

groupedBBMH: Grouped Beta-Binomial Model with Metropolis-Hastings

Sumonkanti Das

2025-11-07

Introduction

The **groupedBBMH** R package provides tools for fitting **nested beta-binomial hierarchical models** designed for **group-testing data in biosecurity surveillance**, explicitly accounting for imperfect testing conditions. In routine inspections of imported agricultural consignments, only group-level binary outcomes—positive or negative—are observed, without information on the exact number of contaminated items. This package extends traditional beta-binomial modeling by incorporating the **nested structure of the population** as well as **imperfect test sensitivity and specificity**, offering **exact inference** via a Metropolis-Hastings (MH) algorithm and **Maximum Likelihood Estimation**. For faster computation, an approximate **beta-binomial** model can be developed using standard R packages such as **brms** (Bayesian) or **VGAM** (MLE). The **groupedBBMH** package also allows users to specify a **biologically meaningful minimum prevalence threshold k** , ensuring that estimates respect the lower bound of plausible contamination levels. Overall, the package supports estimation of contamination prevalence, quantification of the risk of undetected contamination, and facilitates risk-based decision-making through model-based simulations that account for test accuracy and minimum prevalence thresholds.

groupedBBMH implements the methods developed in *Das et al. (2025)*, providing exact and approximate estimation for beta-binomial models with censored group-testing data while explicitly accounting for group-level test inaccuracy (sensitivity, Δ and specificity Λ) and biologically meaningful minimum prevalence threshold. While existing R packages such as **brms** or **VGAM** can fit approximate beta-binomial models in case of perfect test ($\Delta = \Lambda = 1$) and zero threshold ($k = 0$), **groupedBBMH** enables fitting the **exact model** according to the nested structure of the population, test inaccuracy and minimum prevalence threshold.

This vignette is organized as follows:

1. Mathematical Background & Notation

Definitions of t_{yi} , b , m , T_{Xi} , L_i , and the beta-binomial population models in exact and approximate form

under test accuracy (Δ and Λ) and minimum positive prevalence k .

2. Case Study & Basic Usage of Package Functions

Describing a case study and then demonstration of model fitting under different scenarios and then calculation of leakage parameters.

3. Model-based simulation

Assessing community-risk based on model-based simulation using the estimated BB model parameters from the case study.

4. Session Info & References

`sessionInfo()` output and full bibliographic entries.

Installation

You can install the development version of `groupedHG` as below.

```
install.packages("devtools")
devtools::install_github("sumon148/groupedBBMH")

library(groupedBBMH)
library(ggplot2)
library(ggpubr)
library(MASS)
library(bayesplot)
#> This is bayesplot version 1.13.0
#> - Online documentation and vignettes at mc-stan.org/bayesplot
#> - bayesplot theme set to bayesplot::theme_default()
#>   * Does _not_ affect other ggplot2 plots
#>   * See ?bayesplot_theme_set for details on theme setting
```

Mathematical Background

The `groupedBBMH` package provides tools for analyzing group (pooled) sampling data under both perfect and imperfect testing conditions. It supports inference on contamination prevalence, quantifies the risk of undetected contamination (leakage), and facilitates risk-based decision-making through model-based simulations. The package accounts for test imperfections and allows specifying a biologically meaningful minimum positive

prevalence. Both exact and approximate beta-binomial distributions of the group-testing outcomes are implemented to calculate leakage probability and the expected number of contaminated items per batch.

To illustrate the underlying nested model, consider a biosecurity scenario involving the importation of frozen seafood into a country. Suppose there are D consignments imported annually, each consisting of approximately B groups, with each group containing m items — so that the total number of items per batch is $N = mB$.

From each batch, a simple random sample of b groups is selected for group testing in screening procedure. These tests, which screen for pathogens or contaminants, are typically imperfect. We assume a constant sensitivity Δ — the probability that a test correctly detects infection when it is present — and a constant specificity Λ — the probability that a test correctly identifies no infection when it is absent.

For each batch, the only observed outcome is the number of positive groups (t_{yi}) among the b tested groups. The below notations are used to develop the nested beta-binomial models.

Notation

- N : Number of units (e.g., prawns) in a consignment
- B : Number of groups (pools) in a consignment
- m : Group (pool) size, $m = N/B$
- b : Number of groups selected for testing
- $X_{ij\ell}$: Indicator for contamination of unit ℓ in group j of consignment i
- X_{ij} : Number of contaminated units in group j of consignment i , values $0, 1, \dots, m$
- Y_{ij} : Presence of contamination in group j of consignment i , $Y_{ij} = I(X_{ij} > 0)$
- \tilde{Y}_{ij} : Observed test result for group j of consignment i , accounting for measurement error
- T_{Xi} : Total number of contaminated units in consignment i , $T_{Xi} = \sum_{j=1}^B X_{ij}$
- t_{xi} : Number of contaminated units among sampled groups, $t_{xi} = \sum_{j=1}^b X_{ij}$
- t_{yi} : Number of positive groups among sampled groups, $t_{yi} = \sum_{j=1}^b Y_{ij}$
- \tilde{t}_{yi} : Number of positive tests among sampled groups, $\tilde{t}_{yi} = \sum_{j=1}^b \tilde{Y}_{ij}$
- L_i : Leakage in consignment i , defined as $L_i = T_{Xi} I(\tilde{t}_{yi} = 0)$
- k : Minimum biologically plausible positive prevalence

Nested Beta-binomial models

Now let p_i denote the true prevalence of contamination in batch i , and define $\phi_i = 1 - (1 - p_i)^m$ as the probability that a group of m items is contaminated. Assuming items are randomly distributed among bags, we have:

$$X_{ij} \mid p_i \stackrel{\text{i.i.d.}}{\sim} \text{Bin}(m, p_i), \quad p_i \stackrel{\text{i.i.d.}}{\sim} \text{Beta}(\alpha, \beta).$$

The group-level test outcome \tilde{Y}_{ij} follows:

$$\tilde{Y}_{ij} \mid p_i \sim \text{Bernoulli}(\tilde{\phi}_i),$$

and the observed number of positive groups in the sample follows:

$$\tilde{t}_{yi} \mid p_i \sim \text{Binomial}(b, \tilde{\phi}_i).$$

where the effective probability of a positive test is:

$$\tilde{\phi}_i = \Delta\phi_i + (1 - \Lambda)(1 - \phi_i); \quad \phi_i = 1 - (1 - p_i)^m.$$

Under perfect specificity ($\Lambda = 1$), this simplifies to:

$$\tilde{\phi}_i = \Delta\phi_i.$$

When $\beta \gg \alpha$, the contamination prevalence βp_i is approximately Gamma distributed, leading to:

$$\tilde{\phi}_i \sim (1 - \Lambda) + \text{Beta}\left(\alpha, \frac{\beta}{m(\Delta + \Lambda - 1)}\right).$$

Then two important special cases can be considered as:

- **Perfect testing** ($\Delta = 1, \Lambda = 1$): $\tilde{\phi}_i \sim \text{Beta}(\alpha, \beta/m)$,
- **Perfect specificity** ($\Lambda = 1$): $\tilde{\phi}_i \sim \text{Beta}(\alpha, \frac{\beta}{m\Delta})$.

In either case, \tilde{t}_{yi} approximately follows a Beta-Binomial distribution when the infection is rare:

$$\tilde{t}_{yi} \sim \text{Beta-Binomial}\left(b, \alpha, \frac{\beta}{m\Delta}\right).$$

When a regulatory threshold (k) is defined such that contamination levels below a certain prevalence cut-off are considered unlikely, the effective prevalence can then be expressed as:

$$p_i^* = p_i \cdot \mathbb{I}(p_i > k), \quad p_i \sim \text{Beta}(\alpha, \beta).$$

Consideration of the new prevalence in the above-mentioned **BB model** leads to **truncated BB model**. Using the truncated model, the **probability of leakage** can be expressed as:

$$\Pr[L_i > 0] = \frac{B_{(k_1,1)}(\alpha, \frac{\beta}{m\Delta} + b)}{B(\alpha, \frac{\beta}{m\Delta})} - \frac{B_{(k,1)}(\alpha, \beta + MB)}{B(\alpha, \beta)},$$

with $k_1 = \Delta(1 - (1 - k)^m)$. Under perfect testing:

$$\Pr[L_i > 0] = \frac{B_{(k,1)}(\alpha, \beta + mb)}{B(\alpha, \beta)} - \frac{B_{(k,1)}(\alpha, \beta + MB)}{B(\alpha, \beta)}.$$

The **expected leakage** under threshold k is:

$$\mathbb{E}(L_i) = (B - b)M \cdot \mathbb{E}[(1 - p_i)^{bm\Delta} \cdot p_i \cdot \mathbb{I}(p_i > k)],$$

which simplifies to:

$$\mathbb{E}(L_i) = (B - b)M \cdot \frac{B_{(k,1)}(\alpha + 1, \beta + bm\Delta)}{B(\alpha, \beta)}.$$

When $k = 0$, these derivations reflect cases for the standard beta-binomial case.

Functions Overview

The `groupedBBMH` package provides a set of functions for fitting, simulating, and evaluating grouped beta-binomial models for group-testing data. Key functions include:

- `loglik_group_bb`: Log-likelihood for grouped BB model with test error
- `loglik_group_trbb`: Log-likelihood for truncated (minimum positive prevalence) grouped BB model with test error
- `fit_GroupedBB`: Fit grouped BB model using MLE
- `fit_trGroupedBB`: Fit truncated grouped BB model using MLE
- `MH_Sampler_BB`: Metropolis-Hastings sampler for fitting grouped (truncated) BB models with test errors
- `summary_mcmc`: Summarize MCMC outcomes with diagnostics
- `create_mcmc_BP`: Create MCMC array for visualization using `bayesplot` R package

- `DunnSmythTestBB`: Dunn-Smyth residual diagnostics for BB models
- `estimate_leakage`: Estimate expected leakage and probability of leakage
- `post_pred_sn`: Posterior predictive simulation for grouped BB model with imperfect sensitivity
- `ppc_barplot`: Posterior predictive check (PPC) through creating barplots for observed and simulated counts and proportions
- `post_pred_aprox_model`: Posterior predictive simulation for grouped BB models under perfect testing
- `simulate_infection_sampling`: Simulate infection sampling with (truncated) grouped BB models accounting sensitivity and minimum prevalence adjustments
- `run_infection_simulations`: Run the simulation of infection sampling using `simulate_infection_sampling` function
- `compute_simulated_leakage`: Compute expected leakage and probability of leakage from model-based simulation output

These functions allow users to fit exact models by incorporating minimum prevalence thresholds, test imperfections, and perform simulation-based evaluation and visualization of group-testing data.

