Variational Bayes: Unified M1/M3 Analysis

VI-UNIFIED-VBGibbsOnly.Rmd

Contents

```r
# ============================================================================
# MODEL TYPE SELECTION
# M1: Linear regression with beta and sigma^2 (residual variance)
#     - No random effects, no variance component

# M3: Hierarchical model with beta, u (random effects), tau_e, tau_u
#     - Demonstrates variance component under-dispersion
#     - Two scenarios: 30 sparse groups vs 6 rich groups
model_type <- "M3"  # "M1" for linear regression, "M3" for hierarchical

# SHARED PARAMETERS (both scenarios)
n <- 500  # Total observations (Dr John's Model 3 design)
p <- 4    # Number of fixed effects (beta_0, beta_1, beta_2, beta_3)

# True Parameter Values
tau_e_true <- 5
tau_u_true <- if (model_type == "M1") NULL else 0.5  # Lower = more variance between groups
beta_true  <- c(5, 0.5, 0.5, 0.5)

# Prior Hyperparameters
alpha_e <- 0.01
gamma_e <- 0.01
alpha_u <- 1.0   # Prior for tau_u centred at 0.5
gamma_u <- 2.0   # Mean = alpha_u / gamma_u = 1.0 / 2.0 = 0.5

# 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 <- TRUE  # TRUE to show function code, FALSE to hide

# Define scenarios: Q (number of groups) and nq (observations per group)
# N=500 constant, vary Q = 5, 10, 20, 50 (all divide evenly into 500)
# Only used when model_type == "M3"
scenarios <- if (model_type == "M1") {
  list(list(q = 0, nq = 0, name = "Linear Regression"))
} else {
  list(
    list(q = 5,  nq = 100, name = "Scenario 1: Q=5 (n=100 per group)"),
    list(q = 10, nq = 50,  name = "Scenario 2: Q=10 (n=50 per group)"),
    list(q = 20, nq = 25,  name = "Scenario 3: Q=20 (n=25 per group)"),
    list(q = 50, nq = 10,  name = "Scenario 4: Q=50 (n=10 per group)")
  )
}

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

```r
# Full Gibbs sampler (MCMC gold standard)
# - M1: beta | tau_e, y  and  tau_e | beta, y
# - M3: (beta,u) | tau_e,tau_u,y  and  tau_e,tau_u | (beta,u),y
# - Returns post-burnin samples for posterior inference
run_gibbs_sampler <- function(X, Z, y, p, q, n, alpha_e, gamma_e, alpha_u, gamma_u,
                              model_type = "M3", n_iter = 5000, n_burnin = 1000) {

  set.seed(82171165)

  if (model_type == "M1") {
    # M1: Sample beta | tau_e, y and tau_e | beta, y
    n_save <- n_iter - n_burnin
    samples <- matrix(NA, nrow = n_save, ncol = p + 1)

    tau_e <- alpha_e / gamma_e
    beta <- rep(0, p)

    XtX <- t(X) %*% X
    Xty <- t(X) %*% y

    for (iter in 1:n_iter) {
      # Sample beta | tau_e, y
      Sigma_beta <- solve(tau_e * XtX + diag(p) / 100)
      mu_beta <- tau_e * Sigma_beta %*% Xty
      beta <- MASS::mvrnorm(1, mu_beta, Sigma_beta)

      # Sample tau_e | beta, y
      residuals <- y - X %*% beta
      a_post <- alpha_e + n / 2
      b_post <- gamma_e + 0.5 * sum(residuals^2)
      tau_e <- rgamma(1, shape = a_post, rate = b_post)

      if (iter > n_burnin) {
        samples[iter - n_burnin, ] <- c(beta, tau_e)
      }
    }

    colnames(samples) <- c(paste0("beta", 0:(p-1)), "tau_e")

  } else {
    # M3: Sample (beta, u) | tau_e, tau_u, y and tau_e, tau_u | (beta, u), y
    n_save <- n_iter - n_burnin
    samples <- matrix(NA, nrow = n_save, ncol = p + q + 2)

    K <- diag(q)
    K_inv <- solve(K)
    XZ <- cbind(X, Z)
    XZtXZ <- t(XZ) %*% XZ
    XZty <- t(XZ) %*% y

    penalty_matrix <- rbind(
      cbind(matrix(0, p, p), matrix(0, p, q)),
      cbind(matrix(0, q, p), K_inv)
```

```
    )

    tau_e <- alpha_e / gamma_e
    tau_u <- alpha_u / gamma_u
    betau <- rep(0, p + q)

    for (iter in 1:n_iter) {
      # Sample (beta, u) | tau_e, tau_u, y
      Sigma_betau <- solve(tau_e * XZtXZ + tau_u * penalty_matrix + diag(p + q) / 100)
      mu_betau <- tau_e * Sigma_betau %*% XZty
      betau <- MASS::mvrnorm(1, mu_betau, Sigma_betau)

      beta <- betau[1:p]
      u <- betau[(p + 1):(p + q)]

      # Sample tau_e | (beta, u), y
      residuals <- y - XZ %*% betau
      a_e_post <- alpha_e + n / 2
      b_e_post <- gamma_e + 0.5 * sum(residuals^2)
      tau_e <- rgamma(1, shape = a_e_post, rate = b_e_post)

      # Sample tau_u | u
      a_u_post <- alpha_u + q / 2
      b_u_post <- gamma_u + 0.5 * as.numeric(t(u) %*% K_inv %*% u)
      tau_u <- rgamma(1, shape = a_u_post, rate = b_u_post)

      if (iter > n_burnin) {
        samples[iter - n_burnin, ] <- c(beta, u, tau_e, tau_u)
      }
    }

    colnames(samples) <- c(paste0("beta", 0:(p-1)), paste0("u", 1:q), "tau_e", "tau_u")
  }

  return(samples)
}
```

```
# plot_vb_posteriors_gibbs_only.R
# Function to create VB vs Gibbs posterior comparison plots
plot_vb_posteriors <- function(mu_beta, Sigma_betau,
                               gibbs_samples, p, q, beta_true, u_true,
                               tau_e_true, tau_u_true, E_tau_e, E_tau_u,
                               a_e_new, b_e_new, a_u_new, b_u_new,
                               gibbs_tau_e, gibbs_tau_u, run_gibbs, model_type = "M3") {

  beta_means <- mu_beta[1:p]
  mu_u <- mu_beta[(p+1):(p+q)]

  plot_list <- list()
  plot_idx <- 1

  # Beta parameters
  for (i in 1:p) {
    if (run_gibbs) {
```

```
    gibbs_col_name <- paste0('beta', i-1)
    gibbs_range <- range(density(gibbs_samples[, gibbs_col_name])$x)
    vb_width <- 3*sqrt(Sigma_betau[i,i])
    x_min <- max(gibbs_range[1], beta_means[i] - 2*vb_width)
    x_max <- min(gibbs_range[2], beta_means[i] + 2*vb_width)
  } else {
    vb_sd <- sqrt(Sigma_betau[i,i])
    if (!is.finite(vb_sd) || vb_sd <= 0) {
      cat(sprintf("Warning: Invalid SD for beta[%d]: %f\n", i-1, vb_sd))
      vb_sd <- 1
    }
    x_min <- beta_means[i] - 3*vb_sd
    x_max <- beta_means[i] + 3*vb_sd
  }

