\_dose),  
 response = c(  
 stats::rnorm(  
 n = dplyr::n(),  
 mean = mean\_response,  
 sd = ifelse(replica == 1, .2, .8))))

Note that the tidyverse framework—and especially the dplyr package—uses what is called non-standard-evaluation, where column names (e.g., ec50 and log\_dose in the dplyr::mutate() call) should be treated as variables without putting them in quotation marks.

**Prior**: A key principle of Bayesian models is that they require specifying priors. For newcomers, understanding how these priors should be specified and justified tends to be one of the more challenging parts of the modeling process. From a practical perspective, priors can be thought of as just defining a weighted region of parameter space over which to optimize the model to best fit the data. In particular, the more compact and closely aligned the priors are with the data, the easier it is for the model to fit the data. Therefore, for setting up and getting started with a new model fit, the stronger (more constrained) the priors are, and the closer they align with the data, the easier the model is to fit. From a scientific perspective, since priors and posterior distributions can be interpreted as capturing the uncertainty in parameters before and after observing the data, the weaker (less constrained) priors, the less biased the inference, thus “letting the data speak for itself”. Ultimately, however, in Bayesian modeling, it is not possible to completely remove the bias due to the prior. This means that in a complete Bayesian analysis, some substantive argument should be made to justify that inferences from the model are not sensitive to reasonable choices of the prior. In a way, this actually makes choosing the prior less stressful as there is no singular best prior choice, and instead it reflects the scientific questions of the modeling process. For a deeper discussion and practical advice, see: the [prior choice recommendations](https://github.com/stan-dev/stan/wiki/Prior-Choice-Recommendations) page on the Stan wiki. To facilitate specifying priors for each of the BayesPharma models, the BayesPharma package implements helper functions, for example

# This formula  
demo\_prior <- BayesPharma::sigmoid\_agonist\_prior()  
  
# will generate the equivalent formula as this  
demo\_prior\_alt <- c(  
 brms::prior(prior = normal(-6, 2.5), nlpar = "ec50"),  
 brms::prior(prior = normal(1, 1), nlpar = "hill", lb = 0.01),  
 brms::prior(prior = normal(1, 0.5), nlpar = "top"),  
 brms::prior(prior = normal(0, 0.5), nlpar = "bottom"))

The brms::prior function takes in Stan code (in this case the [Normal distribution](https://mc-stan.org/docs/2_20/functions-reference/normal-distribution.html), which is defined in the Stan documentation), the nlpar defines the non-linear parameter, and optional arguments ub and lb give upper or lower bounds. For each BayesPharma prior helper function, individual priors can either be explicitly given to override the defaults, or specified as numeric constants to fix them to a particular value. For example,

# This formula overrides some of the default parameter priors  
demo\_custom\_prior <- BayesPharma::sigmoid\_agonist\_prior(  
 ec50 = brms::prior(prior = normal(-8, 2.5), nlpar = "ec50"),  
 hill = 1,  
 bottom = 0)  
  
# will generate the equivalent formula as this  
demo\_custom\_prior\_alt <- c(  
 brms::prior(prior = normal(-8, 2.5), nlpar = "ec50"),  
 brms::prior(prior = constant(1), nlpar = "hill"),  
 brms::prior(prior = normal(1, 0.5), nlpar = "top"),  
 brms::prior(prior = constant(0), nlpar = "bottom"))

**Init**: To estimate the posterior distribution using sampling-based inference, the initial parameters values must be given. Then during inference, the parameters are simulated so that they hopefully converge to a sample from the posterior distribution. Therefore, the initial parameter values should at least be in a feasible region of parameter space to get the simulation to rapidly mix. For each model, the BayesPharma package provides default initialization values. Typically, if a different prior is given, the initial values need to be adjusted along with the updated prior.

# This formula  
demo\_init <- BayesPharma::sigmoid\_agonist\_init()  
  
# will generate the equivalent initial values as this  
demo\_init\_alt <- function() {  
 list(  
 b\_ec50 = function(){as.array(-6)},  
 b\_hill = function(){as.array(1)},  
 b\_top = function(){as.array(1)},  
 b\_bottom = function(){as.array(0)})  
}

Note that the b\_ in the function list corresponds to how the parameter names are translated by brms from the input formula to the Stan modeling language.

| Name | Treatments | Parameters | Response |
| --- | --- | --- | --- |
| sigmoid\_agonist | log\_dose | top, bottom, ec50, hill | response |
| sigmoid\_antagonist | log\_dose | top, bottom, ic50, hill | response |
| growth\_sigmoid | time | K, K0, rate, lambda | response |
| growth\_richards | time | K, K0, rate, lambda, nu | response |
| MuSyC | logd1, logd2 | logE[0-3], logC[1,2], h[1,2], logalpha | response |
| tQ | series\_index, time, ET, ST | Kcat, kM | P |

:Model types implemented in BayesPharma {#tbl-model-types}

#### Model fitting

Once the components of the model have been specified, each model type provides a function to integrate the formula, data, prior, init, and additional arguments to build and fit the model.

# This model ...  
demo\_model <- BayesPharma::sigmoid\_model(  
 formula = demo\_formula,  
 data = demo\_data,  
 prior = demo\_prior,  
 init = demo\_init)  
  
# is equivalent to:  
demo\_model\_alt <- brms::brm(  
 formula = demo\_formula,  
 data = demo\_data,  
 prior = demo\_prior,  
 init = demo\_init,  
 control = list(adapt\_delta = 0.99),  
 iter = 8000,  
 stanvars = BayesPharma::sigmoid\_stanvar())  
  
demo\_model\_alt$bayes\_pharma\_info <- list(  
 model\_type = "sigmoid\_agonist",  
 treatment\_variable = "log\_dose",  
 treatment\_units = "Log[Molar]",  
 response\_variable = "response",  
 response\_units = NULL)  
  
brms::expose\_functions(demo\_model\_alt, vectorize = TRUE)

The call to brms::expose\_functions() allows calling the compiled model functions in R, which is needed for the downstream analysis described in the next section.

#### Model Evaluation

Following the principled Bayesian workflow, evaluation of the model fit involves evaluating not only the parameter estimates, but also confirming inference convergence, checking how sensible the prior and posterior distributions are with the scientific understanding of the problem, selecting among alternative models, and testing hypotheses. We briefly review these points here:

**Convergence**: For sampling-based inference, while simulations are guaranteed to converge to the posterior eventually, for any finite sample, there is a risk that the samples may be biased by the initial values. To assess convergence, a key strategy is to run multiple chains and compare the within chain correlation against the between chain covariation in parameter estimates. If the chains have not converged, then the between chain covariation will be high. To quantify(Gelman and Rubin 1992), define , where values greater than 1 (e.g., 1.1 and larger) indicate lack of convergence.

To mitigate this bias, there are two key strategies. First, sample the model for longer, or second, reparametrize the model to make it easier to sample. A benefit of the NUTs sampling algorithm used by Stan is that it will report when the sampling has gotten stuck and give warnings. Michael Betancourt has a detailed discussion about these warnings, what they mean, and how to resolve them on his [website](https://betanalpha.github.io/writing/), his chapter on [Identifiability and Divergences](https://betanalpha.github.io/assets/case_studies/identifiability.html) is particularly helpful.

**Prior and Posterior Predictive Checks**: A key strategy in developing and checking the quality of a model is to sample from the prior and posterior distributions and visualize the resulting outcomes. In sampling from the prior, the goal is to make sure that the resulting distributions are consistent with the scientific understanding of the model. For example, if the collected data are counts, but the default Normal distribution family is used, then since the Normal distribution has infinite support, the prior model may generate samples with negative counts, which does not make sense. Seeing these negative counts would motivate using either a family bounded below by zero, or setting a lower bound of zero by setting lb = 0 in the prior specification.

For posterior predictive distributions, the goal is to generate samples from the posterior and qualitatively evaluate if it is consistent with the observed data. For example, if the observed data are multi-modal, but the posterior samples are unimodal, this may suggest that a different functional form is needed. Using plots and visualizations can reveal these and other issues with the model that may need to be explicitly handled (See Model Selection below). Note that in contrast with typical frequentist modelling, the goal is not to construct the simplest meaningful model in order to have the most power to reject it, the goal is to construct a model that best fits the data in order to give the best interpretation of the observations.

**Model Selection**: A general modeling strategy is to begin with simpler models and incrementally increase model complexity to handle nuance as needed. Simpler models are not only easier to interpret, but given a simple model and complex model that explain the data equally, Occam’s razor says that the more simple model should be preferred, in part because it is more likely to generalize to unseen contexts. Evaluating model fit while considering model complexity is non-trivial for non-Bayesian models and can be done in ad hoc ways through the number of parameters, deviance scores, etc. However, for Bayesian models, explicitly modeling the model uncertainty gives a principled approach by measuring the marginal likelihood of the data given the model, quantified by expected log-posterior density (ELPD) of held out data. Instead of splitting the data into train/test split, k-fold or leave-one-out (LOO) cross-validation can be used, where the model is fit to a portion of the data and the ELPD is measured on the rest, across multiple partitions. Re-fitting the model for each data-point is computationally intensive, but it can be approximated through Pareto Smoothed Importance Sampling (PSIS) and is implemented in the loo package(Vehtari, Gelman, and Gabry 2017b). By default, the BayesPharma package models compute the loo criterion. Checking the model$criteria$loo will summarize the outlier data points (Pareto k-statistics > 0.7). See [Loo interpretation](https://mc-stan.org/loo/articles/online-only/faq.html#elpd_interpretation) for more guidance in how to interpret the summary statistics. Then, given two models for the same data, calling brms::loo\_compare(model1, model2, ...), will rank the models based on their ELPD. See the case-studies for concrete examples. An alternative strategy to compare models is to use Bayesian Model Averaging (BMA), where multiple models are fit separately and then averaged depending on their ability to explain the data. The models that receive weights greater than zero contribute to explaining the overall data distribution.

