Towards a principled statistical workflow for the study of wildlife and environmental contaminants in the presence of method detection and quantification limits

Allen Bush-Beaupré Marc Bélisle

Source: Article Notebook

0.1 Introduction

Human industrial and agricultural intensification has resulted in the release of an astronomical diversity and quantity of contaminants in the environment. The impact of these contaminants, such as mercury and pesticides used in agriculture has been the subject of many discussions in the scientific and public sphere. In her seminal book, Rachel Carson BLABLA. Much effort has been invested in quantifying the impact of such contaminants on various life traits in controlled conditions (eg. BLABLA). Studies such as these are a critical first step in determining contaminant concentration thresholds at which we may observe the detrimental impact of human industrial and agricultural activities. However, the results of these laboratory and semi-field studies can be difficult to fully extrapolate to the real-life environment conditions to which organisms are exposed. Organisms are exposed to a diversity of contaminants depending on their environmental conditions. Additionally, we would expect an increased level of contamination in organisms at higher trophic levels due to bioaccumulation.

As emphasized in a review by Tison et al. (2024), the mechanisms and consequences by which pesticides accumulate across trophic levels are understudied. The initial step in studying such phenomena is a proper quantification of the chemicals in collected samples. Contamination concentrations are obtained through a variety of laboratory methods. These methods, however, have varying degrees of precision especially when concentrations are low. Thus arises the limit of detection (LOD), the concentration threshold under which the analytical method can not accurately determine if the contaminant is present in the sample or not. Additionally, many analytical methods commonly used have a limit of quantification (LOQ) such that concentrations between the LOD and LOQ are known to be present, but the actual concentration is only known to be within the aforementioned interval. As will be exemplified here, much effort has been made to deal with these complicated data to generate meaningful statistics for over 50 years (ie Leese, 1973). Methods have been proposed, validated and their performance compared by multiple authors (ie Ganser & Hewett, 2010; Huynh et al. 2014, 2015; Busschaert et al. 2011; Gilliom, 1986; She, 1997; Hornung & Reed, 1990; Antweiler & Taylor, 2008; Hewett & Ganser, 2007; Shoari 2016; Sinha, 2006; Succop et al. 2004; Vlachonikolis & Marriott, 1995;). However, practitioners in the field of ecology often lack the statistical training to choose and implement an appropriate method for their specific use cases. Authors often refer to heuristics proposed by others through simulation studies without considering the specific conditions involved in those simulations. We propose here a general data analysis workflow that may be adapted by practitioners for their data at hand using the most recent advances in statistical software and methods. Such workflow will encourage researchers to tailor their statistical models to the specific characteristics of their collected data and make for easier contrasts among

results obtained across various studies.

The accuracy and relevance of the results obtained from modeling efforts rely on statistical models based on the data-generating process. Such endeavor implies profound thinking about the characteristics of the data collected. An example of this in regression models is the use of likelihood distributions mathematically sound for the data at hand. The lognormal distribution is often recommended for concentration data as BLABLA REFS. In the case of contaminants found in wildlife samples, however, for a concentration level to be known, the contaminant must be present in the sample in the first place. Thus arises a need for quantifying the probability of contamination along with the concentration level if present. Generalized Linear Model (GLM) likelihoods that may account for this process are part of the hurdle or zero-inflated class of models. Such models are used, for example, in occupancy and distribution models (refs). EXAMPLE REF. An additional aspect of the data-generating process is the limitations imposed by the methods for detection and quantification of contaminants (the previously mentioned LOD and LOQ). A proper modeling of the data-generating process mentioned here may at first glance appear complex and tedious. The next sections will focus on elaborating the formulation of a statistical model incorporating the entirety of these specific data characteristics with reference to approaches taken by researchers in recent publications (2020-2024) to deal with contaminant presence (often termed exposure) and concentrations in the presence of LODs and LOQs.

0.1.1 Data-generating process: limits of detection and quantification

Concentration values under LOD and between LOD and LOQ are considered as censored observations. Censoring in the context of data analysis refers to inferring BLABLA. These methods are often used in survival analysis where, for example, we attempt to estimate the survival time of an individual in a population, but we don't know exactly when death happened. Such cases arise when we observe individuals over time but do not get the opportunity to confirm its time of death due to a variety of reasons such as it leaving the study site, or the study period did not last long enough to observe the individual's death. Such data are said to be right-censored as we do not observe certain values on the right-side of the distribution of survival times. Although we did not observe the time of death, we know the individual survived at least until the last time we recorded it. ACTUAL EXAMPLE.

