Simulating cetacean responses to sonar exposure within a Bayesian hierarchical modelling framework — R code description

R code description

Phil Bouchet, Catriona Harris, Len Thomas

10 April, 2020

Contents

1 Introduction

This document describes R functions for simulating cetacean responses to various sonar exposure contexts, within a Bayesian hierarchical modelling framework. The functions are designed to support a sensitivity analysis of the effects of uncertainty in acoustic dose measurements (i.e. received sound levels) on the probability of behavioural response. This work is motivated by the need to assess the utility of different tag types in improving our understanding of dose-response relationships (Schick *et al.* 2019). For more details, please refer to the accompanying technical report (Bouchet *et al.* 2020).

Note: The code implements adaptations of the Bayesian hierarchical dose-response model described in Miller *et al.* (2014), and written in the JAGS dialect of the BUGS language. Model files are stored in the `jags` folder within the working directory and are named: `BayesianBRSModel_JAGSsim_v1.txt` for scenario 1, `BayesianBRSModel_JAGSsim_v2.txt` for scenario 2 etc.

1.1 Loading packages

The `pacman` package (Rinker and Kurkiewicz 2018) is a handy tool to load the required libraries, as shown below. The `p_load()` function, used here, automatically installs any packages not already present on the user's machine. Setting the seed of R's random number generator (using `set.seed()`, followed by any number) guarantees that simulations or random objects can be reproduced.

Note: `pacman` can be installed from CRAN via `install.packages("pacman")`. A development version is also available on Github and can be installed by running the command below:

```
library(devtools); devtools::install_github("trinker/pacman")
```

```
#'------------------------------------------------------------------
# Load required libraries
#'------------------------------------------------------------------
pacman::p_load(rjags,          # Bayesian graphical models using MCMC
               coda,           # Output analysis and diagnostics for MCMC
               MASS,           # Support functions for applied statistics
               truncnorm,      # Truncated Normal
               tidyverse,      # R packages for data science
               lubridate,      # Dates and times made easy
               parallel,       # Support for parallel computation
               doParallel,     # Parallel backend for the foreach/%dopar% function
               utils,          # R utility functions
```

```
                pals,           # Comprehensive palettes and palette evaluation tools
                viridisLite,    # Viridis colour palette
                ArgumentCheck,  # Improved communication re problems in function arguments
                crayon,         # Colored terminal output
                ggnewscale,     # Multiple colour scales in ggplot2
                MCMCvis,        # Visualise, manipulate, and summarise MCMC output
                HDInterval,     # Highest (Posterior) Density Intervals
                reshape2,       # Flexibly reshape data
                bayesplot,      # Plotting for Bayesian Models
                gridExtra,      # Miscellaneous functions for "grid" graphics
                ggpubr,         # Tools for publication ready plots
                plyr,           # The split-apply-combine paradigm for R
                rlang,          # Tools for core language features of R and the tidyverse
                ggridges,       # Ridgeline plots with ggplot2
                hms,            # Pretty time of day
                cowplot,        # Add-on for complex ggplot
                grDevices,      # R Graphics Devices and Support for Colours and Fonts
                tictoc)         # Functions for timing

set.seed(75) # Set the random seed
```

1.2 Importing bespoke functions

All required functions are stored in the `BayesianBR_simulation_functions.R` file, and must first be imported into the R workspace.

```
#'----------------------------------------------------------------------
# Load required functions
#'----------------------------------------------------------------------
source('../BayesianBR_simulation_functions.R')
```

1.3 Simulation scenarios

Four simulation scenarios of increasing complexity are considered (see Bouchet *et al.* 2020). These differ in (i) their observation model, i.e. the treatment of errors relating to the measurements of acoustic dose (either fixed for all animals, or varying by tag type), and (ii) the complexity of the model structure, i.e. which either allows covariate effects and multiple exposure sessions, or not. The true underlying mean response threshold (for all whales), μ, is chosen as the received sound pressure level (SPL) corresponding to a p(response) = 0.5 for a "typical odontocete", in line with existing biphasic response curves (for relevant code, see https://github.com/pjbouchet/dose-response/tree/master/biphasic). This gives a true μ = 165 dB re 1 μPa, which is broadly consistent with the values used in Miller *et al.* (2014), Schick *et al.* (2019), and other studies. Between-animal variation is set to ϕ = 20 dB re 1 μPa by default (but can be changed, if necessary). Likewise, the within-whale variation takes the value of σ = 25 dB. As variances add, the overall combined variation ω is equal to $\sqrt{\phi^2 + \sigma^2} \approx$ 30 dB. A range of sample sizes is tested, from n = 5 to n = 40 individuals. Errors in dose measurements are expressed as standard deviations in received SPL, and fall along a spectrum from 2.5 dB to 35 dB. The lower bound reflects typical errors observed on DTAGs (Isojunno & Wensveen, personal communication), while the upper bound is consistent with estimates from sattelite-tagged whales (Schick *et al.* 2019; Joyce *et al.* 2020).

2 Code

2.1 Overview

The code proceeds in 10 steps:

- **Step 1: Perform initial checks.** A number of sanity checks are implemented to ensure that input parameters are correctly specified. For instance, an error is returned if `lower.bound` exceeds `upper.bound`, and a warning issued if the desired number of cores exceeds the capacity of the processor (by default, the maximum number of cores is then used in this case). In addition, relevent folders are created on disk (if not already present) to store simulation outputs. These contain an index (.txt) file, used to assign a unique identifying number to each simulation run.
- **Step 2: Run simulations.** The simulations are run in parallel on as many cores as specified by `parallel.cores`. Simulated observations are generated for every combination of `n.whales` x `uncertainty.dose` (in scenarios 1 and 2) or `n.whales` x `prop.sat` (in scenarios 3 and 4), and the hierarchical model corresponding to the relevant `scenario` is compiled and executed using the `rjags` package (Plummer 2019).

