M3: Hierarchical Logistic Regression with Diagnostics

M3-Hierarchical-Logistic-Diagnostic.Rmd | Compiled: 2026-01-26 09:41:26

Contents

```r
# ============================================================================
# M3: HIERARCHICAL LOGISTIC REGRESSION PARAMETERS
# Model: y ~ Bernoulli(logit^-1(X  + Zu))
# Binary response with random effects, no error precision

# PARAMETERS
n <- 300  # Total observations
p <- 3    # Number of fixed effects (beta_0, beta_1, beta_2)

# True Parameter Values (Dr John's exact values)
tau_u_true <- 1.0   # Random effect precision (variance = 1) - matching M0/M2
beta_true  <- c(0.5, -2, 3)  # Regression coefficients

# Prior Hyperparameters (flat priors for tau_u only)
alpha_u <- 0
gamma_u <- 0

# VB Settings
vb_iter    <- 500
vb_epsilon <- 1e-4

# Gibbs Sampler Settings
run_gibbs    <- TRUE   # Set FALSE for quick testing, TRUE for full run
gibbs_iter   <- 5000
gibbs_burnin <- 1000

# Display Settings
RENDER_FUNCTIONS <- FALSE  # TRUE to show function code, FALSE to hide

# SCENARIO 1: Q=5 groups (n=60 per group)
q_s1  <- 5
nq_s1 <- 60

# SCENARIO 2: Q=10 groups (n=30 per group)
q_s2  <- 10
nq_s2 <- 30

# SCENARIO 3: Q=20 groups (n=15 per group)
q_s3  <- 20
nq_s3 <- 15

# SCENARIO 4: Q=50 groups (n=6 per group)
q_s4  <- 50
nq_s4 <- 6

# SCENARIO 5: Q=100 groups (n=3 per group)
q_s5  <- 100
nq_s5 <- 3

# ============================================================================
```

```r
# Hierarchical Logistic Gibbs sampler with Metropolis-Hastings
# y ~ Bernoulli(logit⁻¹(X  + Zu))
# Metropolis-Hastings for ( ,u) |  ,y
# Conjugate Gamma update for   | u
logisticmm.Gibbs <- function(iter, Z, X, y, burnin, tauu_0, Kinv, a.u, b.u) {
  n <- length(y)
  p <- dim(X)[2]
  q <- dim(Z)[2]
  W <- cbind(X, Z)

  # Initialize
  tauu <- tauu_0
  beta0 <- rnorm(p, 0, 0.1)
  u0 <- rnorm(q, 0, sd = 1/sqrt(tauu))
  betau <- c(beta0, u0)

  # Storage
  par <- matrix(0, iter, p + q + 1)  # No tau_e for logistic
  accept_count <- 0

  # Kinv for penalty
  I0 <- diag(p + q)
  diag(I0)[1:p] <- 0
  I0[-c(1:p), -c(1:p)] <- Kinv

  # MH proposal variance (tuned for ~25% acceptance)
  proposal_scale <- 0.05

  for (i in 1:iter) {
    # Update   (conjugate Gamma)
    u_current <- betau[(p+1):(p+q)]
    uKinvu <- t(u_current) %*% Kinv %*% u_current
    tauu <- rgamma(1, a.u + 0.5*q, b.u + 0.5*as.numeric(uKinvu))

    # Update ( ,u) via Metropolis-Hastings
    # Current log-posterior
    eta_current <- as.vector(W %*% betau)
    p_current <- 1 / (1 + exp(-eta_current))
    log_lik_current <- sum(y * log(p_current + 1e-10) + (1-y) * log(1 - p_current + 1e-10))
    log_prior_current <- -0.5 * tauu * as.numeric(t(betau) %*% I0 %*% betau)
    log_post_current <- log_lik_current + log_prior_current

    # Proposal
    betau_prop <- betau + rnorm(p + q, 0, proposal_scale)

    # Proposed log-posterior
    eta_prop <- as.vector(W %*% betau_prop)
    p_prop <- 1 / (1 + exp(-eta_prop))
    log_lik_prop <- sum(y * log(p_prop + 1e-10) + (1-y) * log(1 - p_prop + 1e-10))
    log_prior_prop <- -0.5 * tauu * as.numeric(t(betau_prop) %*% I0 %*% betau_prop)
    log_post_prop <- log_lik_prop + log_prior_prop

    # Accept/reject
```

```
    log_alpha <- log_post_prop - log_post_current
    if (log(runif(1)) < log_alpha) {
      betau <- betau_prop
      if (i > burnin) accept_count <- accept_count + 1
    }

    # Store
    par[i, ] <- c(betau, tauu)
  }

  # Post-burnin samples
  par <- par[-c(1:burnin), ]
  colnames(par) <- c(paste0('beta', 0:(p-1)), paste0('u', 1:q), 'tau_u')

  # Acceptance rate (post-burnin)
  accept_rate <- accept_count / (iter - burnin)

  mresult <- list(par, accept_rate)
  names(mresult) <- c('par', 'accept_rate')
  return(mresult)
}
```

```
# Logistic regression uses VB.logisticmm_with_history only
# No non-history version needed for M3
```

```
# Hierarchical Logistic VB with history tracking
# Mean-field approximation for y ~ Bernoulli(logit⁻¹(X β + Zu))
# Tracks E[β], ELBO, E[τ], E[u] per iteration
VB.logisticmm_with_history <- function(epsilon, iter, Kinv, Z, X, y, tauu_0, u0, beta0, a.u, g.u) {
  n <- dim(X)[1]
  p <- dim(X)[2]
  q <- dim(Z)[2]
  W <- cbind(X, Z)

  # Initialize
  ub <- c(beta0, u0)
  Vub <- diag(p + q) * 0.1
  Kinvall <- matrix(0, p+q, p+q)
  Kinvall[-c(1:p), -c(1:p)] <- Kinv

  # History storage
  E_tau_u_history <- numeric(iter)
  elbo_history <- numeric(iter)
  E_beta_history <- matrix(NA, nrow = iter, ncol = p)
  E_u_history <- matrix(NA, nrow = iter, ncol = q)

  for (i in 1:iter) {
    # E-step: Update q(β,u) using Laplace approximation
    eta <- as.vector(W %*% ub)
    p_logistic <- 1 / (1 + exp(-eta))
    w_diag <- p_logistic * (1 - p_logistic) + 1e-6  # diagonal weights

    # Precision matrix
    Prec_ub <- crossprod(W * sqrt(w_diag), W * sqrt(w_diag)) + tauu_0 * Kinvall
```

```r
    Vub <- solve(Prec_ub)

    # Mean update (Newton-Raphson step)
    grad <- crossprod(W, y - p_logistic) - tauu_0 * Kinvall %*% ub
    ub <- ub + as.vector(Vub %*% grad)

    # M-step: Update
    TrKinvub <- sum(diag(Kinvall %*% Vub))
    uKinvub <- t(ub) %*% Kinvall %*% ub
    tauu <- (a.u + 0.5*q) / (g.u + 0.5*as.numeric(uKinvub) + 0.5*TrKinvub)
    tauu <- as.numeric(tauu)

    # Store history
    E_tau_u_history[i] <- tauu
    E_beta_history[i, ] <- ub[1:p]
    E_u_history[i, ] <- ub[(p+1):(p+q)]

    # ELBO (approximate)
    eta <- as.vector(W %*% ub)
    p_logistic <- 1 / (1 + exp(-eta))
    log_lik <- sum(y * log(p_logistic + 1e-10) + (1-y) * log(1 - p_logistic + 1e-10))
    log_prior_u <- 0.5 * q * (digamma(a.u + 0.5*q) - log(g.u + 0.5*as.numeric(uKinvub) + 0.5*TrKinvub))
    entropy <- 0.5 * determinant(Vub, logarithm = TRUE)$modulus[1]
    elbo_history[i] <- log_lik + log_prior_u + entropy

    # Convergence check
    if (i > 1) {
      diffub <- sqrt((ub - ub0)^2) / (abs(ub) + 0.01)
      difftu <- abs(tauu_0 - tauu) / (tauu + 0.01)
      diff.all <- c(diffub, difftu)
      if (max(diff.all) < epsilon) break
    }
    ub0 <- ub
    tauu_0 <- tauu
  }

  # Trim history
  E_tau_u_history <- E_tau_u_history[1:i]
  elbo_history <- elbo_history[1:i]
  E_beta_history <- E_beta_history[1:i, , drop = FALSE]
  E_u_history <- E_u_history[1:i, , drop = FALSE]

  tauu.param <- c((a.u + 0.5*q), (g.u + 0.5*uKinvub + 0.5*TrKinvub))
  param <- list(ub, Vub, NULL, tauu.param, i, NULL, E_tau_u_history, elbo_history, E_beta_history, E_u_
  names(param) <- c('betau_mean', 'betau_var', 'tau_e', 'tau_u', 'iter', 'E_tau_e_history', 'E_tau_u_his
  return(param)
}
```

1 scenario 1: Conditional on Model Type

```
cat(glue("\n=== Scenario 1: Q={q_s1} groups (n={nq_s1} per group) ===\n"))
```

```
## === Scenario 1: Q=5 groups (n=60 per group) ===
```

```
# Generate design matrices
X_s1 <- cbind(1, matrix(rnorm(n*(p-1)), nrow=n, ncol=p-1))

# Generate random effects (standardized)
u_true_s1 <- rnorm(q_s1, 0, 1)
u_true_s1 <- scale(u_true_s1)

# Create group assignment matrix
Z_s1 <- table(1:n, rep(1:q_s1, n/q_s1))
K_s1 <- diag(q_s1)

# Linear predictor
eta_s1 <- X_s1 %*% beta_true + Z_s1 %*% u_true_s1

# Logistic transformation
p_s1 <- 1 / (1 + exp(-eta_s1))

# Binary outcomes
y_s1 <- rbinom(n, size = 1, prob = as.vector(p_s1))
```

```
scenario_name <- glue("SCENARIO 1: {q_s1} levels ({nq_s1} obs each)")
cat(glue("\n=== {scenario_name} ===\n"))
```

