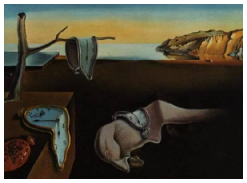


Journal of Statistical Software

MMMMMM YYYY, Volume VV, Issue II. *doi: 10.18637/jss.v000.i00*

BayesPharma: Bayesian methods for pharmacology models

Madeline J. Martin
University of Oregon

Elayne Vieira Diaz
University of California, San Francisco

P. Walter German
University of California, San Francisco

Elyssa B. Margolis
University of California, San Francisco

Matthew J. O'Meara
University of Michigan

Abstract

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 help analyze these experiments. BayesPharma is an `R` package to facilitate applying a principled Bayesian workflow to analyze pharmacology data, built around the `Stan` ecosystem. BayesPharma can be used out of the box to fit and analyze several foundational pharmacology models; as a pedagogical framework for learning Bayesian methods; and as a starting point for building and analyzing sophisticated pharmacological models.

Keywords: Bayesian, Pharmacology, R.

Introduction

As pharmacology experiments increase in complexity, it becomes increasingly challenging to analyze them. While there are many frameworks for fitting common curves in pharmacology ranging from the user friendly GraphPad Prism, the `drc` dose response curves `R` package**??**, to generic `nlm` packages, a principled approach is to use Bayesian statistics to quantify model uncertainty before and after collecting data. In practice this requires defining a functional form, a prior distribution over the model parameters, and likelihood function that models the data generation process. Then using Markov Chain Monte-Carlo (MCMC) or variational inference the prior and the likelihood are combined to generate samples from the posterior

distribution over the parameters.

Excitingly, there has been rapid progress in developing computational frameworks to facilitate developing and applying Bayesian models. A key example is the Stan ecosystem, which defines a model description language, inference engines, front-end interfaces to many common languages, and a suite of tools to analyze fit models ([Carpenter, Gelman, Hoffman, Lee, Goodrich, Betancourt, Brubaker, Guo, Li, and Riddell (2017), Bürkner (2017), Vehtari, Gelman, and Gabry (2017), Gabry and Mahr (2017), Kay (2018), Wickham (2009), Wickham, Averick, Bryan, Chang, McGowan, François, Grolemund, Hayes, Henry, Hester, Kuhn, Pedersen, Miller, Bache, Müller, Ooms, Robinson, Seidel, Spinu, Takahashi, Vaughan, Wilke, Woo, and Yutani (2019), Team and Others (2013)). To facilitate rapid application model development, the Bayesian Regression Modeling using Stan (BRMS) package in R implements a formula-based interface for Stan similar to `lme4`, that support defining linear and nonlinear predictors, hierarchical models, and a range of pre-specified or custom response functions. Once specified these models are translated into the Stan modeling language, where they are compiles and run.

Here we describe the BayesPharma R package, a novel Bayesian pharmacology modeling framework building on the `Stan` and `BRMS`. After reviewing basic Bayesian modeling concepts in the context of dose-response experiments, we will describe the package architecture and demonstrate the utility through several case studies.

1. Related work

Dimensions of evaluation - Methodology - Features / functionality - Generality - Usability

Methods

2. Bayesian Modeling Workflow

Bayesian statistics is 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. Bayes theorem can be derived from factoring the joint probability distribution over parameters and data, $P(\theta, D)$ into conditional probability distributions two different ways, $P(\theta|D)P(D)$, and $P(D|\theta)P(\theta)$. Setting them equal to each other and dividing through by the evidence, $P(D)$, gives:

$$P(\theta|D) = \frac{P(D|\theta) \cdot P(\theta)}{P(D)}$$

$$\text{POSTERIOR} = \frac{\text{LIKELIHOOD} \cdot \text{PRIOR}}{\text{EVIDENCE}}$$

While the prior and likelihood distributions are explicitly defined in the model, the evidence is only implicitly defined by integrating the likelihood over the entire prior, and is typically intractable to compute explicitly. One way to conceptualize the evidence is simply as

the (unique) constant that allows the posterior to be properly normalized and integrate to one. While range of techniques have been developed to estimate the evidence, an important technique that sidesteps the issues is to note that the relative posterior probability of two different parameters $P(\theta_1|D)$ vs. $P(\theta_2|D)$ reduces to the computing the likelihood ratios $P(D|\theta_1)/P(D|\theta_2)$ as the prior and the evidence cancel out. By taking small steps in the parameter space and repeatedly evaluating the relative posterior probability it is possible to (in the long run) sample points from the posterior probability distribution. Intuitively, this is akin to running dynamical simulations using the posterior as the Boltzmann distribution.

The challenge in estimating the evidence is akin to estimating the partition function in the Boltzmann distribution from statistical mechanics. Another

is In general it is not possible to solve for the posterior distribution analytically.

Alternatively, one can draw samples from the posterior using Markov chain Monte Carlo.

Stan provides a programming language and inference engine to define

, draw samples samples from the

The scientific method can be conceptualized through building and analyzing models. Consider we are interested in interacting with a complex system such as an rocket ship, ecosystem, person with an illness etc., but we are cautious because doing so may be dangerous, expensive, slow, or unethical. Therefore we may develop a tractable model system to interact with instead (such as a very small rocket ship, zebrafish, or molecular simulation, etc.). If the model system corresponds with to the system of interest in relevant ways, then poking and prodding the model system can be use to anticipate how the system of interest will respond when it is in turn poked and prodded. A key step in establishing this correspondence is to collect a small amount of data from the system of interest and use it design the model system to be representative.

When scientific model systems are constructed by composing conditional probability distributions, they are often called Bayesian models. This is because the key step of going from a nebulous set of possible models to more precise set of models based on observed data uses the celebrated Bayes theorem. While basic Bayesian theory is covered in range of introductory and advanced textbooks. To complement those totally reasonable expositions, here we will give correct, but informal derivation of Bayesian probability, to helpfully build intuition and facilitate communication with practitioners. Let $\theta \in \Theta$ be a vector of model parameters and let $\mathrm{D} \in \mathbb{D}$ observed data. For concreteness, you can think of $\theta = (\mathrm{Top}, \mathrm{Bottom}, \mathrm{IC}_{50}, \mathrm{Slope})$ as the parameters for specific sigmoid function, and $\mathrm{D} = \{(\mathrm{dose}_i, \mathrm{response}_i)_{i \in N}\}$, a set of dose-response measurements. A probability distribution naturally models an event or an occurrence. The event is sometimes called a random variable. Given a region, the chance that the event happens in that region is the probability density over it. Assuming the event definitely happens, then the total density must integrate to unity. A joint distribution over two random variables is just the probability that both events occur. A conditional probability distribution can be thought of as a stochastic function, where evaluation corresponds to setting the values to be conditioned and obtaining the remaining values. Concretely if you have a scatter plot, you could sample points by first sampling along the x-axis and within the narrow range of the chosen x-value, pick a y-value. Or visa versa, first sample along the y-axis and then within the narrow range of the chose y-value, pick the x-value. Then the joint probabil-

Stochastic:A stochastic function is a mathematical function that involves some element of randomness or probability. In other words, its output is not deterministic and may vary each time the function is evaluated. One common way to model a stochastic function is to use probability distributions. For example, a stochastic function might use a normal distribution to add some random noise to a deterministic input, or it might use a Poisson distribution to model the probability of rare events occurring.

ity distribution over parameters and data, $P(\theta, D)$ can be written as conditional probability distributions two different ways, $P(\theta|D)P(D)$, and $P(D|\theta)P(\theta)$. Setting them equal to each other and dividing through by $P(D)$ gives Bayes Theorem:

$$P(\theta|D) = \frac{P(D|\theta)P(\theta)}{P(D)}.$$

With that in mind, lets look at each term in the is equation. The easiest is the $P(\theta)$, this called the prior must be specified before hand. We'll come back to how to chose priors below. The term $P(D|\theta)$ is called the likelihood, and can thought of as the data generation process, the "model system" if you will. On the left is the $P(\theta|D)$, which is called the posterior. This is a little trickier to understand, but thinking of it as a stochastic function, we can at least inspect the signature: it takes in data, and produces random samples of parameters. Finally the $P(D)$ is called the evidence, and is basically there to make the left and the right be equal to each other. As we'll see below, we don't actually need to evaluate it, but if did we could use a trick called marginalization.