Setting `verbose = TRUE` ensures that a text-based summary of simulation parameters/steps is printed to the R console during execution (Figure 1). As a reminder, analyses in `rjags` follow a standard structure:

1. Define the model using the BUGS language in a separate text file (here, model files are saved in the in the `jags` folder).
2. Read in the model file using the `jags.model()` function. This creates an object of class `jags`.
3. Update the model using the `update` method for `jags` objects. This constitutes a 'burn-in' period.
4. Extract samples from the model object using the `coda.samples()` function. This creates an object of class `mcmc.list` which can be used to summarize the posterior distribution.

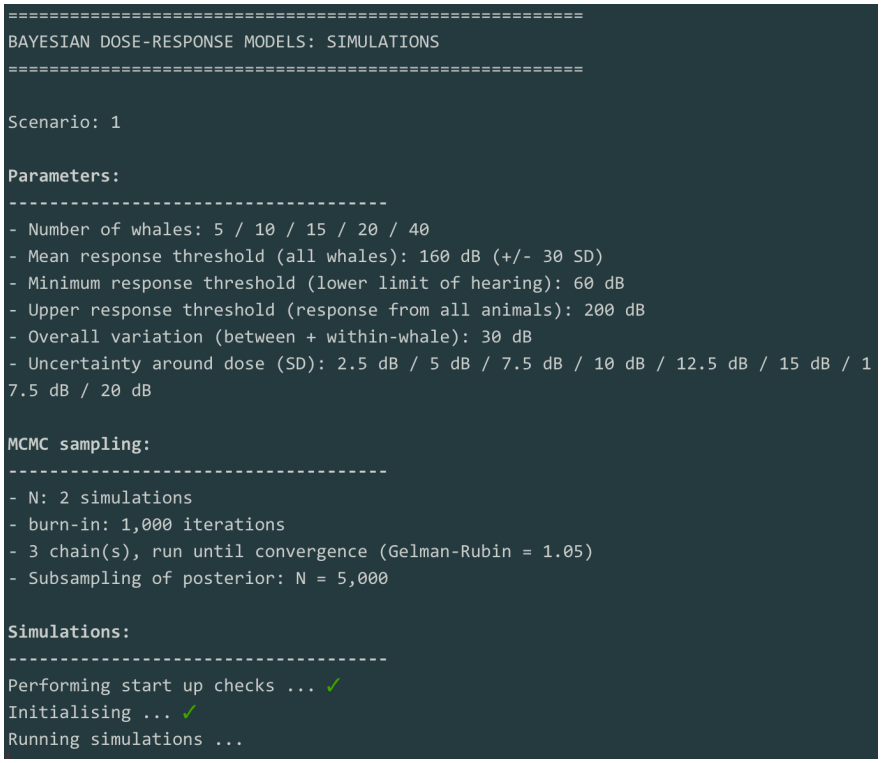

Figure 1: A simulation overview is returned in the R console when `verbose = TRUE`.

Note that automatic MCMC run length and convergence diagnostics are available from the `runjags` package (Denwood 2016), and can be invoked with `mcmc.auto = TRUE`. This kind of adaptive approach can be helpful here as convergence tends to be slower with increasing errors in dose measurements, such that setting a constant chain length would not guarantee convergence in all simulations (and/or be inefficient). However, in our experience, timeout issues can occur with `mcmc.auto = TRUE` when the number of combinations of `n.whales` x `uncertainty.dose/prop.sat` x `n.sim` is large. One solution is to loop through simulations for each individual pair of `n.whales` x `uncertainty.dose/prop.sat` separately. The results can then be compiled using the `compile_sim()` function (see section 2.3).

- **Step 3: Post-processing and compiling.** This is largely convenience code used to extract and format MCMC outputs. Note that in order to circumvent memory problems during subsequent calculations (particularly when running long chains and a large number of simulations), the final posterior samples are obtained from an additional update of the Markov chains after convergence (using `mcmc.n` iterations).

- **Step 4: Effective response range.** Next, the code produces posterior estimates of the effective response range (or effective response radius, ERR). The ERR is a novel metric that combines information on animal distribution, dose-response relationships, sound source levels, and models of acoustic propagation, to produce more robust and accurate estimates of impact (Tyack and Thomas 2019). See section 2.3 for more details.

- **Step 5: Dose-response curves.** Similarly, dose-response curves are computed for each simulation from pairs of posterior estimates of μ and ω (or ϕ and σ). Summary statistics (i.e. median, quantiles) are saved and will be used for plotting using `plot_doseresponse()`.

- **Step 6: Effective sample sizes and autocorrelation.** MCMC samples must be representative of the posterior distribution, however most MCMC chains exhibit some level of autocorrelation, such that successive steps in the chains are not independent. This section uses the `effectiveSize()` and the `autocorr.diag()` functions from the `coda` package (Plummer *et al.* 2019) to assess chain autocorrelation and calculate effective sample sizes (ESS) as a metric of chain length for each parameter (Kruschke 2015). The mean ESS for each combination of `n.whales` x `uncertainty.dose` or `n.whales` x `prop.sat` (across all simulations) is calculated, and a warning issued if resulting values are inferior to `min.ESS`. Kruschke (2015) recommends a heuristic threshold of `ESS = 10,000` for achieving numerically stable estimates. However, it may be computationally inefficient to run chains for that long, and `min.ESS` thus defaults to `1,000`. Similarly, a warning is returned if the mean correlation coefficient for any combination of `n.whales` x `uncertainty.dose` (or x `prop.sat`) x lag > `0.2`.