Correspondence to Equations in Das et al. (2025) paper

The following equations from the Barnes et al. (2025) form the statistical foundation of the `groupedHG` package:

- **Equation (1) & (2)**: Two-level BB model for X_{ij}
- **Equation (3), (4) & (5)**: Two-level BB model for \tilde{t}_{yi} under imperfect testing
- **Equation (6)**: PMF of \tilde{t}_{yi} under imperfect testing
- **Equation (12)**: Approximate distribution \tilde{t}_{yi} under imperfect testing
- **Equation (12)**: Approximate distribution \tilde{t}_{yi} under imperfect sensitivity
- **Equation (16)**: Exact distribution \tilde{t}_{yi} under imperfect testing
- **Equation (17)**: Exact $\Pr[L_i > 0]$ under perfect testing
- **Equation (18)**: Exact $E(L_i)$ under perfect testing
- **Equation (19)**: Approximate $\Pr[L_i > 0]$ under imperfect sensitivity
- **Equation (20)**: Approximate $E(L_i)$ under imperfect sensitivity

Case Study: Frozen Seafood Importation in Australia

A real Australian biosecurity data extracted from Frozen Seafood Importation motivate our modeling approach. Each year, approximately 800 consignments (batches) of frozen seafood are imported into Australia. Each batch contains roughly $B = 8000$ bags, with each bag holding $M = 40$ items, yielding a total of $N = MB$ items per batch.

A simple random sample of $b = 13$ bags is selected from each batch for testing. From each selected bag, a group of $m = 5$ items (out of $M = 40$) is randomly chosen, resulting in $n = bm = 65$ sampled items per batch. Each group of m items is tested using a PCR assay, which is assumed to have a sensitivity of approximately $\Delta = 0.80$ and near-perfect specificity, $\Lambda = 1.0$.

The only observation recorded per batch is the number of groups testing positive, out of the $b = 13$ groups tested.

These data motivate the development of models for estimating contamination prevalence, quantifying the risk of undetected contamination (leakage), and evaluating testing strategies under imperfect diagnostic accuracy.

The data used in this package for illustration and reproducibility are publicly available at [Zenodo](#).

For illustration, the frequency distribution of the number of positive test outcomes per batch, \tilde{t}_{yi} , is summarized as $t_y = (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13)$ and $f_y = (2815, 9, 10, 6, 1, 3, 2, 0, 1, 2, 1, 0, 0, 0)$.

The distribution shows that approximately 99% of the batches yielded no positive test results. However, when a batch tested positive, additional positive outcomes within the same batch were more likely to occur. This pattern highlights the sparsity of the data and suggests that, once present, the infection tends to occur at or above a biologically meaningful minimum prevalence level, rather than at very low non-zero values.

Basic Usage of R functions

This vignette illustrates how to perform maximum likelihood estimation (MLE) and MH algorithm for estimating parameters of a grouped Beta-Binomial model. The illustrations are conducted using the frozen seafood data.

Fitting BB model using MLE

For the given data, we first want to estimate parameters of a grouped Beta-Binomial (BB) model under a perfect testing scenario. This is done by maximizing the log-likelihood function that accounts for the grouped structure of the data. The likelihood function, implemented in the `loglik_group_bb()` function, can incorporate diagnostic test sensitivity and specificity. The maximization is carried out through the `optim()` function in R.

The function can be flexibly parameterized to estimate the shape parameters of the Beta distribution, α and β , given known sensitivity and specificity, or to estimate one parameter (such as α) when another (such as μ) is

known. In more general settings, it can be extended to jointly estimate both Beta distribution parameters along with test inaccuracy parameters, depending on the available data and assumptions.

In the first example, we estimate the parameters of the grouped BB model assuming a perfect test (i.e., sensitivity and specificity both equal to 1). The optimization relies on numerical integration using $R = 10^4$ random draws from the Beta distribution. The optimization is performed on the log-scale to ensure positivity of parameters, and the resulting estimates for α and β are obtained by exponentiating the optimized parameter values.

```
ty=c(0:13)
freq=c(2815, 9, 10, 6, 1, 3, 2, 0, 1, 2, 1, 0, 0, 0)
MLE.BB.m1 <- optim(c(0,0), loglik_group_bb, ty=ty, freq=freq, b=13,
                     theta=Inf, m=5, M=40, deviance=FALSE,
                     control=list(reltol=1e-12, fnscale=-1),
                     G=1e4, hessian=FALSE, sensitivity=1, specificity=1)
MLE.BB.m1$mu <- exp(MLE.BB.m1$par[1]) / sum(exp(MLE.BB.m1$par))
MLE.BB.m1$alpha <- exp(MLE.BB.m1$par[1])
MLE.BB.m1$beta <- exp(MLE.BB.m1$par[2])
MLE.BB.m1
#> $par
#> [1] -5.273472  1.929189
#>
#> $value
#> [1] -429.0803
#>
#> $counts
#> function gradient
#>      127       NA
#>
#> $convergence
#> [1] 0
#>
#> $message
#> NULL
#>
#> $mu
#> [1] 0.0007440476
#>
```

```

#> $alpha
#> [1] 0.005125782
#>
#> $beta
#> [1] 6.883926

```

In the second example, we examine whether all the beta parameters and test parameters can be estimate by maximizing the log-likelihood function. The optimization requires specification of lower and upper values for the targeted parameters. The optimization shows that maximum value of log-likelihood value is obtained for $\Delta = 0.74$ and $\Lambda = 1$, when $\alpha = 0.0045$ and $\beta = 3.78$.

In the second example, we fit the model to estimate all parameters simultaneously — that is, both Beta distribution parameters (α and β) and test inaccuracy parameters (Δ and Λ). This general case allows us to evaluate how test imperfections affect the estimated prevalence distribution.

To ensure stable optimization, lower and upper bounds are specified for all parameters. The optimization process searches the parameter space within these limits to find the combination that maximizes the log-likelihood function. The results demonstrate that the maximum log-likelihood occurs when $\Delta = 0.74$ and $\Lambda = 1$, corresponding to $\alpha = 0.0045$ and $\beta = 3.78$. These results indicate moderate overdispersion in the underlying Beta-Binomial model and the test has imperfect sensitivity but near-perfect specificity.

```

MLE.BB.m6 <- optim(
  c(0,0,0,0), loglik_group_bb,
  ty = ty, freq = freq, b = 13,
  theta = Inf, m = 5, M = 40,
  deviance = FALSE,
  control = list(factr = 1e-12, fnscale = -1, trace = TRUE),
  G=1e4, hessian = FALSE,
  method = "L-BFGS-B",
  lower = c(log(0.0001), log(0.0001), log(0.5), log(0.99)),
  upper = c(Inf, Inf, 0, 0)
)
#> iter   10 value 669.843580
#> iter   20 value 429.083634
#> iter   30 value 429.018662
#> iter   40 value 428.595132
#> iter   50 value 428.585925
#> iter   60 value 428.585924

```

```
#> final value 428.585924
#> converged

MLE.BB.m6$mu <- exp(MLE.BB.m6$par[1]) / sum(exp(MLE.BB.m6$par))
MLE.BB.m6$alpha <- exp(MLE.BB.m6$par[1])
MLE.BB.m6$beta <- exp(MLE.BB.m6$par[2])
MLE.BB.m6$sensitivity <- exp(MLE.BB.m6$par[3])
MLE.BB.m6$specificity <- exp(MLE.BB.m6$par[4])
MLE.BB.m6
#> $par
#> [1] -5.3965720  1.3304946 -0.2961164  0.0000000
#>
#> $value
#> [1] -428.5859
#>
#> $counts
#> function gradient
#>      67      67
#>
#> $convergence
#> [1] 0
#>
#> $message
#> [1] "CONVERGENCE: REL_REDUCTION_OF_F <= FACTR*EPSMCH"
#>
#> $mu
#> [1] 0.0008193762
#>
#> $alpha
#> [1] 0.00453209
#>
#> $beta
#> [1] 3.782914
#>
#> $sensitivity
#> [1] 0.7437009
#>
```

```
#> $specificity  
#> [1] 1
```

Overall, these two examples illustrate how the grouped Beta-Binomial likelihood framework, implemented through the `loglik_group_bb()` function, can flexibly accommodate a range of scenarios — from perfect testing conditions to fully parameterized cases that account for test inaccuracy. This approach provides a robust foundation for modeling overdispersed group testing data in the presence of measurement error.

As a user-friendly alternative to manual optimization, the `fit_GroupedBB()` function offers a convenient interface for fitting the grouped Beta-Binomial model. For a given set of sensitivity and specificity values, it estimates the Beta-Binomial parameters by maximizing the likelihood function. The function returns the estimated Beta distribution parameters and, when `leakage = TRUE` is specified, also provides estimates of leakage parameters - expected leakage and probability of leakage per batch. It is noted that for leakage estimation, the number of groups per batch B and the actual group size in the cases study (M) is required.

```
bb.m1 <- fit_GroupedBB(ty = ty, freq = freq, b = 13, m = 5,  
                        sensitivity = 1, specificity = 1)  
bb.m1  
#> $alpha  
#> [1] 0.005125782  
#>  
#> $beta  
#> [1] 6.883926  
#>  
#> $mu  
#> [1] 0.0007440476  
#>  
#> $sensitivity  
#> [1] 1  
#>  
#> $specificity  
#> [1] 1  
#>  
#> $loglik  
#> [1] -429.0803  
#>
```

```
#> $convergence
#> [1] 0

# Fit model with Leakage (requires B and M)
bb.m2 <- fit_GroupedBB(ty = ty, freq = freq, b = 13, m = 5, M = 40, B = 8000,
                        sensitivity = 1, specificity = 1, leakage = TRUE)
bb.m2
#> $alpha
#> [1] 0.005125782
#>
#> $beta
#> [1] 6.883926
#>
#> $mu
#> [1] 0.0007440476
#>
#> $sensitivity
#> [1] 1
#>
#> $specificity
#> [1] 1
#>
#> $loglik
#> [1] -429.0803
#>
#> $convergence
#> [1] 0
#>
#> $P.L
#>      P.L
#> 0.04014339
#>
#> $E.L
#>      E.L
#> 22.49933
```

When a biologically meaningful threshold is introduced for the batch-level infection prevalence (p_i), the model must account for the fact that values below this threshold are biologically implausible. The `groupedBBMH` package accommodates this feature through the truncated beta-binomial model, where the log-likelihood is maximized using the function `loglik_group_trbb()`.