There are two other cases of data censoring. Left-censoring occurs when we do not observe values past the left side of a given threshold of a distribution. This may occur, to return to the previous survival time example, when we do not know the exact age of an individual within a population and the start of the study. We do, however, have methods to approximate its maximal age and thus know that the age of the individual must be below a specific threshold. Lastly, interval-censoring arises when, for example, an event occurs (such as death) between two sampling occasions. We thus know that death occurred between two values of time but are uncertain as to exactly when.

Multiple statistical methods have been developed and used for such data, with varying levels of impact on the estimation of parameters (such as the mean). Shaori & Dubé (2018) reviewed various frequentist and Bayesian methods for this statistical challenge. Readers are referred to the aforementioned review for a detailed overview. We focus here mainly on common practices still observed in recent years by wildlife and environmental practitioners and build up the reader's intuition for understanding such data with the objective to fit regression models evaluating the impact of certain predictors on contaminant presence and concentration.

0.1.2 Ignoring values under LOD and/or LOQ

One way to work with data under the LOD and/or LOQ is to make the assumption that these small values are not biologically relevant for the context at hand. For example, Albert et

al. (2021) measured levels of Mercury in Arctic seabirds and quantified concentration levels several orders of magnitude higher than the LOD. In this case, the authors' assumptions may be justified as the number of samples with concentration values below the LOD would be unlikely to severely bias the estimation of concentration levels. There are many contexts, however, where such assumptions may be detrimental to the precision of the reported estimated concentrations. For example, when concentration values are expected to be low such as BLABLA (refs). Additionally, authors have previously highlighted the bias induced by such methods (Hashimoto & Trussel, 1983). While certain studies can justify the choice of data modifications using toxicological studies in laboratory settings with known contamination levels and thus use calculations based on samples exceeding a certain threshold concentration (eg. Odemer et al. 2023), this does not take into account the half life of these contaminants which can be low in certain cases (such as neonicotinoids; refs). Additionally, ignoring lower concentration levels does not aid in eventually quantifying the effects of these contaminants on the survival and health parameters of the studied organisms in the wild. Nonetheless, ignoring detection and/or quantification limits of contamination levels in wildlife studies remains a common occurrence in recent publications (Spadetto et al. 2024a, 2024b; Esther et al. 2022; Rial-Berriel et al. 2021; Martin-Cruz et al. 2024; Fuentes et al. 2023, 2024a, 2024b, 2024c; Bariod et al. 2024; Tison et al. 2023; Lennon et al. 2020b; Lesch et al. 2024; Badry et al. 2021; Elliot et al. 2024; Drummond et al. 2024).

0.1.3 Data substitution or imputation

Some practitioners recognize the need to incorporate the information contained by the samples with concentrations below the LOD or LOQ and use a variety of methods to substitute values for these points. In recent years, researchers have substituted observations below either limit to the median between zero and the limit (Bishops et al. 2022; Elliot et al. 2022; Perkins et al. 2021), ignored data under the LOD and set values between LOD and LOQ to LOD (Rondeau et al. 2022) and fixed values below LOD to LOD (Lennon et al. 2020, Ito et al. 2020). Other practitioners have used various data imputation methods such as generating random values between zero and the LOD and between the LOD and LOQ (Martin-Cruz et al. 2024b). Bias produced by similar practices have been highlighted in previous publications (Hashimoto & Trussel, 1983). Another imputation method used is the β-substitution method (Ganser and Hewett, 2010), used, for example, by Eng et al. (2020).

0.1.4 MLE

Shoari 2016 quantify uncertainty with bootstrap

There exists, however an alternative method to work with censored data in the Bayesian paradigm. Bayesian models treat uncertainty in a different manner than Frequentist ones and offer probabilistic interpretations of parameter values. In a Bayesian paradigm, censored observations are simply integrated over from the entire data distribution. This allows the uncertainty of the unobserved values to be propagated across the entire model and reflected in the width of confidence intervals generated. When comparing the β-substitution method to Bayesian left-censoring AUTHORS found the two methods to perform similarly at low levels of censoring. However, as the number of censored observations increased, Bayesian left-censoring had a higher predictive performance. Thus, Bayesian left-censoring allows to model censored data in a greater variety of cases. Researchers can thus use this method rather than not model concentrations at all when the level of censored observations is high such as in the case with METABOLITE in Eng et al. (2020). The method has been used in recent publications such as Kerric et al. (2022), MORE.