- **Step 7: Convergence diagnostics.** The code performs both numerical and visual convergence checks (Figure 2), in the form of trace plots (saved to disk) and Gelman-Rubin statistics, respectively. The latter, denoted `Rhat`, compares the between-chain variance to the within-chain variance and is generally taken to indicate successful convergence when `Rhat < 1.1`. For a more conservative assessment, the default value of `Rhat` is set to `mcmc.gelmanrubin = 1.05`. Trace plots are produced using the `mcmc_trace()` function from the `bayesplot` package (Gabry and Mahr 2019) and `Rhat` values obtained using the `gelman.diag()` function from the `coda` package (Plummer *et al.* 2019).

- **Step 8: Posterior predictive checks.** Two types of additional checks are considered next, namely: (1) the posterior prior overlap (PPO), and (2) posterior predictive checks (PPC) (Figure 2). Checking the PPO has particular utility when trying to determine if the parameters in a model are identifiable. If substantial PPO exists, the prior may simply be dictating the posterior distribution, and the data may thus have little influence on the results. By contrast, if a small degree of PPO exists, the data are informative enough to overcome the influence of the prior. In the field of ecology, nonidentifiability is a particular concern in some types of mark-recapture models. Gimenez *et al.* (2009) suggest that a PPO > 35% indicates weak identifiability. PPCs simulate replicated data under the fitted model and then compare these to the observed data. PPCs are useful for determining whether a model gives "valid" predictions about the reality - i.e. whether predictions are consistent with observed data. PPOs are computed using the `MCMCtrace()` function from the `MCMCvis` package (Youngflesh 2018) and a warning issued if the PPO for any simulation is less than 0.35. PPC plots (saved to disk) are generated using the `ppc_densoverlay()` function, which is adapted from `ppc_dens_overlay()` from the `bayesplot` package (Gabry and Mahr 2019).

- **Step 9: Percent relative (median) bias.** This part of the code computes the percent relative (mean/median) bias for each simulation, as a measure of the average tendency of simulated values to depart from the true ones (for each parameter of interest). The mean bias is given as `(posterior mean - true value)/true value * 100`. The median bias is identical, but based on the posterior median rather than the posterior mean. It is defined as `(posterior median - true value)/true value * 100`.

- **Step 10: Save outputs.** Lastly, all relevant outputs are saved to a list object. In addition, a summary of the simulation can be written to disk as a text file by setting `save.textsummary = TRUE`. Optionally, results can also be saved to disk using `save.results = TRUE`.

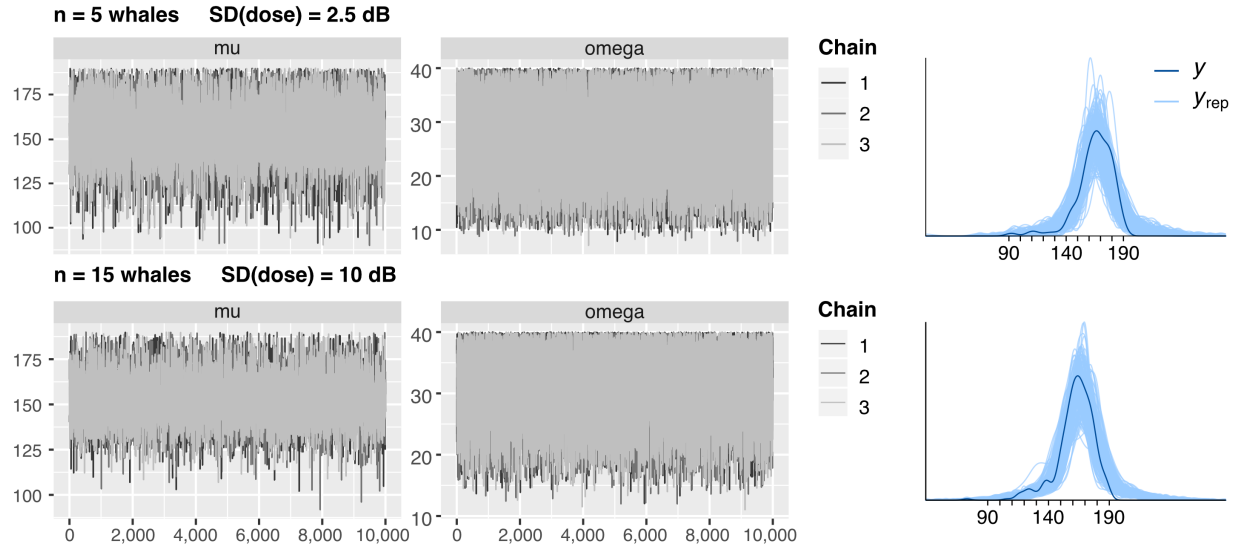

Figure 2: Example graphical model checks performed during simulations. **Left:** Trace plots used in assessing MCMC chain convergence. **Right:** posterior predictive plots comparing the distribution of observations simulated from the fitted model to that of the observed data.