**Hypothesis Testing**: After models have been fit and selected, the posterior distribution can be used to test hypotheses. For example, to test if one parameter is greater than another, the fraction of samples from the posterior in which the condition is true can be interpreted as a p-value.

Together, these stages of model fitting and evaluation are the foundation for a principled work, that is outlined in more detail in (Gelman et al. 2020). Understanding what steps need to be communicated depends on the the context, the case-studies in the following sections illustrate templates that can be for typical Pharmacology analyses.

## Case Studies

In this section, we will consider several models as case studies: the sigmoidal hill model in [Section 2.1](#sec-hill), the MuSyC synergy model in [Section 2.2](#sec-MuSyC)}, and the Michaelis-Menten enzyme progress curve in [Section 2.3](#sec-michaelis_menten)}. For each, we will implement it, apply it to example data by fitting different models, and compare the models based on their fit of the data and inferences that can be made.

### Hill Equation

In this case study, we are going to reanalyze the dose responses of 4 Kappa Opioid receptor (KOR) antagonists from a study performed by Margolis et al. (-Margolis et al. (2020)) using the BayesPharma package. Whole cell electrophysiology in acute rat midbrain slices was used to evaluate the pharmacological properties of four novel KOR antagonists: BTRX-335140, BTRX-395750, PF-04455242, and JNJ-67953964.

Originally, the dose-response analysis was performed using the drc package in R, which implements the minimization of negative log likelihood function and reduces to least square estimation for a continuous response. The data were normalized to % baseline agonist response, measuring the extent of blockade of agonist response in the presence of different concentrations of antagonist. These data were then fit to a 4-parameter log-logistic dose response model, setting the top (max agonist response) to 100% and estimating the IC50, its variance, and the bottom (min agonist response).

#### Fitting the sigmoid model

Using the BayesPharma package, we can re-fit the sigmoid model with a negative slope, and fix the top parameter to 100 as the response is normalized to a no-antagonist baseline.

For the prior, we are going to use a normal distribution because the response values are continuous. First, we will run the analysis with the top (max response) parameter prior set to a constant value of 100 because top is normalized to 100 and is the default broad prior for the ic50, hill, and bottom parameters. Broad priors represent unbiased uncertainty and provide an opportunity for extreme responses.

The level of informativeness of the prior will affect how much influence the prior has on the model. Here is more [information on prior choice recommendations.](https://github.com/stan-dev/stan/wiki/Prior-Choice-Recommendations)

kor\_prior <- BayesPharma::sigmoid\_antagonist\_prior(top = 100)  
kor\_prior

prior class coef group resp dpar nlpar lb ub source  
 normal(-6, 2.5) b ic50 <NA> <NA> user  
 normal(-1, 1) b hill <NA> 0.01 user  
 constant(100) b top <NA> <NA> user  
 normal(0, 0.5) b bottom <NA> <NA> user

#### Prior predictive checks

Following the Bayesian workflow, before fitting the model, it is good to check the prior predictive distributions to see if they are compatible with the domain expertise. So, before running the model, we will verify that the prior distributions cover a plausible range of values for each parameter. To do this, we want to sample only from the prior distributions by adding sample\_prior = "only" as an argument to the sigmoid\_model function. We will use the default response distribution of the model (family = gaussian()).

kor\_sample\_prior <- BayesPharma::sigmoid\_model(  
 data = kor\_antag |> dplyr::select(substance\_id, log\_dose, response),  
 formula = BayesPharma::sigmoid\_antagonist\_formula(),  
 prior = kor\_prior,  
 init = BayesPharma::sigmoid\_antagonist\_init(),  
 sample\_prior = "only")

And then plot of the prior predictive distributions:

kor\_sample\_prior |>  
 BayesPharma::plot\_density\_distribution()

|  |
| --- |
| KOR antagonists prior distribution |

To sample from the model, we will use the Stan NUTs Hamiltonian Monte Carlo, and initialize the parameters to the prior means to help with model convergence, using the default values of ec50 = -9, hill = -1, top = 100, bottom = 0.

kor\_model <- BayesPharma::sigmoid\_model(  
 data = kor\_antag |> dplyr::select(substance\_id, log\_dose, response),  
 formula = BayesPharma::sigmoid\_antagonist\_formula(  
 predictors = 0 + substance\_id),   
 prior = kor\_prior,  
 init = BayesPharma::sigmoid\_antagonist\_init())

#### Analyzing model fit

The brms generated model summary shows the formula that the expected response a is sigmoid function of the log\_dose with four parameters, and a shared Gaussian distribution. Each parameter is dependent on the substance\_id. Since we want to fit a separate model for each substance, we include a 0 + to indicate that there is no common intercept. The data consists of 73 data points and the posterior sampling was done in 4 chains each with 8000 steps with 4000 steps of warm-up. The population effects for each parameter summarize the marginal posterior distributions, as well as the effective sample size in the bulk and tail. This gives an indication of the sampling quality, with an ESS of > 500 samples being good for this type of model.

Family: gaussian   
 Links: mu = identity; sigma = identity   
Formula: response ~ sigmoid(ic50, hill, top, bottom, log\_dose)   
 ic50 ~ 0 + substance\_id  
 hill ~ 0 + substance\_id  
 top ~ 0 + substance\_id  
 bottom ~ 0 + substance\_id  
 Data: data (Number of observations: 73)   
 Draws: 4 chains, each with iter = 8000; warmup = 4000; thin = 1;  
 total post-warmup draws = 16000  
  
Regression Coefficients:  
 Estimate Est.Error l-95% CI u-95% CI Rhat Bulk\_ESS Tail\_ESS  
ic50\_substance\_idBTRX\_335140 -8.84 0.21 -9.20 -8.39 1.00 13221 8520  
ic50\_substance\_idBTRX\_395750 -8.23 0.48 -8.92 -7.40 1.00 9310 4125  
ic50\_substance\_idJNJ -9.15 0.32 -9.76 -8.49 1.00 14009 10195  
ic50\_substance\_idPF -6.14 1.08 -7.65 -3.32 1.00 8772 4951  
hill\_substance\_idBTRX\_335140 -1.47 0.60 -2.87 -0.58 1.00 13725 9962  
hill\_substance\_idBTRX\_395750 -0.89 0.50 -2.20 -0.25 1.00 9046 4910  
hill\_substance\_idJNJ -1.01 0.51 -2.41 -0.40 1.00 15154 11739  
hill\_substance\_idPF -0.31 0.24 -0.92 -0.03 1.00 7704 5291  
bottom\_substance\_idBTRX\_335140 -0.01 0.50 -0.99 0.99 1.00 17386 11381  
bottom\_substance\_idBTRX\_395750 0.01 0.51 -0.97 0.98 1.00 17310 11802  
bottom\_substance\_idJNJ -0.01 0.50 -0.99 0.98 1.00 18067 11820  
bottom\_substance\_idPF 0.01 0.50 -0.96 0.99 1.00 17720 12063  
top\_substance\_idBTRX\_335140 100.00 0.00 100.00 100.00 NA NA NA  
top\_substance\_idBTRX\_395750 100.00 0.00 100.00 100.00 NA NA NA  
top\_substance\_idJNJ 100.00 0.00 100.00 100.00 NA NA NA  
top\_substance\_idPF 100.00 0.00 100.00 100.00 NA NA NA  
  
Further Distributional Parameters:  
 Estimate Est.Error l-95% CI u-95% CI Rhat Bulk\_ESS Tail\_ESS  
sigma 32.18 2.87 27.19 38.44 1.00 13486 10636  
  
Draws were sampled using sampling(NUTS). For each parameter, Bulk\_ESS  
and Tail\_ESS are effective sample size measures, and Rhat is the potential  
scale reduction factor on split chains (at convergence, Rhat = 1).