0.1.5 Data-generating process: Contaminant presence

A measure that is often of interest is the probability of exposure that an environment or organism is subjected to. For example, researchers often discuss and analyse how the number of pesticides that an individual is exposed to affects certain life traits. Anderson et al. (2023) dichotomized samples above and below the LOQ as a measure of exposure probability. Due to a lower LOQ for Imidacloprid than the other pesticides tested, their method of interpretation leads to claiming that individuals had a higher exposure to Imidacloprid. When analysing a 30-year dataset of rodenticide contamination in raptors, Elliot et al. (2022) dealt with detection limits that ranged across several orders of magnitude by considering only the highest limit and thus quantifying "exposure" probability in relation to this value. However, discussing exposure in the aforementioned manner is erroneous; these measures relate to inherent imprecisions in the analytical method rather than the actual data-generating process. Nevertheless, such terminology is pervasive in recent literature (Rondeau et al. 2022; Fuentes et al. 2024a, 2024c; Rial-Berriel et al. 2021; Badry et al. 2021; Bischoff et al. 2023; Martinez et al. 2021; Rodriguez-Aguilar et al. 2022; Roy & Chen, 2023; Tison et al. 2023).

0.1.6 Proposed general statistical framework

Thus far, we have highlighted four major characteristics of contamination concentration data as illustrated through a simulated data distribution in Figure 1. First, there are data that were properly quantified and are thus observed (ie non-censored; purple part of distribution). These observations describe the right tail of the distribution unambiguously. Second, we have observations that lie between the LOD and LOQ (ie interval-censored; gold part of distribution). Such data are ambiguous concerning their exact values but contain valuable information; we know what portion of the distribution within which they reside. Lastly, we have values below the LOD (left-censored; red part of distribution) which can be further characterised into zero-values and values between zero and the LOD. From the proportion of values located at zero, we may derive an exposure probability and, from all values above zero, we may derive an average concentration when the contaminant is present. Furthermore, one may be inclined to calculate a "overall" average thus incorporating the concentration when present and the contamination probability which can be obtained by multiplying the two metrics. Although it may appear counter-intuitive at first, we show here that it is possible to obtain these metrics from data with such characteristics in a variety of conditions. We emphasise, however, that practitioners evaluate the adaptability of the proposed method to their specific use-case following the workflow provided.