To simplify implementation, the high-level function `fit_trGroupedBB()` provides a user-friendly interface for estimating the beta distribution parameters while incorporating a specified threshold value k .

In the first example below, we fit the truncated beta-binomial model with a threshold of $k = 0$ (i.e., no truncation) under perfect test conditions using `fit_trGroupedBB()` with $R = 10^4$ for numerical integration. The results are comparable to those obtained using the standard `fit_GroupedBB()` function.

In the second example, we demonstrate a model fit with a 1% minimum prevalence threshold ($k = 0.01$) under imperfect test conditions, where sensitivity ($\Delta = 0.8$) and specificity ($\Lambda = 1$). This allows us to evaluate how the model responds when small non-zero prevalence values are restricted by biological constraints.

```
# Fit truncated grouped beta-binomial model with cutoff = 0 (no truncation) with perfect test
trbb.m1 <- fit_trGroupedBB(ty, freq, b = 13, m = 5, cutoff = 0,G=1e4)
trbb.m1
#> $alpha
#> [1] 0.005125782
#>
#> $beta
#> [1] 6.883926
#>
#> $sensitivity
#> [1] 1
#>
#> $specificity
#> [1] 1
#>
#> $cutoff
#> [1] 0
#>
#> $loglik
#> [1] -429.0803
#>
#> $convergence
#> [1] 0
```

```

# Fit truncated grouped beta-binomial model with cutoff = 0.01 and imperfect sensitivity
trbb.m2 <- fit_trGroupedBB(ty, freq, b = 13, m = 5, cutoff = 0.01,
                           sensitivity = 0.8, specificity = 1)

trbb.m2
#> $alpha
#> [1] 0.006507405
#>
#> $beta
#> [1] 5.959846
#>
#> $sensitivity
#> [1] 0.8
#>
#> $specificity
#> [1] 1
#>
#> $cutoff
#> [1] 0.01
#>
#> $loglik
#> [1] -427.6794
#>
#> $convergence
#> [1] 0

```

Fitting BB model using MH

Now, we fit the grouped beta-binomial (BB) model using the Metropolis–Hastings (MH) algorithm. The `MH_Sampler_BB()` function implements the MH sampling procedure for specified values of test sensitivity and specificity. Following the setup described in the paper, two prior specifications were considered for the test sensitivity: one with fixed $\Delta = 0.8$, and another where Δ was uniformly distributed between 0.75 and 0.85.

The MH algorithm is constructed based on the frequency distribution of \tilde{t}_{yi} , allowing the weighted log-likelihood of the beta-binomial model to be computed exactly through numerical integration. The priors for the model parameters (e.g., α and μ) were chosen to ensure that parameter estimates remain within expected, interpretable ranges.

The MH algorithm was run on the parameter set θ_1 , θ_2 , and, when Δ was not fixed, θ_3 . Specifically, we defined $\theta_1 = \log(\alpha)$ with prior $\theta_1 \sim U(-A, A)$, and $\mu = \text{expit}(\theta_2)$. When Δ was not fixed, it was parameterized as $\Delta = 0.75 + 0.1 \times \text{expit}(\theta_3)$. These definitions yield induced priors of $\mu \sim U(0, 1)$ and $\Delta \sim U(0.75, 0.85)$.

For proposal distributions, each parameter $(\theta_1, \theta_2, \theta_3)$ was updated as the previous value plus a normal perturbation with mean zero. A proposal standard deviation of $\sigma = 0.2$ provided satisfactory convergence, with low autocorrelation and stable parameter estimates. The model was fitted using 5×10^4 iterations, with the first half discarded as burn-in. From the remaining samples, every fifth draw (`thin = 5`) was retained to reduce autocorrelation in the MCMC chains.

Convergence was evaluated using trace plots and posterior density plots. For demonstration purposes, an example model was also fitted using a smaller number of iterations ($R = 1000$) to illustrate the structure of the MCMC output. Here, R denotes the number of MCMC iterations, while G represents the number of quantiles used in the numerical integration.

We fit the first model using the R code shown below for the fixed sensitivity $\Delta = 0.8$ and examined the summary statistics of the target parameters of the beta distribution. These summaries provide insight into the posterior location and variability of the parameters, helping to assess the model's fit and stability. Additionally, the estimated posterior means and credible intervals can be compared with the maximum likelihood estimates obtained earlier to evaluate the consistency and robustness of the inference.

```
MH.trbb.m1 <- MH_Sampler_BB(par = c(log(0.005), logit(7e-04), logit(0.75)),
cutoff = 0.01,
alpha = TRUE, beta = FALSE, mu = TRUE, rho = FALSE,
G = 500, R = 1000,
sigma = c(0.20, 0.20, 0.20),
num_chains = 3,
burnin = 500, thin = 2,
seed = c(123, 456, 789, 102),
trail = 13,
ty = ty,
wt = freq,
group.size = 5,
sensitivity = TRUE,
specificity = FALSE,
sensitivity.range=c(0.80,0.80),
specificity.range=NULL)
```

Summary statistics of the target parameters, obtained using the `summary_mcmc()` function, provide a comprehensive overview of the posterior distributions, including measures such as the mean, standard deviation, and credible intervals. In addition, the function reports key MCMC diagnostics — such as effective sample size and potential scale reduction factor (R-hat) — which help assess convergence and sampling efficiency. Together, these outputs offer valuable insights into the stability and reliability of the parameter estimates derived from the Metropolis–Hastings algorithm.

```
summ.MH.m1 <- summary_mcmc(MH.trbb.m1$target.parameters$alpha_sample,
                             MH.trbb.m1$target.parameters$beta_sample,
                             MH.trbb.m1$target.parameters$mu_sample,
                             varnames = c("alpha", "beta", "mu"))

(summ.MH.m1$summary_mcmc)
#>      Mean       SD    2.5%     25%     50%     75%
#> alpha 0.005741119 0.0013389398 0.003707111 0.004754688 0.005614421 0.006521045
#> beta  3.510348544 1.1512829094 1.717832592 2.632619276 3.348240220 4.205114082
#> mu    0.001713915 0.0003655358 0.001169679 0.001448233 0.001672446 0.001912894
#>      97.5%   R_hat   n_eff n_eff_ratio
#> alpha 0.008866588 1.011879 129.8161  0.1730881
#> beta  6.040998230 1.025535 152.7556  0.2036742
#> mu    0.002629871 1.073512 167.6818  0.2235757
```

We now fit the second MH-based BB model using the R code below, assuming the sensitivity parameter follows a uniform distribution on $(0.75, 0.85)$. In this setup, we set a proposal standard deviation of $\sigma = 2$ for the sensitivity parameter, while retaining $\sigma = 0.2$ for the α and μ parameters.

```
MH.trbb.m2 <- MH_Sampler_BB(par = c(log(0.005), logit(7e-04), logit(0.75)),
                               cutoff = 0.01,
                               alpha = TRUE, beta = FALSE, mu = TRUE, rho = FALSE,
                               G = 500, R = 1000,
                               sigma = c(0.20, 0.20, 2),
                               num_chains = 3,
                               burnin = 500, thin = 2,
                               seed = c(123, 456, 789, 102),
                               trail = 13,
                               ty = ty,
                               wt = freq,
                               group.size = 5,
```

```

sensitivity = TRUE,
specificity = FALSE,
sensitivity.range=c(0.75,0.85),
specificity.range=NULL)

```

The `summary_mcmc` function provides summary statistics for the MCMC simulations of all targeted parameters, including α , μ , and Δ .

```

summ.MH.m2 <- summary_mcmc(MH.trbb.m2$target.parameters$alpha_sample,
                             MH.trbb.m2$target.parameters$beta_sample,
                             MH.trbb.m2$target.parameters$sensitivity_sample,
                             varnames = c("alpha", "beta", "sensitivity"))

(summ.MH.m2$summary_mcmc)

#>           Mean        SD      2.5%     25%      50%
#> alpha    0.005829232 0.001364334 0.003646279 0.004803597 0.005712284
#> beta     3.546968189 1.135368527 1.784829232 2.696157674 3.384244291
#> sensitivity 0.800936712 0.029296563 0.753141364 0.774247234 0.801606720
#>           75%     97.5%   R_hat   n_eff n_eff_ratio
#> alpha    0.006751844 0.00862891 1.027757 78.66549  0.1048873
#> beta     4.302207121 5.98811544 1.100122 102.27293  0.1363639
#> sensitivity 0.826630004 0.84794184 1.052974 190.36517  0.2538202