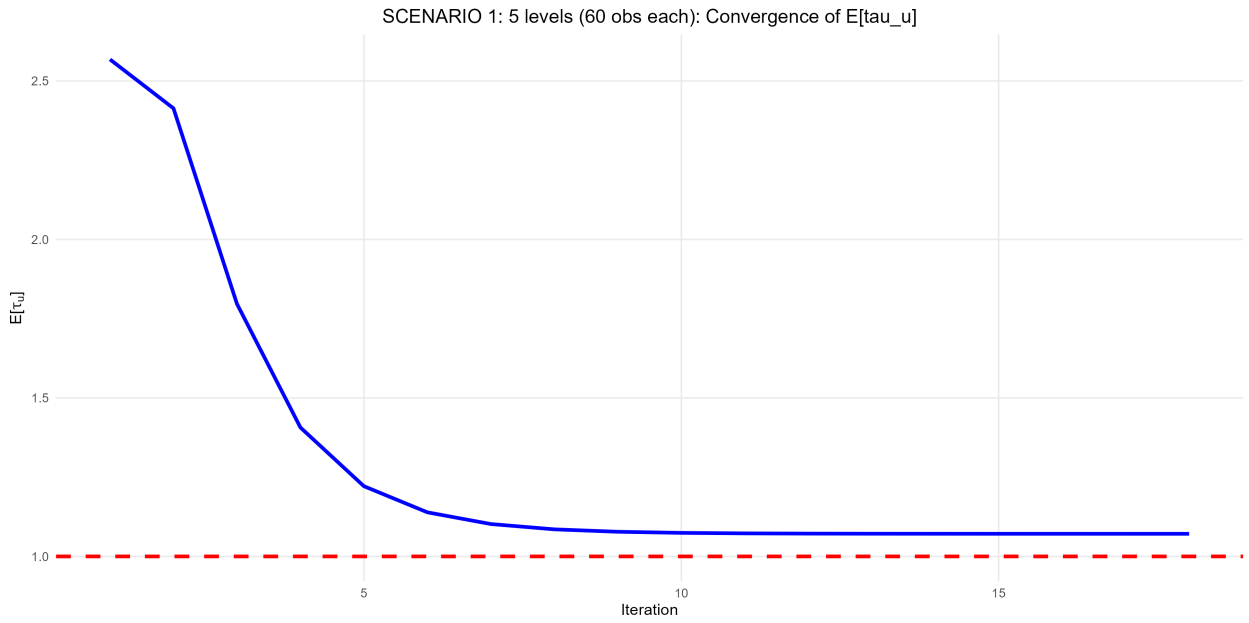
```
## === SCENARIO 1: 5 levels (60 obs each) ===
```

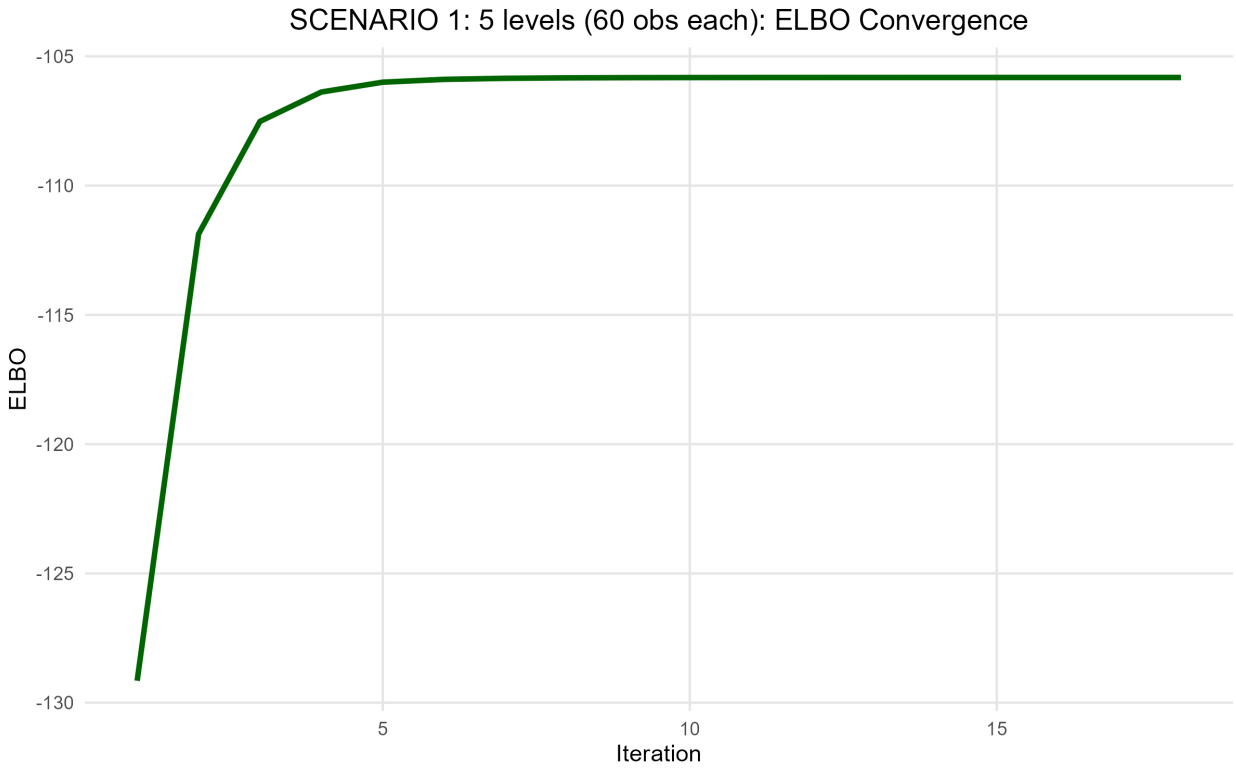
```
results_s1 <- run_vb_algorithm(
  X          = X_s1,
  Z          = Z_s1,
  y          = y_s1,
  K          = K_s1,
  p          = p,
  q          = q_s1,
  n          = n,
  alpha_u    = alpha_u,
  gamma_u    = gamma_u
)
```

```
if (run_gibbs) {
  cat("\nRunning Gibbs sampler...\n")
  gibbs_result <- run_gibbs_sampler(
    X          = X_s1,
    Z          = Z_s1,
    y          = y_s1,
    p          = p,
    q          = q_s1,
    n          = n,
    alpha_u    = alpha_u,
    gamma_u    = gamma_u,
    n_iter     = gibbs_iter,
    n_burnin   = gibbs_burnin
```

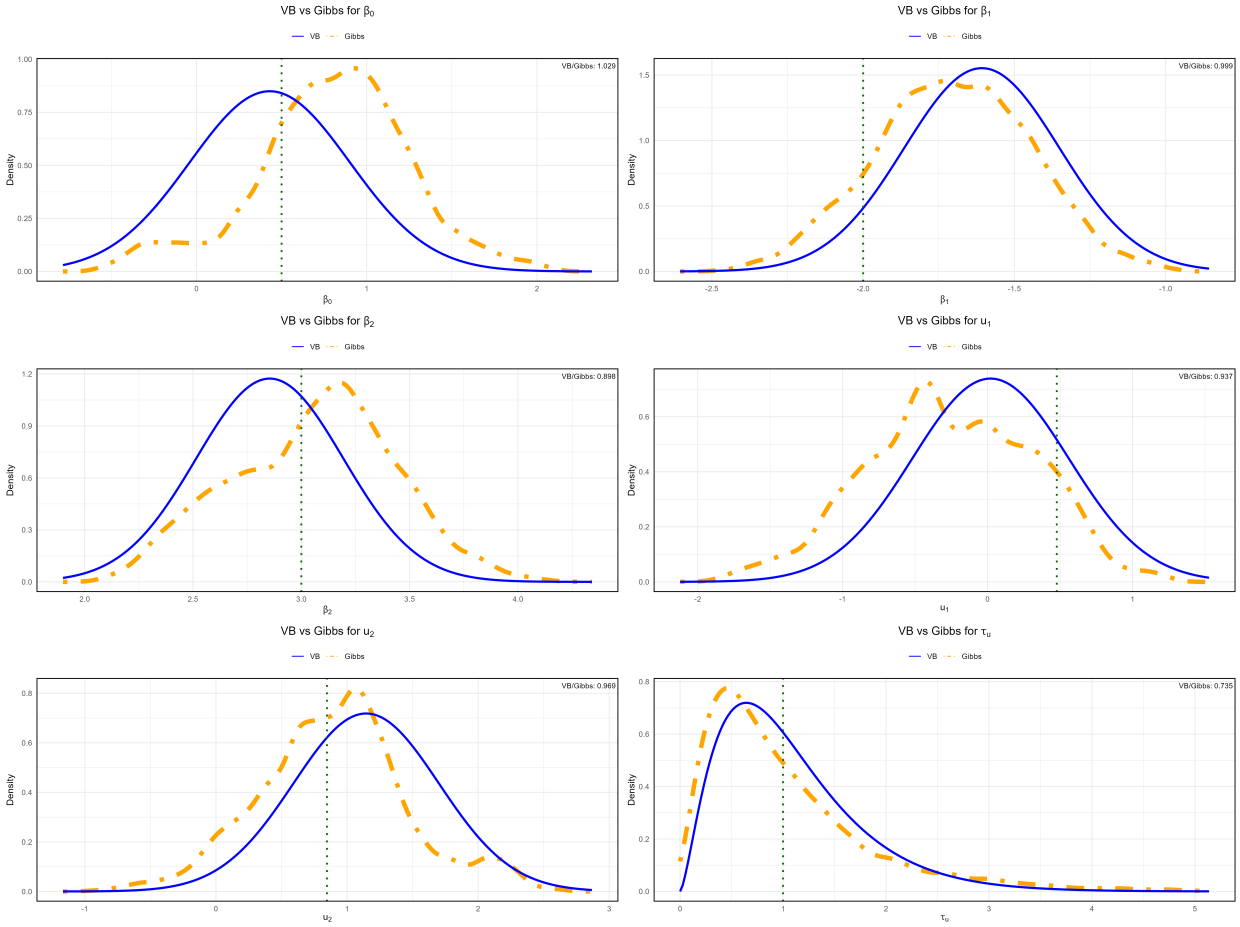
```
  )

  gibbs_s1 <- gibbs_result$samples

  cat("Gibbs posterior means:\n")
  cat("beta:", colMeans(gibbs_s1[, 1:p]), "\n")
  cat("tau_u:", mean(gibbs_s1[, "tau_u"]), "\n")

  gibbs_tau_u_s1 <- gibbs_s1[, "tau_u"]
} else {
  gibbs_s1 <- NULL
  gibbs_tau_u_s1 <- NULL
}
```

```
##
## Running Gibbs sampler...
## Gibbs posterior means:
## beta: 0.7992452 -1.705485 3.057809
## tau_u: 1.059928
```






```
## M3: 5-panel ggplot saved for Scenario 1
```