2.2 Running simulations

2.2.1 run_scenario()

This is the core function used for running simulations using Markov Chain Monte Carlo (MCMC). Its arguments are listed below:

Argument	Description
scenario	Scenario ID. Must be an integer between 1 and 4.
n.sim	Integer. Number of simulations to run.
n.whales	Integer. Number of whales. Can be supplied as a single value or vector of integers, which will be assessed iteratively.
n.trials.per.whale	Integer. Number of exposure sessions per animal.
uncertainty.dose	Uncertainty in acoustic dose, expressed as a standard deviation in received sound levels (in dB). Can be supplied as a single value, or vector of integers, which will be assessed iteratively. Relevant only to scenarios 1 and 2.

(continued)

Argument	Description
prop.sat	Proportion of animals fitted with satellite tags. Can be supplied as a single value, or vector of values between 0 and 100, which will be assessed iteratively. Relevant only to scenarios 3 and 4.
dtag.sd	Uncertainty in dose measurements made on DTAGs within scenarios 3 and 4, expressed as a standard deviation in received sound levels (in dB). Defaults to 2.5 dB.
source.level	Source level of the sonar. Defaults to 210 dB re 1µPa m at a nominal frequency of 3 kHz, as per Tyack & Thomas (2019).
species.argos	Cetacean species to simulate in scenarios 3 and 4. One of either 'Zc' for Cuvier's beaked whale (Ziphius cavirostris) or 'Gm' for short-finned pilot whale (Globicephala macrorhynchus). ARGOS ellipses are simulated for each species based on real data from tagged animals (courtesy of Rob Schick at Duke University).
true.mu	True underlying mean response threshold for all whales. Follows a truncated Normal with lower and upper truncation points given by lower.bound and upper.bound.
lower.bound	Lower bound for true.mu, i.e. minimum response threshold. Defaults to 60 dB, under the conservative assumption that any sound below this threshold will not be heard for a 'typical' odontocete above ambient noise (Schick et al. 2019).
upper.bound	Upper bound for true.mu, i.e. threshold at which all whales are expected to respond. Defaults to 190 dB for a typical odontocete.
true.omega	Combined (between and within-whale) variation (SD). Relevant to scenarios 1 and 3.
omega.upper.bound	Upper bound for true.omega.
true.phi	Between-whale variation (SD). Defaults to 20 dB. Relevant to scenarios 2 and 4.
phi.upper.bound	Upper bound for true.phi.
true.sigma	Within-whale variation (SD). Relevant to scenarios 2 and 4.
sigma.upper.bound	Upper bound for true.sigma.
true.alpha	Parameter representing the effect of previous exposure to sonar on the animals' response threshold. Follows a Normal distribution with standard deviation given by alpha.sd.
alpha.sd	Standard deviation of the exposure effect (true.alpha).

(continued)

Argument	Description
true.beta	Parameter governing the effect of MFAS (Mid-frequency active sonar) relative to LFAS (Low-frequency active sonar) on the animals' response threshold. Follows a Normal distribution with standard deviation given by beta.sd.
beta.sd	Standard deviation of the MFAS effect (true.beta).
censor.right	Right-censoring range. Must be a vector of two values representing the lower and upper bounds, respectively. In each simulation, a value is drawn from a Uniform distribution bounded by this range, and any observation above this threshold is set to NA.
burn.in	Number of Markov Chain Monte Carlo (MCMC) iterations required to achieve chain convergence. These will be discarded (burn-in). When mcmc.auto is set to {TRUE, an initial burn-in of 5,000 is applied, and the autorun.jags function triggered, ensuring that all chains converge before they are returned.
mcmc.adapt	Number of iterations used for adaptation. When a JAGS model is compiled, it may require an initial sampling phase during which the samplers adapt their behaviour to maximize their efficiency (e.g. a Metropolis-Hastings random walk algorithm may change its step size). The sequence of samples generated during this adaptive phase is not a Markov chain, and therefore may not be used for posterior inference on the model.
mcmc.chains	Number of MCMC chains. Defaults to 3.
mcmc.thin	Thinning rate. Thinning is used to reduce autocorrelation in the MCMC posterior sample, by only retaining every mcmc.thin^{th} value.
mcmc.n	Integer. Number of MCMC samples extracted after convergence.
mcmc.auto	Logical. Run the MCMC model in JAGS with automatically run length and convergence diagnostics, using the runjags package.
mcmc.gelmanrubin	Threshold for assessing chain convergence according to the Gelman-Rubin statistic. Defaults to 1.1.
mcmc.save	Logical. Whether to save MCMC samples to disk (in .rds format). Warning: Setting this argument to TRUE may result in large output files, if a high number of simulations is required.
animal.density	Number of whales per square km. Used in the calculation of the effective response range (ERR). Defaults to 1.

(continued)