**Traceplot**: The model ran without warning messages, meaning there were no parameter value problems or MCMC conflicts. The bulk and tail ESS indicate high resolution and stability. The R-hat for each parameter equals 1.00 and the traceplot shows the chains mixed well, indicating the chains converged.

kor\_model |>  
 bayesplot::mcmc\_trace()

|  |
| --- |
|  |

**Compare prior and posterior marginal distributions**: Displayed below is a plot for the prior and posterior distributions of the parameters (prior is pink and posterior is teal). This can be useful for comparing the density distribution of the prior and posterior produced by the model:

BayesPharma::plot\_prior\_posterior\_densities(  
 model = kor\_model,  
 predictors\_col\_name = "substance\_id",  
 half\_max\_label = "ic50",  
 title\_label="")

|  |
| --- |
| KOR antagonists model, compare prior and posterior distributions for each substance |

Displayed below is a plot of the posterior distributions for each parameter with the confidence intervals and mean. This is a useful visual of the model results and can highlight the mode and high-density intervals:

BayesPharma::plot\_posterior\_density(  
 kor\_model,   
 predictors\_col\_name = "substance\_id",   
 half\_max\_label = "ic50",  
 title\_label = "")

|  |
| --- |
| KOR Antagonists, posterior distribution for each substance |

Displayed below is a plot of a sample of 100 sigmoid dose-response curves from the posterior distribution (purple) and the median quantile intervals:

BayesPharma::plot\_posterior\_draws(  
 model = kor\_model,  
 title = "")

|  |
| --- |
| KOR antagonists, posterior draws |

#### Comparing alternative models

To test the sensitivity of the analysis to the prior, we can re-fit the model with a more informative prior:

prior class coef group resp dpar nlpar lb ub source  
 normal(-8.5, 0.5) b ic50 <NA> <NA> user  
 normal(-1, 0.5) b hill <NA> 0.01 user  
 constant(100) b top <NA> <NA> user  
 normal(10, 15) b bottom <NA> <NA> user

Re-fitting the model with the kor\_prior2 gives:

Family: gaussian   
 Links: mu = identity; sigma = identity   
Formula: response ~ sigmoid(ic50, hill, top, bottom, log\_dose)   
 ic50 ~ 0 + substance\_id  
 hill ~ 0 + substance\_id  
 top ~ 0 + substance\_id  
 bottom ~ 0 + substance\_id  
 Data: data (Number of observations: 73)   
 Draws: 4 chains, each with iter = 8000; warmup = 4000; thin = 1;  
 total post-warmup draws = 16000  
  
Regression Coefficients:  
 Estimate Est.Error l-95% CI u-95% CI Rhat  
ic50\_substance\_idBTRX\_335140 -8.82 0.22 -9.23 -8.36 1.00  
ic50\_substance\_idBTRX\_395750 -8.58 0.31 -9.16 -7.96 1.00  
ic50\_substance\_idJNJ -8.95 0.31 -9.52 -8.33 1.00  
ic50\_substance\_idPF -8.18 0.44 -9.06 -7.32 1.00  
hill\_substance\_idBTRX\_335140 -1.19 0.37 -1.99 -0.57 1.00  
hill\_substance\_idBTRX\_395750 -1.05 0.39 -1.90 -0.42 1.00  
hill\_substance\_idJNJ -0.86 0.33 -1.67 -0.39 1.00  
hill\_substance\_idPF -0.73 0.45 -1.75 -0.07 1.00  
bottom\_substance\_idBTRX\_335140 1.79 10.41 -18.98 21.69 1.00  
bottom\_substance\_idBTRX\_395750 15.29 11.15 -7.93 35.93 1.00  
bottom\_substance\_idJNJ -2.68 9.82 -22.88 15.76 1.00  
bottom\_substance\_idPF 31.18 11.54 6.47 51.76 1.00  
top\_substance\_idBTRX\_335140 100.00 0.00 100.00 100.00 NA  
top\_substance\_idBTRX\_395750 100.00 0.00 100.00 100.00 NA  
top\_substance\_idJNJ 100.00 0.00 100.00 100.00 NA  
top\_substance\_idPF 100.00 0.00 100.00 100.00 NA  
 Bulk\_ESS Tail\_ESS  
ic50\_substance\_idBTRX\_335140 12849 10671  
ic50\_substance\_idBTRX\_395750 12481 11281  
ic50\_substance\_idJNJ 11799 11261  
ic50\_substance\_idPF 12643 10249  
hill\_substance\_idBTRX\_335140 13713 10280  
hill\_substance\_idBTRX\_395750 12136 8980  
hill\_substance\_idJNJ 11734 12445  
hill\_substance\_idPF 8345 5967  
bottom\_substance\_idBTRX\_335140 12530 11326  
bottom\_substance\_idBTRX\_395750 12150 10047  
bottom\_substance\_idJNJ 11149 10674  
bottom\_substance\_idPF 9644 9889  
top\_substance\_idBTRX\_335140 NA NA  
top\_substance\_idBTRX\_395750 NA NA  
top\_substance\_idJNJ NA NA  
top\_substance\_idPF NA NA  
  
Further Distributional Parameters:  
 Estimate Est.Error l-95% CI u-95% CI Rhat Bulk\_ESS Tail\_ESS  
sigma 31.91 2.83 26.85 38.02 1.00 14835 10595  
  
Draws were sampled using sampling(NUTS). For each parameter, Bulk\_ESS  
and Tail\_ESS are effective sample size measures, and Rhat is the potential  
scale reduction factor on split chains (at convergence, Rhat = 1).

**Comparing the Two Models Using LOO-Comparison**: One way to evaluate the quality of a model is for each data-point, re-fit the model with remaining points and evaluate the log probability of the point in the posterior distribution. Taking the expectation across all points gives the Expected Log Pointwise predictive Density (ELPD). Since this is computationally challenging to re-fit the model for each point, if the model fits the data reasonably well, then the ELPD can be approximated using the Pareto smoothed importance sampling (PSIS). Using the LOO package, Pareto k value for each data point is computed that indicate how well that data point is fit by the model, where k < 0.5 is good, 0.5 <= k < 0.7 is OK, and 0.7 <= k is bad. Evaluating the model for the KOR antagonists shows that the model fits the data well.

Trying to compile a simple C file

Running /Library/Frameworks/R.framework/Resources/bin/R CMD SHLIB foo.c  
using C compiler: ‘Apple clang version 15.0.0 (clang-1500.3.9.4)’  
using SDK: ‘’  
clang -arch x86\_64 -I"/Library/Frameworks/R.framework/Resources/include" -DNDEBUG -I"/Library/Frameworks/R.framework/Versions/4.3-x86\_64/Resources/library/Rcpp/include/" -I"/Library/Frameworks/R.framework/Versions/4.3-x86\_64/Resources/library/RcppEigen/include/" -I"/Library/Frameworks/R.framework/Versions/4.3-x86\_64/Resources/library/RcppEigen/include/unsupported" -I"/Library/Frameworks/R.framework/Versions/4.3-x86\_64/Resources/library/BH/include" -I"/Library/Frameworks/R.framework/Versions/4.3-x86\_64/Resources/library/StanHeaders/include/src/" -I"/Library/Frameworks/R.framework/Versions/4.3-x86\_64/Resources/library/StanHeaders/include/" -I"/Library/Frameworks/R.framework/Versions/4.3-x86\_64/Resources/library/RcppParallel/include/" -I"/Library/Frameworks/R.framework/Versions/4.3-x86\_64/Resources/library/rstan/include" -DEIGEN\_NO\_DEBUG -DBOOST\_DISABLE\_ASSERTS -DBOOST\_PENDING\_INTEGER\_LOG2\_HPP -DSTAN\_THREADS -DUSE\_STANC3 -DSTRICT\_R\_HEADERS -DBOOST\_PHOENIX\_NO\_VARIADIC\_EXPRESSION -D\_HAS\_AUTO\_PTR\_ETC=0 -include '/Library/Frameworks/R.framework/Versions/4.3-x86\_64/Resources/library/StanHeaders/include/stan/math/prim/fun/Eigen.hpp' -D\_REENTRANT -DRCPP\_PARALLEL\_USE\_TBB=1 -I/opt/R/x86\_64/include -fPIC -falign-functions=64 -Wall -g -O2 -c foo.c -o foo.o  
In file included from <built-in>:1:  
In file included from /Library/Frameworks/R.framework/Versions/4.3-x86\_64/Resources/library/StanHeaders/include/stan/math/prim/fun/Eigen.hpp:22:  
In file included from /Library/Frameworks/R.framework/Versions/4.3-x86\_64/Resources/library/RcppEigen/include/Eigen/Dense:1:  
In file included from /Library/Frameworks/R.framework/Versions/4.3-x86\_64/Resources/library/RcppEigen/include/Eigen/Core:19:  
/Library/Frameworks/R.framework/Versions/4.3-x86\_64/Resources/library/RcppEigen/include/Eigen/src/Core/util/Macros.h:679:10: fatal error: 'cmath' file not found  
#include <cmath>  
 ^~~~~~~  
1 error generated.  
make[1]: \*\*\* [foo.o] Error 1

Warning: Found 1 observations with a pareto\_k > 0.7 in model 'kor\_model'. We  
recommend to set 'moment\_match = TRUE' in order to perform moment matching for  
problematic observations.

Computed from 16000 by 73 log-likelihood matrix.  
  
 Estimate SE  
elpd\_loo -360.5 6.5  
p\_loo 6.9 1.1  
looic 721.0 13.0  
------  
MCSE of elpd\_loo is NA.  
MCSE and ESS estimates assume MCMC draws (r\_eff in [0.3, 1.3]).  
  
Pareto k diagnostic values:  
 Count Pct. Min. ESS  
(-Inf, 0.7] (good) 72 98.6% 3194   
 (0.7, 1] (bad) 1 1.4% <NA>   
 (1, Inf) (very bad) 0 0.0% <NA>   
See help('pareto-k-diagnostic') for details.