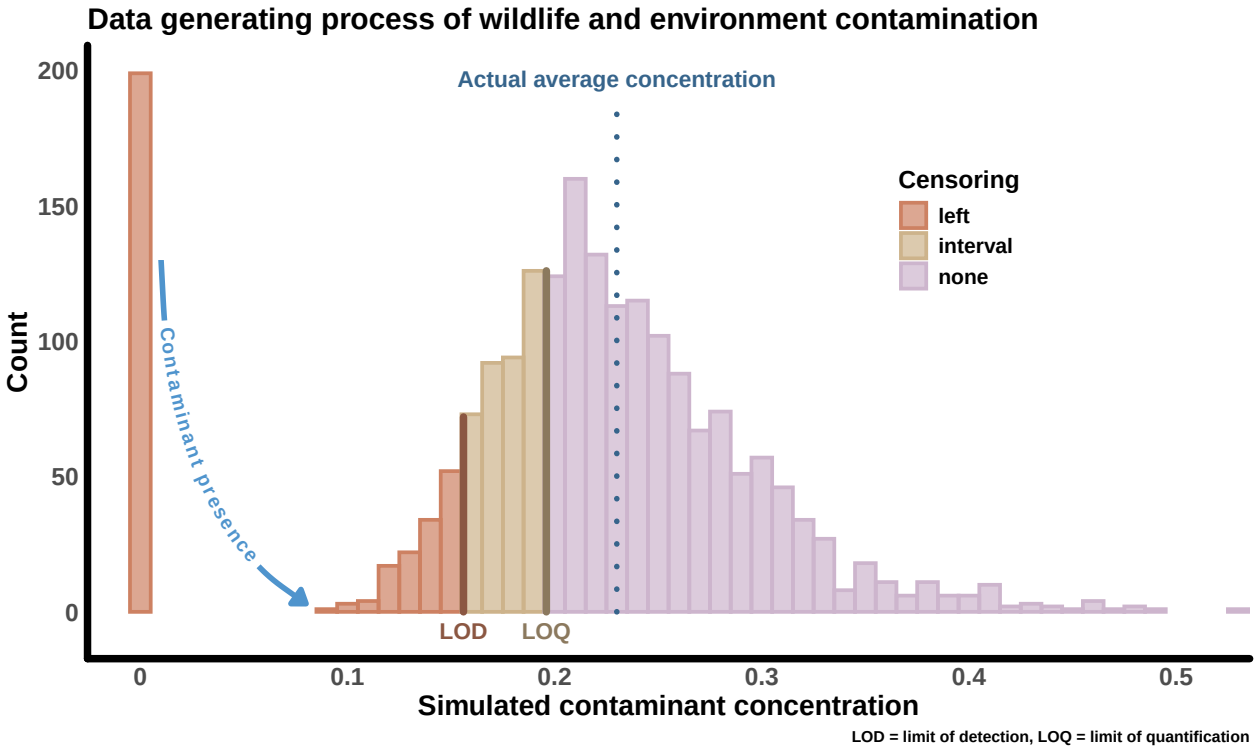

Figure 1: Data generating process of wildlife and environment sample contamination. The plot shows the distribution of simulated contaminant concentrations with different censoring regions highlighted.

Source: Article Notebook

Many of the aforementioned studies focus solely on the number of samples with detected levels of contaminants (refs). Treating the data in such a manner is problematic as previously emphasized. Others assume a lognormal distribution and may use any of the mentioned methods to deal (or not) with the censored data (refs). However, although the lognormal is theoretically sound for concentration data, it does not allow for zero-values. Thus, while such models may infer infinitely small concentrations, an important aspect of the data generating process is neglected; the probability that the contaminant is present in the sample in the first place. Model specifications to include the zero-generating process such as the hurdle lognormal are available and easily implemented using software such as brms. As will be shown using simulations and a case study of pesticide contamination in Tree Swallow boluses, a left- and interval censored hurdle lognormal bayesian hierarchical generalized linear model can be fit to such data to obtain contamination probability and concentration with acceptable levels of precision. As exemplified by the case study, the model can be complexified to include multiple pesticides along with correlations in their contamination probability and concentrations, measures of 'cocktail' contaminations.

Setting up a GLMM with censoring is not trivial and requires a deep understanding of the data-generating process. The next sections will focus on the formulation of the model and the interpretation of the results obtained. We will also provide a step-by-step guide to implement the model using the brms package in R.

0.2 Data simulation

To illustrate the proposed statistical workflow, we will simulate a dataset of contaminant concentrations in wildlife samples. The data will be generated to mimic the characteristics of real-world data, including left-censored observations below the limit of detection (LOD) and interval-censored observations between the LOD and the limit of quantification (LOQ). The data will be generated from a lognormal distribution, which is commonly used to model concentration data. The simulation will also include a binary variable indicating the presence or absence of the contaminant in each sample. The data distribution generated can be observed in Figure 1.

```
n <- 2000       # Number of samples
LOD <- 0.15     # Limit of detection
LOQ <- 0.19     # Limit of quantification
p_zeros <- 0.1  # Proportion of zero values
# Generate simulated data
censored_data <- tibble(
  # Generate lognormal concentrations
  conc = round(rlnorm(n, -1.5, 0.25),2),
  # Presence probability of 0.9
  zeros = rbinom(n, prob = (1 - p_zeros), size = 1),
  # Final concentrations (including zeros)
  Y = conc * zeros)
```

Source: Article Notebook

To analyse censored data, {brms} requires some pre-processing to recognize the censored observations. Values below LOD (left-censored) must be set to LOD while values between LOD and LOQ (interval-censored) must be set to the lower bound of the interval (LOD). Interval-censoring also requires additional column must be created to specify the upper bound of the interval censoring (LOQ) while data that are not censored are left as is. Finally, a column specifying the type of censoring (left, interval, none) must be created. The code below performs these operations on the simulated data.

```
censored_data |>
  mutate(
    # format data
    y_cen = case_when(
    # Set values below LOD to LOD
      Y < LOD ~ LOD,
    # Set values between LOD and LOQ to LOD
      Y >= LOD & Y < LOQ ~ LOD,
    # Leave other values as is
      TRUE ~ Y),
    # Create column for upper bound of interval censoring
    upper_int_cens = ifelse(Y >= LOD & Y < LOQ, LOQ, Y),
    # Create column for censoring type
    censoring = case_when(
    # Left-censored
      Y < LOD ~ "left",
    # Interval-censored
      Y >= LOD & Y < LOQ ~ "interval",
    # Fully observed
      TRUE ~ "none")
) -> censored_data
```