```

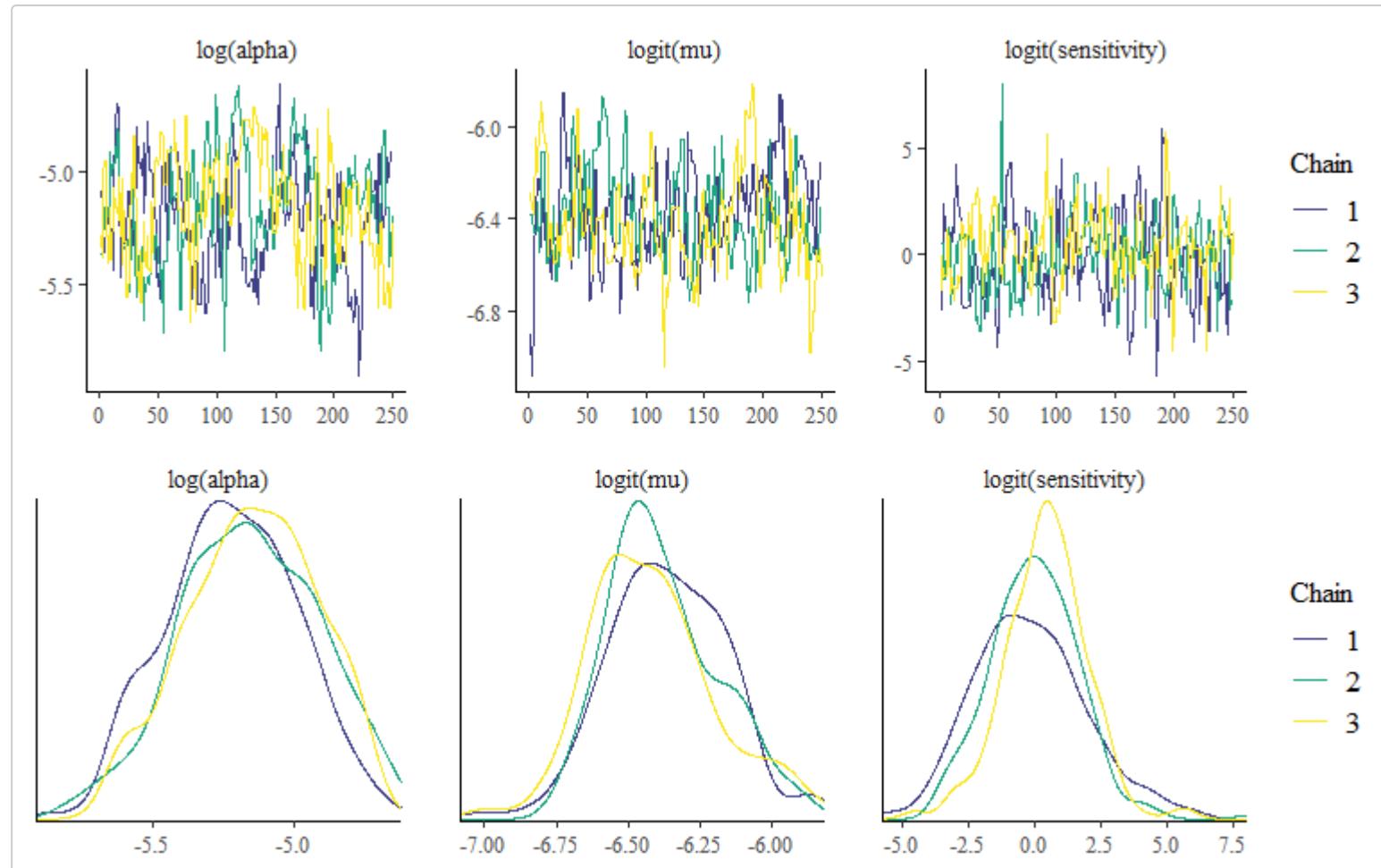
MCMC diagnostics can be conveniently performed using the `bayesplot` R package, which provides a rich set of visualization tools for assessing convergence, autocorrelation, and distribution of MCMC samples. To use `bayesplot`, the MCMC simulations must first be converted into a format compatible with the package. The `create_mcmc_BP()` function in `groupedBBMH` facilitates this conversion by transforming the MCMC output into a three-dimensional array of shape (iterations \times chains \times parameters). Once the array is prepared, `bayesplot` functions can be applied to visualize the posterior distributions, trace plots, and other diagnostics for the transformed parameters across different chains. The R code below illustrates how to convert the MCMC output and generate diagnostic plots to assess convergence and parameter behavior across chains.

```

MCMC.MH.m2 <- lapply(MH.trbb.m2$parameters, function(mat) mat[, 1:3])
var.names <- c("log(alpha)", "logit(mu)", "logit(sensitivity)")
MCMC.MH.m2 <- create_mcmc_BP(MCMC.MH.m2, var.names)
color_scheme_set("viridis")
p1_mcmc_trace <- bayesplot::mcmc_trace(MCMC.MH.m2) # Density overlay for known sensitivity

```

```
p2_dens_overlay <- bayesplot::mcmc_dens_overlay(MCMC.MH.m2) # Density overlay for unknown sensitivity  
ggarrange(p1_mcmc_trace,p2_dens_overlay,nrow = 2)
```



The trace plots and overlay density plots provide visual diagnostics of MCMC convergence, allowing us to assess whether the chains have mixed well and stabilized around the posterior distributions for the considered parameters under varnames.

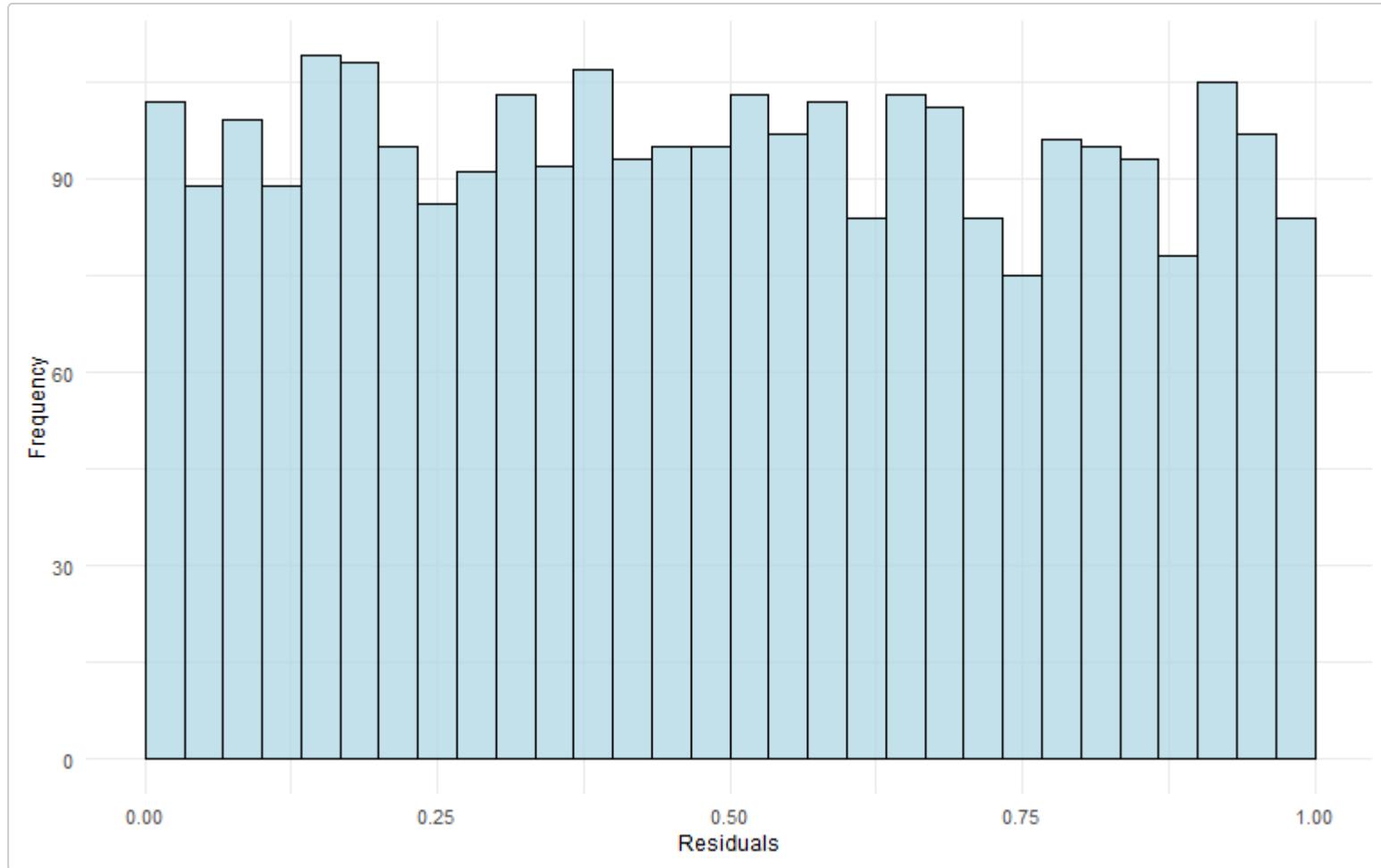
Model Checking

For illustration, we evaluated the MH-fitted BB model with fixed sensitivity $\Delta = 0.8$ using Dunn-Smyth randomized quantile residuals (Dunn and Smyth, 1996) and posterior predictive checks against the observed frequency data. Although similar diagnostics could be applied to ML- or HMC-fitted models, or for positive thresholds k , we focus here on the MH fit with $k = 0$ for simplicity. Posterior uncertainty is incorporated in the Dunn-Smyth residuals through the empirical cumulative distribution of simulated counts, and the residuals are subsequently transformed to a standard normal scale for Q-Q plot evaluation. Further details are provided in Das et al. (2025).