VB vs Gibbs for β_0
VB
Gibbs
VB/Gibbs: 1.029
Density
β_0
VB vs Gibbs for β_1
VB
Gibbs
VB/Gibbs: 0.999
Density
β_1
VB vs Gibbs for β_2
VB
Gibbs
VB/Gibbs: 0.898
Density
β_2
VB vs Gibbs for u_1
VB
Gibbs
VB/Gibbs: 0.937
Density
u_1
VB vs Gibbs for u_2
VB
Gibbs
VB/Gibbs: 0.969
Density
u_2
VB vs Gibbs for τ_u
VB
Gibbs
VB/Gibbs: 0.735
Density
τ_u

2 scenario 2: Conditional on Model Type (M3 only)

```
cat(glue("\n=== Scenario 2: Q={q_s2} groups (n={nq_s2} per group) ===\n"))
```

```
## === Scenario 2: Q=10 groups (n=30 per group) ===
```

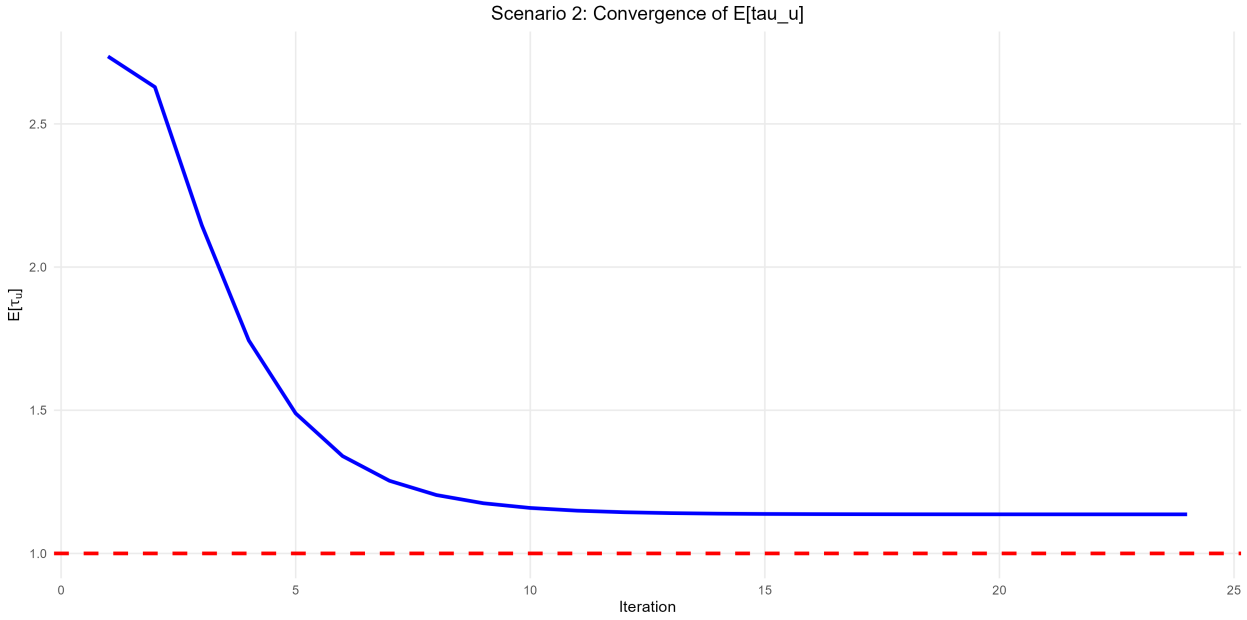
```
# Generate design matrices
X_s2 <- cbind(1, matrix(rnorm(n*(p-1)), nrow=n, ncol=p-1))

# Generate random effects (standardized)
u_true_s2 <- rnorm(q_s2, 0, 1)
u_true_s2 <- scale(u_true_s2)

# Create group assignment matrix
Z_s2 <- table(1:n, rep(1:q_s2, n/q_s2))
K_s2 <- diag(q_s2)

# Linear predictor
eta_s2 <- X_s2 %*% beta_true + Z_s2 %*% u_true_s2

# Logistic transformation
p_s2 <- 1 / (1 + exp(-eta_s2))

# Binary outcomes
y_s2 <- rbinom(n, size = 1, prob = as.vector(p_s2))
```

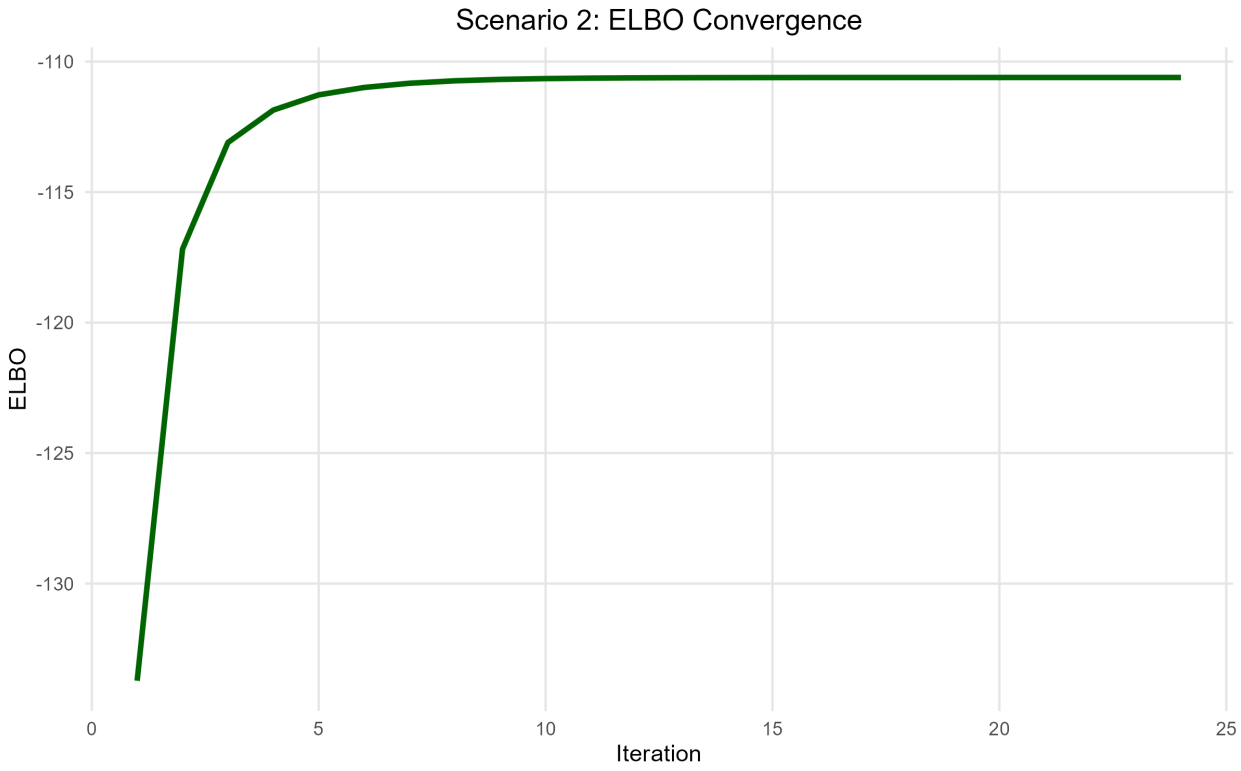
```
results_s2 <- run_vb_algorithm(
  X          = X_s2,
  Z          = Z_s2,
  y          = y_s2,
  K          = K_s2,
  p          = p,
  q          = q_s2,
  n          = n,
  alpha_u    = alpha_u,
  gamma_u    = gamma_u
)
```

```
if (run_gibbs) {
  cat("\nRunning Gibbs sampler for Scenario 2...\n")
  gibbs_result2 <- run_gibbs_sampler(
    X          = X_s2,
    Z          = Z_s2,
    y          = y_s2,
    p          = p,
    q          = q_s2,
    n          = n,
    alpha_u    = alpha_u,
    gamma_u    = gamma_u,
    n_iter     = gibbs_iter,
    n_burnin   = gibbs_burnin
  )

  gibbs_s2 <- gibbs_result2$samples
```

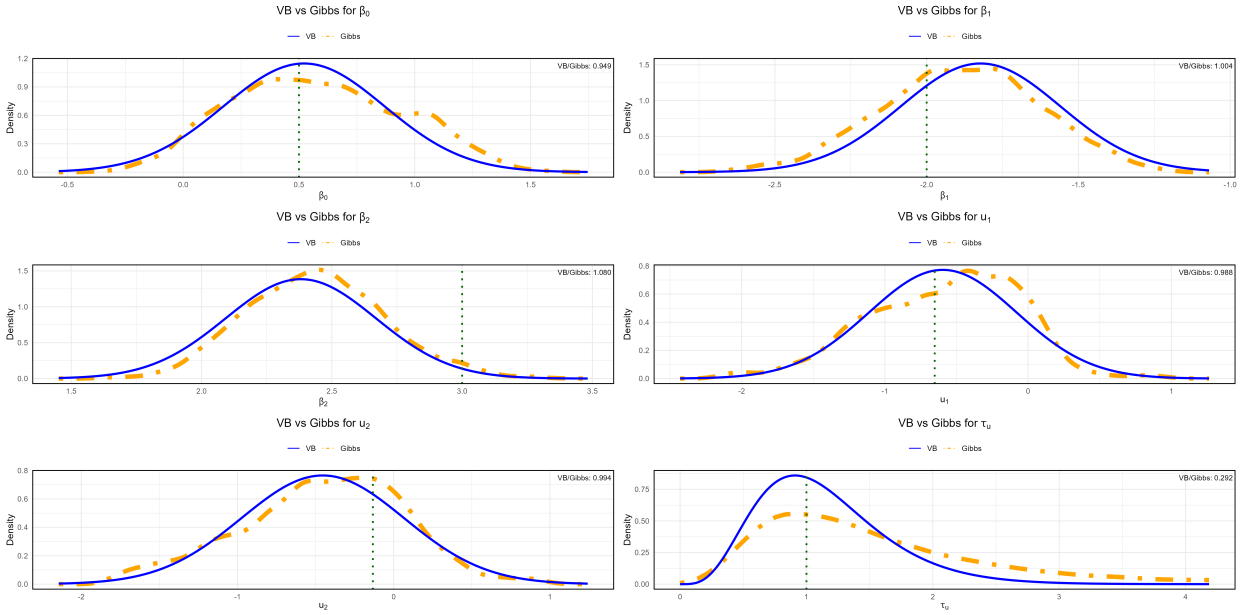
```r
  cat("Gibbs posterior means:\n")
  cat("beta:", colMeans(gibbs_s2[, 1:p]), "\n")
  cat("tau_u:", mean(gibbs_s2[, "tau_u"]), "\n")

  gibbs_tau_u_s2 <- gibbs_s2[, "tau_u"]
} else {
  gibbs_s2 <- NULL
  gibbs_tau_u_s2 <- NULL
}
```

```
##
## Running Gibbs sampler for Scenario 2...
## Gibbs posterior means:
## beta: 0.5771989 -1.888774 2.437932
## tau_u: 1.774233
```






```
## M3: 8-panel ggplot saved for Scenario 2
```