Argument	Description
n.bins	Number of bins used in the calculation of the effective response range (ERR). Defaults to 500.
check.convergence	Logical. Whether to perform convergence checks on MCMC chains, including calculating values of the Gelman-Rubin statistic and producing trace plots. Defaults to TRUE.
gr.multivariate	Logical. If TRUE, the multivariate Gelman-Rubin statistic is returned.
posterior.checks	Logical. Whether to perform posterior checks, including PPO: Prior posterior overlap; PPC: Posterior predictive checks. Defaults to TRUE.
min.ESS	Minimum effective sample size (ESS) considered to be acceptable. A warning will be issued if this value is not attained for each parameter. Defaults to 1000.
correlation.threshold	Threshold for assessing MCMC chain correlation. If this value is exceeded for any lag, a warning will be triggered.
save.trace	Logical. Whether to save a subset of trace plots to disk.
no.tracePlots	Integer between 1 and 5. Number of simulations (chosen at random) for which to produce trace plots when check.convergence is TRUE.
n.yrep	Number of replicate datasets to generate for posterior predictive checking (posterior.checks = TRUE).
hdi.value	Probability mass to include in credible intervals derived as highest density intervals. Must be a number between 0 and 1. Defaults to 0.95 for a 95% HDI.
verbose	Logical. Whether to prints an overview of simulation parameters/steps to the R console during execution. Defaults to TRUE.
parallel.cores	Integer. Number of cores to use when computations are run in parallel. Default is 1 for single core computing.
record.time	Logical. Whether to time the simulations. Defaults to TRUE.
save.textsummary	Logical. If TRUE, a text-based summary of simulation parameters will be saved on disk.
save.results	Logical. Whether to save the simulations outputs as an .RData file on disk. Defaults to TRUE.

2.3 Visualising outputs

The following functions are available for visualising and summarising simulation results.

2.3.1 plot_results()

This function produces forest plots (Figure 3) showing the average (i) posterior median and (ii) credible interval bounds for parameters of interest across simulation runs. A bivariate colour legend is used to characterise combinations of `n.whales` x `uncertainty.dose` or `n.whales` x `prop.sat`, depending on the scenario considered and as defined in `run_scenario()`. Each sample size is given a hue along the `viridis` colour scale, and each measurement error/tag ratio is given a different shade, with lower values shown in lighter tones, and higher values in darker tones. In addition, heat maps (coloured grids) are produced to show patterns in the average (i) credible interval width (in dB re 1μPa or km, as appropriate), (ii) percent relative bias, and (iii) prior posterior overlap (PPO) for each combination of sample size and error/tag ratio, for a given parameter. These plots are saved both individually and in a combined PDF on disk.

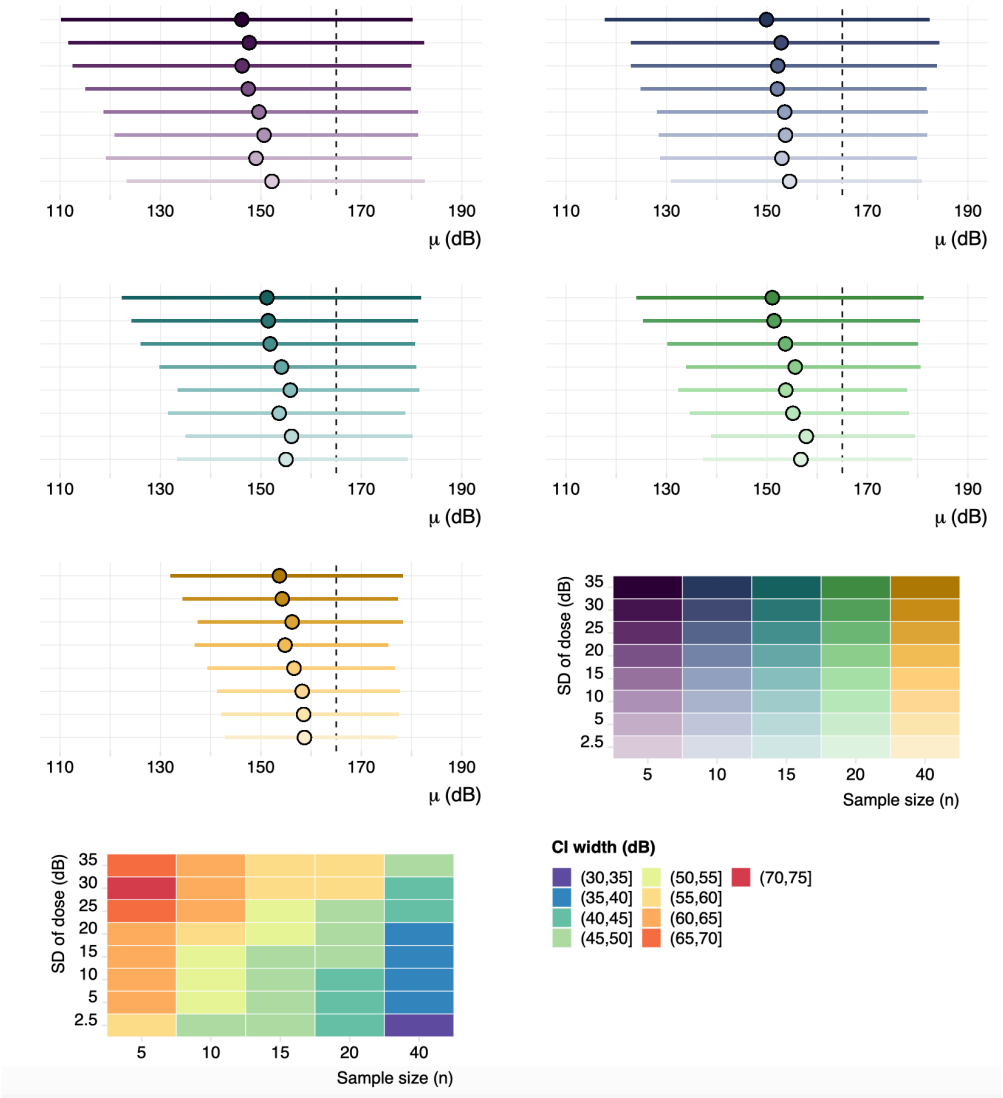

Figure 3: Example forest plot produced by the `plot_results()` function for the mean response threshold across all whales, μ, under scenario 1.