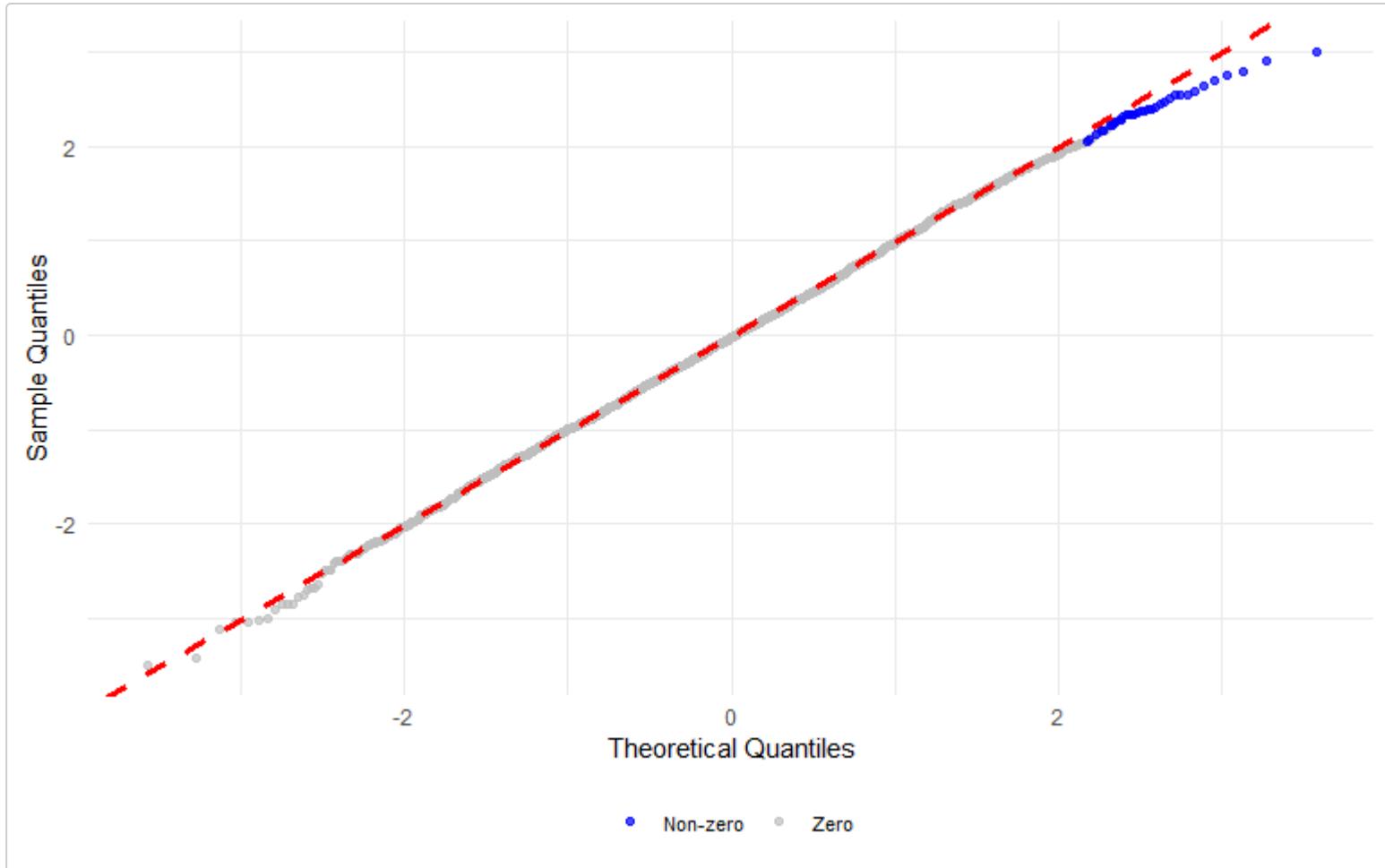
```
# Expand frequency table into a vector of observed responses
ObservedResponse <- rep(ty, times = freq)

# Run Dunn-Smyth residual diagnostic using posterior samples
set.seed(123456)
DSTest_MH_Imperfect <- DunnSmythTestBB_Custom(
  ObservedResponse = ObservedResponse,
  size = 13,
  group_size = 5,
  approximate.model = FALSE,
  alpha = unlist(MH.trbb.m1$target.parameters$alpha_sample),
  beta = unlist(MH.trbb.m1$target.parameters$beta_sample)
)
#> Warning in ks.test.default(DS, "punif"): ties should not be present for the
#> one-sample Kolmogorov-Smirnov test

# Display diagnostic plots
DSTest_MH_Imperfect$DS_hist      # Histogram of Dunn-Smyth residuals
```



```
DSTest_MH_Imperfect$DS_QQ_norm    # Normal Q-Q plot of scaled residuals
```



```
# Print KS test result for Uniformity
DSTest_MH_Imperfect$KS
#>
#> Asymptotic one-sample Kolmogorov-Smirnov test
#>
#> data: DS
#> D = 0.018369, p-value = 0.2914
#> alternative hypothesis: two-sided
```

For doing Posterior predictive check of the fitted model, we use the posterior distribution of α and β obtained from the fitted model `MH.trbb.m1` considering $k = 0.01$ and $\Delta = 0.8$. The comparison of counts and their proportions from the observed outcomes are compared with simulated counts. In the below two bar-plots, the shaded areas represent the observed counts (proportions), while the error bars indicate the 95% credible interval of simulated counts (proportions). For the plot of counts, the left y-axis is scaled from 0 to 3000 for zero value of \tilde{t}_{yi} , while the right y-axis is scaled from 0 to 30 for non-zero values of \tilde{t}_{yi} .

```

# Observed outcomes from D batches
ObservedResponse <- rep(ty, times = freq)

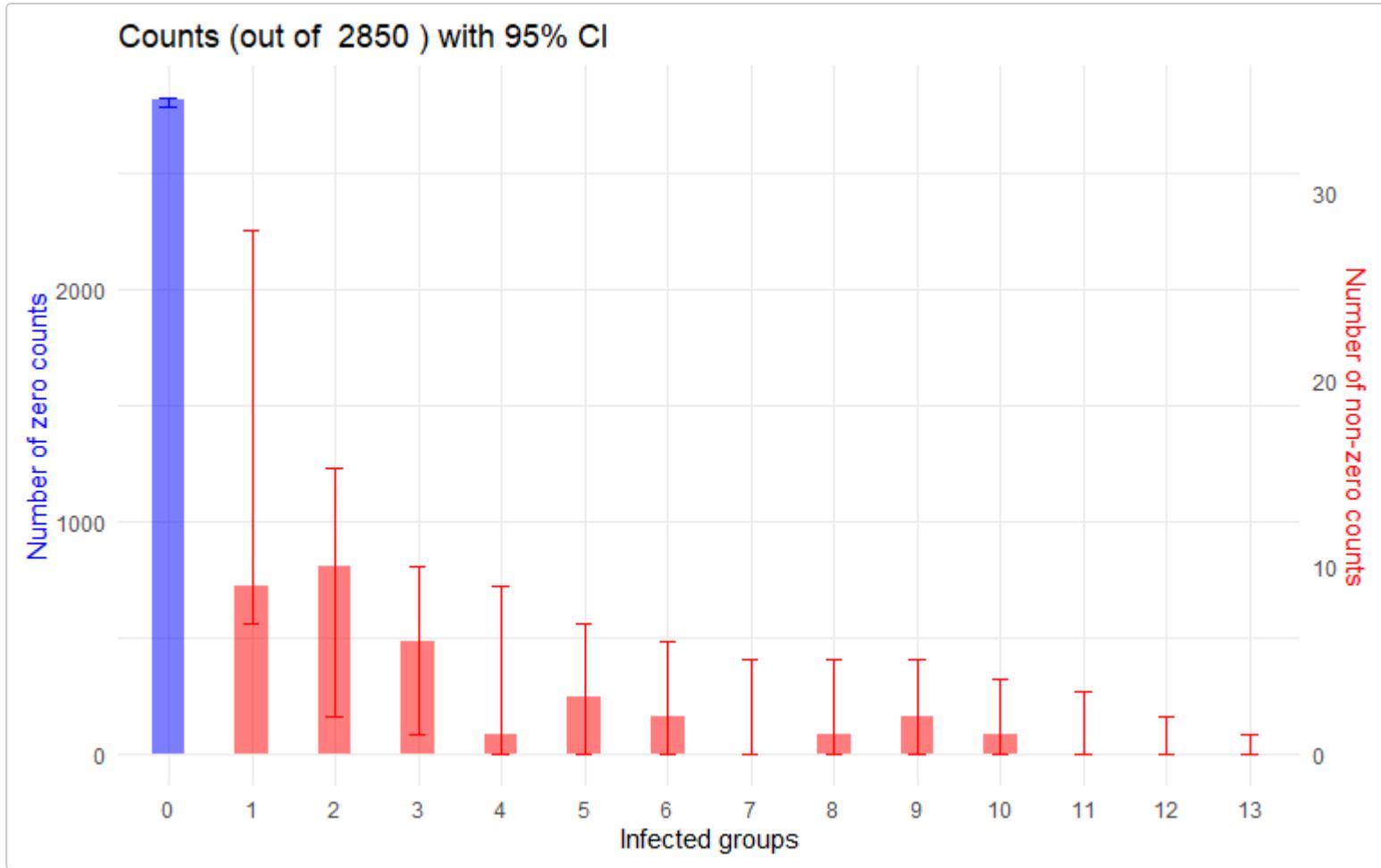
# Generate posterior predictive draws using the 'post_pred_sn' function
predicted.counts.imperfect.case <- post_pred_sn(
  D = length(ObservedResponse),    # Number of batches (observed outcomes)
  n = 13,                         # Number of group tests per batch
  m = 5,                           # Pool size in group test
  alpha = unlist(MH.trbb.m1$target.parameters$alpha_sample),
  beta = unlist(MH.trbb.m1$target.parameters$beta_sample),
  sensitivity = 0.8
)

# Check proportions of observed vs simulated counts (for model diagnostics)
prop.table(table(ObservedResponse))                                # Observed data distribution
#> ObservedResponse
#>      0          1          2          3          4          5
#> 0.9877192982 0.0031578947 0.0035087719 0.0021052632 0.0003508772 0.0010526316
#>      6          8          9         10
#> 0.0007017544 0.0003508772 0.0007017544 0.0003508772
prop.table(table(predicted.counts.imperfect.case$ty_rep))    # Simulated data distribution
#>
#>      0          1          2          3          4          5
#> 9.835312e-01 5.549474e-03 2.743392e-03 1.799766e-03 1.352982e-03 1.099883e-03
#>      6          7          8          9         10         11
#> 8.902924e-04 7.738012e-04 6.877193e-04 5.852632e-04 4.968421e-04 3.026901e-04
#>      12         13
#> 1.501754e-04 3.649123e-05

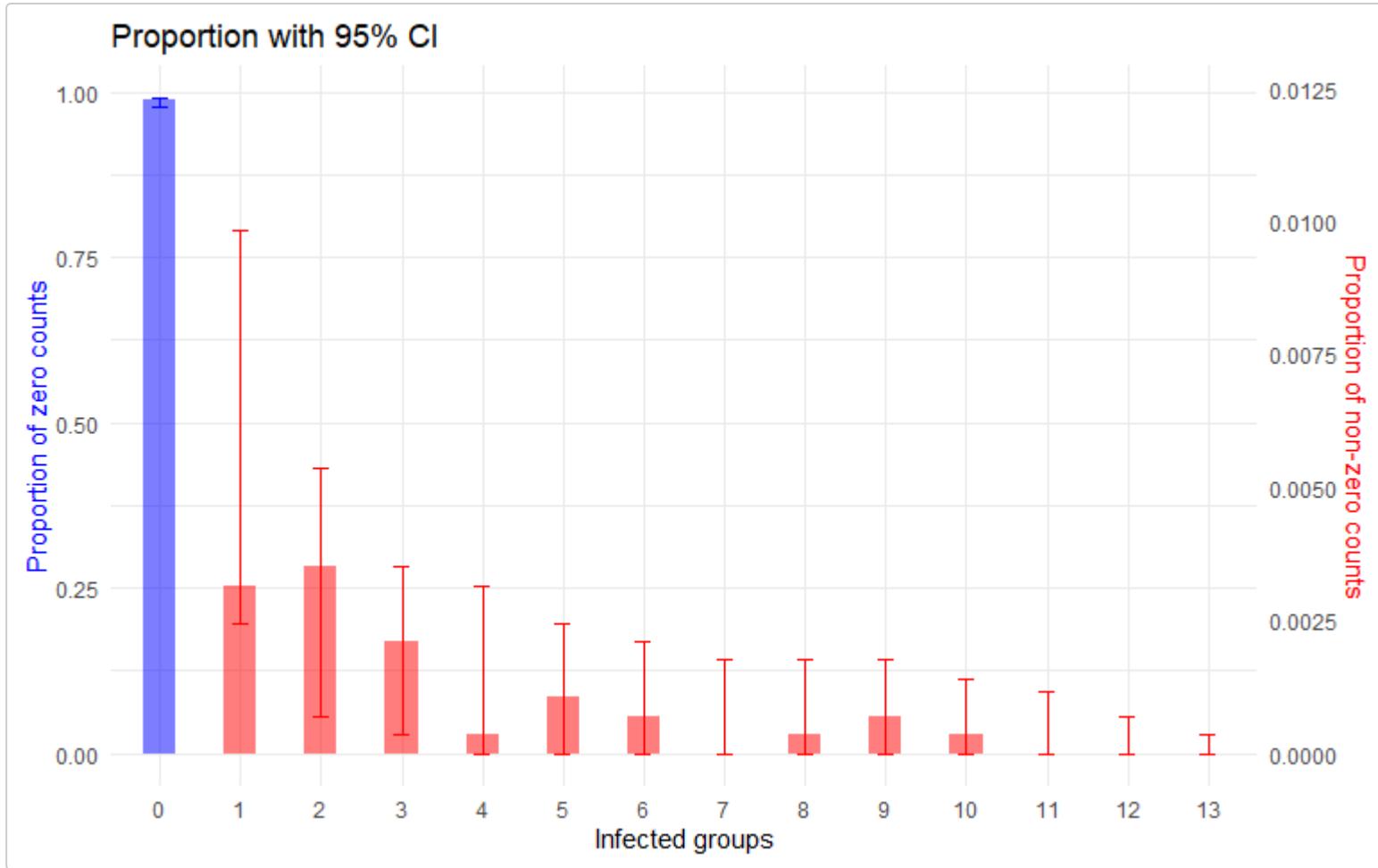
```

```
# -----#
# Posterior Predictive Check Visualization
# -----#

# Generate barplots comparing observed and replicated counts
ppc.barplot.imperfect.case <- ppc_barplot(
  ty = ObservedResponse,
  yrep = predicted.counts.imperfect.case$ty_rep,
  trial = 13
)
ppc.barplot.imperfect.case$p.zero.non.zero      # Combined plot
```



```
ppc.barplot.imperfect.case$prop.zero.non.zero # Zero vs non-zero proportion values
```