VB vs Gibbs for β_0
VB
Gibbs
VB/Gibbs: 0.949
Density
β_0
VB vs Gibbs for β_1
VB
Gibbs
VB/Gibbs: 1.004
Density
β_1
VB vs Gibbs for β_2
VB
Gibbs
VB/Gibbs: 1.080
Density
β_2
VB vs Gibbs for u_1
VB
Gibbs
VB/Gibbs: 0.988
Density
u_1
VB vs Gibbs for u_2
VB
Gibbs
VB/Gibbs: 0.994
Density
u_2
VB vs Gibbs for τ_u
VB
Gibbs
VB/Gibbs: 0.292
Density
τ_u

3 comparison Between Scenarios (M3 only)

```
comparison_df <- data.frame(
  Parameter   = c("E[tau_u]", "sigma^2_u"),
  True_Value  = c(tau_u_true, 1/tau_u_true),
  Scenario_30 = c(results_s1$E_tau_u, 1/results_s1$E_tau_u),
  Scenario_6  = c(results_s2$E_tau_u, 1/results_s2$E_tau_u)
)

print(comparison_df)
```

```
##   Parameter True_Value Scenario_30 Scenario_6
## 1  E[tau_u]          1   1.0713771  1.1364722
## 2 sigma^2_u          1   0.9333782  0.8799159
```

```
cat("\nUnder-dispersion in tau_u estimates:\n")
```

```
##
## Under-dispersion in tau_u estimates:
```

```
cat("Q=5:  VB tau_u =", round(results_s1$E_tau_u, 4),
    "vs True =", tau_u_true,
    "(ratio:", round(results_s1$E_tau_u / tau_u_true, 4), ")\n")
```

```
## Q=5:  VB tau_u = 1.0714 vs True = 1 (ratio: 1.0714 )
```

```
cat("Q=50: VB tau_u =", round(results_s2$E_tau_u, 4),
    "vs True =", tau_u_true,
    "(ratio:", round(results_s2$E_tau_u / tau_u_true, 4), ")\n")
```

```
## Q=50: VB tau_u = 1.1365 vs True = 1 (ratio: 1.1365 )
```

4 sample Size Effects on _u (Multi-Configuration Analysis)

```
# experimental design: Fix N=300, vary Q to show sample size per group effect
# Following Dr John's guidance (Meeting 16 Jan 2026)
# Q = 5 → 60 obs/group (rich data)
# Q = 10 → 30 obs/group
# Q = 20 → 15 obs/group
# Q = 50 → 6 obs/group (sparse data)

cat("\n========================================\n")
```

```
##
## ========================================
```

```
cat("Multi-Configuration _u Analysis\n")
```

```
## Multi-Configuration _u Analysis
```

```
cat("========================================\n")
```

```
## ========================================
```

```
group_configs <- list(
  list(q = 5,   nq = 60, label = "Q=5 (n=60 per group)"),
  list(q = 10,  nq = 30, label = "Q=10 (n=30 per group)"),
```

```
  list(q = 20,  nq = 15, label = "Q=20 (n=15 per group)"),
  list(q = 50,  nq = 6,  label = "Q=50 (n=6 per group)"),
  list(q = 100, nq = 3,  label = "Q=100 (n=3 per group)")
)

results_multi <- list()

for (i in seq_along(group_configs)) {
  config <- group_configs[[i]]
  cat("\n--- Running:", config$label, "---\n")

  # generate data using Dr John's exact method
  q_temp <- config$q
  nq_temp <- config$nq

  # Dr John's method: generate then standardize
  u_true_temp <- rnorm(q_temp, 0, 1)
  u_true_temp <- scale(u_true_temp)

  # Dr John's method: table() for incidence matrix
  Z_temp <- table(1:n, rep(1:q_temp, n/q_temp))

  # Dr John's method: simple X matrix (no correlation)
  X_temp <- cbind(1, matrix(rnorm(n*(p-1)), n, p-1))

  # Dr John's method: hardcoded logit for binary response
  eta_temp <- X_temp %*% beta_true + Z_temp %*% u_true_temp
  p_logistic_temp <- 1 / (1 + exp(-eta_temp))
  y_temp <- rbinom(n, 1, p_logistic_temp)

  # Covariance matrix for random effects
  K_temp <- diag(q_temp)

  # Run VB
  vb_result <- run_vb_algorithm(
    X        = X_temp,
    Z        = Z_temp,
    y        = y_temp,
    K        = K_temp,
    p        = p,
    q        = q_temp,
    n        = n,
    alpha_u  = alpha_u,
    gamma_u  = gamma_u,
    max_iter = vb_iter,
    tol      = vb_epsilon
  )

  # Run Gibbs 3 chains (run_gibbs_sampler handles this internally)
  if (run_gibbs) {
    cat("Running Gibbs sampler (3 chains)...\n")

    gibbs_result_temp <- run_gibbs_sampler(
```

```
      X        = X_temp,
      Z        = Z_temp,
      y        = y_temp,
      p        = p,
      q        = q_temp,
      n        = n,
      alpha_u  = alpha_u,
      gamma_u  = gamma_u,
      n_iter   = gibbs_iter,
      n_burnin = gibbs_burnin
    )

    gibbs_result <- gibbs_result_temp$samples
    cat("Combined", nrow(gibbs_result), "samples from 3 Gibbs chains\n")
  } else {
    gibbs_result <- NULL
  }

  # Store results
  results_multi[[i]] <- list(
    config = config,
    vb     = vb_result,
    gibbs  = gibbs_result
  )

  cat("VB E[tau_u]:", round(vb_result$E_tau_u, 4), "\n")
  if (!is.null(gibbs_result)) {
    cat("HMC E[tau_u]:", round(mean(gibbs_result[, "tau_u"]), 4), "\n")
  }
}
```

```
##
## --- Running: Q=5 (n=60 per group) ---
## Running Gibbs sampler (3 chains)...
## Combined 12000 samples from 3 Gibbs chains
## VB E[tau_u]: 0.5351
## HMC E[tau_u]: 0.4545
##
## --- Running: Q=10 (n=30 per group) ---
## Running Gibbs sampler (3 chains)...
## Combined 12000 samples from 3 Gibbs chains
## VB E[tau_u]: 4.9054
## HMC E[tau_u]: 738.1451
##
## --- Running: Q=20 (n=15 per group) ---
## Running Gibbs sampler (3 chains)...
## Combined 12000 samples from 3 Gibbs chains
## VB E[tau_u]: 1.0896
## HMC E[tau_u]: 1.2843
##
## --- Running: Q=50 (n=6 per group) ---
## Running Gibbs sampler (3 chains)...
## Combined 12000 samples from 3 Gibbs chains
## VB E[tau_u]: 1.3126
```