Source: Article Notebook

Y	y_cen	upper_int_cens	censoring
0.18	0.15	0.19	interval
0.15	0.15	0.19	interval
0.19	0.19	0.19	none
0.28	0.28	0.28	none
0.00	0.15	0.00	left
0.13	0.15	0.13	left

0.3 Model specification with {brms}

Next, we specifiy data censoring in the model equation with bf(censored observations | cens(censoring type, upper bound of interval censoring)). In bf_intercept below, we specify an intercept-only model for both the concentration and zero-probability with a hurdle-lognormal response distribution with identity link for the A parameter, log link for the B parameter and logit link for the zero-probability.

```
library(brms)       # Bayesian GLMMs
```

```
Loading required package: Rcpp
```

```
Loading 'brms' package (version 2.21.0). Useful instructions
can be found by typing help('brms'). A more detailed introduction
to the package is available through vignette('brms_overview').


Attaching package: 'brms'
```

```
The following object is masked from 'package:stats':

    ar
```

```
library(cmdstanr)   # cmdstanr backend for brms
```

```
This is cmdstanr version 0.8.1.9000
```

```
- CmdStanR documentation and vignettes: mc-stan.org/cmdstanr
```

```
- CmdStan path: /home/ubuntu/.cmdstan/cmdstan-2.35.0
```

```
- CmdStan version: 2.35.0
```

```
library(tidybayes)  # Extract and visualize priors/posteriors
```

```
Attaching package: 'tidybayes'
```

```
The following objects are masked from 'package:brms':

    dstudent_t, pstudent_t, qstudent_t, rstudent_t
```

```
bf_intercept <- bf(y_cen | cens(censoring , upper_int_cens) ~ 1,
                                hu ~ 1,
                           family = hurdle_lognormal(link = "identity", link_sigma = "log"
)

get_prior(bf_intercept, data = censored_data)
```

```
               prior     class coef group resp dpar nlpar lb ub  source
```

student_t(3, -1.5, 2.5) Intercept default student_t(3, 0, 2.5) sigma 0 default logistic(0, 1) Intercept hu default

```
priors_intercept <- c(
  prior(normal(0,2), class = "Intercept", dpar = "hu"),
  prior(normal(0, 2.5), class = "Intercept"),
  prior(exponential(0.5), class = "sigma")
)

prior_sim_intercept <- brm(formula = bf_intercept,
                 data = censored_data,
                 prior = priors_intercept,
                 sample_prior = "only",
                 iter = 2000, warmup = 1000, chains = 4, cores = 4,
                 threads = threading(4, grainsize = 100),
                 backend = "cmdstanr")
```

```
Start sampling
```

Running MCMC with 4 parallel chains, with 4 thread(s) per chain...