  if (!is.finite(x_min) || !is.finite(x_max)) {
    cat(sprintf("Error: Non-finite range for beta[%d]: [%f, %f]\n", i-1, x_min, x_max))
    cat(sprintf("  beta_means[%d] = %f, SD = %f\n", i, beta_means[i], sqrt(Sigma_betau[i,i])))
    next
  }

  x_seq <- seq(x_min, x_max, length = 200)

  vb_data <- data.frame(
    x       = x_seq,
    density = dnorm(x_seq, beta_means[i], sqrt(Sigma_betau[i,i])),
    method  = "VB"
  )

  if (run_gibbs) {
    gibbs_density <- density(gibbs_samples[, gibbs_col_name])
    gibbs_interp <- approx(gibbs_density$x, gibbs_density$y, xout = x_seq, rule = 2)
    gibbs_data <- data.frame(
      x       = x_seq,
      density = gibbs_interp$y,
      method  = "Gibbs"
    )
    combined_data <- rbind(vb_data, gibbs_data)
    color_values <- c("VB" = "blue", "Gibbs" = "orange")
    linetype_values <- c("VB" = "solid", "Gibbs" = "dotdash")
    color_breaks <- c("VB", "Gibbs")
    label_text <- sprintf("VB/Gibbs: %.3f",
                          sqrt(Sigma_betau[i,i]) / sd(gibbs_samples[, gibbs_col_name]))
    plot_title <- bquote(paste("VB vs Gibbs for ", beta[.(i-1)]))
  } else {
    combined_data <- vb_data
    color_values <- c("VB" = "blue")
    linetype_values <- c("VB" = "solid")
    color_breaks <- c("VB")
    label_text <- ""
    plot_title <- bquote(paste("VB posterior for ", beta[.(i-1)]))
  }
```

```
  p_temp <- ggplot(
    combined_data,
    aes(
      x        = x,
      y        = density,
      color    = method,
      linetype = method)) +
    geom_line(aes(linewidth = method)) +
    scale_linewidth_manual(values = c("VB" = 1.2, "Gibbs" = 2.4), guide = "none") +
    geom_vline(
      xintercept = beta_true[i],
      color      = "darkgreen",
      linetype   = "dotted",
      linewidth  = 1) +
    scale_color_manual(
      name   = NULL,
      values = color_values,
      breaks = color_breaks) +
    scale_linetype_manual(
      name   = NULL,
      values = linetype_values,
      breaks = color_breaks) +
    labs(
      x     = bquote(beta[.(i-1)]),
      y     = "Density",
      title = plot_title) +
    theme_minimal() +
    theme(
      legend.position = "top",
      plot.title      = element_text(hjust = 0.5),
      panel.border    = element_rect(color = "black", fill = NA, linewidth = 1))

  if (run_gibbs && label_text != "") {
    p_temp <- p_temp + annotate(
      "text",
      x     = Inf,
      y     = Inf,
      label = label_text,
      hjust = 1.05,
      vjust = 2,
      size  = 3,
      color = "black")
  }

  plot_list[[plot_idx]] <- p_temp
  plot_idx <- plot_idx + 1
}

# u_1 and u_2 posteriors
for (u_idx in 1:2) {
  if (run_gibbs) {
    gibbs_col <- paste0('u', u_idx)
    gibbs_range <- range(density(gibbs_samples[, gibbs_col])$x)
```

```
  vb_width <- 3*sqrt(Sigma_betau[p+u_idx, p+u_idx])
  x_min <- max(gibbs_range[1], mu_u[u_idx] - 2*vb_width)
  x_max <- min(gibbs_range[2], mu_u[u_idx] + 2*vb_width)
} else {
  x_min <- mu_u[u_idx] - 3*sqrt(Sigma_betau[p+u_idx, p+u_idx])
  x_max <- mu_u[u_idx] + 3*sqrt(Sigma_betau[p+u_idx, p+u_idx])
}
x_seq <- seq(x_min, x_max, length = 200)

vb_data <- data.frame(
  x       = x_seq,
  density = dnorm(x_seq, mu_u[u_idx], sqrt(Sigma_betau[p+u_idx, p+u_idx])),
  method  = "VB"
)

if (run_gibbs) {
  gibbs_u_density <- density(gibbs_samples[, gibbs_col])
  gibbs_interp <- approx(gibbs_u_density$x, gibbs_u_density$y, xout = x_seq, rule = 2)
  gibbs_data <- data.frame(
    x       = x_seq,
    density = gibbs_interp$y,
    method  = "Gibbs"
  )
  combined_data <- rbind(vb_data, gibbs_data)
  color_values <- c("VB" = "blue", "Gibbs" = "orange")
  linetype_values <- c("VB" = "solid", "Gibbs" = "dotdash")
  color_breaks <- c("VB", "Gibbs")
  label_text <- sprintf("VB/Gibbs: %.3f",
                        sqrt(Sigma_betau[p+u_idx, p+u_idx]) / sd(gibbs_samples[, gibbs_col]))
  plot_title <- bquote(paste("VB vs Gibbs for ", u[.(u_idx)]))
} else {
  combined_data <- vb_data
  color_values <- c("VB" = "blue")
  linetype_values <- c("VB" = "solid")
  color_breaks <- "VB"
  label_text <- ""
  plot_title <- bquote(paste("VB for ", u[.(u_idx)]))
}

p_u <- ggplot(
  combined_data,
  aes(
    x        = x,
    y        = density,
    color    = method,
    linetype = method)) +
  geom_line(aes(linewidth = method)) +
  scale_linewidth_manual(values = c("VB" = 1.2, "Gibbs" = 2.4), guide = "none") +
  geom_vline(
    xintercept = u_true[u_idx],
    color      = "darkgreen",
    linetype   = "dotted",
    linewidth  = 1) +
```

```
    scale_color_manual(
      name   = NULL,
      values = color_values,
      breaks = color_breaks) +
    scale_linetype_manual(
      name   = NULL,
      values = linetype_values,
      breaks = color_breaks) +
    labs(
      x     = bquote(u[.(u_idx)]),
      y     = "Density",
      title = plot_title) +
    theme_minimal() +
    theme(
      legend.position = "top",
      plot.title      = element_text(hjust = 0.5),
      panel.border    = element_rect(color = "black", fill = NA, linewidth = 1))

  if (run_gibbs && label_text != "") {
    p_u <- p_u + annotate(
      "text",
      x     = Inf,
      y     = Inf,
      label = label_text,
      hjust = 1.05,
      vjust = 2,
      size  = 3,
      color = "black")
  }

  plot_list[[plot_idx]] <- p_u
  plot_idx <- plot_idx + 1
}

# tau_e and tau_u posteriors
for (tau_name in c("tau_e", "tau_u")) {
  if (tau_name == "tau_e") {
    E_tau <- E_tau_e
    a_new <- a_e_new
    b_new <- b_e_new
    tau_true <- tau_e_true
    gibbs_tau <- gibbs_tau_e
  } else {
    E_tau <- E_tau_u
    a_new <- a_u_new
    b_new <- b_u_new
    tau_true <- tau_u_true
    gibbs_tau <- gibbs_tau_u
  }

  if (run_gibbs) {
    gibbs_range <- range(density(gibbs_tau)$x)
    vb_width <- 3*sqrt(a_new/b_new^2)
```