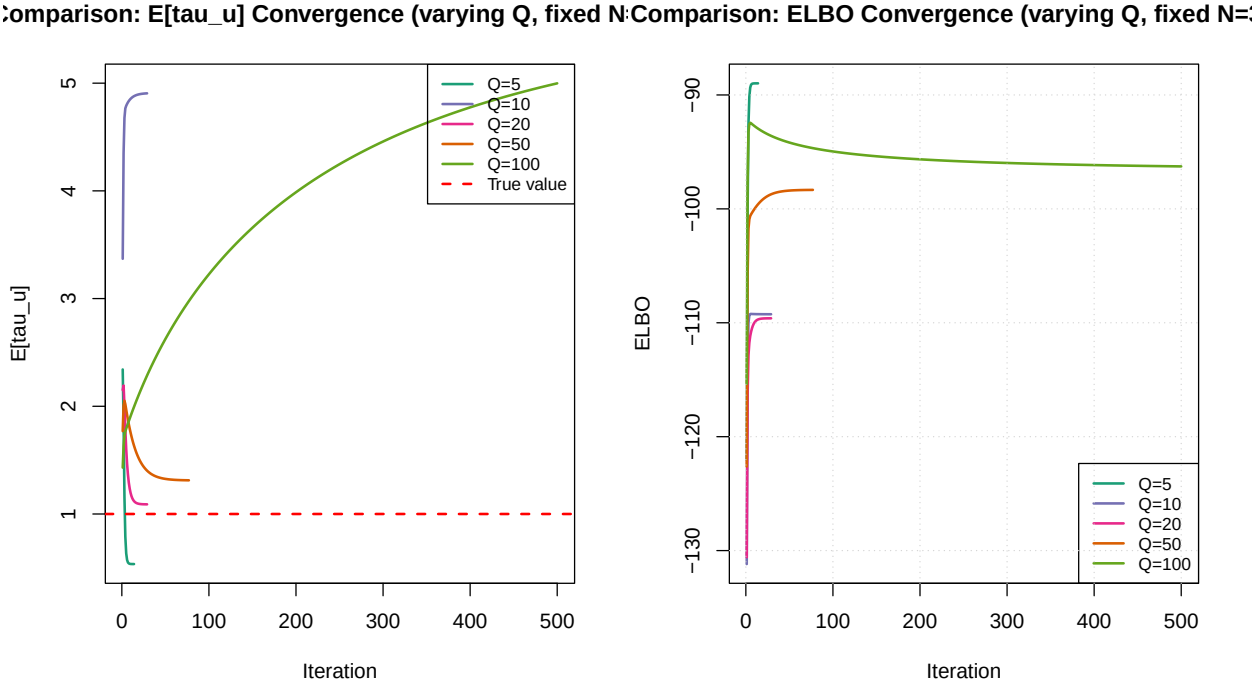
```
## HMC E[tau_u]: 2.3836
##
## --- Running: Q=100 (n=3 per group) ---
## Running Gibbs sampler (3 chains)...
## Combined 12000 samples from 3 Gibbs chains
## VB E[tau_u]: 4.9983
## HMC E[tau_u]: 4.7288
```

```
cat("\n========================================\n")
```

```
##
## ========================================
```

```
# Check if history data is available
if (!is.null(results_multi[[1]]$vb$E_tau_u_history) && length(results_multi[[1]]$vb$E_tau_u_history) > 0
  par(mfrow = c(1, 2))

  # E[tau_u] convergence for all configurations
  max_len_tau <- max(sapply(results_multi, function(r) length(r$vb$E_tau_u_history)))

  plot(
    1:length(results_multi[[1]]$vb$E_tau_u_history),
    results_multi[[1]]$vb$E_tau_u_history,
    type = 'l',
    lwd  = 2,
    col  = '#1b9e77',
    xlab = 'Iteration',
    ylab = 'E[tau_u]',
    main = 'Comparison: E[tau_u] Convergence (varying Q, fixed N=300)',
    xlim = c(1, max_len_tau),
    ylim = range(c(sapply(results_multi, function(r) r$vb$E_tau_u_history), tau_u_true)))

  for (i in 2:length(results_multi)) {
    lines(1:length(results_multi[[i]]$vb$E_tau_u_history),
          results_multi[[i]]$vb$E_tau_u_history,
          col = c('#7570b3', '#e7298a', '#d95f02', '#66a61e')[i-1],
          lwd = 2)
  }

  abline(h = tau_u_true, col = 'red', lty = 2, lwd = 2)

  legend('topright',
         legend = c('Q=5', 'Q=10', 'Q=20', 'Q=50', 'Q=100', 'True value'),
         col    = c('#1b9e77', '#7570b3', '#e7298a', '#d95f02', '#66a61e', 'red'),
         lty    = c(rep(1, 5), 2),
         lwd    = 2,
         cex    = 0.8)

  # ELBO convergence for all configurations
  max_len_elbo <- max(sapply(results_multi, function(r) length(r$vb$elbo_history)))

  plot(
    1:length(results_multi[[1]]$vb$elbo_history),
    results_multi[[1]]$vb$elbo_history,
    type = 'l',
```

```
    lwd  = 2,
    col  = '#1b9e77',
    xlab = 'Iteration',
    ylab = 'ELBO',
    main = 'Comparison: ELBO Convergence (varying Q, fixed N=300)',
    xlim = c(1, max_len_elbo),
    ylim = range(sapply(results_multi, function(r) r$vb$elbo_history)))

  for (i in 2:length(results_multi)) {
    lines(1:length(results_multi[[i]]$vb$elbo_history),
          results_multi[[i]]$vb$elbo_history,
          col = c('#7570b3', '#e7298a', '#d95f02', '#66a61e')[i-1],
          lwd = 2)
  }

  legend('bottomright',
         legend = c('Q=5', 'Q=10', 'Q=20', 'Q=50', 'Q=100'),
         col    = c('#1b9e77', '#7570b3', '#e7298a', '#d95f02', '#66a61e'),
         lty    = 1,
         lwd    = 2,
         cex    = 0.8)
  grid()
} else {
  plot.new()
  text(0.5, 0.5, "Convergence history not available\n(Using Dr John's original functions)",
       cex = 1.5, col = "gray50")
}
```




```
# Create multi-panel comparison plot as requested by Dr John
# Focus on _u posterior distributions across different group sizes
```

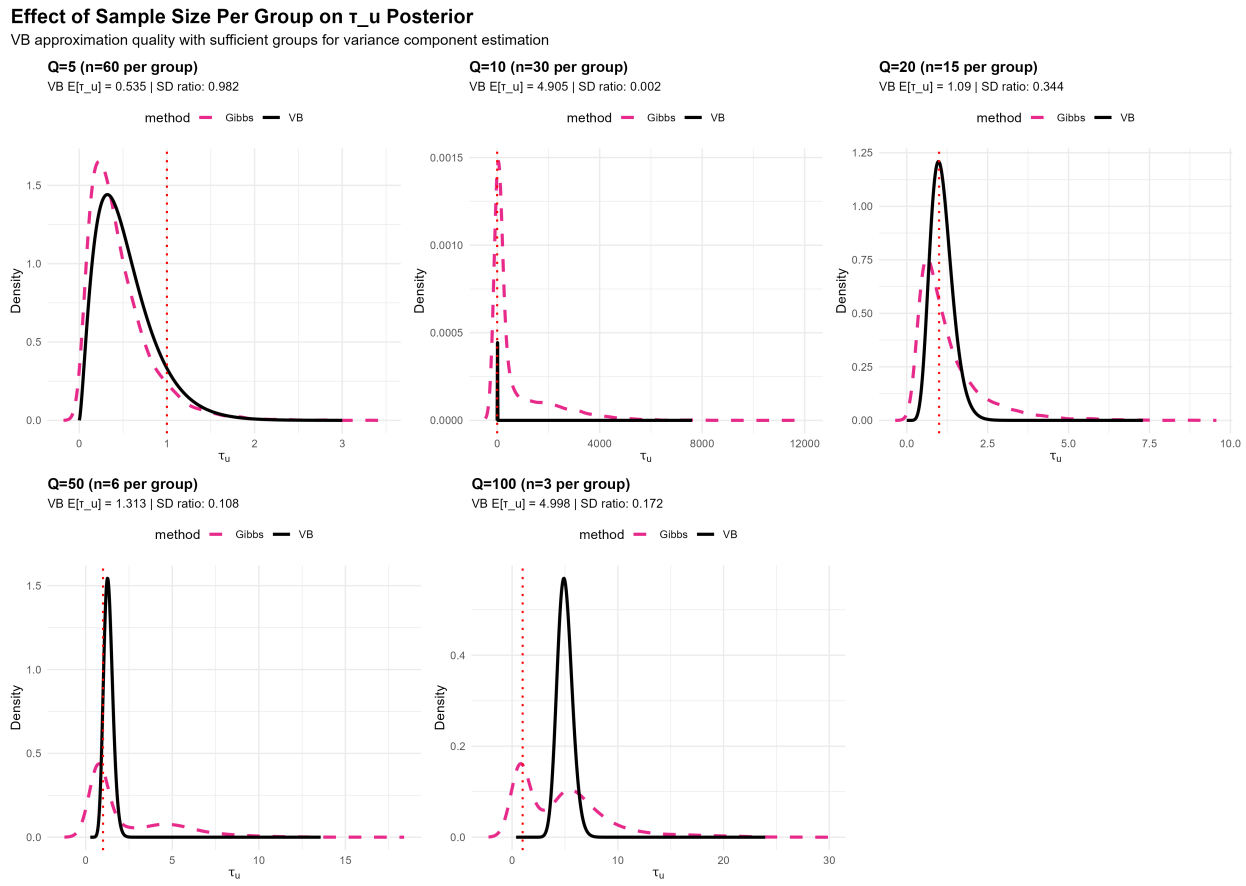
```r
plot_list <- list()

for (i in seq_along(results_multi)) {
  result <- results_multi[[i]]
  config <- result$config

  # VB posterior: Gamma(a_u_new, b_u_new)
  a_vb <- result$vb$a_u_new
  b_vb <- result$vb$b_u_new

  # Calculate VB range
  vb_mean <- a_vb / b_vb
  vb_sd <- sqrt(a_vb) / b_vb
  vb_min <- max(0, vb_mean - 4 * vb_sd)
  vb_max <- vb_mean + 4 * vb_sd

  # If Gibbs available, extend range to include both distributions
  if (!is.null(result$gibbs)) {
    gibbs_tau_u <- result$gibbs[, "tau_u"]
    gibbs_min <- quantile(gibbs_tau_u, 0.001)
    gibbs_max <- quantile(gibbs_tau_u, 0.999)

    x_min <- min(vb_min, gibbs_min, tau_u_true * 0.5)
    x_max <- max(vb_max, gibbs_max, tau_u_true * 3)
  } else {
    x_min <- min(vb_min, tau_u_true * 0.5)
    x_max <- max(vb_max, tau_u_true * 3)
  }

  x_range <- seq(x_min, x_max, length.out = 500)
  vb_density <- dgamma(x_range, shape = a_vb, rate = b_vb)

  df_plot <- data.frame(
    tau_u   = x_range,
    density = vb_density,
    method  = "VB",
    type    = "solid"
  )

  # Add Gibbs if available and calculate SD ratio
  sd_ratio_text <- ""
  if (!is.null(result$gibbs)) {
    dens_gibbs <- density(gibbs_tau_u, adjust = 1.5)

    df_gibbs <- data.frame(
      tau_u   = dens_gibbs$x,
      density = dens_gibbs$y,
      method  = "Gibbs",
      type    = "dashed"
    )

    df_plot <- rbind(df_plot, df_gibbs)
```

```
    # Calculate SD ratio
    vb_sd <- sqrt(a_vb) / b_vb
    gibbs_sd <- sd(gibbs_tau_u)
    sd_ratio <- vb_sd / gibbs_sd
    sd_ratio_text <- glue(" | SD ratio: {round(sd_ratio, 3)}")
  }

  # Create plot
  p_tau <- ggplot(df_plot, aes(x = tau_u, y = density, color = method, linetype = method)) +
    geom_line(linewidth = 1.2) +
    geom_vline(xintercept = tau_u_true, color = "red", linetype = "dotted", linewidth = 0.8) +
    scale_color_manual(
      values = c("VB" = "black", "Gibbs" = "#E7298A")
    ) +
    scale_linetype_manual(
      values = c("VB" = "solid", "Gibbs" = "dashed")
    ) +
    labs(
      title = config$label,
      subtitle = glue("VB E[ _u] = {round(result$vb$E_tau_u, 3)}{sd_ratio_text}"),
      x = expression(tau[u]),
      y = "Density"
    ) +
    theme_minimal() +
    theme(
      legend.position = "top",
      plot.title = element_text(size = 12, face = "bold"),
      plot.subtitle = element_text(size = 10)
    )

  plot_list[[i]] <- p_tau
}

# Combine all 5 plots into a grid (2 rows, 3 columns)
combined_tau_u <- (plot_list[[1]] | plot_list[[2]] | plot_list[[3]]) /
                  (plot_list[[4]] | plot_list[[5]] | plot_spacer()) +
  plot_annotation(
    title = "Effect of Sample Size Per Group on  _u Posterior",
    subtitle = "VB approximation quality with sufficient groups for variance component estimation",
    theme = theme(
      plot.title = element_text(size = 16, face = "bold"),
      plot.subtitle = element_text(size = 12)
    )
  )

# Save plot
ggsave(
  filename = "../figs/M3_tau_u_sample_size_comparison.png",
  plot     = combined_tau_u,
  width    = 14,
  height   = 10,
  dpi      = 300
)
```

```
cat(" _u comparison plot saved to figs/M2_tau_u_sample_size_comparison.png\n")
```

```
##  _u comparison plot saved to figs/M2_tau_u_sample_size_comparison.png
```

```
# Display
img_tau_u <- readPNG("../figs/M3_tau_u_sample_size_comparison.png")
grid.newpage()
grid.raster(img_tau_u)
```




```
# Create 2-panel overlay plot: All Gibbs together, All VB together

# Prepare data for Gibbs panel
if (run_gibbs) {
  gibbs_combined <- data.frame()

  for (i in seq_along(results_multi)) {
    result <- results_multi[[i]]
    config <- result$config
    gibbs_tau_u <- result$gibbs[, "tau_u"]

    dens_gibbs <- density(gibbs_tau_u, adjust = 1.5)

    df_temp <- data.frame(
      tau_u   = dens_gibbs$x,
      density = dens_gibbs$y,
```