Since ELPD is a global measure of model fit, it can be used to compare models. Using loo\_compare from the LOO package returns the elpd\_diff and se\_diff for each model relative the model with the lowest ELPD. The kor\_model2, the model with more informative prior, is the preferred model, but not significantly.

Trying to compile a simple C file

Running /Library/Frameworks/R.framework/Resources/bin/R CMD SHLIB foo.c  
using C compiler: ‘Apple clang version 15.0.0 (clang-1500.3.9.4)’  
using SDK: ‘’  
clang -arch x86\_64 -I"/Library/Frameworks/R.framework/Resources/include" -DNDEBUG -I"/Library/Frameworks/R.framework/Versions/4.3-x86\_64/Resources/library/Rcpp/include/" -I"/Library/Frameworks/R.framework/Versions/4.3-x86\_64/Resources/library/RcppEigen/include/" -I"/Library/Frameworks/R.framework/Versions/4.3-x86\_64/Resources/library/RcppEigen/include/unsupported" -I"/Library/Frameworks/R.framework/Versions/4.3-x86\_64/Resources/library/BH/include" -I"/Library/Frameworks/R.framework/Versions/4.3-x86\_64/Resources/library/StanHeaders/include/src/" -I"/Library/Frameworks/R.framework/Versions/4.3-x86\_64/Resources/library/StanHeaders/include/" -I"/Library/Frameworks/R.framework/Versions/4.3-x86\_64/Resources/library/RcppParallel/include/" -I"/Library/Frameworks/R.framework/Versions/4.3-x86\_64/Resources/library/rstan/include" -DEIGEN\_NO\_DEBUG -DBOOST\_DISABLE\_ASSERTS -DBOOST\_PENDING\_INTEGER\_LOG2\_HPP -DSTAN\_THREADS -DUSE\_STANC3 -DSTRICT\_R\_HEADERS -DBOOST\_PHOENIX\_NO\_VARIADIC\_EXPRESSION -D\_HAS\_AUTO\_PTR\_ETC=0 -include '/Library/Frameworks/R.framework/Versions/4.3-x86\_64/Resources/library/StanHeaders/include/stan/math/prim/fun/Eigen.hpp' -D\_REENTRANT -DRCPP\_PARALLEL\_USE\_TBB=1 -I/opt/R/x86\_64/include -fPIC -falign-functions=64 -Wall -g -O2 -c foo.c -o foo.o  
In file included from <built-in>:1:  
In file included from /Library/Frameworks/R.framework/Versions/4.3-x86\_64/Resources/library/StanHeaders/include/stan/math/prim/fun/Eigen.hpp:22:  
In file included from /Library/Frameworks/R.framework/Versions/4.3-x86\_64/Resources/library/RcppEigen/include/Eigen/Dense:1:  
In file included from /Library/Frameworks/R.framework/Versions/4.3-x86\_64/Resources/library/RcppEigen/include/Eigen/Core:19:  
/Library/Frameworks/R.framework/Versions/4.3-x86\_64/Resources/library/RcppEigen/include/Eigen/src/Core/util/Macros.h:679:10: fatal error: 'cmath' file not found  
#include <cmath>  
 ^~~~~~~  
1 error generated.  
make[1]: \*\*\* [foo.o] Error 1

No problematic observations found. Returning the original 'loo' object.

elpd\_diff se\_diff  
kor\_model2 0.0 0.0   
kor\_model -1.0 1.2

**Comparing against models fit with the drc Package**: Here we will analyze the KOR antagonist data using the drc package and compare it to the results from the BayesPharma analysis.

We will fix the top to 100 and fit the ic50, hill, and bottom.

drc\_models <- kor\_antag |>  
 dplyr::group\_by(substance\_id) |>  
 dplyr::group\_nest() |>  
 dplyr::mutate(  
 model = data |>   
 purrr::map(~drc::drm(  
 response ~ log\_dose,  
 data = .x,  
 fct = drc::L.4(fixed = c(NA, NA, 100, NA),  
 names = c("hill", "bottom", "top", "ic50")))))  
  
drc\_models |>  
 dplyr::mutate(summary = purrr::map(model, broom::tidy, conf.int = TRUE)) |>  
 tidyr::unnest(summary) |>  
 dplyr::arrange(term, substance\_id) |>  
 dplyr::select(-data, -model, -curve)

# A tibble: 12 × 8  
 substance\_id term estimate std.error statistic p.value conf.low conf.high  
 <chr> <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
 1 BTRX\_335140 bottom 1.31 19.4 0.0675 9.47e- 1 -40.0 42.6   
 2 BTRX\_395750 bottom 29.5 9.40 3.14 7.85e- 3 9.20 49.8   
 3 JNJ bottom -18.1 26.7 -0.681 5.04e- 1 -73.7 37.4   
 4 PF bottom 39.4 30.8 1.28 2.22e- 1 -27.0 106.   
 5 BTRX\_335140 hill 4.06 9.20 0.441 6.65e- 1 -15.5 23.7   
 6 BTRX\_395750 hill 9.82 164. 0.0600 9.53e- 1 -344. 364.   
 7 JNJ hill 1.17 0.580 2.02 5.69e- 2 -0.0378 2.38  
 8 PF hill 1.13 1.33 0.855 4.08e- 1 -1.73 4.00  
 9 BTRX\_335140 ic50 -8.91 0.308 -28.9 1.42e-14 -9.57 -8.26  
10 BTRX\_395750 ic50 -8.97 0.505 -17.8 1.70e-10 -10.1 -7.88  
11 JNJ ic50 -8.77 0.670 -13.1 2.89e-11 -10.2 -7.37  
12 PF ic50 -7.96 1.27 -6.29 2.78e- 5 -10.7 -5.23

Displayed below is the comparison of results from drc and BayesPharma for each parameter of the dose-response curve. Here we see that the Bayesian method provides a distribution curve as evidence and has smaller confidence intervals than most of the standard errors provided by the drc method.

Trying to compile a simple C file

Running /Library/Frameworks/R.framework/Resources/bin/R CMD SHLIB foo.c  
using C compiler: ‘Apple clang version 15.0.0 (clang-1500.3.9.4)’  
using SDK: ‘’  
clang -arch x86\_64 -I"/Library/Frameworks/R.framework/Resources/include" -DNDEBUG -I"/Library/Frameworks/R.framework/Versions/4.3-x86\_64/Resources/library/Rcpp/include/" -I"/Library/Frameworks/R.framework/Versions/4.3-x86\_64/Resources/library/RcppEigen/include/" -I"/Library/Frameworks/R.framework/Versions/4.3-x86\_64/Resources/library/RcppEigen/include/unsupported" -I"/Library/Frameworks/R.framework/Versions/4.3-x86\_64/Resources/library/BH/include" -I"/Library/Frameworks/R.framework/Versions/4.3-x86\_64/Resources/library/StanHeaders/include/src/" -I"/Library/Frameworks/R.framework/Versions/4.3-x86\_64/Resources/library/StanHeaders/include/" -I"/Library/Frameworks/R.framework/Versions/4.3-x86\_64/Resources/library/RcppParallel/include/" -I"/Library/Frameworks/R.framework/Versions/4.3-x86\_64/Resources/library/rstan/include" -DEIGEN\_NO\_DEBUG -DBOOST\_DISABLE\_ASSERTS -DBOOST\_PENDING\_INTEGER\_LOG2\_HPP -DSTAN\_THREADS -DUSE\_STANC3 -DSTRICT\_R\_HEADERS -DBOOST\_PHOENIX\_NO\_VARIADIC\_EXPRESSION -D\_HAS\_AUTO\_PTR\_ETC=0 -include '/Library/Frameworks/R.framework/Versions/4.3-x86\_64/Resources/library/StanHeaders/include/stan/math/prim/fun/Eigen.hpp' -D\_REENTRANT -DRCPP\_PARALLEL\_USE\_TBB=1 -I/opt/R/x86\_64/include -fPIC -falign-functions=64 -Wall -g -O2 -c foo.c -o foo.o  
In file included from <built-in>:1:  
In file included from /Library/Frameworks/R.framework/Versions/4.3-x86\_64/Resources/library/StanHeaders/include/stan/math/prim/fun/Eigen.hpp:22:  
In file included from /Library/Frameworks/R.framework/Versions/4.3-x86\_64/Resources/library/RcppEigen/include/Eigen/Dense:1:  
In file included from /Library/Frameworks/R.framework/Versions/4.3-x86\_64/Resources/library/RcppEigen/include/Eigen/Core:19:  
/Library/Frameworks/R.framework/Versions/4.3-x86\_64/Resources/library/RcppEigen/include/Eigen/src/Core/util/Macros.h:679:10: fatal error: 'cmath' file not found  
#include <cmath>  
 ^~~~~~~  
1 error generated.  
make[1]: \*\*\* [foo.o] Error 1

|  |
| --- |
| KOR antagonists conditional effects. The blue lines are samples from the BayesPharma kor\_model posterior distribution, the orange line is the conditional mean, and the purple line is the conditional mean for the drc model fit. |

### MuSyC Epistasis Model

When two different treatments are made in an assay, their combined effect may be stronger or weaker than what would be expected with an additive model, the treatments are said to be epistatic For sigmoidal dose-response models, one treatment may effect of the other in two different ways; by either shifting the maximal response (efficacy) or by shifting the dose needed to cause the response (potency). A range of statistical models have been proposed that capture different aspects of synergy, notably Bliss independence(Bliss 1956) and Loewe additivity(Loewe 1926) models can be used to test for significant efficacy or potency interactions, respectively. The SynergyFinder R package(Ianevski, Giri, and Aittokallio 2022) and the synergy python package(Wooten and Albert 2021) can be used to visualize treatment interactions, compute a range of synergy scores, and test if the interactions are significant.