Chain 1 Iteration: 1 / 2000 [0%] (Warmup) Chain 1 Iteration: 100 / 2000 [5%] (Warmup) Chain 1 Iteration: 200 / 2000 [10%] (Warmup) Chain 1 Iteration: 300 / 2000 [15%] (Warmup) Chain 1 Iteration: 400 / 2000 [20%] (Warmup) Chain 1 Iteration: 500 / 2000 [25%] (Warmup) Chain 1 Iteration: 600 / 2000 [30%] (Warmup) Chain 1 Iteration: 700 / 2000 [35%] (Warmup) Chain 1 Iteration: 800 / 2000 [40%] (Warmup) Chain 1 Iteration: 900 / 2000 [45%] (Warmup) Chain 1 Iteration: 1000 / 2000 [50%] (Warmup) Chain 1 Iteration: 1001 / 2000 [50%] (Sampling) Chain 1 Iteration: 1100 / 2000 [55%] (Sampling) Chain 1 Iteration: 1200 / 2000 [60%] (Sampling) Chain 1 Iteration: 1300 / 2000 [65%] (Sampling) Chain 1 Iteration: 1400 / 2000 [70%] (Sampling) Chain 1 Iteration: 1500 / 2000 [75%] (Sampling) Chain 1 Iteration: 1600 / 2000 [80%] (Sampling) Chain 1 Iteration: 1700 / 2000 [85%] (Sampling) Chain 1 Iteration: 1800 / 2000 [90%] (Sampling) Chain 1 Iteration: 1900 / 2000 [95%] (Sampling) Chain 1 Iteration: 2000 / 2000 [100%] (Sampling) Chain 2 Iteration: 1 / 2000 [0%] (Warmup) Chain 2 Iteration: 100 / 2000 [5%] (Warmup) Chain 2 Iteration: 200 / 2000 [10%] (Warmup) Chain 2 Iteration: 300 / 2000 [15%] (Warmup) Chain 2 Iteration: 400 / 2000 [20%] (Warmup) Chain 2 Iteration: 500 / 2000 [25%] (Warmup) Chain 2 Iteration: 600 / 2000 [30%] (Warmup) Chain 2 Iteration: 700 / 2000 [35%] (Warmup) Chain 2 Iteration: 800 / 2000 [40%] (Warmup) Chain 2 Iteration: 900 / 2000 [45%] (Warmup) Chain 2 Iteration: 1000 / 2000 [50%] (Warmup) Chain 2 Iteration: 1001 / 2000 [50%] (Sampling) Chain 2 Iteration: 1100 / 2000 [55%] (Sampling) Chain 2 Iteration: 1200 / 2000 [60%] (Sampling) Chain 2 Iteration: 1300 / 2000 [65%] (Sampling) Chain 2 Iteration: 1400 / 2000 [70%] (Sampling) Chain 2 Iteration: 1500 / 2000 [75%] (Sampling) Chain 2 Iteration: 1600 / 2000 [80%] (Sampling) Chain 2 Iteration: 1700 / 2000 [85%] (Sampling) Chain 2 Iteration: 1800 / 2000 [90%] (Sampling) Chain 2 Iteration: 1900 / 2000 [95%] (Sampling) Chain 2 Iteration: 2000 / 2000 [100%] (Sampling) Chain 3 Iteration: 1 / 2000 [0%] (Warmup) Chain 3 Iteration: 100 / 2000 [5%] (Warmup) Chain 3 Iteration: 200 / 2000 [10%] (Warmup) Chain 3 Iteration: 300 / 2000 [15%] (Warmup) Chain 3 Iteration: 400 / 2000 [20%] (Warmup) Chain 3 Iteration: 500 / 2000 [25%] (Warmup) Chain 3 Iteration: 600 / 2000 [30%] (Warmup) Chain 3 Iteration: 700 / 2000 [35%] (Warmup) Chain 3 Iteration: 800 / 2000 [40%] (Warmup) Chain 3 Iteration: 900 / 2000 [45%] (Warmup) Chain 3 Iteration: 1000 / 2000 [50%] (Warmup) Chain 3 Iteration: 1001 / 2000 [50%] (Sampling) Chain 3 Iteration: 1100 / 2000 [55%] (Sampling) Chain 3 Iteration: 1200 / 2000 [60%] (Sampling) Chain 3 Iteration: 1300 / 2000 [65%] (Sampling) Chain 3 Iteration: 1400 / 2000 [70%] (Sampling) Chain 3 Iteration: 1500 / 2000 [75%] (Sampling) Chain 3 Iteration: 1600 / 2000 [80%] (Sampling) Chain 3 Iteration: 1700 / 2000 [85%] (Sampling) Chain 3 Iteration: 1800 / 2000 [90%] (Sampling) Chain

3 Iteration: 1900 / 2000 [95%] (Sampling) Chain 3 Iteration: 2000 / 2000 [100%] (Sampling) Chain 4 Iteration: 1 / 2000 [0%] (Warmup) Chain 4 Iteration: 100 / 2000 [5%] (Warmup) Chain 4 Iteration: 200 / 2000 [10%] (Warmup) Chain 4 Iteration: 300 / 2000 [15%] (Warmup) Chain 4 Iteration: 400 / 2000 [20%] (Warmup) Chain 4 Iteration: 500 / 2000 [25%] (Warmup) Chain 4 Iteration: 600 / 2000 [30%] (Warmup) Chain 4 Iteration: 700 / 2000 [35%] (Warmup) Chain 4 Iteration: 800 / 2000 [40%] (Warmup) Chain 4 Iteration: 900 / 2000 [45%] (Warmup) Chain 4 Iteration: 1000 / 2000 [50%] (Warmup) Chain 4 Iteration: 1001 / 2000 [50%] (Sampling) Chain 4 Iteration: 1100 / 2000 [55%] (Sampling) Chain 4 Iteration: 1200 / 2000 [60%] (Sampling) Chain 4 Iteration: 1300 / 2000 [65%] (Sampling) Chain 4 Iteration: 1400 / 2000 [70%] (Sampling) Chain 4 Iteration: 1500 / 2000 [75%] (Sampling) Chain 4 Iteration: 1600 / 2000 [80%] (Sampling) Chain 4 Iteration: 1700 / 2000 [85%] (Sampling) Chain 4 Iteration: 1800 / 2000 [90%] (Sampling) Chain 4 Iteration: 1900 / 2000 [95%] (Sampling) Chain 4 Iteration: 2000 / 2000 [100%] (Sampling) Chain 1 finished in 0.0 seconds. Chain 2 finished in 0.0 seconds. Chain 3 finished in 0.0 seconds. Chain 4 finished in 0.0 seconds.