```r
  x_min <- max(0, gibbs_range[1], E_tau - 2*vb_width)
  x_max <- min(gibbs_range[2], E_tau + 2*vb_width)
} else {
  x_min <- max(0, E_tau - 3*sqrt(a_new/b_new^2))
  x_max <- E_tau + 3*sqrt(a_new/b_new^2)
}
x_seq <- seq(x_min, x_max, length = 200)

vb_data <- data.frame(
  x       = x_seq,
  density = dgamma(x_seq, shape = a_new, rate = b_new),
  method  = "VB"
)

if (run_gibbs) {
  gibbs_tau_density <- density(gibbs_tau)
  gibbs_interp <- approx(gibbs_tau_density$x, gibbs_tau_density$y, xout = x_seq, rule = 2)
  gibbs_data <- data.frame(
    x       = x_seq,
    density = gibbs_interp$y,
    method  = "Gibbs"
  )
  combined_data <- rbind(vb_data, gibbs_data)
  color_values <- c("VB" = "blue", "Gibbs" = "orange")
  linetype_values <- c("VB" = "solid", "Gibbs" = "dotdash")
  color_breaks <- c("VB", "Gibbs")
  label_text <- sprintf("VB/Gibbs: %.3f", sqrt(a_new/b_new^2) / sd(gibbs_tau))
  if (tau_name == "tau_e") {
    plot_title <- expression(paste("VB vs Gibbs for ", tau[e]))
  } else {
    plot_title <- expression(paste("VB vs Gibbs for ", tau[u]))
  }
} else {
  combined_data <- vb_data
  color_values <- c("VB" = "blue")
  linetype_values <- c("VB" = "solid")
  color_breaks <- "VB"
  label_text <- ""
  if (tau_name == "tau_e") {
    plot_title <- expression(paste("VB for ", tau[e]))
  } else {
    plot_title <- expression(paste("VB for ", tau[u]))
  }
}

p_tau <- ggplot(
  combined_data,
  aes(
    x        = x,
    y        = density,
    color    = method,
    linetype = method)) +
  geom_line(aes(linewidth = method)) +
```

```
      scale_linewidth_manual(values = c("VB" = 1.2, "Gibbs" = 2.4), guide = "none") +
      geom_vline(
        xintercept = tau_true,
        color      = "darkgreen",
        linetype   = "dotted",
        linewidth  = 1) +
      scale_color_manual(
        name   = NULL,
        values = color_values,
        breaks = color_breaks) +
      scale_linetype_manual(
        name   = NULL,
        values = linetype_values,
        breaks = color_breaks) +
      labs(
        x     = ifelse(tau_name == "tau_e", expression(tau[e]), expression(tau[u])),
        y     = "Density",
        title = plot_title) +
      theme_minimal() +
      theme(
        legend.position = "top",
        plot.title      = element_text(hjust = 0.5),
        panel.border    = element_rect(color = "black", fill = NA, linewidth = 1))

    if (run_gibbs && label_text != "") {
      p_tau <- p_tau + annotate(
        "text",
        x     = Inf,
        y     = Inf,
        label = label_text,
        hjust = 1.05,
        vjust = 2,
        size  = 3,
        color = "black")
    }

    plot_list[[plot_idx]] <- p_tau
    plot_idx <- plot_idx + 1
  }

  # Combine all plots
  total_plots <- p + 4
  n_cols <- 2
  combined_plot <- wrap_plots(plot_list, ncol = n_cols)

  return(combined_plot)
}
```

```
# Mean-field VB with coordinate ascent
# - Updates: q(beta,u), q(tau_e), q(tau_u) iteratively
# - Tracks ELBO for convergence monitoring
# - Returns variational posterior parameters
run_vb_algorithm <- function(X, Z, y, K, p, q, n, alpha_e, gamma_e, alpha_u, gamma_u,
                             model_type = "M3", max_iter = 100, tol = 1e-5) {
```

```r
if (model_type == "M1") {
  penalty_matrix <- matrix(0, p, p)
  XZ      <- X
  XZ_t_XZ <- t(X) %*% X
  XZ_t_y  <- t(X) %*% y
} else {
  K_inv          <- solve(K)
  penalty_matrix <- rbind(
    cbind(matrix(0, p, p), matrix(0, p, q)),
    cbind(matrix(0, q, p), K_inv)
  )
  XZ      <- cbind(X, Z)
  XZ_t_XZ <- t(XZ) %*% XZ
  XZ_t_y  <- t(XZ) %*% y
}

E_tau_e <- alpha_e / gamma_e
E_tau_u <- alpha_u / gamma_u

E_tau_e_history <- numeric(max_iter)
E_tau_u_history <- numeric(max_iter)
elbo_history    <- numeric(max_iter)

mu_betau_old    <- NA
Sigma_betau_old <- NA

for (iter in 1:max_iter) {

  precision_betau <- E_tau_e * XZ_t_XZ + E_tau_u * penalty_matrix
  Sigma_betau     <- solve(precision_betau)
  mu_betau        <- E_tau_e * Sigma_betau %*% XZ_t_y

  mu_beta  <- mu_betau[1:p]
  Sigma_beta <- Sigma_betau[1:p, 1:p]

  if (model_type == "M3") {
    mu_u     <- mu_betau[(p+1):(p+q)]
    Sigma_uu <- Sigma_betau[(p+1):(p+q), (p+1):(p+q)]
  } else {
    mu_u <- NULL
    Sigma_uu <- NULL
  }

  residuals <- y - XZ %*% mu_betau
  SSR       <- sum(residuals^2)
  trace_e   <- sum(diag(XZ_t_XZ %*% Sigma_betau))

  a_e_new <- alpha_e + n/2
  b_e_new <- gamma_e + 0.5 * (SSR + trace_e)

  E_tau_e_old <- E_tau_e
  E_tau_e     <- a_e_new / b_e_new
```

```
  if (model_type == "M3") {
    quad_form <- as.numeric(t(mu_u) %*% K_inv %*% mu_u)
    trace_u   <- sum(diag(K_inv %*% Sigma_uu))
    a_u_new <- alpha_u + q/2
    b_u_new <- gamma_u + 0.5 * (quad_form + trace_u)
    E_tau_u_old <- E_tau_u
    E_tau_u     <- a_u_new / b_u_new
  } else {
    quad_form <- 0
    trace_u   <- 0
    a_u_new <- alpha_u
    b_u_new <- gamma_u
    E_tau_u_old <- 0
    E_tau_u     <- 0
  }