Leakage Calculation

Using the derived formulas presented in **Equations (17)–(20)** of the manuscript, the leakage quantities $E(L_i)$ and $\Pr(L_i > 0)$ can be estimated from the posterior distributions of the model parameters α and β . These quantities are computed for specified values of the minimum positive propensity (k) and test sensitivity (Δ), which together determine the expected rate and probability of undetected infection (leakage) in consignments.

For example, the posterior samples obtained from the fitted model `MH.trbb.m1` can be used to estimate leakage parameters via the function `estimate_leakage()`. This function integrates over the posterior distribution to produce summaries of both the **expected leakage** — the average number of infected items that escape detection per

consignment — and the **probability of leakage**, representing the likelihood that at least one infected item remains undetected.

Using the Bayesian algorithm allows us to explore the full posterior distribution of the leakage parameters, rather than relying only on point estimates. This approach captures the uncertainty around both the expected leakage, $E(L_i)$, and the probability of leakage, $\Pr(L_i > 0)$, providing a richer understanding of potential biosecurity risks.

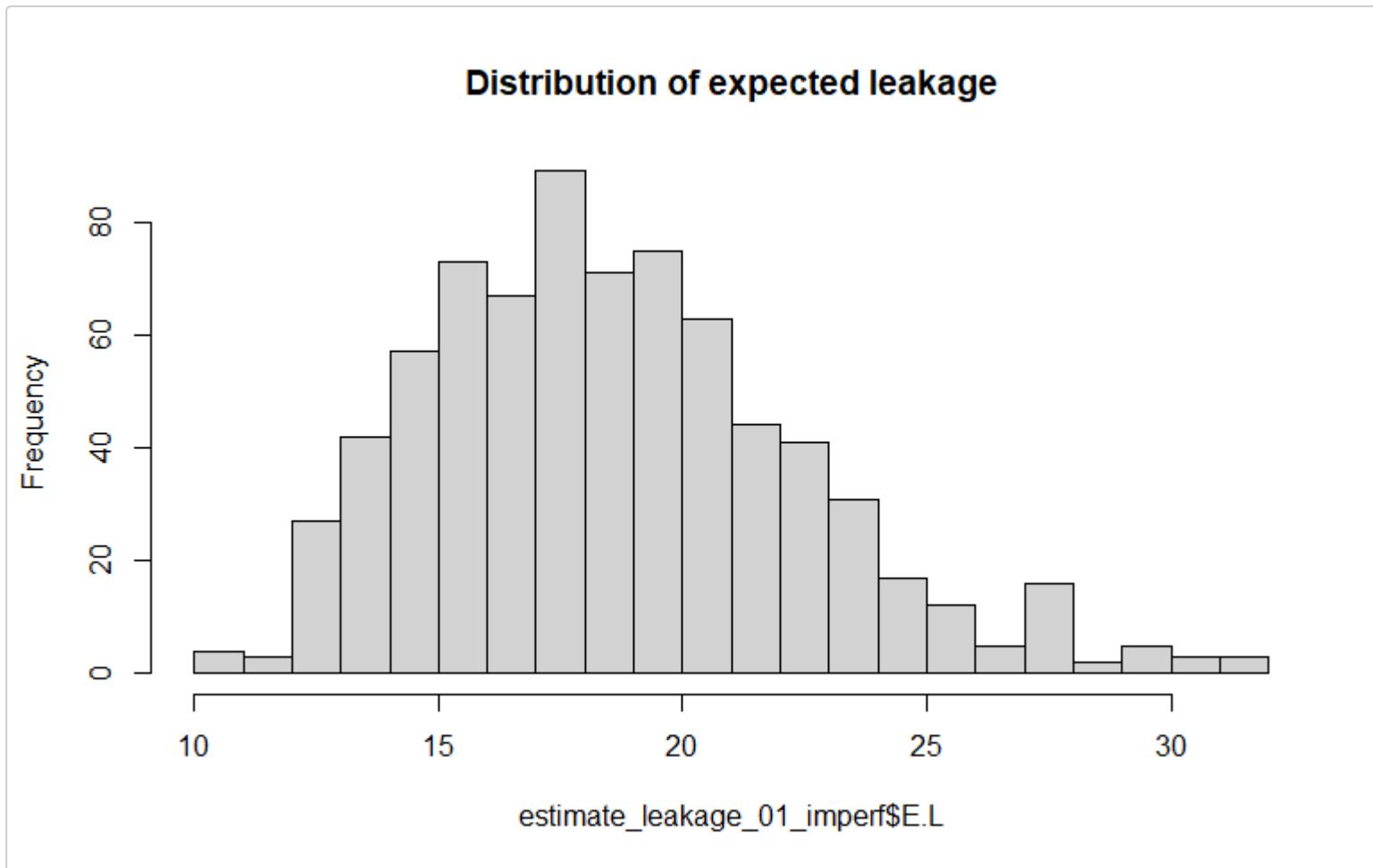
In the below example, the function `estimate_leakage()` is applied using posterior samples of α and β from the model `MH.trbb.m1`. Two scenarios are compared: one assuming perfect test sensitivity $\Delta = 1$ and another assuming imperfect sensitivity $\Delta = 0.8$. The summaries below present the posterior estimates for both the expected leakage and the probability of leakage under each scenario.

```
estimate_leakage_01_perf <- estimate_leakage(B=8000,b=13,M=40,m=5,
                                              alpha=unlist(MH.trbb.m1$target.parameters$alpha_sample),
                                              beta=unlist(MH.trbb.m1$target.parameters$beta_sample),
                                              cutoff = 0.01,
                                              sensitivity = 1)
estimate_leakage_01_imperf <- estimate_leakage(B=8000,b=13,M=40,m=5,
                                                alpha=unlist(MH.trbb.m1$target.parameters$alpha_sample),
                                                beta=unlist(MH.trbb.m1$target.parameters$beta_sample),
                                                cutoff = 0.01,
                                                sensitivity = 0.8)
summary(estimate_leakage_01_perf$E.L)
#>   Min. 1st Qu. Median Mean 3rd Qu. Max.
#> 7.125 11.176 12.957 13.216 14.858 22.428
summary(estimate_leakage_01_perf$P.L)
#>   Min. 1st Qu. Median Mean 3rd Qu. Max.
#> 0.001157 0.001830 0.002130 0.002175 0.002450 0.003727
summary(estimate_leakage_01_imperf$E.L)
#>   Min. 1st Qu. Median Mean 3rd Qu. Max.
#> 10.11 15.76 18.25 18.59 20.88 31.35
summary(estimate_leakage_01_imperf$P.L)
#>   Min. 1st Qu. Median Mean 3rd Qu. Max.
#> 0.001556 0.002459 0.002860 0.002919 0.003287 0.004985
```