Argument	Description
mcmc.object	List object containing simulation outputs, as returned by simulate_scenario().
layout.ncol	Number of columns used in the final plot layout. Must be an integer between 1 and 3.
pars.to.plot	Parameter(s) of interest. By default, the function will produce plots for every monitored parameter (param = NULL).
summary.method	Character vector. One of 'mean' or 'median'. Whether to calculate the average or median value of posterior statistics across simulations.

start.shade	Number between 0 and 1 indicating the shade of the lightest to use on the Y-axis. Higher values indicate darker shades. Defaults to 0.2.
n.cols	Integer. Minimum number of colours used to define the colour palettes of the heat plots.
darken.bars	Logical. Whether to add dark lines to the plot to enhance the legibility of the lightest colours. Defaults to FALSE.
save.to.disk	Logical. Whether to save the plots to disk. Defaults to FALSE.
output.format	Output file type. Defaults to 'pdf'.

2.3.2 plot_doseresponse()

This function generates dose-response plots such as those presented in Miller *et al.* (2014) and Harris *et al.* (2018) (Figure 4). By default, one plot is produced per combination of `n.whales` x `uncertainty.dose` or `n.whales` x `prop.sat`, however various parameters are available to select combinations of interest and modify the layout of the output file.

Argument	Description
mcmc.object	List object containing simulation outputs, as returned by simulate_scenario().
select.n	Subset of sample sizes to display. All values are shown when set to NULL (the default).
select.obs	Subset of values to display for the observation model parameter (i.e. uncertainty.dose in scenarios 1 and 2, prop.sat in scenarios 3 and 4). All values are shown when set to NULL (the default).
concatenate	Logical. By default, plots for each sample size are saved on separate pages when output to PDF, which can result in unnecessary empty space depending on the type of layout chosen (n.row x n.col). Set this argument to TRUE to combine plots across a minimum number of pages.
n.row	Number of rows used in the plot layout. Values greater than 1 (default) allow multiple plots to be arranged in the same plotting space.
n.col	Number of columns used in the plot layout. Values greater than 1 (default) allow multiple plots to be arranged in the same plotting space.
save.to.disk	Logical. Whether to save the plots to disk. Defaults to FALSE.
output.format	Output file type. Defaults to 'pdf'.
plot.width	Width of the output plot (in pixels).
plot.height	Height of the output plot (in pixels).

plot.res	Resolution of the output plot (in dpi).

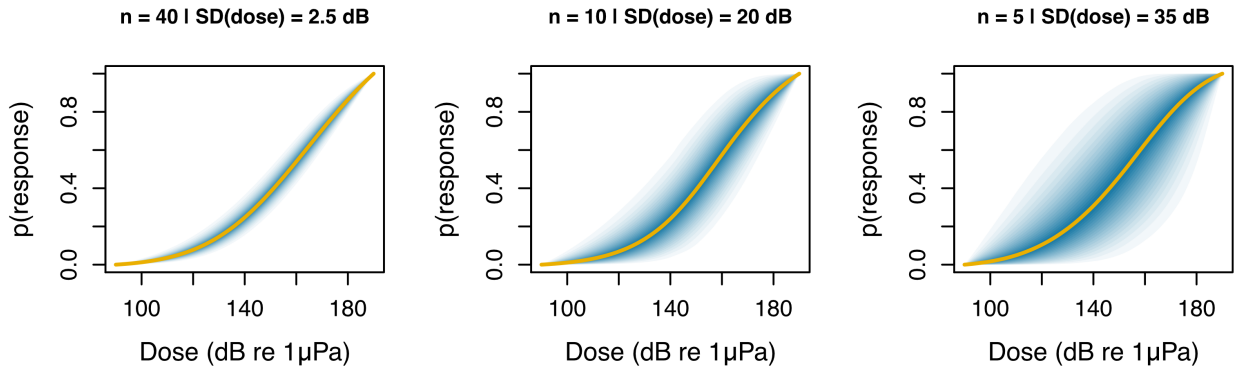

Figure 4: Example dose-response curves produced by the `plot_doseresponse()` function.

2.4 Convenience functions

2.4.1 extra_sim()

This function can be called to run additional simulations, for combinations of `n.whales` x `uncertainty.dose` or `n.whales` x `prop.sat` in which not all MCMC chains converged for all parameters. A tally of convergence failures is saved by default in the output from `run_scenario()` under `$convergence`. The function launches an internal call to `run_scenario()` with `mcmc.auto` set to `TRUE`, such that posterior samples are automatically returned after convergence has been achieved. The `replace.sims` argument allows users to update the simulation results as appropriate (see Figure 5).

Argument	Description
mcmc.object	List object containing simulation outputs, as returned by simulate_scenario().
replace.sims	Logical. If FALSE, the function only returns the output of run_scenario() for the combinations of n.whales x uncertainty.dose/prop.sat for which convergence was not achieved. If TRUE, the entire mcmc.object is updated with the new results (i.e. including posterior values, summaries, etc).
update.dr	Logical. If TRUE, dose-response curves are also updated. Note: This requires MCMC sample objects to have been saved on disk (mcmc.save = TRUE in run_scenario()).