  E_log_p_y <- -0.5*n*log(2*pi) +  # Expected log-likelihood (Gaussian with precision tau_e)
               0.5*n*(digamma(a_e_new) - log(b_e_new)) -
               0.5*E_tau_e*(SSR + trace_e)

  if (model_type == "M3") {
    E_log_p_betau <- -0.5*q*log(2*pi) +  # Expected log-prior for beta and u (random effects with tau
                     0.5*q*(digamma(a_u_new) - log(b_u_new)) -
                     0.5*E_tau_u*(quad_form + trace_u)
    E_log_p_tau_u <- alpha_u*log(gamma_u) - lgamma(alpha_u) +  # Expected log-prior for tau_u (Gamma
                     (alpha_u-1)*(digamma(a_u_new) - log(b_u_new)) -
                     gamma_u*E_tau_u
    entropy_tau_u <- a_u_new - log(b_u_new) + lgamma(a_u_new) +  # Entropy of q(tau_u) ~ Gamma(a_u, b
                     (1-a_u_new)*digamma(a_u_new)
  } else {
    E_log_p_betau <- 0
    E_log_p_tau_u <- 0
    entropy_tau_u <- 0
  }

  E_log_p_tau_e <- alpha_e*log(gamma_e) - lgamma(alpha_e) +  # Expected log-prior for tau_e (Gamma pr
                   (alpha_e-1)*(digamma(a_e_new) - log(b_e_new)) -
                   gamma_e*E_tau_e

  dim_param <- if (model_type == "M1") p else (p + q)
  entropy_betau <- 0.5*determinant(Sigma_betau, logarithm=TRUE)$modulus +  # Entropy of q(beta) or q(
                   0.5*dim_param*(1 + log(2*pi))

  entropy_tau_e <- a_e_new - log(b_e_new) + lgamma(a_e_new) +  # Entropy of q(tau_e) ~ Gamma(a_e, b_e
                   (1-a_e_new)*digamma(a_e_new)

  elbo <- E_log_p_y + E_log_p_betau + E_log_p_tau_e + E_log_p_tau_u +
          entropy_betau + entropy_tau_e + entropy_tau_u

  E_tau_e_history[iter] <- E_tau_e
  E_tau_u_history[iter] <- E_tau_u
  elbo_history[iter]    <- elbo
```

```
    if (iter > 1) {
      diff_betau <- sqrt((mu_betau - mu_betau_old)^2) / (abs(mu_betau) + 0.01)
      diff_tau_e <- abs(E_tau_e - E_tau_e_old) / (E_tau_e + 0.01)
      diff_tau_u <- abs(E_tau_u - E_tau_u_old) / (E_tau_u + 0.01)
      diff_Sigma <- sqrt((diag(Sigma_betau) - diag(Sigma_betau_old))^2) /
                    (diag(Sigma_betau) + 0.01)

      diff_all <- c(diff_betau, diff_tau_e, diff_tau_u, diff_Sigma)

      if (max(diff_all) < tol) {
        cat("Converged at iteration", iter, "\n")
        cat("Max relative change:", sprintf("%.2e", max(diff_all)), "\n")
        E_tau_e_history <- E_tau_e_history[1:iter]
        E_tau_u_history <- E_tau_u_history[1:iter]
        elbo_history    <- elbo_history[1:iter]
        break
      }
    }

    mu_betau_old    <- mu_betau
    Sigma_betau_old <- Sigma_betau
  }

  list(
    mu_betau        = mu_betau,
    Sigma_betau     = Sigma_betau,
    mu_beta         = mu_beta,
    mu_u            = mu_u,
    Sigma_beta      = Sigma_beta,
    Sigma_uu        = Sigma_uu,
    E_tau_e         = E_tau_e,
    E_tau_u         = E_tau_u,
    a_e_new         = a_e_new,
    b_e_new         = b_e_new,
    a_u_new         = a_u_new,
    b_u_new         = b_u_new,
    E_tau_e_history = E_tau_e_history,
    E_tau_u_history = E_tau_u_history,
    elbo_history    = elbo_history
  )
}
```

```
# VB parameter convergence diagnostic
# - Plots E[tau_e] and E[tau_u] vs iteration
# - Shows convergence to true values
# - Helps identify convergence issues
plot_convergence <- function(results, scenario_name, tau_e_true, tau_u_true, model_type = "M3") {

  # tau_e convergence plot
  df_e <- data.frame(
    iteration = 1:length(results$E_tau_e_history),
    value     = results$E_tau_e_history
  )
```

```
  p_e <- ggplot(df_e, aes(x = iteration, y = value)) +
    geom_line(color = "blue", linewidth = 1.2) +
    geom_hline(yintercept = tau_e_true, color = "red", linetype = "dashed", linewidth = 1.2) +
    labs(
      x     = "Iteration",
      y     = expression(E*"["*tau[e]*"]"),
      title = glue("{scenario_name}: Convergence of E[tau_e]")
    ) +
    theme_minimal() +
    theme(
      plot.title       = element_text(hjust = 0.5),
      panel.grid.minor = element_blank()
    )

  if (model_type == "M3") {
    # tau_u convergence plot
    df_u <- data.frame(
      iteration = 1:length(results$E_tau_u_history),
      value     = results$E_tau_u_history
    )

    p_u <- ggplot(df_u, aes(x = iteration, y = value)) +
      geom_line(color = "blue", linewidth = 1.2) +
      geom_hline(yintercept = tau_u_true, color = "red", linetype = "dashed", linewidth = 1.2) +
      labs(
        x     = "Iteration",
        y     = expression(E*"["*tau[u]*"]"),
        title = glue("{scenario_name}: Convergence of E[tau_u]")
      ) +
      theme_minimal() +
      theme(
        plot.title       = element_text(hjust = 0.5),
        panel.grid.minor = element_blank()
      )

    # Combine plots side by side
    combined_plot <- p_e + p_u
    return(combined_plot)
  } else {
    return(p_e)
  }
}
```

```
# ELBO trajectory plot
# - Evidence Lower BOund over iterations
# - Should be monotonically increasing
# - Flattening indicates convergence
plot_elbo <- function(results, scenario_name) {
  df <- data.frame(
    iteration = 1:length(results$elbo_history),
    elbo      = results$elbo_history
  )

  p_elbo <- ggplot(df, aes(x = iteration, y = elbo)) +
```

```
    geom_line(color = "darkgreen", linewidth = 1.2) +
    labs(
      x     = "Iteration",
      y     = "ELBO",
      title = glue("{scenario_name}: ELBO Convergence")
    ) +
    theme_minimal() +
    theme(
      plot.title       = element_text(hjust = 0.5),
      panel.grid.major = element_line(color = "gray90"),
      panel.grid.minor = element_blank()
    )

  return(p_elbo)
}
```


1 Loop through all scenarios

```
# Ensure required directories exist
if (!dir.exists("../figs")) dir.create("../figs", recursive = TRUE)
if (!dir.exists("../results")) dir.create("../results", recursive = TRUE)

# Storage for all scenario results
all_results <- list()

for (scenario_idx in seq_along(scenarios)) {
  scenario <- scenarios[[scenario_idx]]
  q_current  <- scenario$q
  nq_current <- scenario$nq
  scenario_name <- scenario$name

  cat("\n\n")
  cat(glue("# {scenario_name}\n\n"))