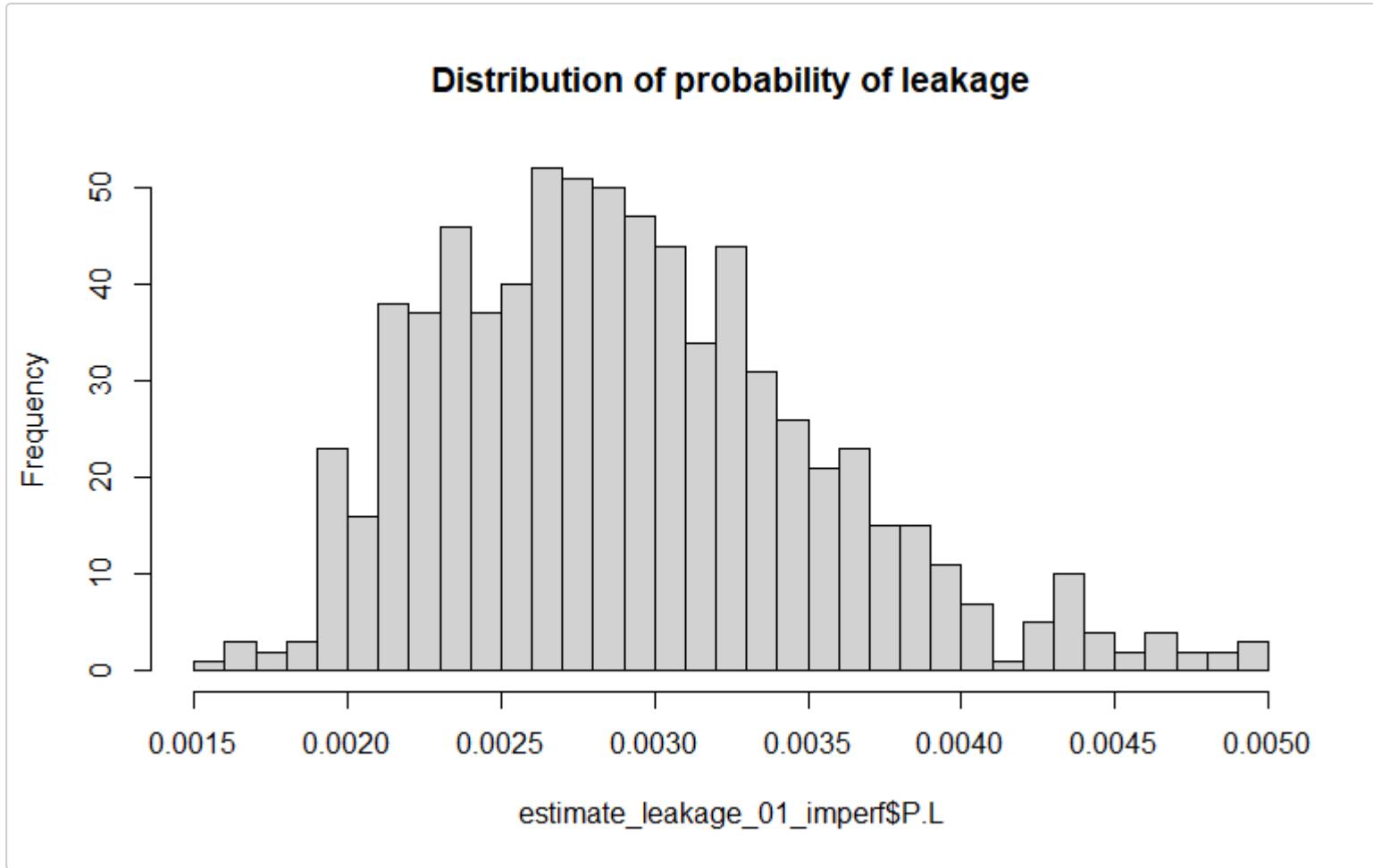
The following histograms illustrate the posterior distributions of the leakage quantities under imperfect sensitivity. These distributions highlight the variability in the model-based leakage estimates, demonstrating how imperfect

detection increases both the mean and uncertainty of leakage outcomes. Such results emphasize the importance of accounting for test performance when evaluating biosecurity risks and setting inspection thresholds.

```
hist(estimate_leakage_01_imperf$E.L, breaks=30, main="Distribution of expected leakage")
```



```
hist(estimate_leakage_01_imperf$P.L, breaks=30, main="Distribution of probability of leakage")
```



These leakage estimates, derived through approximate analytical formulas for a given set of beta distribution parameters, provide a direct link between statistical inference and its practical implications for biosecurity management. By quantifying both the magnitude and likelihood of undetected infection, they help decision-makers assess inspection performance and refine testing strategies to reduce leakage risk.

Model-based Communicating Risk Assessment

The R package provides a framework for model-based simulation to support the communication and interpretation of biosecurity risk. In many biosecurity applications, the practical implications of the assumed model and parameter estimates for policy and risk management are not always straightforward to interpret. While

quantities such as the expected leakage rate per consignment, $E(L_i)$, and the probability that a consignment contains undetected infection, $\Pr(L_i > 0)$, have clear meanings, they are simplified summaries that do not fully describe the distribution of L_i . The leakage rate $E(L_i)$, in particular, may be difficult to interpret when L_i is zero with high probability.

To provide a more intuitive and visual representation of leakage, we summarize the simulated number of infected items in a consignment, T_{Xi} , classified by detection status: true positives (infection correctly detected), false negatives (infection undetected), and true negatives (no infection). Leakage arises in consignments that are false negatives, where infection is present but not detected.

To make these results more tangible, the package allows users to simulate expected outcomes for a notional set of 1,000 consignments, representing approximately one year of import activity. Outcomes are generated by simulating both T_{Xi} and \tilde{t}_{yi} from the assumed model, using parameters drawn from their Metropolis–Hastings posterior distributions, with test sensitivity fixed at $\Delta = 0.8$. Users can explore alternative values of the minimum positive propensity, k (e.g., 0%, 1%, or 2%). These simulation capabilities help translate statistical inference into interpretable measures of risk, improving how uncertainty and leakage potential are communicated in biosecurity decision-making.

For illustration, we use posterior samples of the parameters α and β estimated from the frozen seafood data, assuming a 1% minimum positive propensity. These posterior samples are used to generate infection outcomes X_{ij} and group test results Y_{ij} for $D = 1000$ batches, each containing $B = 8000$ groups. Following the case study, we also consider $m = 5$, $M = 40$, and $b = 13$. The function `run_infection_simulations()` can be used to generate the population and perform sampling while adjusting the test sensitivity (e.g., $\Delta = 0.80$). This function compares sampled group test outcomes with the true infection or contamination status and classifies batches into three categories: true negatives (TN), true positives (TP), and false negatives (FN). The simulated outcomes are then used to estimate (i) the **expected leakage** — the average number of infected items missed due to undetected infection — and (ii) the **probability of leakage**, defined as the proportion of batches resulting in FN outcomes. Increasing the number of posterior samples of α and β along with the number of batches D provide better results.

```
alpha_TrBB <- sample(unlist(MH.trbb.m1$target.parameters$alpha_sample), 500)
beta_TrBB <- sample(unlist(MH.trbb.m1$target.parameters$beta_sample), 500)
cut_off <- 0.01
summary(alpha_TrBB)
#>   Min. 1st Qu. Median Mean 3rd Qu. Max.
#> 0.002897 0.004643 0.005607 0.005708 0.006497 0.010264
summary(beta_TrBB)
#>   Min. 1st Qu. Median Mean 3rd Qu. Max.
#> 1.133 2.638 3.376 3.558 4.237 8.076
```

```

Sim_Tx_TrBB <- run_infection_simulations(alpha=alpha_TrBB,beta=beta_TrBB, cutoff = cut_off, D = 5000,B
= 8000, M = 40,m = 5,b = 13,sensitivity_values = c(0.8,1.0),NSim = length(alpha_TrBB),seed =
2015)
prop.table(table(Sim_Tx_TrBB$`0.8`$Simulation_Infection_Ty_Results)) # Proportion of batches with TP,
# TN and FN
#>
#>      1          2          3
#> 0.014194 0.983000 0.002806

```

We can calculate **leakage estimates** based on the simulated observations using the function `compute_simulated_leakage`. This function produces both the expected leakage and the probability of leakage per batch, summarizing the outcomes of the simulation under different assumptions of test sensitivity. From the model-based simulations, additional leakage summaries, including the median of non-zero leakage, $Q_2(L_i > 0)$, and the pseudo-expected leakage, $\text{E}_p(L_i) = P(L_i > 0) \times Q_2(L_i > 0)$ can be computed using `compute_simulated_leakage` function. These measures provide complementary insights by quantifying the typical size of leakage events when they occur and the combined effect of their frequency and magnitude, offering a more nuanced view of leakage behavior beyond the mean-based summaries.

```

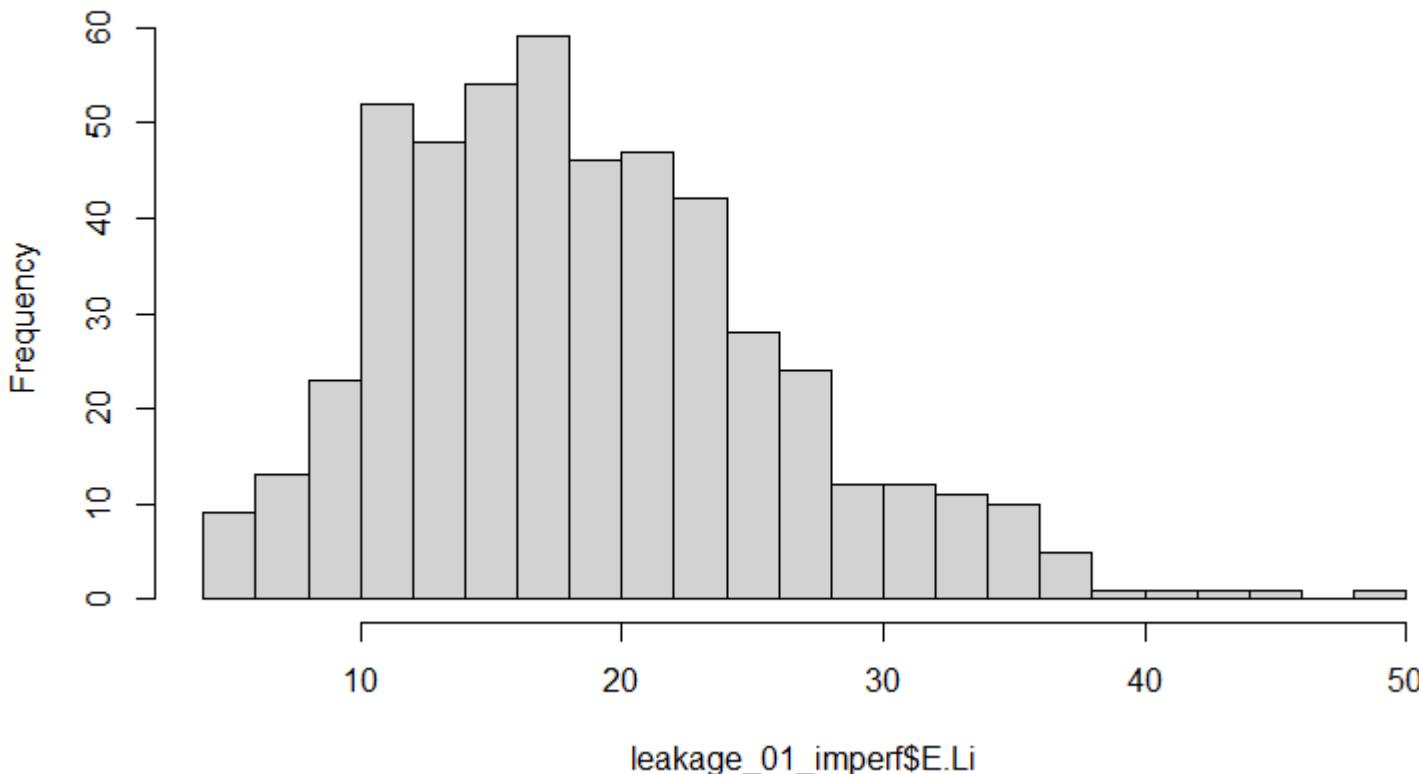
leakage_01_perf <- compute_simulated_leakage(Sim_Tx_TrBB$`1`)
leakage_01_imperf <- compute_simulated_leakage(Sim_Tx_TrBB$`0.8`)
mean(leakage_01_perf$E.Li)
#> [1] 12.9772
summary(leakage_01_perf$P.Li)
#>   Min. 1st Qu. Median Mean 3rd Qu. Max.
#> 0.000400 0.001600 0.002000 0.002133 0.002600 0.005000
summary(leakage_01_imperf$E.Li)
#>   Min. 1st Qu. Median Mean 3rd Qu. Max.
#> 4.466 13.201 17.573 18.744 23.057 48.552
summary(leakage_01_imperf$P.Li)
#>   Min. 1st Qu. Median Mean 3rd Qu. Max.
#> 0.000800 0.002000 0.002600 0.002806 0.003400 0.006400