How do we sample parameter values from the posterior distribution? One way to conceptualize problem is to use Boltzmann's equation to convert the probability distribution to an energy function and temperature. Then to sample, begin at a location in parameter space and simulate the kinematic trajectory as it would move through the parameter space over time. Given enough time, if the parameter can access all parts of the probability distribution, the parameter will equilibrate and give sample from the probability distribution. While this is guaranteed to happen eventually, in practice a common concern is that for any fixed amount of equilibration, it is unclear if there has been sufficient mixing. We'll come back to strategies to detect and handle issues with mixing below. A basic kinematic sampling algorithm called Markov Chain Monte Carlo (MCMC) says to take a step, and if it has lower energy, accept, if it has higher energy accept with a probability proportional to the difference in probabilities between the two states and the simulation temperature.* A key insight is that because only relative probabilities are needed, the global evidence normalization factors cancel and therefore they don't even need to be computed. More sophisticated algorithms take into account the not only the specific energy values but also the energy gradient. Thinking of the object as having inertia, the sampling allows it to overcome certain types of local minima. to take steps that are likely to be accepted, sometimes called Hamiltonian Monte Carlo. A particularly annoying pat of HMC is that in some cases, such as a narrow ravine in the energy landscape, the trajectory will vibrate back and forth without making much forward progress. A clever heuristic that is used by the Stan, called No U-turn sampling (NUTs), tries to dampen these types of unproductive moves.

The principled Bayesian workflow (Gelman, Vehtari, Simpson, Margossian, Carpenter, Yao, Kennedy, Gabry, Bürkner, and Modrák 2020, Van de Schoot, Veen, Smeets, Winter, and Depaoli (2020)), describes a general protocol 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 *posterior* distribution over the parameters using Hamiltonian Markov Chain Monte Carlo.

2. Check for sampling convergence.

* To actually generate the samples, we use a method called Markov Chain Monte Carlo (MCMC), which involves taking a step in the parameter space and accepting the new location based on the difference in probability densities between the current and proposed locations. This process is repeated many times to generate a sequence of samples from the posterior distribution.

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.

3. BayesPharma package design

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

To provide data to the BayesPharma package, the user provides R `data.frame` with columns for the response, treatments, and optionally additional covariates such as `drug_id` or `batch_id`. This is passed to the model function optionally including 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 `brms::brmsfit` object is returned to the user and can be used for analysis. To illustrate

3.1. Model specification

Here we will describe the components that are needed to to specify the model.

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 equations that declares how the response on the left side is sampled from a parameterized distribution on the right side. For the default linear formulas the right side specifies mean response with a linear combination of covariates added together with implicitly defined model parameters. For example, the formula

$$\text{response} \sim 1 + \text{drug\_id}$$

says that `response` is sampled from a distribution with mean $\beta_0 \cdot 1 + \beta_1 \cdot \text{drug\_id}_1 + \beta_2 \cdot \text{drug\_id}_2 + ... \beta_n \cdot \text{drug\_id}_n$ where β_i are scalar parameters and drug_id_i is an indicator variable for drug i. By default the sampling distribution is a Gaussian, but other distributions can be specified from the distribution family with a link function using `family` argument. For example, to model count data, which is strictly positive, setting `family=brms::poissson()`. To model more general sampling equations, `brms::formula` can be specified as non-linear by setting `nl=TRUE`, and all model parmaeters must be explicitly defined. Building on this framework, the `brms` package supports a wide range types of regression models including, hierarchical models or random effects models and 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 each model type, 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,
  family = brms::student)

# 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 XXX 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 including `dplyr`, which can `filter` subsets of rows, `select` subsets of columns, compute new columns rowwise with `mutate` or perform split-apply-combine to operate over subsets of rows, and data in different `data.frame` objects can be merged using SQL-like joins like `left_join`. If data is organized e.g. in plate-layout with multiple observations per-row, the `tidyr` package has `pivot_longer` and `pivot_wider` which can be used to transform the shape of `data.frame` objects. When observation measurements and covariates come in in non-standard format, string manipulation is possible through the `stringr` package. Finally, for exploratory visualization, the `ggplots2` package implements the grammar-of-graphics workflow for mapping data in a tidy format to aesthetic attributes of geometric objects on in the plot such as the coordinates of points or lines. For a good introduction to data manipulation using the tidyverse, the R for Data Science (2e) book and website (https://r4ds.hadley.nz/) are quite good. Throughput each of the vigenttes below we will make use of the tidyverse data manipulation.

```
demo_data <- tibble::tibble(
  drug_id = c("C1", "C2", "C3"),
  ac50 = c(-8, -7, -6)) |>
  dplyr::cross_join(
    tibble::tibble(log_dose = seq(-7, -5, length.out = 30))) |>
  dplyr::mutate(
    mean_response = BayesPharma::sigmoid(
      ac50 = ac50,
      hill = 1,
      top = 1,
      bottom = 0,
      log_dose = log_dose),
    response = stats::rnorm(n = dplyr::n(), mean = mean_response, sd = 0.2))
```

Prior: A key principle of Bayesian models is that they require specifying priors. For new comers, understanding how determine how they 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. So for setting up and getting started with a new model fit, it is best to give as strong (more constrained) of priors as possible. From a scientific perspective priors and posterior distributions can be interpreted as capturing the uncertainty in parameters before and after observing the data. So, from this perspective, weaker (less constrained) priors are preferred in order to "let 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 that the inferences from the model are not sensitive to reasonable choices of the prior. In a way this actually makes choosing the prior less stressful there is no singular best prior choice, and instead reflect the scientific questions of the modeling process. For a deeper discussion and practical advice on selecting priors, see (https://github.com/stan-dev/stan/wiki/Prior-Choice-Recommendations) and the references therein. 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 the above case the normal function is defined here (https://mc-stan.org/docs/2_20/functions-reference/normal-distribution.html), the variable name, and optional arguments such as the 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.