  # ========== Data Generation ==========
  cat(glue("\n=== {scenario_name}: Q={q_current} groups (n={nq_current} per group, N={n}) ===\n"))

  X_current <- cbind(1, matrix(rnorm(n*(p-1)), nrow=n, ncol=p-1))

  if (model_type == "M3") {
    u_true_current <- rnorm(q_current, 0, 1/sqrt(tau_u_true))
    Z_current <- table(1:n, rep(1:q_current, each=nq_current))
    K_current <- diag(q_current)
    linear_predictor_current <- X_current %*% beta_true + Z_current %*% u_true_current
  } else {
    u_true_current <- NULL
    Z_current <- matrix(0, nrow=n, ncol=1)
    K_current <- matrix(1)
    linear_predictor_current <- X_current %*% beta_true
  }

  residuals_true_current <- rnorm(n, 0, 1/sqrt(tau_e_true))
```

```r
y_current <- as.vector(linear_predictor_current + residuals_true_current)

# ========== VB Algorithm ==========
cat(glue("\nRunning VB algorithm...\n"))
results_current <- run_vb_algorithm(
  X          = X_current,
  Z          = Z_current,
  y          = y_current,
  K          = K_current,
  p          = p,
  q          = q_current,
  n          = n,
  alpha_e    = alpha_e,
  gamma_e    = gamma_e,
  alpha_u    = alpha_u,
  gamma_u    = gamma_u,
  model_type = model_type
)

# ========== Gibbs Sampler ==========
if (run_gibbs) {
  cat("\nRunning Gibbs sampler...\n")
  gibbs_current <- run_gibbs_sampler(
    X          = X_current,
    Z          = Z_current,
    y          = y_current,
    p          = p,
    q          = q_current,
    n          = n,
    alpha_e    = alpha_e,
    gamma_e    = gamma_e,
    alpha_u    = alpha_u,
    gamma_u    = gamma_u,
    model_type = model_type,
    n_iter     = gibbs_iter,
    n_burnin   = gibbs_burnin
  )

  cat("Gibbs posterior means:\n")
  cat("beta:", colMeans(gibbs_current[, 1:p]), "\n")
  cat("tau_e:", mean(gibbs_current[, "tau_e"]), "\n")
  if (model_type == "M3") {
    cat("tau_u:", mean(gibbs_current[, "tau_u"]), "\n")
  }

  gibbs_tau_e_current <- gibbs_current[, "tau_e"]
  gibbs_tau_u_current <- if (model_type == "M3") gibbs_current[, "tau_u"] else NULL
} else {
  gibbs_current <- NULL
  gibbs_tau_e_current <- NULL
  gibbs_tau_u_current <- NULL
}
```

```r
# ========== Convergence Plot ==========
conv_plot <- plot_convergence(results_current, scenario_name, tau_e_true, tau_u_true, model_type)

plot_width <- if (model_type == "M1") 6 else 12
conv_filename <- glue("../figs/s{scenario_idx}_convergence.png")
ggsave(
  filename = conv_filename,
  plot     = conv_plot,
  width    = plot_width,
  height   = 6,
  dpi      = 300
)

cat(glue("\n\n```{{r echo=FALSE, fig.width={plot_width}, fig.height=6}}\n"))
cat(glue('img_conv <- readPNG("{conv_filename}")\n'))
cat("grid.newpage()\n")
cat("grid.raster(img_conv)\n")
cat("```\n\n")

# ========== ELBO Plot ==========
elbo_plot <- plot_elbo(results_current, scenario_name)

elbo_filename <- glue("../figs/s{scenario_idx}_elbo.png")
ggsave(
  filename = elbo_filename,
  plot     = elbo_plot,
  width    = 8,
  height   = 5,
  dpi      = 300
)

cat(glue("\n\n```{{r echo=FALSE, fig.width=8, fig.height=5}}\n"))
cat(glue('img_elbo <- readPNG("{elbo_filename}")\n'))
cat("grid.newpage()\n")
cat("grid.raster(img_elbo)\n")
cat("```\n\n")

# ========== Posterior Panels Plot ==========
combined_plot <- plot_vb_posteriors(
  mu_beta          = results_current$mu_betau,
  Sigma_betau      = results_current$Sigma_betau,
  gibbs_samples    = gibbs_current,
  p                = p,
  q                = q_current,
  beta_true        = beta_true,
  u_true           = u_true_current,
  tau_e_true       = tau_e_true,
  tau_u_true       = tau_u_true,
  E_tau_e          = results_current$E_tau_e,
  E_tau_u          = results_current$E_tau_u,
  a_e_new          = results_current$a_e_new,
  b_e_new          = results_current$b_e_new,
  a_u_new          = results_current$a_u_new,
```

```
    b_u_new          = results_current$b_u_new,
    gibbs_tau_e      = gibbs_tau_e_current,
    gibbs_tau_u      = gibbs_tau_u_current,
    run_gibbs        = run_gibbs,
    model_type       = model_type
  )

  total_plots <- if (model_type == "M1") (p + 1) else (p + 4)
  n_rows <- ceiling(total_plots / 2)

  panels_filename <- if (model_type == "M1") {
    "../figs/vb_linear_panels.png"
  } else {
    glue("../figs/vb_Q{q_current}_8Panel.png")
  }

  ggsave(
    filename = panels_filename,
    plot     = combined_plot,
    width    = 20,
    height   = 5 * n_rows,
    dpi      = 300
  )

  cat(glue("{model_type}: {total_plots}-panel ggplot saved for {scenario_name}\n"))

  cat(glue("\n\n```{{r echo=FALSE, fig.width=8, fig.height=11}}\n"))
  cat(glue('img_panels <- readPNG("{panels_filename}")\n'))
  cat("grid.newpage()\n")
  cat("grid.raster(img_panels)\n")
  cat("```\n\n")