All 4 chains finished successfully. Mean chain execution time: 0.0 seconds. Total execution time: 0.4 seconds.

```
epred_rvars(prior_sim_intercept,
            dpar = TRUE,
            newdata = tibble(.rows = 1))
```

1 A tibble: 1 x 4

.epred	mu	hu	sigma
<rvar[1d]>	<rvar[1d]>	<rvar[1d]>	<rvar[1d]>
1 1.9e+35 ± 1.2e+37	-0.015 ± 2.5	0.51 ± 0.31	2 ± 1.9

```
mod_intercept <- brm(formula = bf_intercept,
                     data = censored_data,
                     prior = priors_intercept,
                     sample_prior = "yes",
                     iter = 2000, warmup = 1000, chains = 4, cores = 4,
                     threads = threading(4, grainsize = 100),
                     backend = "cmdstanr")
```

```
Start sampling
```

Running MCMC with 4 parallel chains, with 4 thread(s) per chain...

Chain 1 Iteration: 1 / 2000 [0%] (Warmup) Chain 2 Iteration: 1 / 2000 [0%] (Warmup) Chain 3 Iteration: 1 / 2000 [0%] (Warmup) Chain 4 Iteration: 1 / 2000 [0%] (Warmup) Chain 1 Iteration: 100 / 2000 [5%] (Warmup) Chain 2 Iteration: 100 / 2000 [5%] (Warmup) Chain 1 Iteration: 200 / 2000 [10%] (Warmup) Chain 3 Iteration: 100 / 2000 [5%] (Warmup) Chain 1 Iteration: 300 / 2000 [15%] (Warmup) Chain 2 Iteration: 200 / 2000 [10%] (Warmup) Chain 3 Iteration: 200 / 2000 [10%] (Warmup) Chain 1 Iteration: 400 / 2000 [20%] (Warmup) Chain 2 Iteration: 300 / 2000 [15%] (Warmup) Chain 1 Iteration: 500 / 2000 [25%] (Warmup) Chain 2 Iteration: 400 / 2000 [20%] (Warmup) Chain 3 Iteration: 300 / 2000 [15%] (Warmup) Chain 4 Iteration: 100 / 2000 [5%] (Warmup) Chain 1 Iteration: 600 / 2000 [30%] (Warmup) Chain 2 Iteration: 500 / 2000 [25%] (Warmup) Chain 3 Iteration: 400 / 2000 [20%] (Warmup) Chain 2 Iteration: 600 / 2000 [30%] (Warmup) Chain 2 Iteration: 700 / 2000 [35%] (Warmup) Chain 3 Iteration: 500 / 2000 [25%] (Warmup) Chain 1 Iteration: 700 / 2000 [35%] (Warmup) Chain 2 Iteration: 800 / 2000 [40%] (Warmup) Chain 3 Iteration: 600 / 2000 [30%] (Warmup) Chain 4 Iteration: 200 / 2000 [10%] (Warmup) Chain 2 Iteration: 900 / 2000 [45%] (Warmup) Chain 3 Iteration: 700 / 2000 [35%] (Warmup) Chain 1 Iteration: 800 / 2000 [40%] (Warmup) Chain 2 Iteration: 1000