Init: To estimate the posterior distribution using markov chain Monte Carlo sampling, initial parameters values must be given. From these initial parameters, the parameters are simulated in multiple independent chains in such a way that they converge to a sample from the posterior distribution. The the initial parameter values should 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 then the initial values can be adjusted alogn 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)})
}
```

To summarize the model

Name	Type	Treatments	Parameters	Response
Sigmoid	one treatment	log_dose	top, bottom, AC50, hill	response
MuSyC	two treatments	logd1, logd2	logE[0-3], logC[1,2], h[1,2], logalpha	response
tQ	enzyme kinetics	series_index, time, ET, ST	Kcat, kM	P

3.2. Model fitting

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

```
# this is model...
demo_model <- BayesPharma::sigmoid_agonist_model(
  formula = demo_formula,
  data = demo_data)
```

```
Warning: There were 108 transitions after warmup that exceeded the maximum treedepth. Increase max_treedepth above 10. See
https://mc-stan.org/misc/warnings.html#maximum-treedepth-exceeded

Warning: Examine the pairs() plot to diagnose sampling problems
```

```
# 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)
```

```
Warning: There were 41 transitions after warmup that exceeded the maximum treedepth. Increase max_treedepth above 10. See
https://mc-stan.org/misc/warnings.html#maximum-treedepth-exceeded

Warning: Examine the pairs() plot to diagnose sampling problems
```

```
brms::expose_functions(demo_model_alt, vectorize = TRUE)
demo_model$model_type <- "sigmoid_agonist"
```

3.3. Model Evaluation

Model Evaluation: To evaluate the model fit involves evaluating the convergence of the simulation and

data model diagnostics formula => model => Model estimate summaries prior - parameter estimate tables init - prior / posterior densities - marginal distributions Model inference - hypothesis testing - model comparison

model implementation - generate | R code to generate synthetic data - stanvars | Stan code implementing the model - formula | build BRMS formula connecting covariates, parameters and response - prior | default priors for model parameters - init | default initial values for parameter simulations - run

4. Model selection

(Vehtari, 2016, https://arxiv.org/abs/1507.04544) predictive accuracy how well would the model generalize to new data? empirical risk minimization of the model log-likelihood leave-k-out cross validation - expensive to re-fit model k times - approximation via importance sampling but may be unstable - Pareto smoothed importance sampling (PSIS) more stable

Case Studies

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

5. Hill Equation

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

Originally paper, the dose-response data analysis was done by 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 was normalized to % baseline then fit to a 4-parameter log-logistic dose response model, setting the top (max response) to 100% and estimating the IC50, its variance, and the bottom (min response).

5.1. Fitting the sigmoid model

Using the BayesPharma package, we can re-fit the sigmoid model with a negative slope, and fixing the top parameter to 100 as the response is normalized to a no-drug 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 top (max response) parameter prior set to a constant value of 100 because top is normalized to 100 and the default broad prior for ic50, hill and bottom. 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.

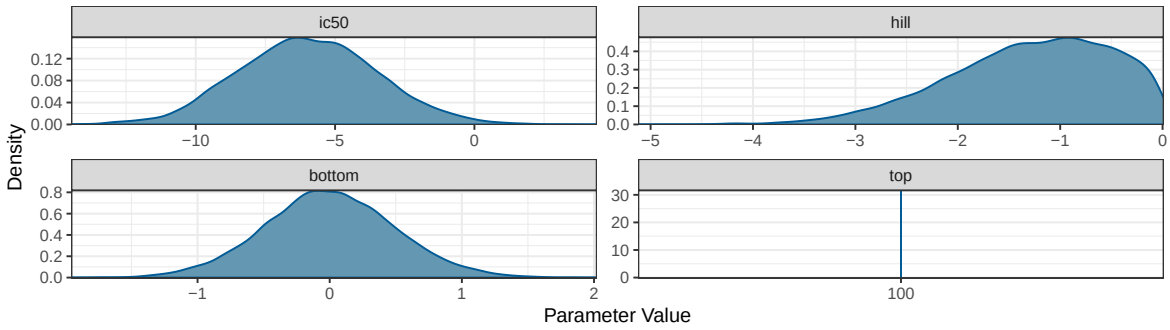


Figure 1: KOR antagonists prior distribution

```
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_antagonist_model` function. We will use the default response distribution of the model (*family = gaussian()*).

```
kor_sample_prior <- BayesPharma::sigmoid_antagonist_model(
  data = kor_antag |> dplyr::select(substance_id, log_dose, response),
  prior = kor_prior,
  sample_prior = "only")
```

just genertaes samples from prior. it is soley based on the prior looks like the posterior but just prior.
check process.

And then plot of the prior predictive distributions:

```
kor_sample_prior |>
  BayesPharma::density_distributions_plot()
```

To sample from the model we will 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_antagonist_model(
  data = kor_antag |> dplyr::select(substance_id, log_dose, response),
  formula = BayesPharma::sigmoid_antagonist_formula(
    predictors = 0 + substance_id),
  prior = kor_prior)
```

5.2. Analyzing model fit

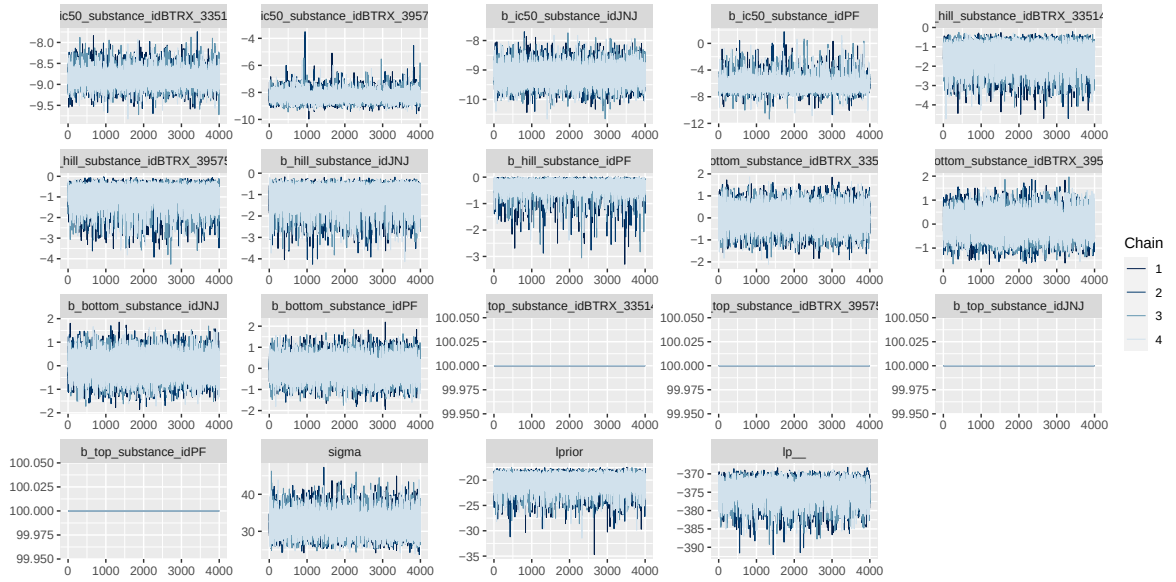
The BRMS generated model summary shows the formula that the expected response is sigmoid function of the log_dose with four parameters, and a shared Gaussian distribution. Each parameter is dependent on the substance_id. Since want to fit a separate model for each substance we include a `0 +` to indicate that there is no common intercept. The 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 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.

The family here indicates what type of connection you want between the mean and your response variables i.e., ic50, top, bottom, log_dose here. To visualized this, in figure 4 you can imagin having a Gaussian around each point, which results in the colored regions. Each color would then indicate how far that region is from the mean 1SD? 2SDs? and so on

```
 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

Population-Level Effects:
                               Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
ic50_substance_idBTRX_335140      -8.84      0.20    -9.20    -8.40 1.00    13542     8700
ic50_substance_idBTRX_395750      -8.24      0.41    -8.92    -7.37 1.00    11435     5487
ic50_substance_idJNJ              -9.15      0.32    -9.77    -8.51 1.00    15085     9371
ic50_substance_idPF               -6.15      1.06    -7.64    -3.33 1.00     8184     4927
hill_substance_idBTRX_335140      -1.47      0.61    -2.92    -0.58 1.00    14622    11291
hill_substance_idBTRX_395750      -0.89      0.49    -2.17    -0.26 1.00    10926     6320
hill_substance_idJNJ              -1.01      0.51    -2.38    -0.40 1.00    14262    10996
hill_substance_idPF               -0.31      0.25    -0.93    -0.03 1.00     7332     5246
bottom_substance_idBTRX_335140    -0.00      0.50    -0.99     0.98 1.00    16477    11542
bottom_substance_idBTRX_395750     0.02      0.50    -0.94     0.99 1.00    19064    12423
bottom_substance_idJNJ            -0.01      0.50    -0.99     0.97 1.00    17501    11293
bottom_substance_idPF              0.01      0.50    -0.97     0.98 1.00    18273    12265
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