  # Store results
  all_results[[scenario_idx]] <- list(
    scenario_name = scenario_name,
    q             = q_current,
    nq            = nq_current,
    results       = results_current,
    gibbs         = gibbs_current,
    u_true        = u_true_current
  )
}
```

2 Scenario 1: Q=5 (n=100 per group)

=== Scenario 1: Q=5 (n=100 per group): Q=5 groups (n=100 per group, N=500) ===Running VB algorithm…Converged at iteration 7 Max relative change: 8.66e-06

Running Gibbs sampler… Gibbs posterior means: beta: 5.140447 0.488648 0.5253486 0.4707969 tau_e: 4.62892 tau_u: 0.433533

{r echo=FALSE, fig.width=12, fig.height=6}img_conv <- readPNG("../figs/s1_convergence.png")grid.newpage grid.raster(img_conv)

{r echo=FALSE, fig.width=8, fig.height=5}img_elbo <- readPNG("../figs/s1_elbo.png")grid.newpage() grid.raster(img_elbo)

M3: 8-panel ggplot saved for Scenario 1: Q=5 (n=100 per group) {r echo=FALSE, fig.width=8, fig.height=11}img_panels <- readPNG("../figs/vb_Q5_8Panel.png")grid.newpage() grid.raster(img_panels)

3 Scenario 2: Q=10 (n=50 per group)

=== Scenario 2: Q=10 (n=50 per group): Q=10 groups (n=50 per group, N=500) ===Running VB algorithm…Converged at iteration 7 Max relative change: 2.85e-06

Running Gibbs sampler… Gibbs posterior means: beta: 5.040312 0.5110523 0.5154391 0.4882652 tau_e: 4.961034 tau_u: 0.85037

{r echo=FALSE, fig.width=12, fig.height=6}img_conv <- readPNG("../figs/s2_convergence.png")grid.newpage grid.raster(img_conv)

{r echo=FALSE, fig.width=8, fig.height=5}img_elbo <- readPNG("../figs/s2_elbo.png")grid.newpage() grid.raster(img_elbo)

M3: 8-panel ggplot saved for Scenario 2: Q=10 (n=50 per group) {r echo=FALSE, fig.width=8, fig.height=11}img_panels <- readPNG("../figs/vb_Q10_8Panel.png")grid.newpage() grid.raster(img_panels)

4 Scenario 3: Q=20 (n=25 per group)

=== Scenario 3: Q=20 (n=25 per group): Q=20 groups (n=25 per group, N=500) ===Running VB algorithm…Converged at iteration 6 Max relative change: 5.90e-06

Running Gibbs sampler… Gibbs posterior means: beta: 5.085117 0.4978812 0.4649996 0.4981764 tau_e: 4.936911 tau_u: 0.4728673

{r echo=FALSE, fig.width=12, fig.height=6}img_conv <- readPNG("../figs/s3_convergence.png")grid.newpage grid.raster(img_conv)

{r echo=FALSE, fig.width=8, fig.height=5}img_elbo <- readPNG("../figs/s3_elbo.png")grid.newpage() grid.raster(img_elbo)

M3: 8-panel ggplot saved for Scenario 3: Q=20 (n=25 per group) {r echo=FALSE, fig.width=8, fig.height=11}img_panels <- readPNG("../figs/vb_Q20_8Panel.png")grid.newpage() grid.raster(img_panels)

5 Scenario 4: Q=50 (n=10 per group)

=== Scenario 4: Q=50 (n=10 per group): Q=50 groups (n=10 per group, N=500) ===Running VB algorithm…Converged at iteration 7 Max relative change: 8.38e-06

Running Gibbs sampler… Gibbs posterior means: beta: 5.223436 0.4823582 0.505106 0.5257552 tau_e: 5.001927 tau_u: 0.3617891

{r echo=FALSE, fig.width=12, fig.height=6}img_conv <- readPNG("../figs/s4_convergence.png")grid.newpage grid.raster(img_conv)

{r echo=FALSE, fig.width=8, fig.height=5}img_elbo <- readPNG("../figs/s4_elbo.png")grid.newpage() grid.raster(img_elbo)

M3: 8-panel ggplot saved for Scenario 4: Q=50 (n=10 per group) {r echo=FALSE, fig.width=8, fig.height=11}img_panels <- readPNG("../figs/vb_Q50_8Panel.png")grid.newpage() grid.raster(img_panels)

6 Comparison Between Scenarios (M3 only)

```
comparison_df <- data.frame(
  Parameter = c("E[tau_e]", "E[tau_u]", "$\\sigma^2_e$", "$\\sigma^2_u$"),
  True      = c(tau_e_true, tau_u_true, 1/tau_e_true, 1/tau_u_true),
  Q5        = c(all_results[[1]]$results$E_tau_e, all_results[[1]]$results$E_tau_u,
                1/all_results[[1]]$results$E_tau_e, 1/all_results[[1]]$results$E_tau_u),
  Q10       = c(all_results[[2]]$results$E_tau_e, all_results[[2]]$results$E_tau_u,
                1/all_results[[2]]$results$E_tau_e, 1/all_results[[2]]$results$E_tau_u),
  Q20       = c(all_results[[3]]$results$E_tau_e, all_results[[3]]$results$E_tau_u,
                1/all_results[[3]]$results$E_tau_e, 1/all_results[[3]]$results$E_tau_u),
  Q50       = c(all_results[[4]]$results$E_tau_e, all_results[[4]]$results$E_tau_u,
                1/all_results[[4]]$results$E_tau_e, 1/all_results[[4]]$results$E_tau_u)
)

print(comparison_df)
```

```
##       Parameter True        Q5       Q10       Q20       Q50
## 1      E[tau_e]  5.0 4.6222758 4.9616379 4.9398101 5.0053012
## 2      E[tau_u]  0.5 0.4275260 0.8508445 0.4695342 0.3611568
## 3 $\\sigma^2_e$  0.2 0.2163436 0.2015463 0.2024369 0.1997882
## 4 $\\sigma^2_u$  2.0 2.3390389 1.1753029 2.1297705 2.7688804
```

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

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

```
for (i in seq_along(all_results)) {
  cat(glue("Q={all_results[[i]]$q}: VB tau_u = {round(all_results[[i]]$results$E_tau_u, 4)} ",
           "vs True = {tau_u_true} ",
           "(ratio: {round(all_results[[i]]$results$E_tau_u / tau_u_true, 4)})\n"))
}
```

```
## Q=5: VB tau_u = 0.4275 vs True = 0.5 (ratio: 0.8551)Q=10: VB tau_u = 0.8508 vs True = 0.5 (ratio: 1.
```

7 Sample Size Effects on τ_u (Multi-Configuration Analysis)

```
# Experimental design: Fix N=500, vary Q to show sample size per group effect
# Q = 5  → 100 obs/group (rich data)
# Q = 10 → 50 obs/group
# Q = 20 → 25 obs/group
# Q = 50 → 10 obs/group (sparse data)
# All values divide evenly into N=500
```

8 Multi-Configuration Comparison (M3 only)

```
cat("\n========================================\n")
cat("RUNNING 4-CONFIGURATION COMPARISON\n")
cat("========================================\n")

group_configs <- list(
  list(q = 5,  nq = 100, label = "Q=5 (n=100 per group)"),
```

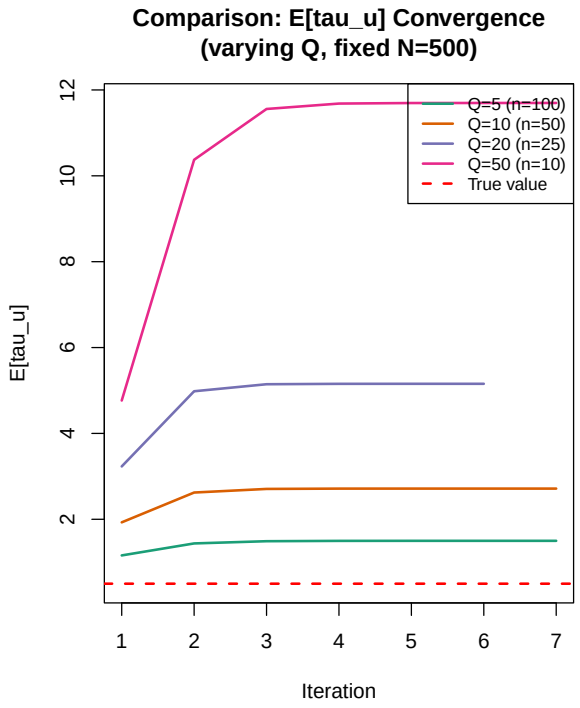
```
  list(q = 10, nq = 50,  label = "Q=10 (n=50 per group)"),
  list(q = 20, nq = 25,  label = "Q=20 (n=25 per group)"),
  list(q = 50, nq = 10,  label = "Q=50 (n=10 per group)")
)

results_multi <- list()

