The Journal of Open Source Software

bayesnec: An R package for Concentration-Response modelling and estimation of No-Effect-Concentrations

Rebecca Fisher[1, 2] **and Diego Barneche**[1, 2]

1 Australian Institute of Marine Science, Australia **2**

DOI:

Software

- Review
- Repository
- Archive

Submitted:
Published:

Summary

The `bayesnec` package in R has been developed to fit concentration(dose) - response curves (C-R) to toxicity data for the purpose of deriving No-Effect-Concentration (*NEC*), No-Significant-Effect-Concentration (*NSEC*), and Effect-Concentration (of specified percentage 'x', *ECx*) thresholds from non-linear models fitted using Bayesian MCMC fitting methods via `brms` (Bürkner, 2017; Bürkner, 2018) and `stan`. The package is an adaptation and extension of an initial package `jagsNEC` (Fisher, Ricardo, & Fox, 2020) which was based on the `R2jags` package (Su & Yajima, 2015) and `jags` (Plummer, 2003). In `bayesnec` it is possible to fit a single model, custom model set, specific model set or all of the available models. When multiple models are specified the `bnec` function returns a model weighted estimate of predicted posterior values. A range of support functions and methods are also included to work with the returned single, or multi- model objects that allow extraction of raw or model averaged predicted, *ECx*, *NEC* and *NSEC* values and to interrogate the fitted model or model set.

Statement of need

Concentration-response (C-R) modelling is fundamental to assessing toxicity and deriving toxicity thresholds used in the risk assessments that underpin protection of human health and the environment. They are widespread in the disciplines of pharmacology, toxicology and ecotoxicoloy. Typically C-R curves are non-linear, which increases the complexity of their numerical estimation. Estimates of uncertainty in parameters and derived thresholds are critical to effective integration in risk assessment and formal decision frameworks. Bayesian methods that allow robust quantification of uncertainty with intuitive and direct probabilistic meaning are therefore an ideal platform for C-R modelling in most settings.

Bayesian model fitting can be difficult to automate across a broad range of usage cases, particularly with respect to specifying valid initial values and appropriate priors. This is one reason the use of Bayesian statistics for *NEC* estimation (or even *ECx* estimation) is not currently widely adopted across the broader ecotoxicological and toxicology community, who rarely have access to specialist statistical expertise. The `bayesnec` package provides an accessible interface specifically for fitting *NEC* and other concentration-response models using Bayesian methods. A range of models can be specified based on the known distribution of the "concentration" or "dose" variable (the predictor, x) as well as the "response" (y) variable. The model formula, including priors and initial values required to call a `brms` model are automatically generated based on information contained in the supplied data.

The Journal of Open Source Software

Technical details and Usage

This project started with an implementation of the *NEC* model based on that described in (Fox, 2010) using R2jags. The package has been further generalised to allow a large range of response variables to be modelled using the appropriate statistical distribution. While the original `jagsNEC` implementation supported gaussian, poisson, binomial, gamma, negative binomial and beta response data, `bayesnec` supports all of these in addition to a custom betabinomial family. Furthermore, the new structure implemented using `brms` means `bayesnec` can be readily extended to include any of the `brms` families. In addition to greater flexibility in the available response families, `bayesnec` includes a range of alternative *NEC* model types, as well as a range of typically used smooth concentration-response models (such as 4-parameter logistic and weibull models) that have no *NEC* 'step' function but simply model the response as a smooth function of concentration. We have now incorporated most of the commonly used models in frequentist packages such as `drc` (Ritz, Baty, Streibig, & Gerhard, 2016) (please see the Model details vignette for more information on the full list of models currently available in `bayesnec`).

Model specification

The main working function in `bayesnec` is `bnec`. We have attempted to make the `bnec` function as easy to use as possible, targeting the novice R user. A `bayesnec` model can be fit as simply as:

```
library(bayesnec)
data(nec_data)
exmp_fit <- bnec(data = nec_data, x_var = "x", y_var = "y", model = "all")
```

Only three arguments must be supplied, including: `data` - a `data.frame` containing the data to use for the model; `x_var` - a `character` indicating the column heading containing the concentration (x) variable; and `y_var` - a `character` indicating the column heading containing the response (y) variable. The next argument `model` is a `character` indicating the desired model (or model set, see details below) and currently defaults to "all" models available. A large range of arguments can be specified manually by the user as required for more advanced users.

While the distribution of the x and y variables can be specified directly, `bayesnec` will automatically 'guess' the correct distribution (family) to use based on the characteristics of the provided data, as well as build the relevant model priors (see more details below) and initial values for the `brms` model. Initial values are selected such that they yield predicted values in the range of the observed response (`y_var`) data.