Family Specific Parameters:
      Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
sigma    32.16      2.88    27.15    38.42 1.00    15048    12016

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::prior_posterior_densities_plot(
  model = kor_model,
  predictors_col_name = "substance_id",
  half_max_label = "ic50",
  title_label="")
```

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 to highlight the mode and high density intervals:

```
BayesPharma::posterior_densities_plot(
  kor_model,
  predictors_col_name = "substance_id",
  half_max_label = "ic50",
  title_label = "")
```

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

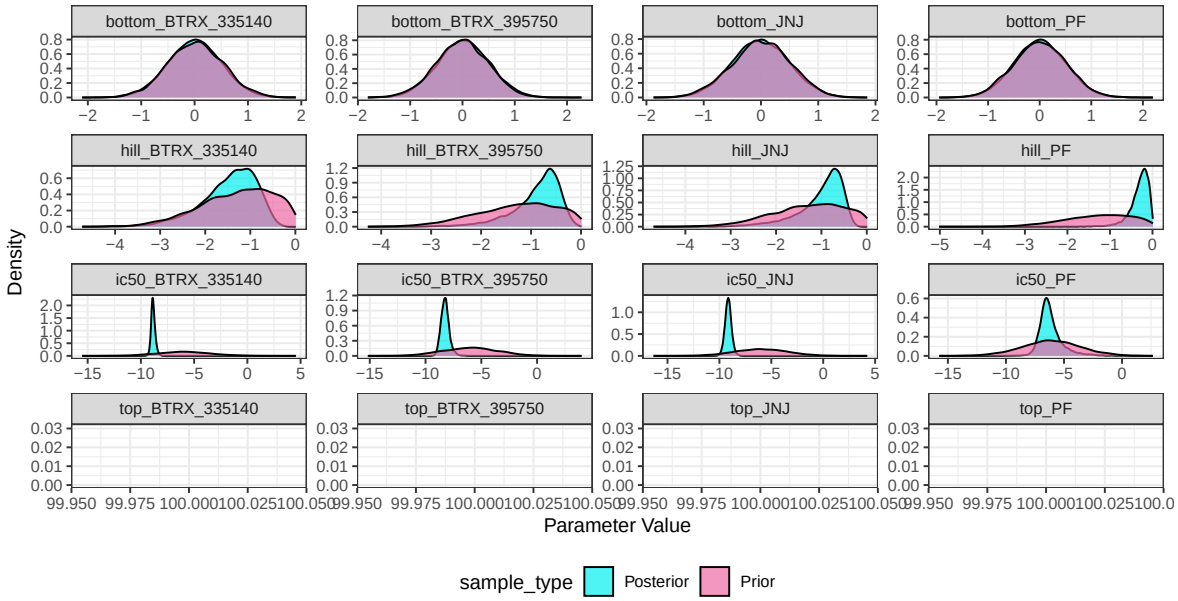


Figure 2: KOR antagonists model, compare prior and posterior distributions for each substance

```
BayesPharma::posterior_draws_plot(
  model = kor_model,
  title = "")
```

5.3. Comparing alternative models

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

```
          prior class coef group resp dpar  nlpar   lb   ub source
 normal(-6, 2.5)     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

6. 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 give 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, the package, Pareto k value for each

PSIS is a method used to approximate the ELPD, which is a measure of how well a Bayesian model predicts new, unseen data. PSIS estimates the ELPD by using importance sampling and smoothing techniques, and it provides Pareto k values for each data point, which indicate how well the model fits that particular data point.

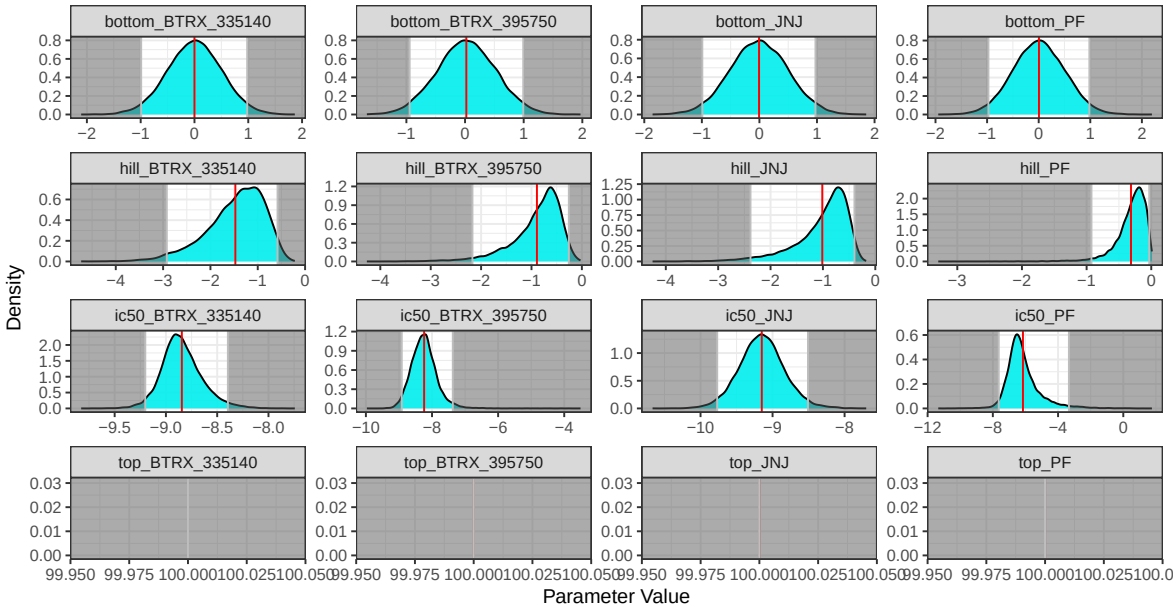


Figure 3: KOR Antagonists, posterior distribution for each substance

data point is computed, where k less than 0.5 is good, between 0.5 and 0.7 is ok, and higher than 0.7 indicates the data point is not fit by the model well. Evaluating the model for the KOR antagonists, shows that the model fits the data well.

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

NULL
```

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.

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

           elpd_diff se_diff
kor_model2 0.0       0.0
kor_model  0.0       0.8
```

In LOO cross-validation, the model is repeatedly fit with all but one data point, and the left-out data point is used to evaluate the model's predictive performance. This process is repeated for all data points, and the average log probability of the left-out data points gives an estimate of the ELPD.