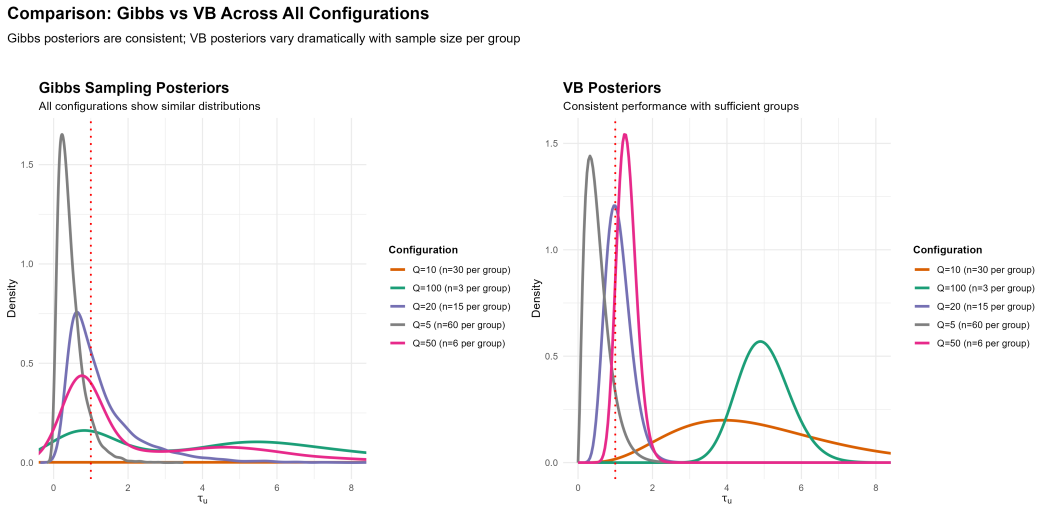
```
      config  = config$label
    )

    gibbs_combined <- rbind(gibbs_combined, df_temp)
  }

  # Gibbs panel
  p_gibbs <- ggplot(gibbs_combined, aes(x = tau_u, y = density, color = config)) +
    geom_line(linewidth = 1.2) +
    geom_vline(xintercept = tau_u_true, color = "red", linetype = "dotted", linewidth = 0.8) +
    scale_color_manual(
      values = c(
        "Q=5 (n=60 per group)"  = "gray50",
        "Q=10 (n=30 per group)" = "#d95f02",
        "Q=20 (n=15 per group)" = "#7570b3",
        "Q=50 (n=6 per group)"  = "#e7298a",
        "Q=100 (n=3 per group)" = "#1b9e77"
      )
    ) +
    coord_cartesian(xlim = c(0, 8)) +
    labs(
      title = "Gibbs Sampling Posteriors",
      subtitle = "All configurations show similar distributions",
      x = expression(tau[u]),
      y = "Density",
      color = "Configuration"
    ) +
    theme_minimal() +
    theme(
      legend.position = "right",
      legend.text = element_text(size = 9),
      legend.title = element_text(size = 10, face = "bold"),
      plot.title = element_text(size = 14, face = "bold"),
      plot.subtitle = element_text(size = 11)
    )
}

# Prepare data for VB panel
vb_combined <- data.frame()

for (i in seq_along(results_multi)) {
  result <- results_multi[[i]]
  config <- result$config

  a_vb <- result$vb$a_u_new
  b_vb <- result$vb$b_u_new

  # Use broad range to show all VB distributions
  x_range <- seq(0, 20, length.out = 500)
  vb_density <- dgamma(x_range, shape = a_vb, rate = b_vb)

  df_temp <- data.frame(
    tau_u   = x_range,
```

```
    density = vb_density,
    config  = config$label
  )

  vb_combined <- rbind(vb_combined, df_temp)
}

# VB panel
p_vb <- ggplot(vb_combined, aes(x = tau_u, y = density, color = config)) +
  geom_line(linewidth = 1.2) +
  geom_vline(xintercept = tau_u_true, color = "red", linetype = "dotted", linewidth = 0.8) +
  scale_color_manual(
    values = c(
      "Q=5 (n=60 per group)"  = "gray50",
      "Q=10 (n=30 per group)" = "#d95f02",
      "Q=20 (n=15 per group)" = "#7570b3",
      "Q=50 (n=6 per group)"  = "#e7298a",
      "Q=100 (n=3 per group)" = "#1b9e77"
    )
  ) +
  coord_cartesian(xlim = c(0, 8)) +
  labs(
    title = "VB Posteriors",
    subtitle = "Consistent performance with sufficient groups",
    x = expression(tau[u]),
    y = "Density",
    color = "Configuration"
  ) +
  theme_minimal() +
  theme(
    legend.position = "right",
    legend.text = element_text(size = 9),
    legend.title = element_text(size = 10, face = "bold"),
    plot.title = element_text(size = 14, face = "bold"),
    plot.subtitle = element_text(size = 11)
  )

# Combine panels
if (run_gibbs) {
  combined_overlay <- p_gibbs | p_vb
  plot_title <- "Comparison: Gibbs vs VB Across All Configurations"
  plot_subtitle <- "Gibbs posteriors are consistent; VB posteriors vary dramatically with sample size p
  plot_width <- 14
} else {
  combined_overlay <- p_vb
  plot_title <- "VB Posteriors Across All Configurations"
  plot_subtitle <- "VB posterior quality varies dramatically with sample size per group"
  plot_width <- 8
}

combined_overlay <- combined_overlay +
  plot_annotation(
    title = plot_title,
```

```
    subtitle = plot_subtitle,
    theme = theme(
      plot.title = element_text(size = 16, face = "bold", margin = margin(b = 10)),
      plot.subtitle = element_text(size = 12, margin = margin(b = 20))
    )
  ) &
  theme(
    legend.position = "right",
    legend.direction = "vertical",
    legend.box = "vertical",
    legend.margin = margin(l = 10),
    plot.margin = margin(t = 15, r = 10, b = 10, l = 10)
  )

# Save plot
ggsave(
  filename = "../figs/M3_tau_u_overlay_comparison.png",
  plot     = combined_overlay,
  width    = plot_width,
  height   = 7,
  dpi      = 300
)

cat(" _u overlay comparison plot saved to figs/M2_tau_u_overlay_comparison.png\n")
```

```
##  _u overlay comparison plot saved to figs/M2_tau_u_overlay_comparison.png
```

```
# Display
img_overlay <- readPNG("../figs/M3_tau_u_overlay_comparison.png")
grid.newpage()
grid.raster(img_overlay)
```




```
# Generate MY plot in Dr John's exact style
# Single panel with all Gibbs (solid) and VB (dashed) overlaid

cat("\n==========================================\n")
```

```
##
```

```
## ========================================
```

```
cat("My  _u Comparison (Dr John's Style)\n")
```

```
## My  _u Comparison (Dr John's Style)
```

```
cat("========================================\n\n")
```

```
## ========================================
```

```
# Prepare data: combine all Gibbs chains for each Q
gibbs_combined <- list()
vb_params <- list()

for (i in seq_along(results_multi)) {
  cfg <- group_configs[[i]]
  q_val <- cfg$q

  if (run_gibbs) {
    # results_multi[[i]]$gibbs is a matrix from run_gibbs_sampler
    # tau_u column name is "tau_u"
    gibbs_matrix <- results_multi[[i]]$gibbs
    gibbs_combined[[paste0("q", q_val)]] <- gibbs_matrix[, "tau_u"]
  }

  # VB gamma parameters from results_multi
  vb_result <- results_multi[[i]]$vb
  a_param <- vb_result$a_u_new
  b_param <- vb_result$b_u_new
  vb_params[[paste0("q", q_val)]] <- list(a = a_param, b = b_param)
}

# Create plot matching Dr John's style
png(filename = "../figs/M3_my_tau_u_comparison.png", width = 10, height = 8, units = "in", res = 300)

# Colors for all 5 Q values
colors <- c("black", "red", "green3", "blue", "purple")
q_values <- c(5, 10, 20, 50, 100)

# Start with first Gibbs density
if (run_gibbs) {
  plot(density(gibbs_combined$q5),
       xlab = expression(tau['u']),
       main = '',
       ylim = c(0, 2.5),
       xlim = c(0, 8),
       lwd = 2,
       col = colors[1])

  # Add remaining Gibbs densities
  lines(density(gibbs_combined$q10), col = colors[2], lwd = 2)
  lines(density(gibbs_combined$q20), col = colors[3], lwd = 2)
  lines(density(gibbs_combined$q50), col = colors[4], lwd = 2)
  lines(density(gibbs_combined$q100), col = colors[5], lwd = 2)

  # Add VB approximations (dashed)
```