If not supplied, the appropriate family to use will be determined based on the characteristics of the input data. This includes evaluation of class (integer or numeric) as well as the observed range through `check_data`. Guesses include all of the available families (see above) except negative binomial and betabinomimal because these require knowledge on whether the data are over-dispersed, which cannot be assessed prior to model fitting. By default `bayesnec` will use the default link function for the guessed family. Note that in `bnec` the default link for the gamma family has been set to `log`. If other link functions are required for any of the implemented families, such as Identity in the case of a beta family for example, this will need to be specified manually using:

```
family = Beta(link = "identity")
```

Data that are numeric and scaled from `0` to `1` or `0` to `+ve` are assumed to be `beta` and `gamma` distributed respectively. Because neither distribution takes true `0`, if necessary a value that is`0` values are increased by 1/10th of the next smallest non-sero value. Similarly, as `beta` does not take `1`, where `y_var` data = 1 they are substituted with 1-0.001. Data scaled from or `-ve` to `+ve` are modelled using a `gaussian` distribution. If `y_var` data are integers and a `trials_var` argument is supplied (must also be integer) a `binomial` distribution is assumed. If no `trials_var` argument is supplied the `y_var` data are assumed to be `poisson`. Similar checks are used to assign a likely family to the `x_var` (concentration) data. In this case of `x-var` data the assigned family only influences the priors on `x_var` scaled model parameters, which can be overidden by setting manual priors (see more information on priors below).

Specific models can be fit directly using `bnec`, in which case an object of class `bayesnecfit` is returned. Alternatively it is possible to fit a custom model set, specific model set or all of the available models. The default in `bayesnec` is to use `model = "all"` which fits all of the available models.

When multiple models are fit using `bnec` an object of class `bayesmanecfit` is returned that includes a model weighted estimate of predicted posterior values. `bayesnec` also includes a set of `bayesmanecfit` methods for predicting, plotting and extracting effect concentrations and related threshold values that return model weighted estimates.

Multi-model inference can be useful where there are a range of plausible models that could be used (Burnham & Anderson, 2002) and has been recently adopted in ecotoxicology for SSD model inference (Fox et al., 2021; Thorley & Schwarz, 2018). The approach may have considerable value in concentration-response modelling because there is often no a priori knowledge of the functional form that the response relationship should take. In this case model averaging can be a useful way of allowing the data to drive the model selection process, with weights proportional to how well the individual models fit the data. Well fitting models will have high weights, dominating the model averaged outcome. Conversely, poorly fitting models will have very low model weights and will therefore have little influence on the outcome. Were multiple models fit the data equally well, these can equally influence the outcome, and the resultant posterior predictions reflect that model uncertainty. It is possible to specify the "stacking" method (Yao, Vehtari, Simpson, & Gelman, 2018) for model weights if desired (through the argument `loo_controls`) which aim to minimise prediction error. We do not currently recommend using stacking weights given the typical sample sizes associated with most concentration-response experiments, and because the main motivation for model averaging within the `bayesnec` package is to properly capture model uncertainty rather than reduce prediction error. By default `bayesnec` uses the "pseudobma" method with Bayesian bootstrap through `loo_model_weights` (Vehtari et al., 2020; Vehtari, Gelman, & Gabry, 2017). These are reasonably analogous to the way model weights are generated using AIC or AICc (Burnham & Anderson, 2002).

Model diagnostics

A range of tools are available to assess model fit, including an estimate of overdispersion (for relevant families), an extension of the `brms` `rhat` function that can be applied to both `bayesnecfit` and `bayesmanecfit` model objects, and a function `check_chains` that can be used to visually assess chain mixing.

All diagnostic functions available in `brms` and `rstan` can be used on the underlying `brm` model fit by extracting the fitted `brms` model from the `bayenecfit` or `bayesmanecfit` model object. For example, we can use the default `brms` plotting method to obtain a diagnostic plot of the individual fit of the **nec4param** model using:

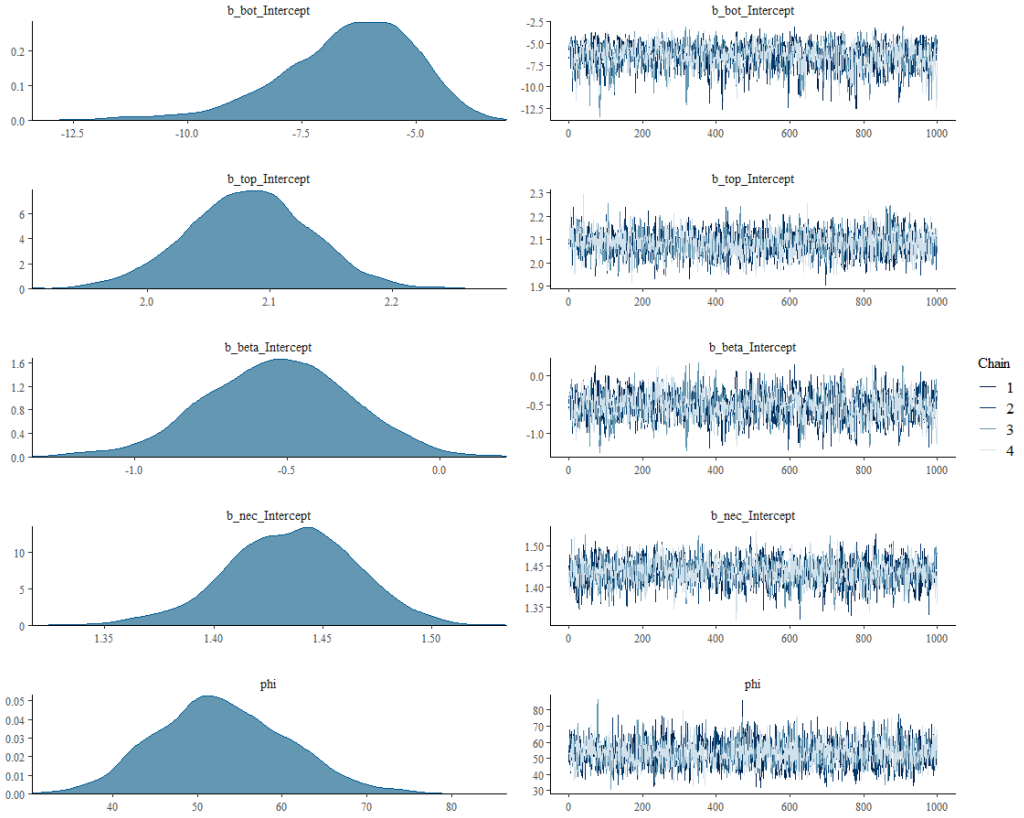


Figure 1: Default brms plot of the nec4param model showing the posterior probability densities and chain mixing for each of the included parameters.

```
plot(exmp_fit$mod_fits$nec4param$fit)
```

which yields a plot the posterior densities and chains plot for each parameter in the specified model as shown in Fig. @ref(fig:brmsplot).

The default number of total iterations in `bayesnec` is 10,000 per chain, with 9,000 of these used as warm up (or burn-in) across 4 chains. If the `bnec` call returns `brms` warning messages the number of iterations and warm-up samples can be adjusted through `iter` and `warmup`. A range of other arguments can be further adjusted to improve convergence, see the rich set of Resources available for the `brms` package for further information.

Several helper functions have been included that allow the user to add or drop models from a `bayesmanecfit` object , or change the model weighting method (`amend`); extract a single or subset of models from the `bayesmanecfit` object (`pull_out`); and examine the priors use for model fitting (`pull_prior`, `sample_prior` and `check_priors`).