Recently Meyer et al.Wooten et al. (2021) derived an integrated functional synergistic sigmoidal dose-response called the Multi-dimensional Synergy of Combinations (MuSyC) method, which has the Loewe and Bliss models as special cases. They implemented a Bayesian model-fitting strategy in Matlab, and a maximum likelihood model-fitting into the synergy python package. To make the model more accessible to the pharmacology community, in this section, we briefly review the MuSyC functional form, describe a Bayesian implementation in Stan/BRMS, and illustrate using the model to re-analyze how drugs and voltage may interact to modulate the current through a potassium channel.

**MuSyC Functional Form**: The functional form for the MuSyC model gives an equation for the response $\color{brown}{E\_d}$ at doses of $\color{teal}{d\_1}$ and $\color{teal}{d\_2}$ of the two treatments and has free parameters $\theta = \left({\color{purple}{C\_1}}, {\color{purple}{C\_2}}, {\color{brown}{E\_0}}, {\color{brown}{E\_1}}, {\color{brown}{E\_2}}, {\color{brown}{E\_3}}, {\color{purple}{h\_1}}, {\color{purple}{h\_2}}, {\color{purple}{\alpha}}\right)$

The parameters $\color{brown}{E\_0}$, $\color{brown}{E\_3}$, give response values for the extreme values for the doses: $\mbox{MuSyC}({\color{teal}{0}}, {\color{teal}{0}}; \theta) = \color{brown}{E\_0}$ and $\mbox{MuSyC}({\color{teal}{\infty}}, {\color{teal}{\infty}}; \theta) = \color{brown}{E\_3}$. Setting one of the doses to zero e.g., ${\color{teal}{d\_2}} = 0$, the MuSyC functional form reduces to a sigmoid function of the other, where $\mbox{MuSyC}({\color{teal}{d\_1}}, {\color{teal}{0}}; \theta) = Sigmoid({\color{teal}{d\_1}}; \phi)$, where the half maximal activity is $\mbox{AC}\_{50} = \color{purple}{C\_1}$ and the slope at the half maximal activity is ${\color{purple}{\mbox{hill}}} = {\color{purple}{h\_1}}$ and if , then ${\color{brown}{\mbox{top}}} = {\color{brown}{E\_1}}$ and ${\color{brown}{\mbox{bottom}}} = {\color{brown}{E\_0}}$, otherwise the assignment is reversed. See Appendix XXX for a derivation.

To interpret these parameters if we set $\color{teal}{d\_2}=0$, then which is the Hill equation, which we modeled above . If we then additionally set $\color{teal}{d\_1}=0$ then $\color{brown}{E\_d}=\color{brown}{E\_0}$, in the limit as ${\color{teal}{d\_1}}\rightarrow \infty$ then ${\color{brown}{E\_d}}\rightarrow {\color{brown}{E\_1}}$, and if ${\color{teal}{d\_1}}=\color{purple}{C\_1}$ then ${\color{brown}{E\_d}} = ({\color{brown}{E\_0}} + {\color{brown}{E\_2}})/2$, which is the half maximal response (either the $\color{brown}{\mbox{IC}\_{50}}$ if treatment is an inhibitor or $\color{brown}{\mbox{EC}\_{50}}$ if treatment is agonist). The slope at ${\color{teal}{d\_1}}={\color{purple}{C\_1}}$ is

The evaluation of the functional form for ${\color{brown}{E\_d}}$ is numerically unstable due to the exponentiation. To transform using the trick, let

Then for a vector , let . Then

$$
{\color{brown}{E\_d}} = \mbox{exp}\!\left(\texttt{log-sum-exp}(\texttt{numerator\\_parts}) - \texttt{log-sum-exp}(\texttt{denominator\\_parts})\right).
$$

To implement the model in Stan we use the following parameterization.

# the brms MuSyC formula with given covariates  
synergy\_formula <- MuSyC\_formula(predictors = covariates)  
  
# will generate a formula like this  
synergy\_formula\_alt <- brms::brmsformula(  
 # The Stan MuSyC function is defined in BayesPharma::MuSyC\_stanvar()  
 response ~ MuSyC(  
 logd1 - logd1scale,  
 logd2 - logd2scale,  
 logE0,  
 logC1, logE1, h1,  
 logC2, logE2, h2,  
 logE3, logalpha),  
 nl = TRUE) +  
 # The free parameters are regressed against the given covariates  
 brms::lf(logE0 ~ covariates) +  
 brms::lf(logC1 + logE1 + h1 ~ covariates) +  
 brms::lf(logC2 + logE2 + h2 ~ covariates) +  
 brms::lf(logE3 + logalpha ~ covariates)

Note that if the logd1scale and logd1scale values are not provided in the the data, when the model is run, they are automatically computed as the mean value of the doses and is used to make the model easier to fit.

#### MuSyC model BayesPharma

### Enzyme Progress Kinetics

#### Enzyme Kinetic Modeling

Enzymes are proteins that catalyze chemical reactions. Not only do they facilitate producing virtual all biological matter, but they are crucial for regulating biological processes. In the early 20th century Michaelis and Menten described a foundational kinematic model for enzymes, where the substrate and enzyme reversibly bind, the substrate is converted to the product and then released.

$$
E + S \xrightleftharpoons[k\_b]{k\_f} C \xrightarrow{k\_{cat}} E + P
$$

where the free enzyme (E) reversibly binds to the substrate (S) to form a complex (C) with forward and backward rate constants of kf and kb, which is irreversibly catalyzed into the product (P), with rate constant of kcat, releasing the enzyme to catalyze additional substrate. The total enzyme concentration is defined to be the ET := E + C. The total substrate and product concentration is defined to be ST := S + C + P. The Michaelis constant is the defined to be the kM := (kb + kcat) / kf. The kcat rate constant determines the maximum turn over at saturating substrate concentrations, Vmax := kcat \* ET. The rate constants kcat and kM can be estimated by monitoring the product accumulation over time (enzyme progress curves), by varying the enzyme and substrate concentrations.

By assuming that the enzyme concentration is very low (ET << ST), they derived their celebrated Michaelis-Menten kinetics. Since their work, a number of groups have developed models for enzyme kinetics that make less stringent assumptions. Recently (Choi, Rempala, and Kim 2017), described a Bayesian model for the total QSSA model. To make their model more accessible, we have re-implemented it in the Stan/BRMS framework and made it available through the BayesPharma package.

Next we will formally define the problem and formulate the model as the solution to an ordinary differential equation. To illustrate, we will consider a a toy system where we assuming the kcat and kM are known and simulate a sequence of measurements using deSolve. We will then implement the ODE in Stan/BRMS using stanvars and show how the parameters of the toy system can be estimated. Since it is common to vary the enzyme and substrate concentrations in order to better estimate the kinematic parameters, we will show how we can improve the Stan/BRMS model to allow multiple observations, each with an arbitrary number of measurements. Then finally, we will consider a real enzyme kinetics data set and use the Stan/BRMS model to estimate the kinematic parameters. We will compare estimated parameters with those fit using standard approaches.

**Problem Statement**: Implement the total QSSA model in Stan/BRMS, a refinement of the classical Michaelis-Menten enzyme kinetics ordinary differential equation described in (Choi, Rempala, and Kim 2017). From their equation 2:

Observed data:  
 M = number of measurements # The product concentration Pt is measured  
 t[M] = time # at M time points t  
 Pt[M] = product #   
 ST = substrate total conc. # Substrate and enzyme concentrations are  
 ET = enzyme total conc. # assumed to be given for each observation  
  
 Model parameters:  
 kcat # catalytic constant  
 kM # Michaelis constant  
  
 ODE formulation:  
 dPdt = kcat \* ( # Change in product concentration at time t  
 ET + kM + ST - Pt +   
 -sqrt((ET + kM + ST - Pt)^2 - 2\* ET \* (ST - Pt))) / 2  
 initial condition:  
 P := 0 # There is zero product at time 0

In (Choi, Rempala, and Kim 2017) they prove, that the tQ model is valid when

where K = kb/kf is the dissociation constant.