/ 2000 [50%] (Warmup) Chain 2 Iteration: 1001 / 2000 [50%] (Sampling) Chain 3 Iteration: 800 / 2000 [40%] (Warmup) Chain 4 Iteration: 300 / 2000 [15%] (Warmup) Chain 2 Iteration: 1100 / 2000 [55%] (Sampling) Chain 3 Iteration: 900 / 2000 [45%] (Warmup) Chain 1 Iteration: 900 / 2000 [45%] (Warmup) Chain 2 Iteration: 1200 / 2000 [60%] (Sampling) Chain 3 Iteration: 1000 / 2000 [50%] (Warmup) Chain 3 Iteration: 1001 / 2000 [50%] (Sampling) Chain 4 Iteration: 400 / 2000 [20%] (Warmup) Chain 1 Iteration: 1000 / 2000 [50%] (Warmup) Chain 1 Iteration: 1001 / 2000 [50%] (Sampling) Chain 2 Iteration: 1300 / 2000 [65%] (Sampling) Chain 4 Iteration: 500 / 2000 [25%] (Warmup) Chain 2 Iteration: 1400 / 2000 [70%] (Sampling) Chain 4 Iteration: 600 / 2000 [30%] (Warmup) Chain 2 Iteration: 1500 / 2000 [75%] (Sampling) Chain 3 Iteration: 1100 / 2000 [55%] (Sampling) Chain 4 Iteration: 700 / 2000 [35%] (Warmup) Chain 1 Iteration: 1100 / 2000 [55%] (Sampling) Chain 2 Iteration: 1600 / 2000 [80%] (Sampling) Chain 4 Iteration: 800 / 2000 [40%] (Warmup) Chain 4 Iteration: 900 / 2000 [45%] (Warmup) Chain 2 Iteration: 1700 / 2000 [85%] (Sampling) Chain 3 Iteration: 1200 / 2000 [60%] (Sampling) Chain 4 Iteration: 1000 / 2000 [50%] (Warmup) Chain 4 Iteration: 1001 / 2000 [50%] (Sampling) Chain 1 Iteration: 1200 / 2000 [60%] (Sampling) Chain 2 Iteration: 1800 / 2000 [90%] (Sampling) Chain 2 Iteration: 1900 / 2000 [95%] (Sampling) Chain 4 Iteration: 1100 / 2000 [55%] (Sampling) Chain 3 Iteration: 1300 / 2000 [65%] (Sampling) Chain 1 Iteration: 1300 / 2000 [65%] (Sampling) Chain 2 Iteration: 2000 / 2000 [100%] (Sampling) Chain 3 Iteration: 1400 / 2000 [70%] (Sampling) Chain 4 Iteration: 1200 / 2000 [60%] (Sampling) Chain 4 Iteration: 1300 / 2000 [65%] (Sampling) Chain 2 finished in 2.2 seconds. Chain 1 Iteration: 1400 / 2000 [70%] (Sampling) Chain 1 Iteration: 1500 / 2000 [75%] (Sampling) Chain 3 Iteration: 1500 / 2000 [75%] (Sampling) Chain 4 Iteration: 1400 / 2000 [70%] (Sampling) Chain 1 Iteration: 1600 / 2000 [80%] (Sampling) Chain 3 Iteration: 1600 / 2000 [80%] (Sampling) Chain 4 Iteration: 1500 / 2000 [75%] (Sampling) Chain 1 Iteration: 1700 / 2000 [85%] (Sampling) Chain 3 Iteration: 1700 / 2000 [85%] (Sampling) Chain 4 Iteration: 1600 / 2000 [80%] (Sampling) Chain 1 Iteration: 1800 / 2000 [90%] (Sampling) Chain 3 Iteration: 1800 / 2000 [90%] (Sampling) Chain 4 Iteration: 1700 / 2000 [85%] (Sampling) Chain 1 Iteration: 1900 / 2000 [95%] (Sampling) Chain 3 Iteration: 1900 / 2000 [95%] (Sampling) Chain 4 Iteration: 1800 / 2000 [90%] (Sampling) Chain 1 Iteration: 2000 / 2000 [100%] (Sampling) Chain 3 Iteration: 2000 / 2000 [100%] (Sampling) Chain 4 Iteration: 1900 / 2000 [95%] (Sampling) Chain 4 Iteration: 2000 / 2000 [100%] (Sampling) Chain 1 finished in 2.8 seconds. Chain 3 finished in 2.8 seconds. Chain 4 finished in 2.8 seconds.

All 4 chains finished successfully. Mean chain execution time: 2.7 seconds. Total execution time: 3.0 seconds.

```
epred_rvars(mod_intercept,
            dpar = TRUE,
            newdata = tibble(.rows = 1))
```

2 A tibble: 1 x 4

.epred	mu	hu	sigma
<rvar[1d]>	<rvar[1d]>	<rvar[1d]>	<rvar[1d]>
1 0.21 ± 0.0022	-1.5 ± 0.0074	0.085 ± 0.0095	0.25 ± 0.0058

References