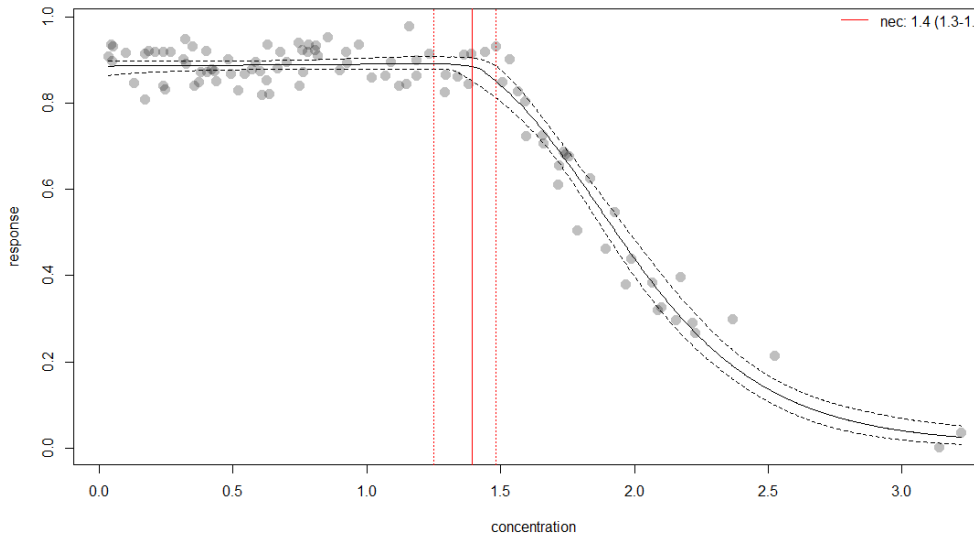
Model inference

`bayesnec` includes a summary method for both `bayesnecfit` and `bayesmanecfit` model objects, providing the usual summary of model parameters and any relevant model fit statistics as returned in the underlying `brm` model fits. This includes a list of fitted models, and model weights are provided and a model averaged NEC, reported with a warning in this case indicating it contains NSEC values. A warning message also indicates that the **ecxll5** model may have convergence issues according to the default brms Rhat criteria.

```
summary(exmp_fit)
# Object of class bayesmanecfit containing the following non-linear models:
#   -  nec4param
#   -  nechorme4
#   -  neclin
#   -  neclinhorme
#   -  nechorme4pwr
#   -  ecxlin
#   -  ecx4param
#   -  ecxwb1
#   -  ecxwb2
#   -  ecxll5
#   -  ecxll4
#   -  ecxhormebc5
#
# Distribution family: beta
# Number of posterior draws per model:  4000
#
# Model weights (Method: pseudobma_bb_weights):
#                  waic   wi
# nec4param     -332.75 0.37
# nechorme4     -330.31 0.13
# neclin        -332.05 0.29
# neclinhorme   -329.80 0.14
# nechorme4pwr -328.40 0.06
# ecxlin        -188.03 0.00
# ecx4param     -302.35 0.00
# ecxwb1        -294.30 0.00
# ecxwb2        -317.12 0.00
# ecxll5        -317.78 0.00
# ecxll4        -302.92 0.00
# ecxhormebc5   -310.25 0.00
#
# Summary of weighted NEC posterior estimates:
# NB: Model set contains the ECX models: ecxlin;ecx4param;ecxwb1;ecxwb2;ecxll5;ecx
#     Estimate Q2.5 Q97.5
# NEC     1.39 1.26  1.48
#
# Warning message:
# In print.manecsummary(x) :
#   The following model had Rhats > 1.05 (no convergence):
#   -  ecxll5
# Consider dropping them (see ?amend)
```

Base R (`plot`) and ggplot2 (`ggbnec`) plotting methods, as well as predict methods have also been developed for both `bayesnecfit` and `bayesmanecfit` model classes. In addition, there are method based functions for extracting *ECx* (`ecx`), *NEC* (`nec`) and *NSEC* (`nsec`) threshold values. In all cases the posterior samples that underpin these functions are achieved through `posterior_epred` from the `brms` package. An example base plot of a `bayesmanecfit` model fit can be seen in Fig. @ref(fig:baseplot).

```
plot(exmp_fit)
```



\begin{figure}
\caption{Base plot of the exmp_fit model averaged curve, showing the fitted median of the posterior prediction (solid line), 95% credible intervals (dashed lines), and the estimated *NEC* value (red vertical lines).} \end{figure}

By default the plot shows the fitted posterior curve with 95% credible intervals, along with an estimate of the η = NEC value. Please see the vignettes for more examples using `bayesnec` models for inference.

Models in `bayesnec`

The argument `model` in the function `bnec` is a character string indicating the name(s) of the desired model. If a recognised model name is provided a single model of the specified type is fit, and `bnec` returns a model object of class `bayesnecfit`. If a vector of two or more of the available models are supplied, `bnec` returns a model object of class `bayesmanecfit` containing Bayesian model averaged predictions for the supplied models, providing they were successfully fitted.

The `model` argument may also be one of "all", meaning all of the available models will be fit; "ecx" meaning only models excluding the η = NEC step parameter will be fit; or "nec" meaning only models with the η = NEC step parameter (see below **Model parameters**) will be fit. There are a range of other pre-define model groups available. The full list of currently implemented model groups can be seen using:

```
library("bayesnec")
models()
```

Please see the Model details vignette or `?model("all")` for more information on all the models available in `bayesnec` and their specific formulation.

Parameter definitions

Where possible we have aimed for consistency in the interpretable meaning of the individual parameters across models. Across the currently implemented model set, models contain from two (basic linear or exponential decay, see **ecxlin** or **ecxexp**) to five possible parameters (e.g. **nechorme4**, **ecxhormebc5**), including:

- τ = top, usually interpretable as either the y-intercept or the upper plateau representing the mean concentration of the response at zero concentration;
- η = NEC, the no-effect-concentration value (the x concentration value where the breakpoint in the regression is estimated at, see **Model types for *NEC* and *ECx* estimation** and (Fox, 2010) for more details on parameter based *NEC* estimation);
- β = beta, generally the exponential decay rate of response, either from 0 concentration or from the estimated η value, with the exception of the **neclinhorme** model where it represents a linear decay from η because slope (α) is required for the linear increase;
- δ = bottom, representing the lower plateau for the response at infinite concentration;
- α = slope, the linear decay rate in the models **neclin** and **ecxlin**, or the linear increase rate prior to η for all hormesis models;
- ec50 notionally the 50% effect concentration but may be influenced by scaling and should therefore not be strictly interpreted; and
- ϵ = d, the exponent in the **ecxsigm** and **necisgm** models.