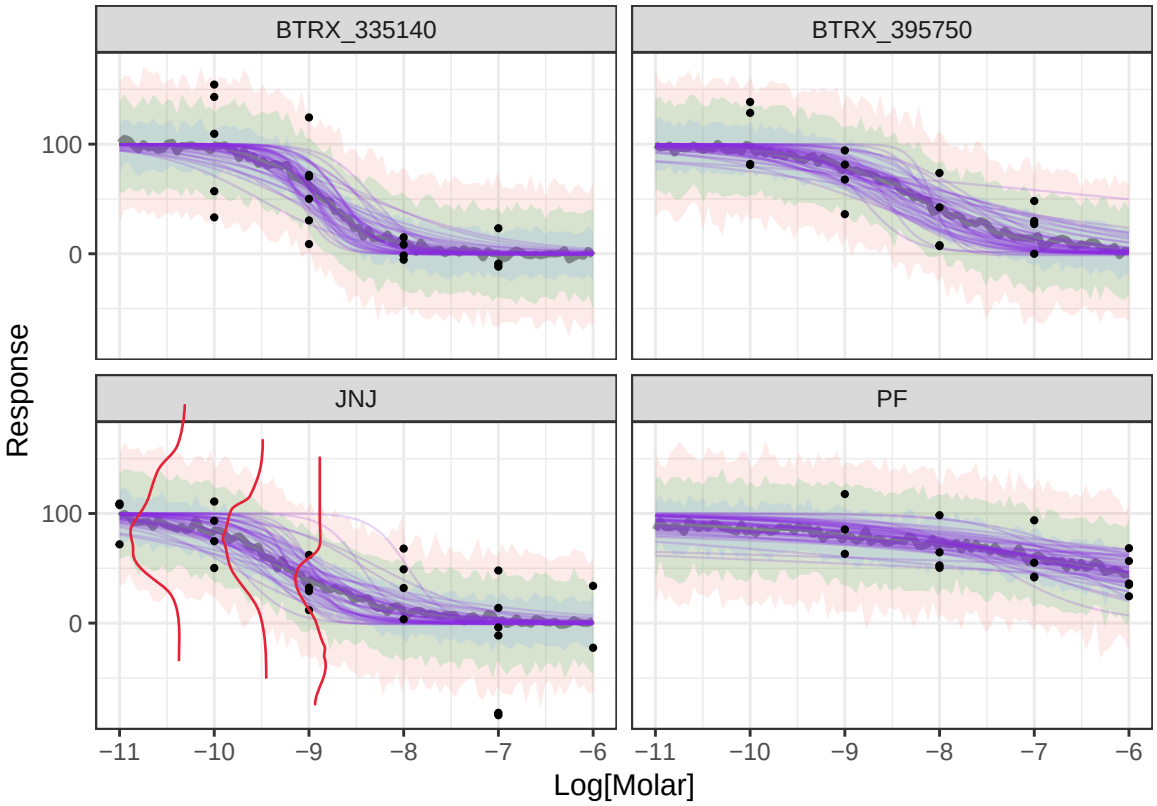
##Analysis Using the drc Package

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(
```

ELPD is a global measure of model fit that takes into account the predictive performance of the model for all data points. It is computed by averaging the log probability of each data point in the posterior distribution, obtained by refitting the model with the remaining data points excluded, which is similar to the process used in LOO cross-validation. So, in Bayesian model evaluation, LOO is often used as a practical way to estimate the ELPD, which provides an overall assessment of the model's predictive performance, taking into account all data points.



I repeat here as well. To visualized why the Gaussian familty in the summary above, you can imagin having a Gaussian around each point, which results in the colored regions. Each color would then indicate how far that region is from the mean 1SD? 2SDs? and so on

Figure 4: KOR antagonists, posterior draws

```
        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 x 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
```

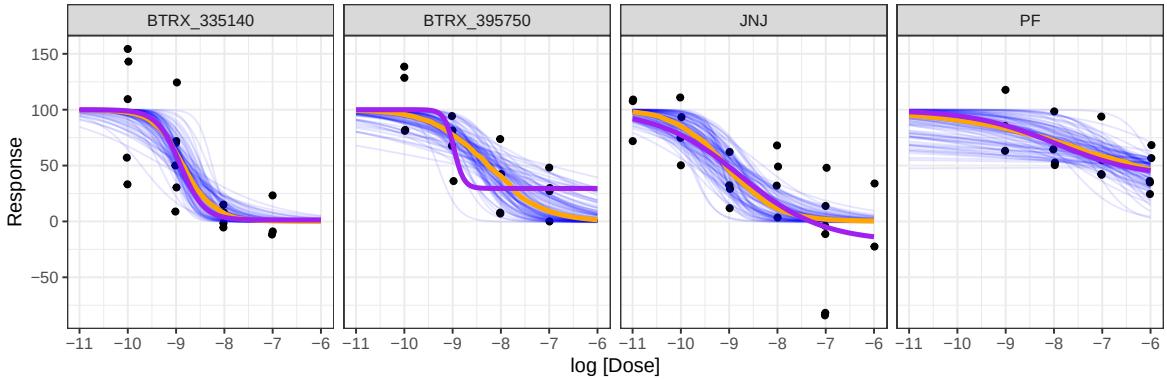


Figure 5: KOR antagonists conditional effects. The blue lines are samples from the BayesPharma kor_model posterior distribution, the orange line is the conditional mean, and the purple line is the conditional mean for the DRC model fit.

```
 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.

```
Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
i Please use `linewidth` instead.
```

7. MuSyC synergy model

When two different treatments are combined they may interact to cause a response. For end-point assays, if the response is stronger or weaker than what would be expected with an additive model, the treatments are said to be epistatic. For sigmoidal dose-response models, however, the analysis may be more complicated. One drug may not only may shift the maximal response (efficacy) of the other, but it may also shift the effective dose and shape of the response (potency). Historically a range of models have been proposed that capture different aspects of synergy, for example the Bliss independenceBliss (1956) and Loewe additivityLoewe (1926) are null-models for no synergistic efficacy or potency, respectively. The `SynergyFinder` R packageIanevski, Giri, and Aittokallio (2022) and the `synergy` python packageWooten 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.Meyer, Wooten, Paudel, Bauer, Hardeman, Westover, Lovly, Harris, Tyson, and Quaranta (2019); Wooten, Meyer, Lubbock, Quaranta, and Lopez (2021) derived an integrated functional synergistic sigmoidal dose-response, which has the Loewe and Bliss models as special cases. They implemented a Bayesian model-fitting strategy in Matlab, and a maximum likelihood model-ftting 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.

7.1. MuSyc Functional Form

The functional form for the MuSyC model gives an equation for the response E_d at doses of d_1 and d_2 of the two treatments and has 9 free parameters C_1, C_2, E_0, E_1, E_2, E_3, h_1, h_2, α:

$$E_d = \frac{{C_1}^{h_1}{C_2}^{h_2}E_0 + {d_1}^{h_1}{C_2}^{h_2}E_1 + {C_1}^{h_1}{d_2}^{h_2}E_2 + {d_1}^{h_1}{d_2}^{h_2}\alpha E_3}{{C_1}^{h_1}{C_2}^{h_2} + {d_1}^{h_1}{C_2}^{h_2} + {C_1}^{h_1}{d_2}^{h_2} + {d_1}^{h_1}{d_2}^{h_2}\alpha} \tag{1}$$