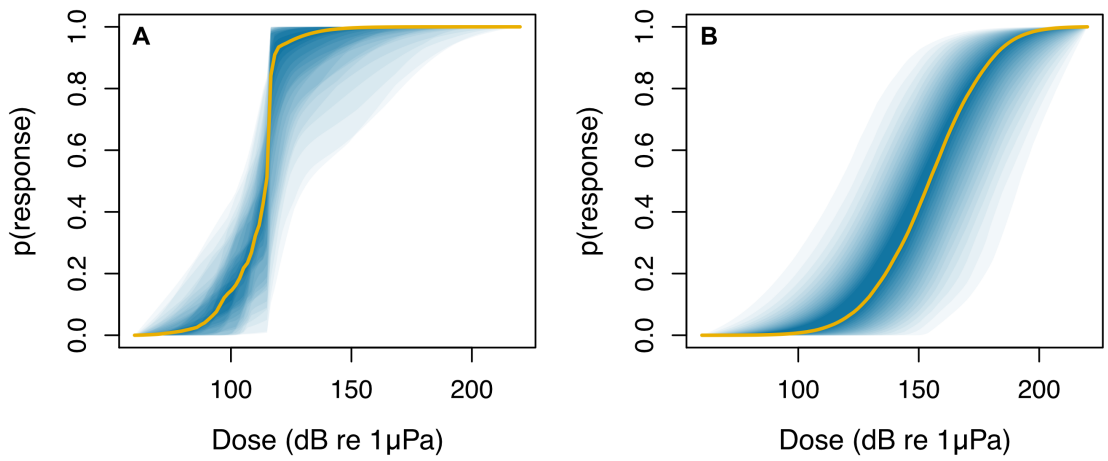

Figure 5: Dose-response curves obtained before and after updates from the `extra_sim()` function. **(A)** Curves derived from simulations in which MCMC chains did not converge for all parameters. **(B)** The same data, after the relevant simulations were corrected to achieved convergence.

2.4.2 effective_range()

This function is called by `run_scenario()` to calculate the effective response range (or effective response radius, ERR), a novel metric that combines information on animal distribution, dose-response relationships, sound source levels, and models of acoustic propagation, to produce more robust and accurate estimates of impact (Tyack and Thomas 2019). By default, the function assumes a simple inverse-squared circular transmission loss model, and performs calculations over a maximum range given by the distance at which received levels drops below `lower.bound` (i.e. such that the probability of response is zero).

Argument	Description
response.threshold	Mean response threshold (received sound level at which a response is trigerred), in dB re 1uPa rms.
response.sd	Uncertainty around the response threshold, expressed as a standard deviation of response.threshold.
response.lowerbound	Received level below which no animals respond, in dB re 1uPa rms.
response.upperbound	Received level at which all animals are expected to exhibit a response, in dB re 1uPa rms.
received.level	Received levels, reflecting a given source level and transmission loss model.
D	Animal density. Defaults to 1 animal per km2, as per the example presented in Tyack & Thomas (2019).
n.bins	Number of bins with which to divide the maximum.rge. The number of whales showing a response is evaluated in each bin.
absorption.loss	Maximum range for sound propagation (km).

2.4.3 xy_error()

This function calculates the uncertainty (SD) in the acoustic dose received by animals fitted with satellite tags, whilst accounting for positional uncertainty on a 2D plane. This is done by simulating candidate ARGOS (x,y) locations within plausible error ellipses around position estimates for a given received level, assuming a simple inverse-squared spherical transmission loss model. See Bouchet *et al.* (2020) for details.

Argument	Description
argos.data	Input ARGOS data.
received.lvl	Received level.
source.lvl	Sound pressure level of the noise source.
multi	Logical. If TRUE, repeat the estimation for multiple ellipses.
n.ellipses	Number of ellipses to simulate when multi = TRUE.
plot.ellipse	Logical. If TRUE, create plot a random realisation of the bivariate normal from which simulated errors are drawn.

2.4.4 run_argos_example()

This function is used to produce Figure 16 in the accomnpanying technical report (Bouchet *et al.* 2020). It simulates animals at incremental distances from the sonar source and computes the variation in received levels experienced by these animals, accounting for variable positional error from Argos-linked satellite tags.

Argument	Description
N	Number of simulated animals.
range.min	Minimum distance from the noise source.
range.max	Maximum distance from the noise source.
range.increment	Increments by which to divide the range.
source.lvl	Level of the noise source.

2.4.5 load_sim() and compile_sim()

These two functions are used to load the results from a single simulation run (provided they have been saved to disk), and compile the results from multiple simulation runs, respectively. The latter is useful when multiple combinations of `n.whales` x `uncertainty.dose` or `n.whales` x `prop.sat` have been run separately, and need to be collated.

Argument	Description
scenario	Scenario ID.
index	Simulation index ID.

2.4.6 hexa2hex()

This function returns the HEX colour code equivalent to an input colour subjected to a given level of transparency. This is useful for emulating different opacity levels for overlapping plot elements without actually making them transparent. `hexa2hex()` is called internally by `plot_results()`.

Argument	Description
input.colour	Initial colour.
opacity	Desired level of transparency. Must be a number between 0 and 1, with higher values representing more opaque colours.
bg.colour	Colour of the background. Defaults to 'white'.

2.4.7 addlzero() and removelzero()

These are convenience functions for adding and removing leading zeroes, respectively. They are used to ensure that combinations of `n.whales` x `uncertainty.dose` or `n.whales` x `prop.sat`, as specified in `run_scenario()`, are ordered correctly when plotted or summarised.

```
# Quick example
addlzero(5)
```

```
## [1] "005"
```

```
removelzero(025)
```

```
##
## "25"
```

2.4.8 start_cluster() and stop_cluster()

These two functions are shorthands for launching and stopping parallel clusters, respectively. The only argument to `start_cluster()` is `n.cores`, indicating the number of cores required for parallel processing.