**Simulate one observation**: Using the deSolve package we can simulate data following the total QSSA model. Measurements are made with random Gaussian noise with mean 0 and variance of 0.5. To visualize, the true enzyme progress curve is shown in blue, and the enzyme progress curve fit to the noisy measurements with a smooth loess spline is shown in orange. While the smooth fits well, we cannot estimate the parameters for the curve from it.

tQ\_model\_generate <- function(  
 time, kcat, kM, ET, ST) {  
 ode\_tQ <- function(time, Pt, theta) {  
 list(c(theta[1] \* (  
 ET + theta[2] + ST - Pt -  
 sqrt(  
 (ET + theta[2] + ST - Pt)^2 -  
 4 \* ET \* (ST - Pt))) / 2))  
 }  
 deSolve::ode(  
 y = 0, times = time, func = ode\_tQ,  
 parms = c(kcat, kM))  
}  
  
data\_single <- tQ\_model\_generate(  
 time = seq(0.00, 3, by=.05),  
 kcat = 3, kM = 5, ET = 10, ST = 10) |>  
 as.data.frame() |>  
 dplyr::rename(P\_true = 2) |>  
 dplyr::mutate(  
 P = rnorm(dplyr::n(), P\_true, 0.5), # add some observational noise  
 ST = 10, ET = 10)  
  
head(data\_single)

time P\_true P ST ET  
1 0.00 0.0000000 0.4583681 10 10  
2 0.05 0.7311578 0.9634292 10 10  
3 0.10 1.4243598 1.7089984 10 10  
4 0.15 2.0794197 2.2313994 10 10  
5 0.20 2.6964485 2.8697773 10 10  
6 0.25 3.2758537 3.0942174 10 10

|  |
| --- |
| Simulated enzyme progress curve with parameters kcat=3, kM=5, and total substrace and enzyme concentrations to 10 nM, simulated over 3 seconds |

**Fitting a single ODE observation in BRMS**: To implement in BRMS, we can use the stanvars to define custom functions. The key idea is call the ODE solver, in this case the backward differentiation formula (bdf) used to solve stiff ODEs, passing a function ode\_tQ that returns dP/dt, the change in product at time t. The ode\_tQ function depends on the product at time t as the state vector, the kinematic parameters to be estimated kcat and kM and the user-provided data of the enzyme and substrate concentrations ET and ST. To call ode\_dbf we pass in the initial product concentration and time (both equal to zero), measured time-points, parameters and user defied data. Finally we, extract the vector of sampled vector of product concentrations which we return.

stanvars\_tQ\_ode <- brms::stanvar(scode = paste("  
vector tQ\_ode(  
 real time,  
 vector state,  
 vector params,  
 data real ET,  
 data real ST) {  
   
 real Pt = state[1]; // product at time t  
 real kcat = params[1];  
 real kM = params[2];  
 vector[1] dPdt;  
 dPdt[1] = kcat \* (  
 ET + kM + ST - Pt  
 -sqrt((ET + kM + ST - Pt)^2 - 4 \* ET \* (ST - Pt))) / 2;  
 return(dPdt);  
}  
", sep = "\n"),  
block = "functions")

stanvars\_tQ\_single <- brms::stanvar(scode = paste("  
vector tQ\_single(  
 data vector time,  
 vector vkcat,  
 vector vkM,  
 data vector vET,  
 data vector vST) {  
   
 vector[2] params;  
 params[1] = vkcat[1];  
 params[2] = vkM[1];  
 vector[1] initial\_state;  
 initial\_state[1] = 0.0;  
 real initial\_time = 0.0;  
 int M = size(time);  
  
 vector[1] P\_ode[M] = ode\_bdf( // Function signature:  
 tQ\_ode, // function ode  
 initial\_state, // vector initial\_state  
 initial\_time, // real initial\_time  
 to\_array\_1d(time), // array[] real time  
 params, // vector params  
 vET[1], // ...  
 vST[1]); // ...  
   
 vector[M] P; // Need to return a vector not array  
 for(i in 1:M) P[i] = P\_ode[i,1];  
 return(P);  
}  
", sep = "\n"),  
block = "functions")

To use this function, we define kcat and kM as parameters and that we wish to sample P ~ tQ(...). Since all the data points define a single observation, we set loop = FALSE. We use gamma priors for kcat and kM with the shape parameter alpha=4 and the rate parameter beta=1. The prior mean is alpha/beta = 4/1 = 4 and the variance is alpha/beta^2 = 4/1 = 4. We also bound the parameters from below by 0. We initialize each chain at the prior mean and use cmdstanr version 2.29.2 as the backend, and use the default warmup of 1000.

model\_single <- brms::brm(  
 formula = brms::brmsformula(  
 P ~ tQ\_single(time, kcat, kM, ET, ST),  
 kcat + kM ~ 1,  
 nl = TRUE,  
 loop=FALSE),  
 data = data\_single |> dplyr::filter(time > 0),  
 prior = c(  
 brms::prior(prior = gamma(4, 1), lb = 0, nlpar = "kcat"),  
 brms::prior(prior = gamma(4, 1), lb = 0, nlpar = "kM")),  
 init = function() list(kcat = 4, kM = 4),  
 stanvars = c(  
 stanvars\_tQ\_ode,  
 stanvars\_tQ\_single))

Fitting the model takes ~15 seconds, with Rhat = 1 and effective sample size for the bulk and tail greater than 1400 for both parameters. The estimates and 95% confidence intervals are good.

Family: gaussian   
 Links: mu = identity; sigma = identity   
Formula: P ~ tQ\_single(time, kcat, kM, ET, ST)   
 kcat ~ 1  
 kM ~ 1  
 Data: dplyr::filter(data\_single, time > 0) (Number of observations: 60)   
 Draws: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;  
 total post-warmup draws = 4000  
  
Regression Coefficients:  
 Estimate Est.Error l-95% CI u-95% CI Rhat Bulk\_ESS Tail\_ESS  
kcat\_Intercept 2.81 0.45 2.10 3.86 1.00 1419 1547  
kM\_Intercept 4.43 2.16 1.25 9.59 1.00 1418 1514  
  
Further Distributional Parameters:  
 Estimate Est.Error l-95% CI u-95% CI Rhat Bulk\_ESS Tail\_ESS  
sigma 0.47 0.04 0.39 0.57 1.00 1555 1504  
  
Draws were sampled using sampling(NUTS). For each parameter, Bulk\_ESS  
and Tail\_ESS are effective sample size measures, and Rhat is the potential  
scale reduction factor on split chains (at convergence, Rhat = 1).

To visualize the posterior distribution vs. the prior distribution, we first sample from the prior, using the same brms::brm call with sample\_prior = "only" the argument.

model\_single\_prior <- model\_single |>  
 stats::update(  
 sample\_prior = "only",  
 iter = 2000)

And to plot, we use tidybayes to gather the draws and ggplot2 to map them to curves, with the prior as the orange curve, posterior as the blue curve, and the true parameter marked as a vertical line.

|  |
| --- |
|  |

Next, we plot the prior and posterior samples as a scatter plot. Note that the high correlation of the kcat and kM parameters in the posterior. This is expected, and typically better estimates require varying the enzyme and substrate concentrations.

|  |
| --- |
|  |

## Fitting multiple observations

Next we will extend the BRMS model to allow fitting common kcat, kM concentrations based on multiple replicas, or varying substrate/enzyme concentrations using BRMS. To demonstrate, we varying the enzyme and substrate concentrations, to better fit the kinematic parameters.

data\_multiple <- tidyr::expand\_grid(  
 kcat = 3,  
 kM = 5,  
 ET = c(3, 10, 30),  
 ST = c(3, 10, 30)) |>  
 dplyr::mutate(observation\_index = dplyr::row\_number()) |>  
 dplyr::rowwise() |>  
 dplyr::do({  
 data <- .  
 time <- seq(0.05, 3, by=.05)  
 data <- data.frame(data,  
 time = time,  
 P = tQ\_model\_generate(  
 time = time,  
 kcat = data$kcat,  
 kM = data$kM,  
 ET = data$ET,  
 ST = data$ST)[,2])  
 }) |>  
 dplyr::mutate(P = rnorm(dplyr::n(), P, 0.5))

|  |
| --- |
|  |

Next we will implement tQ\_multiple as a brms::stanvar object.

stanvars\_tQ\_multiple <- brms::stanvar(scode = paste("  
vector tQ\_multiple(  
 array[] int replica,  
 data vector time,  
 vector vkcat,  
 vector vkM,  
 data vector vET,  
 data vector vST) {  
  
 int N = size(time);  
 vector[N] P;  
 int begin = 1;  
 int current\_replica = replica[1];  
 for (i in 1:N){  
 if(current\_replica != replica[i]){  
 P[begin:i-1] = tQ\_single(  
 time[begin:i-1],  
 vkcat,  
 vkM,  
 vET[begin:i-1],  
 vST[begin:i-1]);  
 begin = i;  
 current\_replica = replica[i];  
 }  
 }  
 P[begin:N] = tQ\_single(time[begin:N], vkcat, vkM, vET[begin:N], vST[begin:N]);  
 return(P);  
}", sep = "\n"),  
block = "functions")

Then we will use it to fit multiple measurements for the same enzyme

model\_multiple <- brms::brm(  
 formula = brms::brmsformula(  
 P ~ tQ\_multiple(observation\_index, time, kcat, kM, ET, ST),  
 kcat + kM ~ 1,  
 nl = TRUE,  
 loop=FALSE),  
 data = data\_multiple,  
 prior = c(  
 brms::prior(prior = gamma(4, 1), lb = 0, nlpar = "kcat"),  
 brms::prior(prior = gamma(4, 1), lb = 0, nlpar = "kM")),  
 init = function() list(kcat = 4, kM = 4),  
 stanvars = c(  
 stanvars\_tQ\_ode,  
 stanvars\_tQ\_single,  
 stanvars\_tQ\_multiple))  
  
model\_multiple

To assess the model fit, we will re-fit the model just sampling from the prior

model\_multiple\_prior <- model\_multiple |>  
 stats::update(  
 sample\_prior = "only",  
 iter = 2000)

|  |
| --- |
|  |

Next we will sample enzyme progress curves from the posterior

|  |
| --- |
|  |

## Over dispersed negative binomial response

The response of an assay results from a measurement of the experimental system. Often the measurements are normalized so that the response for negative control is 1 (e.g. diseased) and the positive control is 0 (e.g. healthy). However when the robustness of the measurement depends on the measured value, this normalization can make it difficult to combine different measurements. An alternative approach is to model the measurements directly, to take into account the uncertainty associated with the response. To illustrate, if the measurement is the relative number of cells having a phenotype, then five out of ten cells and five thousand out of ten thousand cells will have the same response of 0.5, but the former will a less reliable measurement.