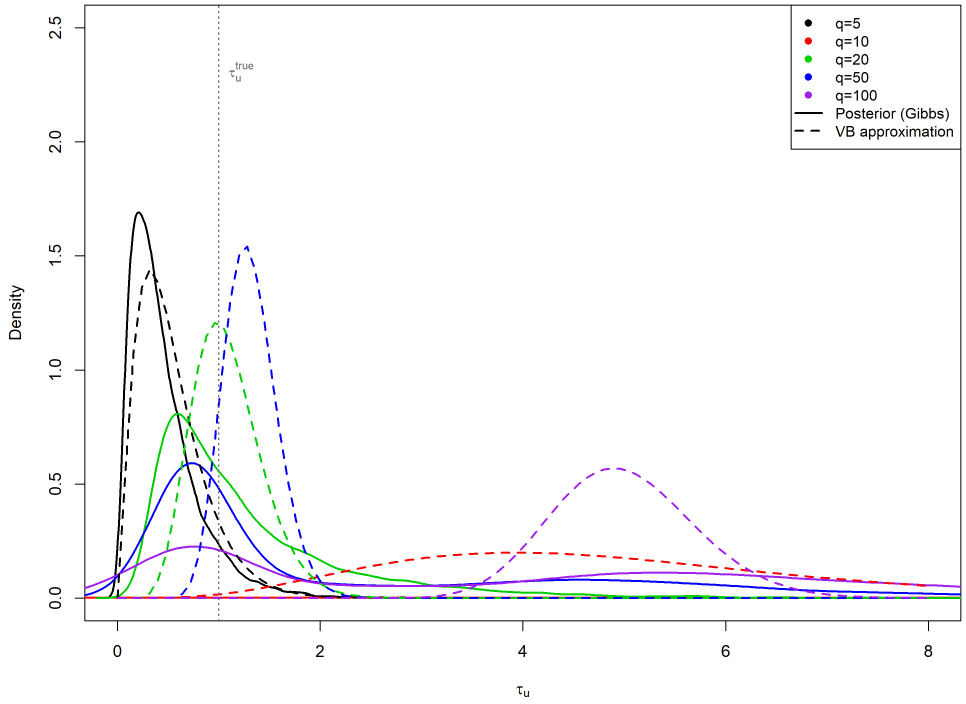
```
  curve(dgamma(x, vb_params$q5$a, vb_params$q5$b),
        add = TRUE, lty = 2, lwd = 2, col = colors[1])
  curve(dgamma(x, vb_params$q10$a, vb_params$q10$b),
        add = TRUE, lty = 2, lwd = 2, col = colors[2])
  curve(dgamma(x, vb_params$q20$a, vb_params$q20$b),
        add = TRUE, lty = 2, lwd = 2, col = colors[3])
  curve(dgamma(x, vb_params$q50$a, vb_params$q50$b),
        add = TRUE, lty = 2, lwd = 2, col = colors[4])
  curve(dgamma(x, vb_params$q100$a, vb_params$q100$b),
        add = TRUE, lty = 2, lwd = 2, col = colors[5])

  # Legend
  legend('topright',
         col = c(colors, "black", "black"),
         lty = c(0, 0, 0, 0, 0, 1, 2),
         lwd = c(NA, NA, NA, NA, NA, 2, 2),
         pch = c(19, 19, 19, 19, 19, NA, NA),
         legend = c('q=5', 'q=10', 'q=20', 'q=50', 'q=100', 'Posterior (Gibbs)', 'VB approximation'),
         cex = 0.9)
} else {
  # VB only version
  curve(dgamma(x, vb_params$q5$a, vb_params$q5$b),
        from = 0, to = 8,
        xlab = expression(tau['u']),
        ylab = "Density",
        main = '',
        ylim = c(0, 2.5),
        lwd = 2,
        col = colors[1])

  curve(dgamma(x, vb_params$q10$a, vb_params$q10$b),
        add = TRUE, lwd = 2, col = colors[2])
  curve(dgamma(x, vb_params$q20$a, vb_params$q20$b),
        add = TRUE, lwd = 2, col = colors[3])
  curve(dgamma(x, vb_params$q50$a, vb_params$q50$b),
        add = TRUE, lwd = 2, col = colors[4])
  curve(dgamma(x, vb_params$q100$a, vb_params$q100$b),
        add = TRUE, lwd = 2, col = colors[5])

  legend('topright',
         col = colors,
         lty = 1,
         lwd = 2,
         legend = c('q=5', 'q=10', 'q=20', 'q=50', 'q=100'),
         cex = 0.9)
}

# Add vertical line at true value
abline(v = tau_u_true, lty = 3, col = "gray40", lwd = 1.5)
text(tau_u_true, 2.3, labels = expression(tau[u]^true), pos = 4, cex = 0.9, col = "gray40")

dev.off()
```

```
## pdf
```

```
##   2
```

```
cat("My plot saved to figs/M2_my_tau_u_comparison.png\n")
```

```
## My plot saved to figs/M2_my_tau_u_comparison.png
```




```
# Load Dr John's reference results from RDS
cat("\n=========================================\n")
```

```
##
## =========================================
```

```
cat("Loading Dr John's Reference Data (RDS)\n")
```

```
## Loading Dr John's Reference Data (RDS)
```

```
cat("=========================================\n\n")
```

```
## =========================================
```

```
# Use absolute path to ensure it works during knitting
project_root <- "d:/github/VI1"
rds_path <- file.path(project_root, "results", "dr_john_reference_tau_u.rds")

if (file.exists(rds_path)) {
  cat("Loading:", rds_path, "\n")
  dr_john_ref <- readRDS(rds_path)

  # Check structure
  cat("\nVB structure check:\n")
  cat("  q5 VB class:", class(dr_john_ref$vb$q5), "\n")
  if (is.list(dr_john_ref$vb$q5)) {
```

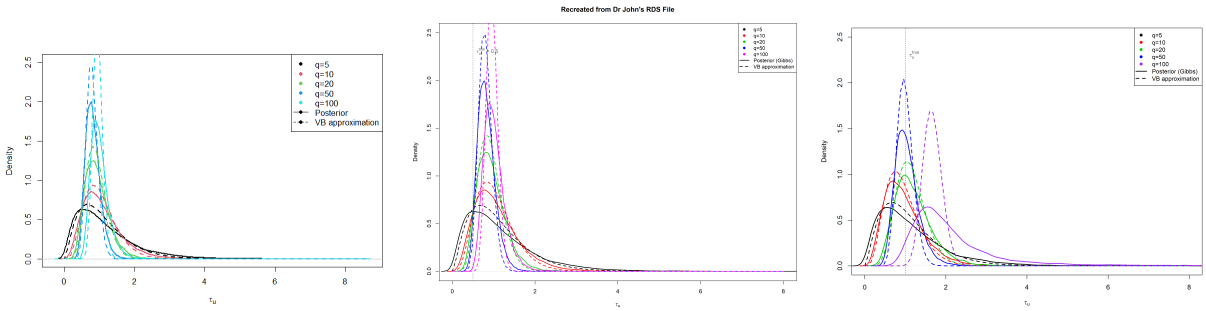
```
  cat("  q5 VB names:", names(dr_john_ref$vb$q5), "\n")
  cat("  q5$a =", dr_john_ref$vb$q5$a, "  q5$b =", dr_john_ref$vb$q5$b, "\n")
} else {
  cat("  q5 VB values:", dr_john_ref$vb$q5, "\n")
}

cat("\nCreating plot from Dr John's saved RDS data...\n")

colors_rds <- c("black", "red", "green3", "blue", "magenta")

# Save plot to PNG file
png(filename = file.path(project_root, "figs", "my_from_rds_tau_u_comparison.png"),
    width = 3500, height = 2800, res = 300)

plot(density(dr_john_ref$gibbs$q5),
     xlab=expression(tau['u']),
     main='Recreated from Dr John\'s RDS File',
     ylim=c(0, 2.5),
     xlim=c(0, 8),
     lwd=2,
     col=colors_rds[1])

lines(density(dr_john_ref$gibbs$q10), col=colors_rds[2], lwd=2)
lines(density(dr_john_ref$gibbs$q20), col=colors_rds[3], lwd=2)
lines(density(dr_john_ref$gibbs$q50), col=colors_rds[4], lwd=2)
lines(density(dr_john_ref$gibbs$q100), col=colors_rds[5], lwd=2)

# Add VB approximations - check if data is vector or list
if (is.list(dr_john_ref$vb$q5)) {
  # VB stored as list(a=, b=)
  curve(dgamma(x, dr_john_ref$vb$q5$a, dr_john_ref$vb$q5$b),
        add = TRUE, lty = 2, lwd = 2, col = colors_rds[1])
  curve(dgamma(x, dr_john_ref$vb$q10$a, dr_john_ref$vb$q10$b),
        add = TRUE, lty = 2, lwd = 2, col = colors_rds[2])
  curve(dgamma(x, dr_john_ref$vb$q20$a, dr_john_ref$vb$q20$b),
        add = TRUE, lty = 2, lwd = 2, col = colors_rds[3])
  curve(dgamma(x, dr_john_ref$vb$q50$a, dr_john_ref$vb$q50$b),
        add = TRUE, lty = 2, lwd = 2, col = colors_rds[4])
  curve(dgamma(x, dr_john_ref$vb$q100$a, dr_john_ref$vb$q100$b),
        add = TRUE, lty = 2, lwd = 2, col = colors_rds[5])
} else {
  # VB stored as vector c(a, b)
  curve(dgamma(x, dr_john_ref$vb$q5[1], dr_john_ref$vb$q5[2]),
        add = TRUE, lty = 2, lwd = 2, col = colors_rds[1])
  curve(dgamma(x, dr_john_ref$vb$q10[1], dr_john_ref$vb$q10[2]),
        add = TRUE, lty = 2, lwd = 2, col = colors_rds[2])
  curve(dgamma(x, dr_john_ref$vb$q20[1], dr_john_ref$vb$q20[2]),
        add = TRUE, lty = 2, lwd = 2, col = colors_rds[3])
  curve(dgamma(x, dr_john_ref$vb$q50[1], dr_john_ref$vb$q50[2]),
        add = TRUE, lty = 2, lwd = 2, col = colors_rds[4])
  curve(dgamma(x, dr_john_ref$vb$q100[1], dr_john_ref$vb$q100[2]),
        add = TRUE, lty = 2, lwd = 2, col = colors_rds[5])
}
```

```
  legend('topright',
         col=c(colors_rds, "black", "black"),
         lty=c(0,0,0,0,0,1,2),
         lwd=c(NA,NA,NA,NA,NA,2,2),
         pch=c(19,19,19,19,19,NA,NA),
         legend=c('q=5', 'q=10', 'q=20', 'q=50', 'q=100', 'Posterior (Gibbs)', 'VB approximation'),
         cex=0.9)

  abline(v=0.5, lty=3, col="gray40", lwd=1.5)
  text(0.5, 2.3, labels=expression(tau[u]^true*" = 0.5"), pos=4, cex=0.9, col="gray40")

  dev.off()

  cat("Plot from RDS saved to figs/M2_my_from_rds_tau_u_comparison.png\n")

} else {
  cat("WARNING: RDS file not found at:", rds_path, "\n")
  cat("Run 'Code for David Ewing Random intercept example.R' to generate it.\n")
}
```

```
## Loading: d:/github/VI1/results/dr_john_reference_tau_u.rds
##
## VB structure check:
##   q5 VB class: numeric
##   q5 VB values: 2.5 2.249205
##
## Creating plot from Dr John's saved RDS data...
## Plot from RDS saved to figs/M2_my_from_rds_tau_u_comparison.png
```