To interpret these parameters if we set $d_2 = 0$, then

$$E_d = \frac{{C_1}^{h_1}E_0 + {d_1}^{h_1}E_1}{{C_1}^{h_1} + {d_1}^{h_1}} \tag{2}$$

which is the Hill equation, which we modeled above **??**. If we then additionally set $d_1 = 0$ then $E_d = E_0$, in the limit as $d_1 \to \infty$ then $E_d \to E_1$, and if $d_1 = C_1$ then $E_d = (E_0 + E_2)/2$, which is the half maximal response (either the IC_{50} if treatment 1 is an inhibitor or EC_{50} if treatment 1 is agonist). The slope at $d_1 = C_1$ is

$$\begin{aligned}
\frac{\mathrm{d}\,E_d}{\mathrm{d}d_1} &= {C_1}^{v}E_0\frac{\mathrm{d}}{\mathrm{d}d_1}\frac{1}{{C_1}^{h_1} + {d_1}^{h_1}} + E_1\frac{\mathrm{d}}{\mathrm{d}d_1}\frac{{d_1}^{h_1}}{{C_1}^{h_1} + {d_1}^{h_1}} \\
&= {C_1}^{h_1}E_0\frac{h_1{d_1}^{h_1-1}}{\left({C_1}^{h_1} + {d_1}^{h_1}\right)^2} + E_1\frac{{C_1}^{h_1}h_1{d_1}^{h_1-1}}{\left({C_1}^{h_1} + {d_1}^{h_1}\right)^2} \\
&= (E_0 + E_1)
\end{aligned}$$

The evaluation of the functional form for E_d is numerically unstable. To transform using the log_sum_exp trick, let

$$\begin{aligned}
\text{numerator_parts} = [& \\
& h_1\log(C_1) + h_2\log(C_2) + \log(E_0), \\
& h_1\log(d_1) + h_2\log(C_2) + \log(E_1), \\
& h_1\log(C_1) + h_2\log(d_2) + \log(E_2), \\
& h_1\log(d_1) + h_2\log(d_2) + \log(E_3) + \log(\alpha)] \\
\text{denominator_parts} = [& \\
& h_1\log(C_1) + h_2\log(C_2), \\
& h_1\log(d_1) + h_2\log(C_2), \\
& h_1\log(C_1) + h_2\log(d_2), \\
& h_1\log(d_1) + h_2\log(d_2)]
\end{aligned}$$

Then

$$E_d = \exp(\text{log\_sum\_exp}(\text{numerator\_parts}) - \text{log\_sum\_exp}(\text{denominator\_parts}))\,.$$

7.2. Implementation and usage of the MuSyC model in Stan/BRMS

8. Michaelis-Menten enzyme progress model

9. 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 Michaels 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.

$$E + S \underset{k_b}{\overset{k_f}{\rightleftharpoons}} C \xrightarrow{k_{cat}} E + P$$

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.

10. Outline for Vignette

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.

11. 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, et al., 2017, DOI: 10.1038/s41598-017-17072-z). 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 *et al.* 2017) they prove, that the tQ model is valid when

$$\frac{K}{(2 \cdot ST)} \cdot \frac{(ET + kM + ST)}{\sqrt{(ET + kM + ST + P)^2 - 4 \cdot ET(ST - P)}} \ll 1,$$

where `K = kb/kf` is the dissociation constant.

12. 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(
```

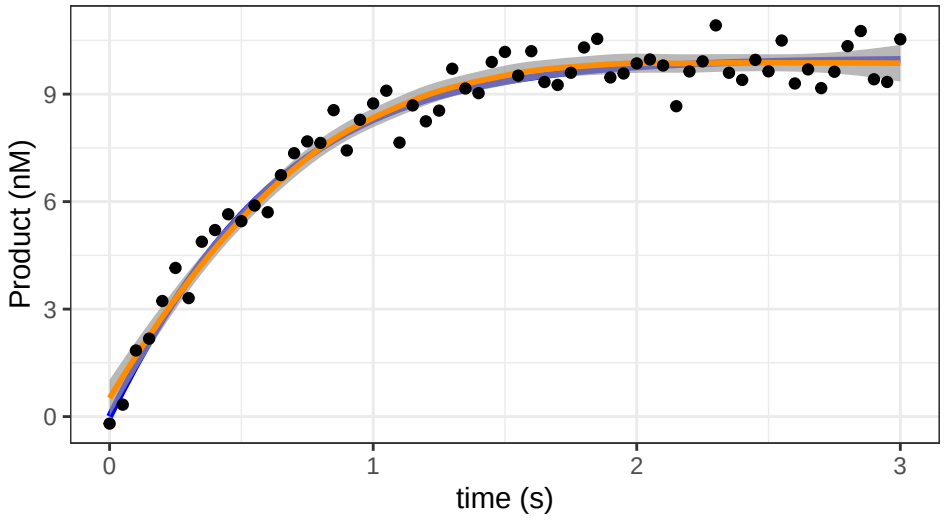


Figure 6: Simulated enzyme progress curve with parameters kcat=3, kM=5, and total substrace and enzyme concentrations to 10 nM, simulated over 3 seconds

```
    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 -0.1952277 10 10
2 0.05 0.7311578  0.3337647 10 10
3 0.10 1.4243598  1.8453044 10 10
4 0.15 2.0794197  2.1755056 10 10
5 0.20 2.6964485  3.2259417 10 10
6 0.25 3.2758537  4.1491231 10 10
```

13. 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

Population-Level Effects:
             Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
kcat_Intercept     2.97      0.47     2.20     4.03 1.00     1232     1536
kM_Intercept       4.32      2.09     1.17     9.20 1.00     1201     1387

Family Specific Parameters:
      Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
sigma     0.52      0.05     0.44     0.62 1.00     1931     1578

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).
```

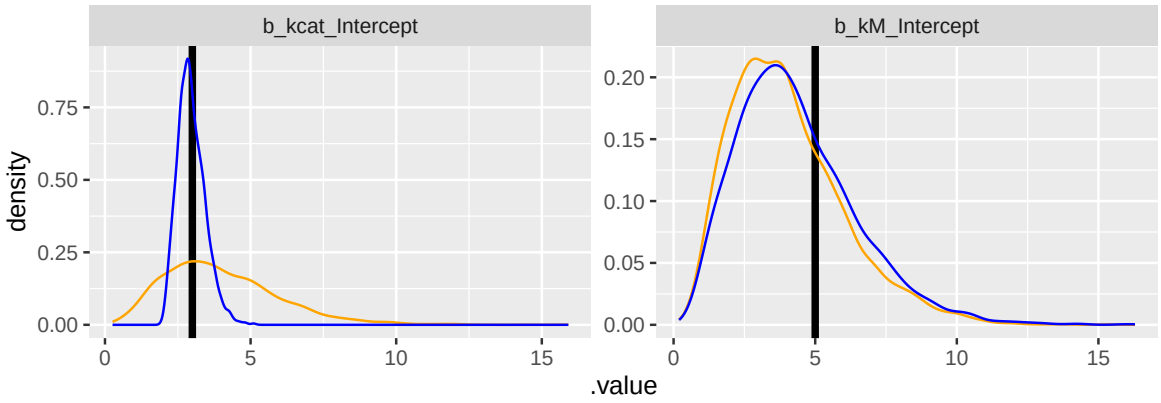
To visualize the posterior distribution vs. the prior distribution, we first sample from the prior, using the same `brms::brm` call with `sample_prior = "only"` the argument.

```
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.

14. Fitting multiple observations

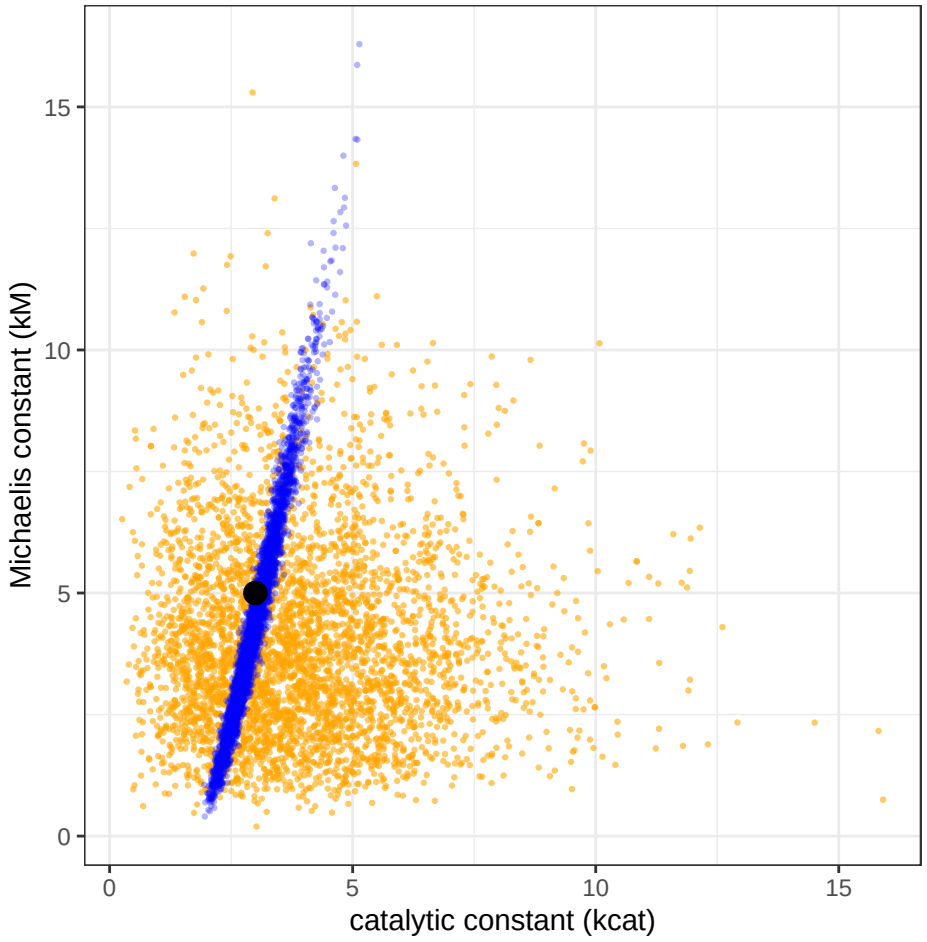


Next we will extend the BRMS model to allow fitting common `kcat`, `kM` concentrations based on multiple replicas, or varying substrate/enzyme concentrations using BRMS. To demonstrate, we varying the enzyme and substrate concentrations, to better fit the kinematic parameters.