In addition to the model parameters, all `nec...` models have a step function used to define the breakpoint in the regression, which can be defined as

$$f(x_i, \eta) = \begin{cases} x_i - \eta < 0, & 0 \\ x_i - \eta \geq 0, & 1 \end{cases}$$

Model types for *NEC* and *ECx* estimation

All models provide an estimate of the No-Effect-Concentration (*NEC*). For model types with "nec" as a prefix, the *NEC* is directly estimated as parameter η = NEC in the model, as per (Fox, 2010). Models with "ecx" as a prefix are continuous curve models, typically used for extracting *ECx* values from concentration response data. In this instance the *NEC* reported is actually the No-Significant-Effect-Concentration (*NSEC*), defined as the concentration at which there is a user supplied percentage certainty (see `sig_val`, based on a Bayesian posterior probability) that the response falls below the estimated value of the upper asymptote (τ = top) of the response (i.e. the response value is significantly lower than that expected in the case of no exposure). The default value for `sig_val` is 0.01, which corresponds to an alpha value (Type 1 error rate) of 0.01 for a one-sided test of significance. See `?nsec` for more details. We currently recommend only using the "nec" model set for estimation of *NEC* values, as the *NSEC* concept has yet to be formally peer-reviewed.

ECx estimates can be equally validly obtained from both "nec" and "ecx" models. *ECx* estimates will usually be lower (more conservative) for "ecx" models fitted to the same data as "nec" models (see the Comparing posterior predictions vignette for an example. However, we recommend using "all" models where *ECx* estimation is required because "nec" models can fit some datasets better than "ecx" models and the model averaging approach will place the greatest weight for the outcome that best fits the supplied data. This approach will yield *ECx* estimates that are the most representative of the underlying relationship in the dataset.

There is ambiguity in the definition of *ECx* estimates from hormesis models (these allow an initial increase in the response (see (Mattson, 2008)) and include models with the character string `horme` in their name), as well as those that have no natural lower bound

on the scale of the response (models with the string `lin` in their name, in the case of gaussian response data). For this reason the `ecx` function has arguments `hormesis_def` and `type`, both character vectors indicating the desired behaviour. For `hormesis_def = "max"` ECx values are calculated as a decline from the maximum estimates (i.e. the peak at η = NEC); and `hormesis_def = "control"` (the default) indicates ECx values should be calculated relative to the control, which is assumed to be the lowest observed concentration. For `type = "relative"` ECx is calculated as the percentage decrease from the maximum predicted value of the response (τ = top) to the minimum predicted value of the response (ie, 'relative' to the observed data). For `type = "absolute"` (the default) ECx is calculated as the percentage decrease from the maximum value of the response (τ = top) to 0 (or δ = bottom for models with that parameter). For `type = "direct"` a direct interpolation of y on x is obtained.

Model suitability for response types

Models that have an exponential decay (most models with parameter β = beta) with no δ = bottom parameter are `0` bounded and are not suitable for the gaussian family, or any family modelled using a logit or log link because they cannot generate predictions of negative y (response). Conversely models with a linear decay (containing the string "lin" in their name) are not suitable for modelling families that are `0` bounded (gamma, poisson, negative binomial, beta, binomial, betabinomial) using an identity link. These restrictions do not need to be controlled by the user as a call to `bnec` with `models = "all"` will simply exclude inappropriate models, albeit with a warning.

Strictly speaking, models with a linear hormesis increase are not suitable for modelling families that are `0`, `1` bounded (binomial, beta and betabinomial2), however they are currently allowed in `bayesnec`, with a reasonable fit achieved through a combination of the appropriate family being applied to the response, and the `bayesnec` `make_inits` function that ensures initial values passed to `brms` yield response values within the range of the observed `y_var` data.

Priors on model parameters

The default priors used in `bayesnec` can generally be considered "weakly informative". Priors are constructed for each parameter of each model being fitted based on the characteristics of either the input `x_var` or `y_var` data, depending on which is relevant to the specific parameter scaling. In the case of parameters that scale with `y_var` (the response), priors are constructed based on the relevant link scaling, whether that be identity, the default, or user specified link function for a specific family. The priors are constructed by `bnec` internally calling the function `define_prior`, which takes the arguments `model`, `family` (including the relevant link function), `predictor` (`x_var` data), and `response` (`y_var` data).

Priors for response (`y_var`) scaled parameters

Only the parameters τ = top and δ = bottom scale specifically with the response (`y_var` data) family. For gaussian `y_var` data (or any `y_var` data for which the link ensures valid values of the response can take from `+Inf` to `-Inf`, including `log` and `logit`) priors are `normal` with a standard deviation of `2.5` and a mean set at the 90th and 10th quantiles for τ = top and δ = bottom respectively. In this way `bayesnec` attempts to construct a prior that scales appropriatly with the observed data, with greatest density near the most likely region of the response for the τ = top and δ = bottom parameters, whilst remaining broad enough to have little influence on the each parameter's posterior density.