4.1 Three-Panel Validation Comparison

Panel status: - Dr John's baseline PNG: - My plot from RDS: - My new run with matched params:



Left: Dr John's baseline PNG (from his .R file)
Centre: My recreation from his RDS data
Right: My new run with matched parameters

```
# Diagnostic: Posterior variance / Prior variance ratio for u's
# As requested by Dr John [0:15:49]
# "When you do badly with tau_u, this ratio will be high. When you do well, this ratio will be low."

cat("\nDiagnostic: Var_posterior(u) / Var_prior(u)\n")
```

```
##
## Diagnostic: Var_posterior(u) / Var_prior(u)
```

```
cat("Diagnostic: Var_posterior(u) / Var_prior(u)\n")
```

```
## Diagnostic: Var_posterior(u) / Var_prior(u)
```

```
# Prior variance: u_i ~ N(0, 1/tau_u_true)
var_prior_u <- 1 / tau_u_true

# Calculate ratio for each configuration
ratio_data <- data.frame()

for (i in seq_along(results_multi)) {
  result <- results_multi[[i]]
  config <- result$config
  q <- config$q

  # VB posterior variances: diagonal of Sigma_betau for u's
  Sigma_betau <- result$vb$Sigma_betau
  var_post_vb_u <- diag(Sigma_betau)[(p+1):(p+q)]
  mean_ratio_vb <- mean(var_post_vb_u / var_prior_u)

  # Gibbs posterior variances if available
  if (!is.null(result$gibbs)) {
    var_post_gibbs_u <- sapply(1:q, function(j) {
      var(result$gibbs[, paste0("u", j)])
    })
    mean_ratio_gibbs <- mean(var_post_gibbs_u / var_prior_u)
  } else {
    mean_ratio_gibbs <- NA
  }

  # Store results
  ratio_data <- rbind(ratio_data, data.frame(
    Q = q,
    n_per_group = config$nq,
    VB_ratio = mean_ratio_vb,
    Gibbs_ratio = mean_ratio_gibbs,
    label = config$label
  ))
}

print(ratio_data)
```

```
##     Q n_per_group  VB_ratio Gibbs_ratio                 label
## 1   5          60 0.5258138  0.46911554  Q=5 (n=60 per group)
## 2  10          30 0.1302980  0.05774511 Q=10 (n=30 per group)
## 3  20          15 0.3932546  0.46209526 Q=20 (n=15 per group)
## 4  50           6 0.5272231  0.61278275  Q=50 (n=6 per group)
## 5 100           3 0.1893677  0.48334537 Q=100 (n=3 per group)
```

```
cat("\nInterpretation:\n")
```

```
##
## Interpretation:
```

```
cat("- Lower ratio = narrower posteriors = more information learnt\n")
```

```
## - Lower ratio = narrower posteriors = more information learnt
```

```
cat("- Narrow posteriors for u → better tau_u estimation in VB\n")
```

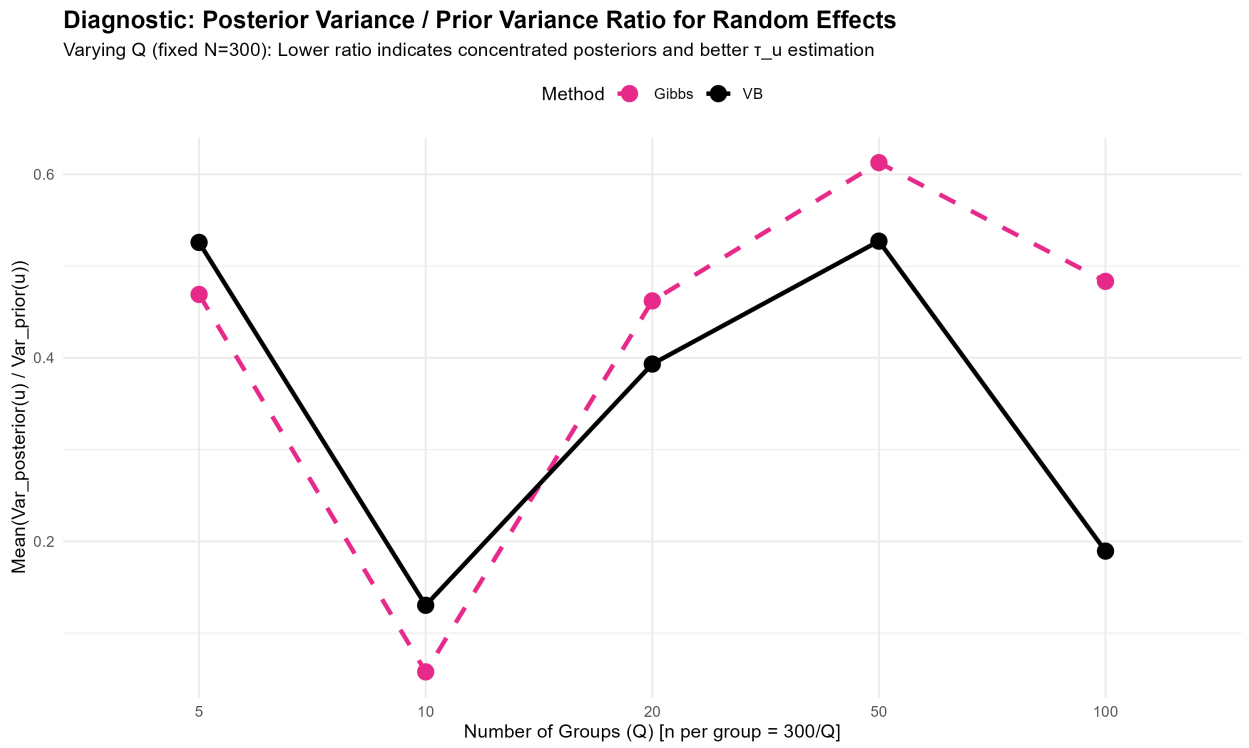
```
## - Narrow posteriors for u → better tau_u estimation in VB
```

```
cat("- As n_per_group increases, VB ratio decreases (posteriors concentrate)\n\n")
```

```
## - As n_per_group increases, VB ratio decreases (posteriors concentrate)
```

```
# Prepare data for plotting
plot_data <- data.frame(
  Q  = ratio_data$Q,
  VB = ratio_data$VB_ratio
)

if (run_gibbs) {
  plot_data$Gibbs <- ratio_data$Gibbs_ratio
  plot_data_long <- tidyr::pivot_longer(plot_data, cols = c(VB, Gibbs),
                                        names_to = "Method", values_to = "Ratio")
} else {
  plot_data_long <- data.frame(
    Q = plot_data$Q,
    Method = "VB",
    Ratio = plot_data$VB
  )
}

# Create diagnostic plot
p_diagnostic <- ggplot(plot_data_long, aes(x = factor(Q), y = Ratio, color = Method, group = Method)) +
  geom_point(size = 4) +
  geom_line(aes(linetype = Method), size = 1.2) +
  scale_color_manual(
    values = c("VB" = "black", "Gibbs" = "#E7298A")
  ) +
  scale_linetype_manual(
    values = c("VB" = "solid", "Gibbs" = "dashed")
  ) +
  labs(
    title = "Diagnostic: Posterior Variance / Prior Variance Ratio for Random Effects",
    subtitle = "Varying Q (fixed N=300): Lower ratio indicates concentrated posteriors and better _u es
    x = "Number of Groups (Q) [n per group = 300/Q]",
    y = "Mean(Var_posterior(u) / Var_prior(u))",
    color = "Method"
  ) +
  theme_minimal() +
  theme(
    legend.position = "top",
    plot.title = element_text(size = 14, face = "bold"),
    plot.subtitle = element_text(size = 11)
  )

ggsave(
  filename = "../figs/M3_diagnostic_variance_ratio.png",
```

```
  plot = p_diagnostic,
  width = 10,
  height = 6,
  dpi = 300
)

img_diagnostic <- readPNG("../figs/M3_diagnostic_variance_ratio.png")
grid.newpage()
grid.raster(img_diagnostic)
```




```
cat("\n========================================\n")
```

```
##
## ========================================
```

```
cat("Key Finding (Dr John's insight):\n")
```

```
## Key Finding (Dr John's insight):
```

```
cat("As n per group increases (Q decreases from 50→5),\n")
```

```
## As n per group increases (Q decreases from 50→5),
```

```
cat("VB posteriors for u become narrower (ratio decreases),\n")
```

```
## VB posteriors for u become narrower (ratio decreases),
```

```
cat("leading to better tau_u estimation.\n")
```

```
## leading to better tau_u estimation.
```

```
cat("========================================\n")
```

```
## ========================================
```

```
##
## Saved M3 SD ratios for 5 configurations
```