2.4.9 nth_element()

This function is used to extract the n^{th} element from a vector, based on a chosen start position.

Argument	Description
vector	Input vector.
starting.position	Starting position.
n	Integer. Sampling interval.

2.4.10 TL()

This function returns the transmission loss for a given range, assuming a simple inverse-square transmission loss model. **Note:** This is how much is lost, **NOT** the sound level after loss.

Argument	Description
rge	Range in km.
a	Sound absorption coefficient, in dB per km. This is frequency-dependent, and takes a value of 0.185 for a 3 kHz signal under normal sea conditions.

2.4.11 range_finder()

This function returns the range corresponding to a given received level, assuming a simple inverse-square transmission loss model.

Argument	Description
rge	Range in km.
SL	Sound pressure level of the noise source.
target.L	Target noise level for which a range must be estimated.

2.4.12 removelabels()

This is a simple convenience function for removing appropriate labels in a tibble.

2.4.13 name_list()

The simulation code returns lists of MCMC objects. This function is used to assign appropriate names to each list element.

Argument	Description
scenario.id	Scenario ID.
input.list	Input list.
Nwhales	Number of whales. Corresponds to n.whales in run_scenario().
Nsim	Number of simulations. Corresponds to n.sim in run_scenario().
dose.or.ratio	Values of the observation model parameter (i.e. uncertainty.dose in scenarios 1+2 or prop.sat in scenarios 3+4).

2.4.14 ppc.dens.overlay()

This function is a modified version of the `ppc_dens_overlay()` function from the `bayesplot` package (Gabry and Mahr 2019), which does **not** return an error when performing graphical posterior predictive checks from posterior samples containing NAs. This is necessary, as the observation model we are using entails a right-censoring component (and therefore generates some NA values). The function is called internally during simulations.

3 References

Bouchet PJ, Harris CM, Thomas L (2020). Simulating cetacean responses to sonar exposure within a Bayesian hierarchical modelling framework: Technical report. University of St Andrews; Double MOCHA Report, 39 p.

Denwood MJ (2016). runjags: An R package providing interface utilities, model templates, parallel computing methods and additional distributions for MCMC models in JAGS. *Journal of Statistical Software* **71**, 1–25. DOI: 10.18637/jss.v071.i09

Gabry J, Mahr T (2019). Bayesplot: Plotting for bayesian models. R package version 1.7.0. Available at: https://cran.r-project.org/web/packages/bayesplot/index.html

Gimenez O, Morgan BJT, Brooks SP (2009). Weak identifiability in models for mark-recapture-recovery data. *In*: 'Modeling demographic processes in marked populations', pp. 1055–1067 (Eds D. L. Thomson, E. G. Cooch, and M. J. Conroy). Springer US. DOI: 10.1007/978-0-387-78151-8_48

Harris CM, Thomas L, Falcone EA, Hildebrand J, Houser D, Kvadsheim PH, Lam F-PA, Miller PJO, Moretti DJ, Read AJ, Slabbekoorn H, Southall BL, Tyack PL, Wartzok D, Janik VM (2018). Marine mammals and sonar: Dose-response studies, the risk-disturbance hypothesis and the role of exposure context. *Journal of Applied Ecology* **55**, 396–404. DOI: 10.1111/1365-2664.12955

Joyce TW, Durban JW, Claridge DE, Dunn CA, Hickmott LS, Fearnbach H, Dolan K, Moretti D (2020). Behavioral responses of satellite tracked Blainville's beaked whales (*Mesoplodon densirostris*) to mid-frequency active sonar. *Marine Mammal Science* **36**, 29–46. DOI: 10.1111/mms.12624

Kruschke JK (2015). Doing Bayesian Data Analysis: A Tutorial with R and BUGS 2nd ed. Academic Press, Oxford, 759 p.

Miller PJO, Antunes RN, Wensveen PJ, Samarra FIP, Alves AC, Tyack PL, Kvadsheim PH, Kleivane L, Lam F-PA, Ainslie MA, Thomas L (2014). Dose-response relationships for the onset of avoidance of sonar by free-ranging killer whales. *Journal of the Acoustical Society of America* **135**, 975–993. DOI: 10.1121/1.4861346

Plummer M (2019). Rjags: Bayesian graphical models using mcmc. R package version 4-9. Available at: https://CRAN.R-project.org/package=rjags

Plummer M, Best N, Cowles K, Vines K (2019). CODA: Convergence diagnosis and output analysis for mcmc. R package version 19-3. Available at: https://cran.r-project.org/web/packages/coda/index.html

Rinker TW, Kurkiewicz D (2018). pacman: Package management for R. R package version 0.5.0. Available at: http://github.com/trinker/pacman

Schick RS, Bowers M, DeRuiter S, Friedlaender A, Joseph J, Margolina T, Nowacek DP, Southall BL (2019). Accounting for positional uncertainty when modeling received levels for tagged cetaceans exposed to sonar. *Aquatic Mammals* **45**, 675–690. DOI: 10.1578/AM.45.6.2019.675

Tyack PL, Thomas L (2019). Using dose–response functions to improve calculations of the impact of anthropogenic noise. *Aquatic Conservation: Marine and Freshwater Ecosystems* **29**, 242–253. DOI: 10.1002/aqc.3149

Youngflesh C (2018). MCMCvis: Tools to visualize, manipulate, and summarize mcmc output. *Journal of Open Source Software* **3**, 640. DOI: 10.21105/joss.00640