## Conclusion

The BayesPharma package aims to make Bayesian modeling for pharmacology more accessible. To do this, it makes easy to use interfaces for foundational dose-response type models. By building on the powerful Stan/BRMS framework, BayesPharma can be light-weight and adaptable to more advanced use cases while being computationally efficient and powerful.

Arezooji, Dorsa Mohammadi. 2020. “A Markov Chain Monte-Carlo Approach to Dose-Response Optimization Using Probabilistic Programming (RStan),” November. <https://arxiv.org/abs/2011.15034>.

Ashby, Deborah. 2006. “Bayesian Statistics in Medicine: A 25 Year Review.” *Stat. Med.* 25 (21): 3589–3631. <https://doi.org/10.1002/sim.2672>.

Bates, Douglas, Martin Mächler, Ben Bolker, and Steve Walker. 2015. “Fitting Linear Mixed-Effects Models Using Lme4.” *J. Stat. Softw.* 67 (October): 1–48. <https://doi.org/10.18637/jss.v067.i01>.

Betancourt, Michael. 2023. “Michael Betancourt: Writing.” <https://betanalpha.github.io/writing/>.

Bezanson, Jeff, Alan Edelman, Stefan Karpinski, and Viral B Shah. 2017. “Julia: A Fresh Approach to Numerical Computing.” *SIAM Rev.* 59 (1): 65–98. <https://doi.org/10.1137/141000671>.

Blei, David M, Alp Kucukelbir, and Jon D McAuliffe. 2017. “Variational Inference: A Review for Statisticians.” *J. Am. Stat. Assoc.* 112 (518): 859–77. <https://doi.org/10.1080/01621459.2017.1285773>.

Bliss, C I. 1956. “The Calculation of Microbial Assays.” *Bacteriol. Rev.* 20 (4): 243–58. <https://doi.org/10.1128/br.20.4.243-258.1956>.

Bürkner, Paul-Christian. 2017. “Brms: An R Package for Bayesian Multilevel Models Using Stan.” *J. Stat. Softw.* 80 (1): 1–28. <https://doi.org/10.18637/jss.v080.i01>.

Campbell, Gregory. 2017. “Bayesian Methods in Clinical Trials with Applications to Medical Devices.” *Communications for Statistical Applications and Methods* 24 (6): 561–81.

Carpenter, Bob, Andrew Gelman, Matthew D Hoffman, Daniel Lee, Ben Goodrich, Michael Betancourt, Marcus A Brubaker, Jiqiang Guo, Peter Li, and Allen Riddell. 2017. “Stan: A Probabilistic Programming Language.” *J. Stat. Softw.* 76 (January). <https://doi.org/10.18637/jss.v076.i01>.

Choi, Boseung, Grzegorz A Rempala, and Jae Kyoung Kim. 2017. “Beyond the Michaelis-Menten Equation: Accurate and Efficient Estimation of Enzyme Kinetic Parameters.” *Sci. Rep.* 7 (1): 17018. <https://doi.org/10.1038/s41598-017-17072-z>.

Cooner, F. 2019. “Considerations and Bayesian Applications in Pharmaceutical Development for Rare Diseases.” *Bayesian Applications in Pharmaceutical Development*. <https://doi.org/10.1201/9781315099798-17/considerations-bayesian-applications-pharmaceutical-development-rare-diseases-freda-cooner>.

Depaoli, Sarah, and Rens van de Schoot. 2017. “Improving Transparency and Replication in Bayesian Statistics: The WAMBS-Checklist.” *Psychol. Methods* 22 (2): 240–61. <https://doi.org/10.1037/met0000065>.

Duane, Simon, A D Kennedy, Brian J Pendleton, and Duncan Roweth. 1987. “Hybrid Monte Carlo.” *Phys. Lett. B* 195 (2): 216–22. <https://doi.org/10.1016/0370-2693(87)91197-X>.

Faya, Paul, Perceval Sondag, Steven Novick, Dwaine Banton, John W Seaman, Jr, James D Stamey, and Bruno Boulanger. 2021. “The Current State of Bayesian Methods in Nonclinical Pharmaceutical Statistics: Survey Results and Recommendations from the DIA / ASA‐BIOP Nonclinical Bayesian Working Group.” *Pharmaceutical Statistics*. <https://doi.org/10.1002/pst.2072>.

Gabry, Jonah, and Tristan Mahr. 2017. “Bayesplot: Plotting for Bayesian Models.” *R Package Version* 1 (0).

Gelman, Andrew, John B Carlin, Hal S Stern, David B Dunson, Aki Vehtari, and Donald B Rubin. 2013. *Bayesian Data Analysis*. Chapman; Hall/CRC.

Gelman, Andrew, and Jennifer Hill. 2006. *Data Analysis Using Regression and Multilevel/Hierarchical Models*. Cambridge University Press.

Gelman, Andrew, and Donald B Rubin. 1992. “Inference from Iterative Simulation Using Multiple Sequences.” *Stat. Sci.* 7 (4): 457–72.

Gelman, Andrew, Aki Vehtari, Daniel Simpson, Charles C Margossian, Bob Carpenter, Yuling Yao, Lauren Kennedy, Jonah Gabry, Paul-Christian Bürkner, and Martin Modrák. 2020. “Bayesian Workflow,” November. <https://arxiv.org/abs/2011.01808>.

Geman, S, and D Geman. 1984. “Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images.” *IEEE Trans. Pattern Anal. Mach. Intell.* 6 (6): 721–41. <https://doi.org/10.1109/tpami.1984.4767596>.

Gould, A Lawrence. 2019. “BMA-Mod: A Bayesian Model Averaging Strategy for Determining Dose-Response Relationships in the Presence of Model Uncertainty.” *Biom. J.* 61 (5): 1141–59. <https://doi.org/10.1002/bimj.201700211>.

GraphPad Software, LLC. 2023. *GraphPad Prism*.

Grieve, Andrew P. 2007. “25 Years of Bayesian Methods in the Pharmaceutical Industry: A Personal, Statistical Bummel.” *Pharm. Stat.* 6 (4): 261–81. <https://doi.org/10.1002/pst.315>.

Hastings, W K. 1970. “Monte Carlo Sampling Methods Using Markov Chains and Their Applications.” *Biometrika* 57 (1): 97–109. <https://doi.org/10.1093/biomet/57.1.97>.

Herbert Lee. 2023. “Coursera: Bayesian Statistics Specialization.” <https://www.coursera.org/specializations/bayesian-statistics>.

Hoffman, Matthew D, Andrew Gelman, and Others. 2014. “The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo.” *J. Mach. Learn. Res.* 15 (1): 1593–623.

Ianevski, Aleksandr, Anil K Giri, and Tero Aittokallio. 2022. “SynergyFinder 3.0: An Interactive Analysis and Consensus Interpretation of Multi-Drug Synergies Across Multiple Samples.” *Nucleic Acids Research*. <https://doi.org/10.1093/nar/gkac382>.

James Bradbury. 2018. “JAX: Composable Transformations of Python+NumPy Programs.”

Johnson, Alicia A, Miles Q Ott, and Mine Dogucu. 2022. *Bayes Rules!: An Introduction to Applied Bayesian Modeling*. CRC Press.

Johnstone, Ross H, David J Gavaghan, Ross H Johnstone, Rémi Bardenet, David J Gavaghan, and Gary R Mirams. 2017. “Hierarchical Bayesian Inference for Ion Channel Screening Dose-Response Data [Version 2; Peer Review: 2 Approved].” *Wellcome Open Research* 1.

Kay, M. 2018. “Tidybayes: Tidy Data and Geoms for Bayesian Models.” *R Package Version* 1 (3).

Kruschke, John K. 2021. “Bayesian Analysis Reporting Guidelines.” *Nat Hum Behav* 5 (10): 1282–91. <https://doi.org/10.1038/s41562-021-01177-7>.

Kucukelbir, Alp, Rajesh Ranganath, Andrew Gelman, and David Blei. 2015. “Automatic Variational Inference in Stan.” In *Advances in Neural Information Processing Systems 28*, edited by C Cortes, N D Lawrence, D D Lee, M Sugiyama, and R Garnett, 568–76. Curran Associates, Inc.