```
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.

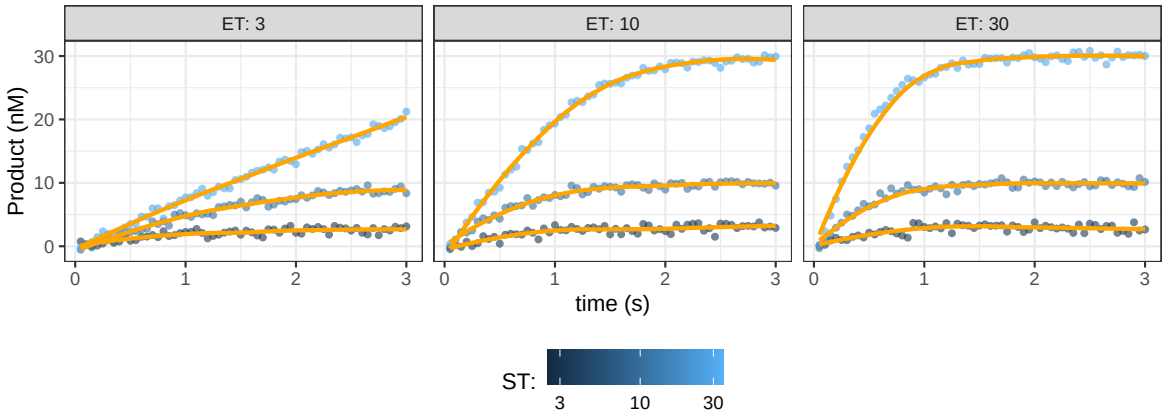
```
stanvars_tQ_multiple <- brms::stanvar(scode = paste("
vector tQ_multiple(
  array[] int replica,
  data vector time,
  vector vkcat,
  vector vkM,
  data vector vET,
  data vector vST) {

  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)
```

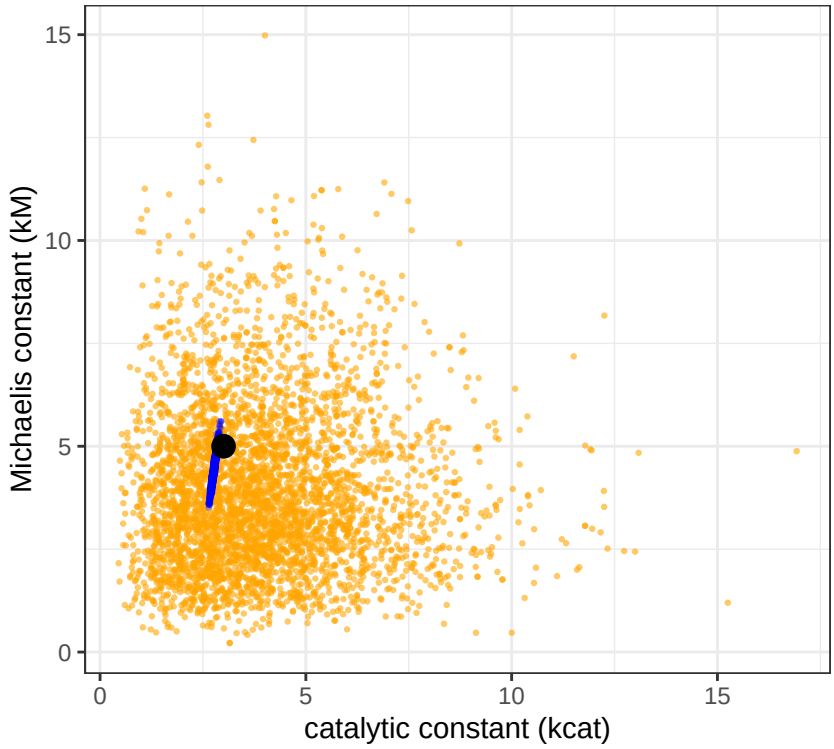
Next we will sample enzyme progress curves from the posterior

15. 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.

References

Bliss CI (1956). "The calculation of microbial assays." *Bacteriol. Rev.*, **20**(4), 243–258. ISSN 0005-3678. doi:10.1128/br.20.4.243-258.1956.



Bürkner PC (2017). "brms: An R Package for Bayesian Multilevel Models Using Stan." *J. Stat. Softw.*, **80**(1), 1–28. ISSN 1548-7660, 1548-7660. doi:10.18637/jss.v080.i01.

Carpenter B, Gelman A, Hoffman MD, Lee D, Goodrich B, Betancourt M, Brubaker MA, Guo J, Li P, Riddell A (2017). "Stan: A Probabilistic Programming Language." *J. Stat. Softw.*, **76**. ISSN 1548-7660. doi:10.18637/jss.v076.i01.

Choi B, Rempala GA, Kim JK (2017). "Beyond the Michaelis-Menten equation: Accurate and efficient estimation of enzyme kinetic parameters." *Sci. Rep.*, **7**(1), 17018. ISSN 2045-2322. doi:10.1038/s41598-017-17072-z.