For poisson, negative binomial and gamma `y_var` data the response is bounded by `0` and normal priors are unsuitable. Instead we use `Gamma` priors, with a mean scaled to correspond to the 75th and 25th quantiles for τ = top and δ = bottom respectively. The mean (μ) is linked mathematically to the shape (s) and rate parameters (r) by the equation

$$\mu = s * (1/r)$$

with the gamma shape parameter is set at 2.

For the binomial, beta, and beta binomial families estimates for τ = top and δ = bottom must necessarily be constrained between `0` and `1` when modelled on the identity link. Because of this constraint there is no need to adjust scaling based on the response. In this case `bayesnec` uses `beta(5, 1)` and `beta(1, 5)` priors to provide a broad density centred across the upper and lower `0` to `1` range for the τ = top and δ = bottom parameters respectively.

Priors for predictor (`x_var`) scaled parameters

The parameters η = NEC and η = ec50 scale with respect to the predictor (`x_var` data), because both of these are estimated in units of the predictor (`x_var`, usually concentration). To stabilise model fitting the η = NEC and η = ec50 parameters are bounded to the upper and lower observed range in the predictor, under the assumption that the range of concentrations in the experiment were sufficient to cover the full range of the response outcomes. The priors used reflect the characteristics of the observed data that are used to guess the appropriate family. If the `x_var` data are bounded to `0` and `>1` a gamma prior is used, with maximum density (μ, see above) at the median value of the predictor, and a shape parameter of 5. If the `x_var` data are bounded to `0` and `1` a `beta(2, 2)` prior is used. For `x_var` data ranging from `-ve` to `+ve` a normal prior is used, with a mean set at the median of the `x_var` values and a standard deviation of 2.5.

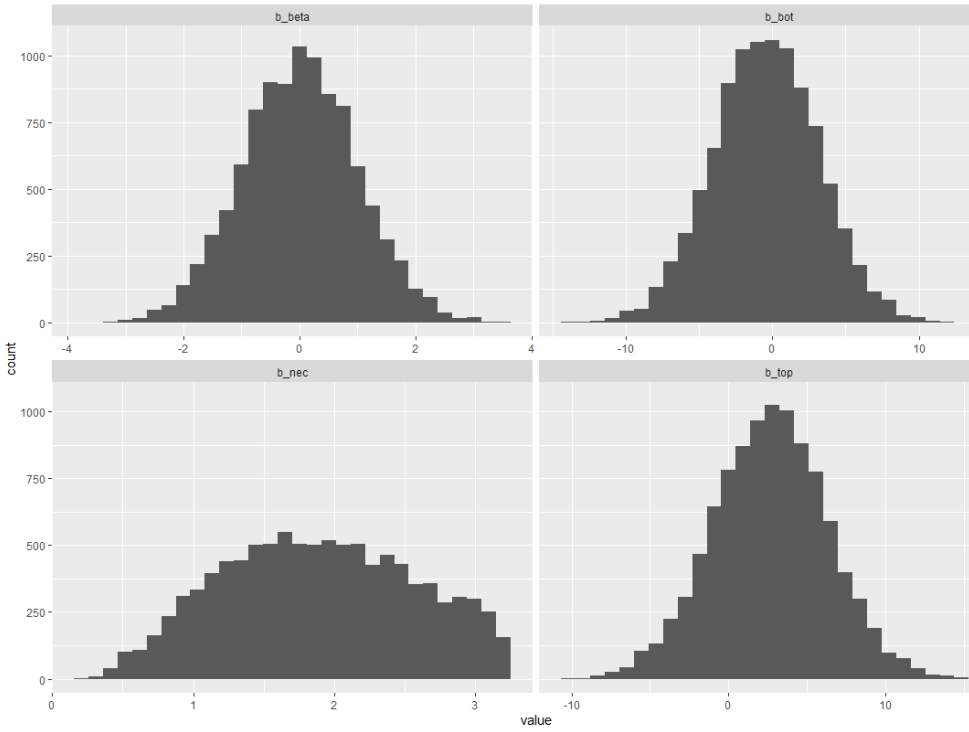
Priors for other parameters

For the parameters β = beta, α = slope and ϵ = d we first ensured any relevant transformations in the model formula such that theoretical values of `-Inf` to `+Inf` are allowable, and a `normal(0, 1)` prior is used. For example in the `nec3param` model β = beta is an exponential decay parameter, which must by definition be bounded to `0` and `+ve`. Calling `exp(beta)` in the model formula ensures the exponent meets these requirements. Note also that a mean of `0` and sd of `1` represents a relatively broad prior on this exponential scaling, so even though we use an sd of 1, this is relatively uninformative in practice.