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

  # Generate data
  q_temp <- config$q
  nq_temp <- config$nq
  n_temp <- q_temp * nq_temp  # Will equal 500 for all configs

  set.seed(82171165 + i)

  # Design matrix for random effects
  Z_temp <- model.matrix(~ 0 + factor(rep(1:q_temp, each = nq_temp)))
  u_true_temp <- rnorm(q_temp, 0, sqrt(1/tau_u_true))

  # Correlation matrix for X
  Sigma_X <- matrix(c(
    1.0, 0.3, 0.2,
    0.3, 1.0, 0.4,
    0.2, 0.4, 1.0
  ), nrow = 3)

  X_raw <- mvtnorm::rmvnorm(n_temp, mean = rep(0, 3), sigma = Sigma_X)
  X_temp <- cbind(1, X_raw)

  # Generate response
  eta_temp <- X_temp %*% beta_true + Z_temp %*% u_true_temp
  mu_temp <- 1 / (1 + exp(-eta_temp))
  y_temp <- rbinom(n_temp, size = 1, prob = mu_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_temp,
    alpha_e = alpha_e,
    gamma_e = gamma_e,
    alpha_u = alpha_u,
    gamma_u = gamma_u,
```

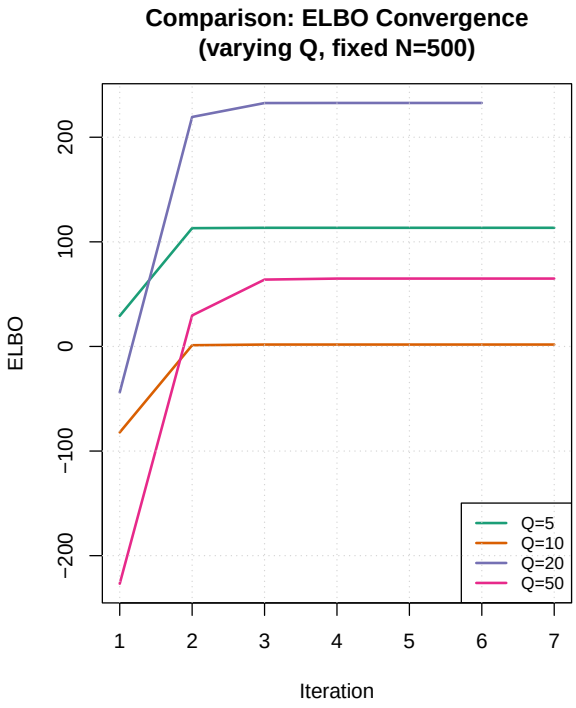
```
    model_type = "M3",
    tol     = 1e-4,
    max_iter = 500
  )

  # Run Gibbs if enabled
  if (run_gibbs) {
    gibbs_result <- run_gibbs_sampler(
      X        = X_temp,
      Z        = Z_temp,
      y        = y_temp,
      p        = p,
      q        = q_temp,
      n        = n_temp,
      alpha_e  = alpha_e,
      gamma_e  = gamma_e,
      alpha_u  = alpha_u,
      gamma_u  = gamma_u,
      model_type = "M3",
      n_iter   = gibbs_iter,
      n_burnin = gibbs_burnin
    )
  } 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("Gibbs E[tau_u]:", round(mean(gibbs_result[, "tau_u"]), 4), "\n")
  }
}

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

```
par(mfrow = c(1, 2))

# E[tau_u] convergence for all 4 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\n(varying Q, fixed N=500)',
  xlim = c(1, max_len_tau),
  ylim = range(c(sapply(results_multi, function(r) r$vb$E_tau_u_history), tau_u_true)))

lines(1:length(results_multi[[2]]$vb$E_tau_u_history),
      results_multi[[2]]$vb$E_tau_u_history,
      col = '#d95f02', lwd = 2)
lines(1:length(results_multi[[3]]$vb$E_tau_u_history),
      results_multi[[3]]$vb$E_tau_u_history,
      col = '#7570b3', lwd = 2)
lines(1:length(results_multi[[4]]$vb$E_tau_u_history),
      results_multi[[4]]$vb$E_tau_u_history,
      col = '#e7298a', lwd = 2)

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

legend('topright',
       legend = c('Q=5 (n=100)', 'Q=10 (n=50)', 'Q=20 (n=25)', 'Q=50 (n=10)', 'True value'),
       col    = c('#1b9e77', '#d95f02', '#7570b3', '#e7298a', 'red'),
       lty    = c(1, 1, 1, 1, 2),
       lwd    = 2,
       cex    = 0.8)

# ELBO convergence for all 4 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\n(varying Q, fixed N=500)',
  xlim = c(1, max_len_elbo),
  ylim = range(sapply(results_multi, function(r) r$vb$elbo_history)))

lines(1:length(results_multi[[2]]$vb$elbo_history),
      results_multi[[2]]$vb$elbo_history,
      col = '#d95f02', lwd = 2)
lines(1:length(results_multi[[3]]$vb$elbo_history),
      results_multi[[3]]$vb$elbo_history,
      col = '#7570b3', lwd = 2)
lines(1:length(results_multi[[4]]$vb$elbo_history),
      results_multi[[4]]$vb$elbo_history,
      col = '#e7298a', lwd = 2)

legend('bottomright',
       legend = c('Q=5', 'Q=10', 'Q=20', 'Q=50'),
       col    = c('#1b9e77', '#d95f02', '#7570b3', '#e7298a'),
       lty    = 1,
       lwd    = 2,
```

```
       cex    = 0.8)
grid()
```