```

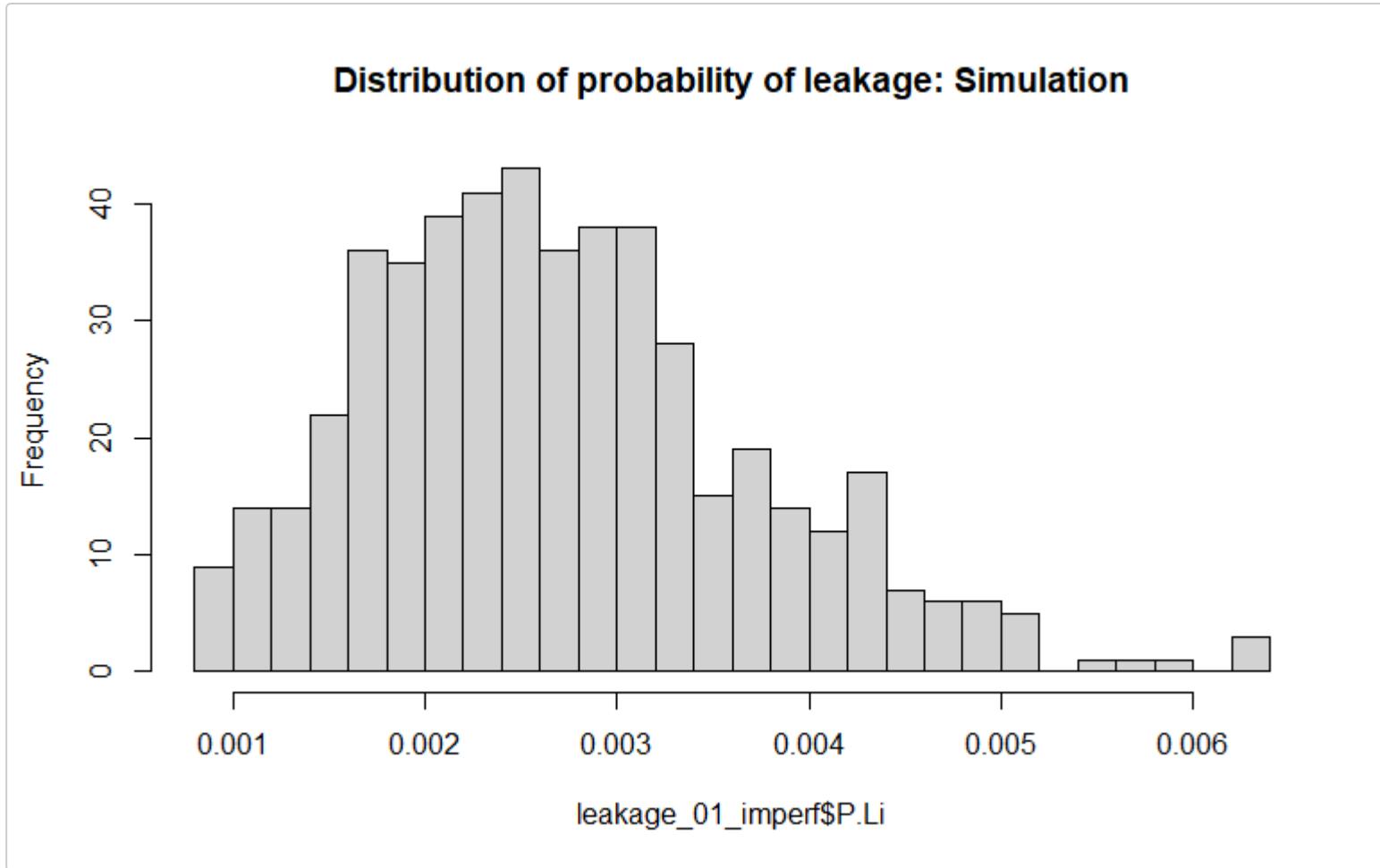
The posterior distributions of the model-based leakage quantities can also be explored visually to assess uncertainty and distributional behavior.

```
hist(leakage_01_imperf$E.Li, breaks=30, main="Distribution of expected leakage: Simulation")
```

Distribution of expected leakage: Simulation



```
hist(leakage_01_imperf$P.Li, breaks=30, main="Distribution of probability of leakage: Simulation")
```



The estimated leakage values ($E(L_i)$ and $\Pr(L_i > 0)$), derived from the posterior distributions of α and β through analytic formulas (17) - (20), are generally consistent with those obtained from the model-based simulation. Under the assumption of perfect test sensitivity, the approximate analytic formula provides estimates that closely align with the simulated values, confirming the robustness of the analytic approach for practical applications. Minor deviations between the analytic and simulation-based estimates primarily reflect stochastic variability introduced through simulation and posterior sampling, as well as approximations in the analytical expressions under imperfect sensitivity.

Session Info & References

```
sessionInfo()
#> R version 4.5.1 (2025-06-13 ucrt)
#> Platform: x86_64-w64-mingw32/x64
#> Running under: Windows 11 x64 (build 26100)
#>
#> Matrix products: default
#>   LAPACK version 3.12.1
#>
#> Locale:
#> [1] LC_COLLATE=English_Australia.utf8  LC_CTYPE=English_Australia.utf8
#> [3] LC_MONETARY=English_Australia.utf8 LC_NUMERIC=C
#> [5] LC_TIME=English_Australia.utf8
#>
#> time zone: Australia/Sydney
#> tzcode source: internal
#>
#> attached base packages:
#> [1] stats      graphics   grDevices utils      datasets   methods    base
#>
#> other attached packages:
#> [1] coda_0.19-4.1     bayesplot_1.13.0   MASS_7.3-65       ggpubr_0.6.1
#> [5] ggplot2_3.5.2     groupedBBMH_0.1.0
#>
#> Loaded via a namespace (and not attached):
#> [1] tensorA_0.36.2.1   gmp_0.7-5          sass_0.4.10
#> [4] generics_0.1.4     tidyverse_1.3.1    rstatix_0.7.2
#> [7] stringi_1.8.7      lattice_0.22-7    digest_0.6.37
#> [10] magrittr_2.0.3     evaluate_1.0.4    grid_4.5.1
#> [13] RColorBrewer_1.1-3 fastmap_1.2.0     plyr_1.8.9
#> [16] jsonlite_2.0.0     backports_1.5.0   Formula_1.2-5
#> [19] gridExtra_2.3       purrrr_1.1.0      scales_1.4.0
#> [22] jqueryLib_0.1.4    abind_1.4-8       cli_3.6.5
#> [25] Rmpfr_1.1-1       rlang_1.1.6      cowplot_1.2.0
#> [28] splines_4.5.1      withr_3.0.2      cachem_1.1.0
#> [31] yaml_2.3.10        tools_4.5.1      reshape2_1.4.4
#> [34] checkmate_2.3.2    ggsignif_0.6.4   dplyr_1.1.4
#> [37] VGAM_1.1-13       broom_1.0.9      vctrs_0.6.5
```

```
#> [40] posterior_1.6.1      R6_2.6.1           stats4_4.5.1
#> [43] lifecycle_1.0.4     stringr_1.5.1      car_3.1-3
#> [46] pkgconfig_2.0.3     pillar_1.11.0      bslib_0.9.0
#> [49] gtable_0.3.6       Rcpp_1.1.0          glue_1.8.0
#> [52] xfun_0.52          tibble_3.3.0       tidyselect_1.2.1
#> [55] rstudioapi_0.17.1   knitr_1.50         farver_2.1.2
#> [58] htmltools_0.5.8.1   labeling_0.4.3     rmarkdown_2.29
#> [61] carData_3.0-5       compiler_4.5.1    distributional_0.5.0
```

References

- Das, S., Clark, R., Parsa, M., & Barnes, B. (2025). *Inferring Leakage in Imports of Frozen Seafood allowing for Censoring, Testing Accuracy and a Minimum Positive Prevalence*. Submitted to *Journal of the American Statistical Association*.
- Clark, R. G., Barnes, B., & Parsa, M. (2024). Clustered and unclustered group testing for biosecurity. *Journal of Agricultural, Biological and Environmental Statistics*, 29(2), 193-211.
- Clark, R. G. (2023). *Deidentified Frozen Seafood Dataset* (1.0) [Data set]. Zenodo.
<https://doi.org/10.5281/zenodo.7762164>