User specified priors

There may be situations were the default `bayesnec` priors to not behave as desired, or the user wants to provide informative priors. For example the default priors may be too informative, yielding unreasonably tight confidence bands (although this is only likely where there are few data or unique values of the `x_var` data). Conversely, priors may be too vague, leading to poor model convergence. Alternatively, the default priors may be of the wrong statistical family if there was insufficient information in the provided data for `bayesnec` to guess correctly the appropriate ones to use. The priors used in the default model fit can be extracted using `pull_prior`, and a sample or plot of prior values can be obtained from the individual `brms` model fits through the function `sample_priors` which samples directly from the `prior` element in the `brm` model fit (see Fig. @ref(fig:priorsplot)).

```
sample_priors(exmp_fit$mod_fits$nec4param$fit$prior)
```

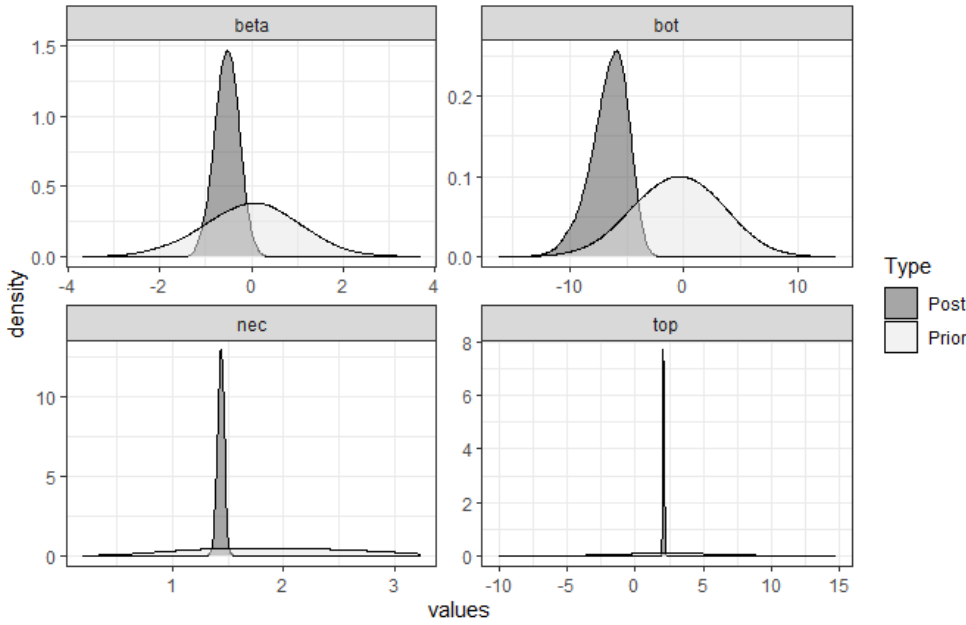


\begin{figure}
\caption{Frequency histograms of samples of the default priors used by bnec for fitting the nec4param model to the example nec_data.} \end{figure}

We can also use the function `check_priors` (based on the `hypothesis` function of `brms`) to assess how the posterior probability density for each parameter differs from that of the prior. Here we show the prior and posterior probability densities for the parameters in the **nec4param** model, extracted from our example fit (see Fig. @ref(fig:checkpriorsplot)). There is also a class `bayesmanecfit` method that can be used to sequentially view all plots in a `bnec` call with multiple models, or write to a pdf as in `check_chains`.

```
# for a single model
exmp_fit_nec4param <- pull_out(exmp_fit, model = "nec4param")
check_priors(exmp_fit_nec4param)

# for all models, writing to a pdf file named Check_priors_plots.pdf
check_priors(exmp_fit, filename = "Check_priors_plots")
```



\begin{figure}
\caption{A comparison of the prior and posterior parameter probability densities for the nec4param model fit to the example nec_data.} \end{figure}

Model comparison

With `bayesnec` we have included a function (`compare_posterior`) that allows bootstrapped comparisons of posterior predictions. This function allows the user to fit several different `bnec` model fits and compare differences in the posterior predictions. Comparisons can be made across the model fits for individual endpoint estimates (e.g. *NEC*, *NSEC* or *ECx*) or across a range of predictor (x) values. Usage is demonstrated in the relevant vignette by comparing different types of models and model sets using a single dataset. However, the intent of this function is to allow comparison across different datasets that might represent, for example, different levels of a fixed factor covariate. For example, this function has been used to compare toxicity of herbicides across three different climate scenarios, to examine the cumulative impacts of pesticides and global warming on corals (Flores, F., Marques, J.A., Uthicke, S., Fisher, R., Patel, F., Kaserzon, S., Negri, 2021).