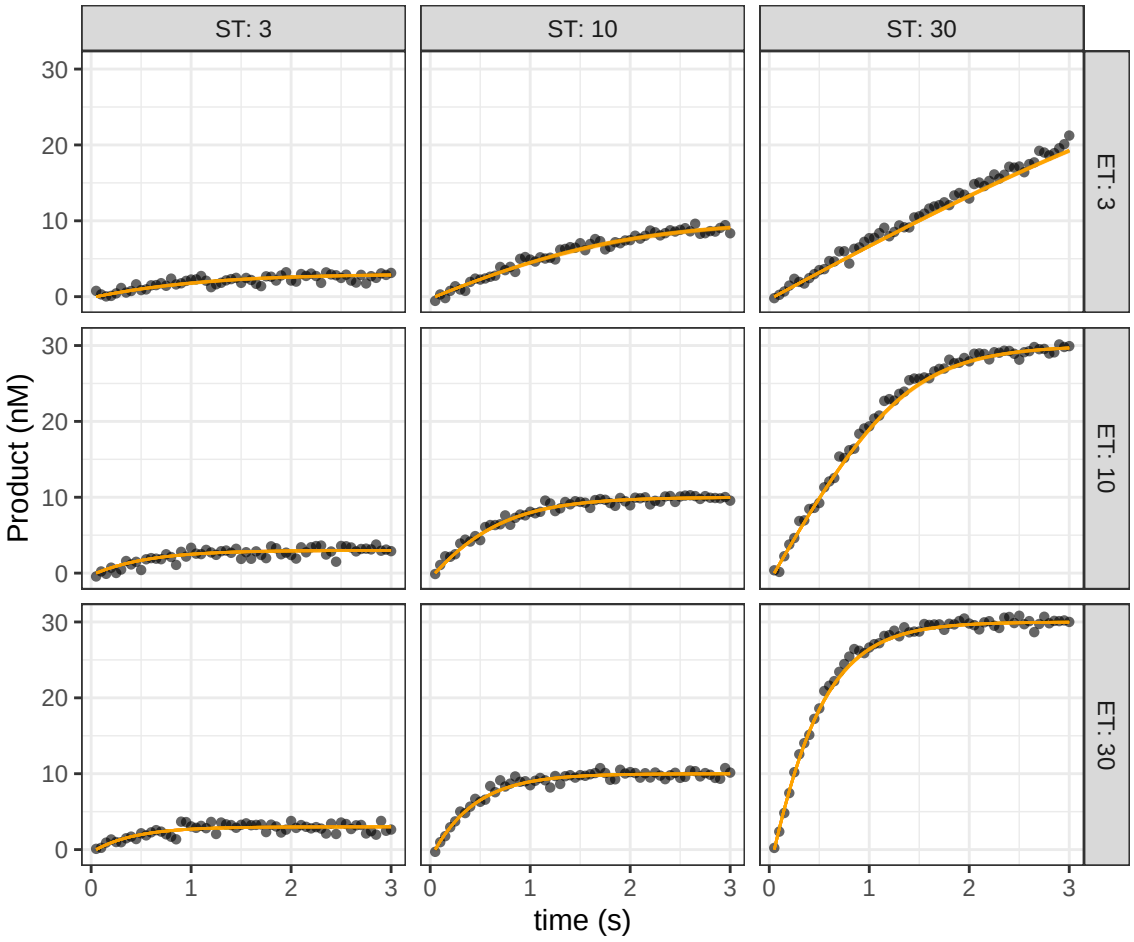
Gabry J, Mahr T (2017). "bayesplot: Plotting for Bayesian models." *R package version*, **1**(0).

Gelman A, Vehtari A, Simpson D, Margossian CC, Carpenter B, Yao Y, Kennedy L, Gabry J, Bürkner PC, Modrák M (2020). "Bayesian Workflow." 2011.01808.

Ianevski A, Giri AK, Aittokallio T (2022). "SynergyFinder 3.0: an interactive analysis and consensus interpretation of multi-drug synergies across multiple samples." doi:10.1093/nar/gkac382.

Kay M (2018). "tidybayes: Tidy data and geoms for Bayesian models." *R package version*, **1**(3).

Loewe S (1926). "Effect of combinations: mathematical basis of problem." *Arch. Exp. Pathol. Pharmakol.*, **114**, 313–326.



Margolis EB, Wallace TL, Van Orden LJ, Martin WJ (2020). "Differential effects of novel kappa opioid receptor antagonists on dopamine neurons using acute brain slice electrophysiology." *PLoS One*, **15**(12), e0232864. ISSN 1932-6203. doi:10.1371/journal.pone.0232864.

Meyer CT, Wooten DJ, Paudel BB, Bauer J, Hardeman KN, Westover D, Lovly CM, Harris LA, Tyson DR, Quaranta V (2019). "Quantifying Drug Combination Synergy along Potency and Efficacy Axes." *Cell Syst*, **8**(2), 97–108.e16. ISSN 2405-4712. doi:10.1016/j.cels.2019.01.003.

Team RC, Others (2013). "R: A language and environment for statistical computing."

Van de Schoot R, Veen D, Smeets L, Winter SD, Depaoli S (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*, pp. 30–49.

Vehtari A, Gelman A, Gabry J (2017). "Erratum to: Practical Bayesian model evaluation using leave-one-out cross-validation and WAIC." *Stat. Comput.*, **27**(5), 1433–1433. ISSN 0960-3174, 1573-1375. doi:10.1007/s11222-016-9709-3.

Wickham H (2009). *Ggplot2: Elegant graphics for data analysis*. Springer, New York, NY. ISBN 9781282509917.

Wickham H, Averick M, Bryan J, Chang W, McGowan L, François R, Grolemund G, Hayes A, Henry L, Hester J, Kuhn M, Pedersen T, Miller E, Bache S, Müller K, Ooms J, Robinson D, Seidel D, Spinu V, Takahashi K, Vaughan D, Wilke C, Woo K, Yutani H (2019). "Welcome to the tidyverse." *J. Open Source Softw.*, **4**(43), 1686. ISSN 2475-9066. doi:10.21105/joss.01686.

Wooten DJ, Albert R (2021). "synergy: a Python library for calculating, analyzing and visualizing drug combination synergy." *Bioinformatics*, **37**(10), 1473–1474. ISSN 1367-4803, 1367-4811. doi:10.1093/bioinformatics/btaa826.

Wooten DJ, Meyer CT, Lubbock ALR, Quaranta V, Lopez CF (2021). "MuSyC is a consensus framework that unifies multi-drug synergy metrics for combinatorial drug discovery." *Nat. Commun.*, **12**(1), 4607. ISSN 2041-1723. doi:10.1038/s41467-021-24789-z.

Affiliation:

Madeline J. Martin
Knight Campus Center for Accelerating Scientific Impact
Eugene USA
E-mail: martin.mjm105@gmail.com

Elayne Vieira Diaz
UCSF Weill Institute for Neurosciences, Department of Neurology
San Francisco USA
E-mail: Elayne.Vieiradias@ucsf.edu

P. Walter German
UCSF Weill Institute for Neurosciences, Department of Neurology
San Francisco USA
E-mail: p.walter.german@gmail.com

Elyssa B. Margolis
UCSF Weill Institute for Neurosciences, Department of Neurology
San Francisco USA
E-mail: elyssa.margolis@ucsf.edu

Matthew J. O'Meara
Department of Computational Medicine and Bioinformatics
Ann Arbor USA
E-mail: maom@umich.edu

Journal of Statistical Software http://www.jstatsoft.org/
published by the Foundation for Open Access Statistics http://www.foastat.org/
MMMMMM YYYY, Volume VV, Issue II *Submitted:* yyyy-mm-dd
doi:10.18637/jss.v000.i00 *Accepted:* yyyy-mm-dd