Labelle, Caroline, Anne Marinier, and Sébastien Lemieux. 2019. “Enhancing the Drug Discovery Process: Bayesian Inference for the Analysis and Comparison of Dose–Response Experiments.” *Bioinformatics* 35 (14): i464–73. <https://doi.org/10.1093/bioinformatics/btz335>.

Lakshminarayanan, Mani, and Fanni Natanegara. 2019. *Bayesian Applications in Pharmaceutical Development*. CRC Press.

Lesaffre, Emmanuel, Gianluca Baio, and Bruno Boulanger. 2020. *Bayesian Methods in Pharmaceutical Research*. CRC Press. <https://doi.org/10.1201/9781315180212>.

Lock, Eric F, and David B Dunson. 2015. “Shared Kernel Bayesian Screening.” *Biometrika* 102 (4): 829–42. <https://doi.org/10.1093/biomet/asv032>.

Loewe, S. 1926. “Effect of Combinations: Mathematical Basis of Problem.” *Arch. Exp. Pathol. Pharmakol.* 114: 313–26.

Ma, Eric J, and Arkadij Kummer. 2021. “Principled Decision-Making Workflow with Hierarchical Bayesian Models of High-Throughput Dose-Response Measurements.” *Entropy* 23 (6). <https://doi.org/10.3390/e23060727>.

Margolis, Elyssa B, Tanya L Wallace, Lori Jean Van Orden, and William J Martin. 2020. “Differential Effects of Novel Kappa Opioid Receptor Antagonists on Dopamine Neurons Using Acute Brain Slice Electrophysiology.” *PLoS One* 15 (12): e0232864. <https://doi.org/10.1371/journal.pone.0232864>.

McElreath, Richard. 2016. *Statistical Rethinking: A Bayesian Course with Examples in R and Stan*. CRC Press/Taylor & Francis Group.

Metropolis, Nicholas, Arianna W Rosenbluth, Marshall N Rosenbluth, Augusta H Teller, and Edward Teller. 1953. “Equation of State Calculations by Fast Computing Machines.” *J. Chem. Phys.* 21 (6): 1087–92. <https://doi.org/10.1063/1.1699114>.

Meyer, Christian T, David J Wooten, B Bishal Paudel, Joshua Bauer, Keisha N Hardeman, David Westover, Christine M Lovly, Leonard A Harris, Darren R Tyson, and Vito Quaranta. 2019. “Quantifying Drug Combination Synergy Along Potency and Efficacy Axes.” *Cell Syst* 8 (2): 97–108.e16. <https://doi.org/10.1016/j.cels.2019.01.003>.

Neal, Radford M. 1996. “Monte Carlo Implementation.” In *Bayesian Learning for Neural Networks*, edited by Radford M Neal, 55–98. New York, NY: Springer New York. <https://doi.org/10.1007/978-1-4612-0745-0\_3>.

Paszke, Adam, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, et al. 2019. “PyTorch: An Imperative Style, High-Performance Deep Learning Library,” December. <https://arxiv.org/abs/1912.01703>.

Plummer, Martyn. 2003. “JAGS: A Program for Analysis of Bayesian Graphical Models Using Gibbs Sampling.” In *3rd International Workshop on Distributed Statistical Computing*. 124th Series.

Polack, Fernando P, Stephen J Thomas, Nicholas Kitchin, Judith Absalon, Alejandra Gurtman, Stephen Lockhart, John L Perez, et al. 2020. “Safety and Efficacy of the BNT162b2 mRNA Covid-19 Vaccine.” *N. Engl. J. Med.* 383 (27): 2603–15. <https://doi.org/10.1056/NEJMoa2034577>.

Posit Software, PBC. 2023. “Bookdown.org Tag: Bayesian.” <https://bookdown.org/home/tags/bayesian/>.

R Core Team. 2023. “R: A Language and Environment for Statistical Computing.”

Ritz, Christian, Florent Baty, Jens C Streibig, and Daniel Gerhard. 2015. “Dose-Response Analysis Using R.” *PLoS One* 10 (12): e0146021. <https://doi.org/10.1371/journal.pone.0146021>.

Ruberg, Stephen J, Francois Beckers, Rob Hemmings, Peter Honig, Telba Irony, Lisa LaVange, Grazyna Lieberman, James Mayne, and Richard Moscicki. 2023. “Application of Bayesian Approaches in Drug Development: Starting a Virtuous Cycle.” *Nat. Rev. Drug Discov.*, February, 1–16. <https://doi.org/10.1038/s41573-023-00638-0>.

Salvatier, John, Thomas V Wiecki, and Christopher Fonnesbeck. 2016. “Probabilistic Programming in Python Using PyMC3.” *PeerJ Comput. Sci.* 2 (April): e55. <https://doi.org/10.7717/peerj-cs.55>.

Semenova, Elizaveta, Maria Luisa Guerriero, Bairu Zhang, Andreas Hock, Philip Hopcroft, Ganesh Kadamur, Avid M Afzal, and Stanley E Lazic. 2021. “Flexible Fitting of PROTAC Concentration–Response Curves with Changepoint Gaussian Processes.” *SLAS DISCOVERY: Advancing the Science of Drug Discovery* 26 (9): 1212–24. <https://doi.org/10.1177/24725552211028142>.

Senn, Stephen. 2022. “The Design and Analysis of Vaccine Trials for COVID-19 for the Purpose of Estimating Efficacy.” *Pharm. Stat.* 21 (4): 790–807. <https://doi.org/10.1002/pst.2226>.

Shterev, Ivo D, David B Dunson, Cliburn Chan, and Gregory D Sempowski. 2021. “BHTSpack: Bayesian Multi-Plate High-Throughput Screening of Compounds.”

Smith, Michael K, and Scott Marshall. 2006. “A Bayesian Design and Analysis for Dose-Response Using Informative Prior Information.” *J. Biopharm. Stat.* 16 (5): 695–709. <https://doi.org/10.1080/10543400600860535>.

Spiegelhalter, D, A Thomas, N Best, and W Gilks. 1996. “BUGS 0.5: Bayesian Inference Using Gibbs Sampling–Manual.” version ii. Medical Research Council Biostatistics Unit, Cambridge.

Tansey, Wesley, Kathy Li, Haoran Zhang, Scott W Linderman, Raul Rabadan, David M Blei, and Chris H Wiggins. 2021. “Dose–Response Modeling in High-Throughput Cancer Drug Screenings: An End-to-End Approach.” *Biostatistics* 23 (2): 643–65. <https://doi.org/10.1093/biostatistics/kxaa047>.

Team, R Core, and Others. 2013. “R: A Language and Environment for Statistical Computing.” Vienna, Austria.

Van de Schoot, Rens, Duco Veen, Laurent Smeets, Sonja D Winter, and Sarah Depaoli. 2020. “A Tutorial on Using the WAMBS Checklist to Avoid the Misuse of Bayesian Statistics.” *Small Sample Size Solutions: A Guide for Applied Researchers and Practitioners; van de Schoot, R. , Miocevic, M. , Eds*, 30–49.

Vehtari, Aki, Andrew Gelman, and Jonah Gabry. 2017a. “Erratum to: Practical Bayesian Model Evaluation Using Leave-One-Out Cross-Validation and WAIC.” *Stat. Comput.* 27 (5): 1433–33. <https://doi.org/10.1007/s11222-016-9709-3>.

———. 2017b. “Practical Bayesian Model Evaluation Using Leave-One-Out Cross-Validation and WAIC.” *Stat. Comput.* 27 (5): 1413–32. <https://doi.org/10.1007/s11222-016-9696-4>.

Wei, Xin, Lin Gao, Xiaolei Zhang, Hong Qian, Karen Rowan, David Mark, Zhengwei Peng, and Kuo-Sen Huang. 2013. “Introducing Bayesian Thinking to High-Throughput Screening for False-Negative Rate Estimation.” *J. Biomol. Screen.* 18 (9): 1121–31. <https://doi.org/10.1177/1087057113491495>.

Wickham, Hadley. 2009. *Ggplot2: Elegant Graphics for Data Analysis*. New York, NY: Springer.

Wickham, Hadley, Mara Averick, Jennifer Bryan, Winston Chang, Lucy McGowan, Romain François, Garrett Grolemund, et al. 2019. “Welcome to the Tidyverse.” *J. Open Source Softw.* 4 (43): 1686. <https://doi.org/10.21105/joss.01686>.

Wooten, David J, and Réka Albert. 2021. “Synergy: A Python Library for Calculating, Analyzing and Visualizing Drug Combination Synergy.” *Bioinformatics* 37 (10): 1473–74. <https://doi.org/10.1093/bioinformatics/btaa826>.

Wooten, David J, Christian T Meyer, Alexander L R Lubbock, Vito Quaranta, and Carlos F Lopez. 2021. “MuSyC Is a Consensus Framework That Unifies Multi-Drug Synergy Metrics for Combinatorial Drug Discovery.” *Nat. Commun.* 12 (1): 4607. <https://doi.org/10.1038/s41467-021-24789-z>.

Yang, Harry, and Steven Novick. 2019. *Bayesian Analysis with R for Drug Development: Concepts, Algorithms, and Case Studies*. CRC Press.