At this time `bnec` does not allow inclusion of an interaction with a fixed factor. Including an interaction term within each of the non-linear models implemented in `bayesnec` is relatively straightforward, and may be introduced in future releases. However, in many cases the functional form of the response may change with different levels of a given factor. The substantial complexity of defining all possible non-linear model combinations at each factor level means it unlikely this could be feasibly implemented in `bayesnec` in the short term. In the meantime the greatest flexibility in the functional form of individual model fits can be readily obtained using models fitted independently to data within each factor level.

Conclusions, caveats and future directions

The `bayesnec` package is a work in progress, and we welcome suggestions and feedback that will improve the package performance and function. Please submit requests through the package Issues on github. Some existing future enhancements include the addition of other custom families, such as the tweedie distribution for data that are `0` to `+Inf` on

the continuous scale, and the addition of random offsets for accommodating hierarchical designs.

Where possible we have tried to set default values to align with those in `brms`. Wherever we deviate we have tended towards being more conservative (for example in the default `iter`) and/or clearly explained our reasoning above (e.g. use of pseudobma rather than stacking weights). We welcome constructive criticism of our selections and users must expect that default settings may change accordingly in later releases.

As described above in detail, we have made considerable effort to ensure that `bayesnec` makes a sensible guess at the appropriate family, constructs appropriate weakly informative priors, and generates sensible initial values. However, this is a difficult task across such a broad range of non-linear models, and across the potential range of ecotoxicological data that may be used. The user must interrogate their model fits using the wide array of helper functions, and use their own judgement regarding the appropriateness of model inferences for their own application.

Like R itself, bayesnec is free software and comes with ABSOLUTELY NO WARRANTY, or even IMPLIED WARRANTY.

Acknowledgements

The development of `bayesnec` was supported by an AIMS internal grant. David Fox and Gerard Ricardo developed some of the initial code on which the `bayesnec` predecessor `jagsNEC` was based. Usage, testing and functionality of both the `jagsNEC` and `bayesnec` packages was substantially aided through input from Joost van Dam, Andrew Negri, Florita Florence, Heidi Luter, Marie Thomas and Mikaela Nordborg.

References

Burnham, K. P., & Anderson, D. R. (2002). *Model Selection and Multimodel Inference; A Practical Information-Theoretic Approach* (2nd ed., p. 488). New York: Springer.

Bürkner, P. C. (2017). brms: An R package for Bayesian multilevel models using Stan. *Journal of Statistical Software*. doi:10.18637/jss.v080.i01

Bürkner, P.-C. (2018). Advanced Bayesian multilevel modeling with the R package brms. *The R Journal*, *10*(1), 395–411. doi:10.32614/RJ-2018-017

Fisher, R., Ricardo, G., & Fox, D. (2020, July). Bayesian concentration-response modelling using jagsNEC. doi:10.5281/ZENODO.3966864

Flores, F., Marques, J.A., Uthicke, S., Fisher, R., Patel, F., Kaserzon, S., Negri, A. P. (2021). Combined effects of climate change and the herbicide diuron on the coral Acropora millepora, *Marine Pol.*

Fox, D. R. (2010). A Bayesian approach for determining the no effect concentration and hazardous concentration in ecotoxicology. *Ecotoxicology and environmental safety*, *73*(2), 123–131.

Fox, D. R., Dam, R. A. van, Fisher, R., Batley, G. E., Tillmanns, A. R., Thorley, J., Schwarz, C. J., et al. (2021). Recent developments in Species Sensitivity Distribution Modeling. *Environmental Toxicology and Chemistry*, *40*(2), 293–308. doi:https://doi.org/10.1002/etc.4925

Mattson, M. P. (2008). Hormesis defined. doi:10.1016/j.arr.2007.08.007

Plummer, M. (2003). JAGS: A program for analysis of Bayesian graphical models using Gibbs sampling. http://citeseer.ist.psu.edu/plummer03jags.html.

Ritz, C., Baty, F., Streibig, J. C., & Gerhard, D. (2016). Dose-Response Analysis Using R. *PLoS ONE*, *10*(12), e0146021. doi:10.1371/journal.pone.0146021

Su, Y.-S., & Yajima, M. (2015). R2jags: Using R to Run 'JAGS'. R package version 0.5-6. http://CRAN.R-project.org/package=R2jags.

Thorley, J., & Schwarz, C. (2018). ssdtools: Species Sensitivity Distributions. R package version 0.0.3. https://CRAN.R-project.org/package=ssdtools.

Vehtari, A., Gabry, J., Magnusson, M., Yao, Y., Bürkner, P.-C., Paananen, T., & Gelman, A. (2020). Loo: Efficient leave-one-out cross-validation and waic for bayesian models. Retrieved from https://mc-stan.org/loo

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

Yao, Y., Vehtari, A., Simpson, D., & Gelman, A. (2018). Using Stacking to Average Bayesian Predictive Distributions (with Discussion). *Bayesian Analysis*. doi:10.1214/17-ba1091