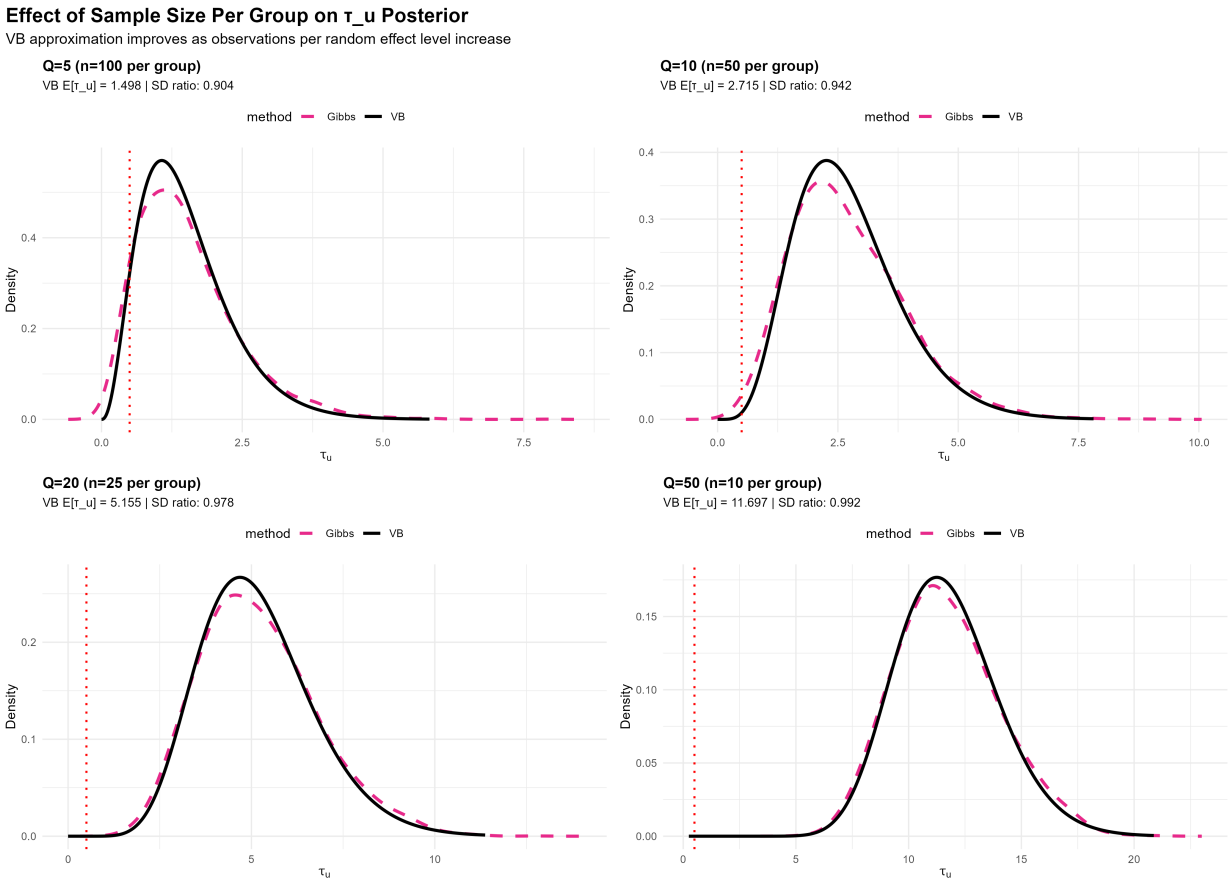

```
# Create multi-panel comparison plot as requested by Dr John
# Focus on _u posterior distributions across different group sizes

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)
```

```r
} 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 into 2x2 grid
combined_tau_u <- (plot_list[[1]] | plot_list[[2]]) /
                  (plot_list[[3]] | plot_list[[4]]) +
  plot_annotation(
    title = "Effect of Sample Size Per Group on  _u Posterior",
    subtitle = "VB approximation improves as observations per random effect level increase",
    theme = theme(
      plot.title = element_text(size = 16, face = "bold"),
      plot.subtitle = element_text(size = 12)
    )
  )

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

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

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

```
# Display
img_tau_u <- readPNG("../figs/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 = 1.0) +
```

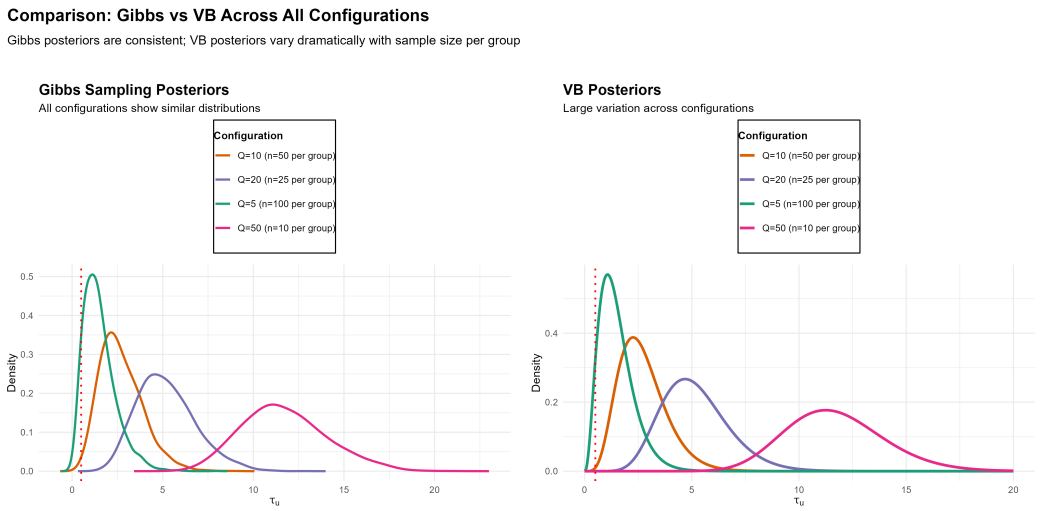
```
    geom_vline(xintercept = tau_u_true, color = "red", linetype = "dotted", linewidth = 0.8) +
    scale_color_manual(
      values = c(
        "Q=5 (n=100 per group)" = "#1b9e77",
        "Q=10 (n=50 per group)" = "#d95f02",
        "Q=20 (n=25 per group)" = "#7570b3",
        "Q=50 (n=10 per group)" = "#e7298a"
      ),
      guide = guide_legend(ncol = 1, title.position = "top")
    ) +
    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.background = element_rect(fill = "white", color = "black", linewidth = 0.5),
      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=100 per group)" = "#1b9e77",
      "Q=10 (n=50 per group)" = "#d95f02",
      "Q=20 (n=25 per group)" = "#7570b3",
      "Q=50 (n=10 per group)" = "#e7298a"
    ),
    guide = guide_legend(ncol = 1, title.position = "top")
  ) +
  labs(
    title = "VB Posteriors",
    subtitle = "Large variation across configurations",
    x = expression(tau[u]),
    y = "Density",
    color = "Configuration"
  ) +
  theme_minimal() +
  theme(
    legend.position = "right",
    legend.background = element_rect(fill = "white", color = "black", linewidth = 0.5),
    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 = "top",
    legend.box = "horizontal",
```

```
    legend.margin = margin(t = 10, b = 15),
    legend.spacing.x = unit(0.5, "cm"),
    plot.margin = margin(t = 15, r = 10, b = 10, l = 10)
  )

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

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

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

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




```
# 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("\n========================================\n")
```

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

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

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

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

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

```r
# 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         100 0.066870609 0.096999401 Q=5 (n=100 per group)
## 2 10          50 0.018854888 0.022753298 Q=10 (n=50 per group)
## 3 20          25 0.005177056 0.005463485 Q=20 (n=25 per group)
## 4 50          10 0.002290598 0.002306124 Q=50 (n=10 per group)
```

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

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

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

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

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

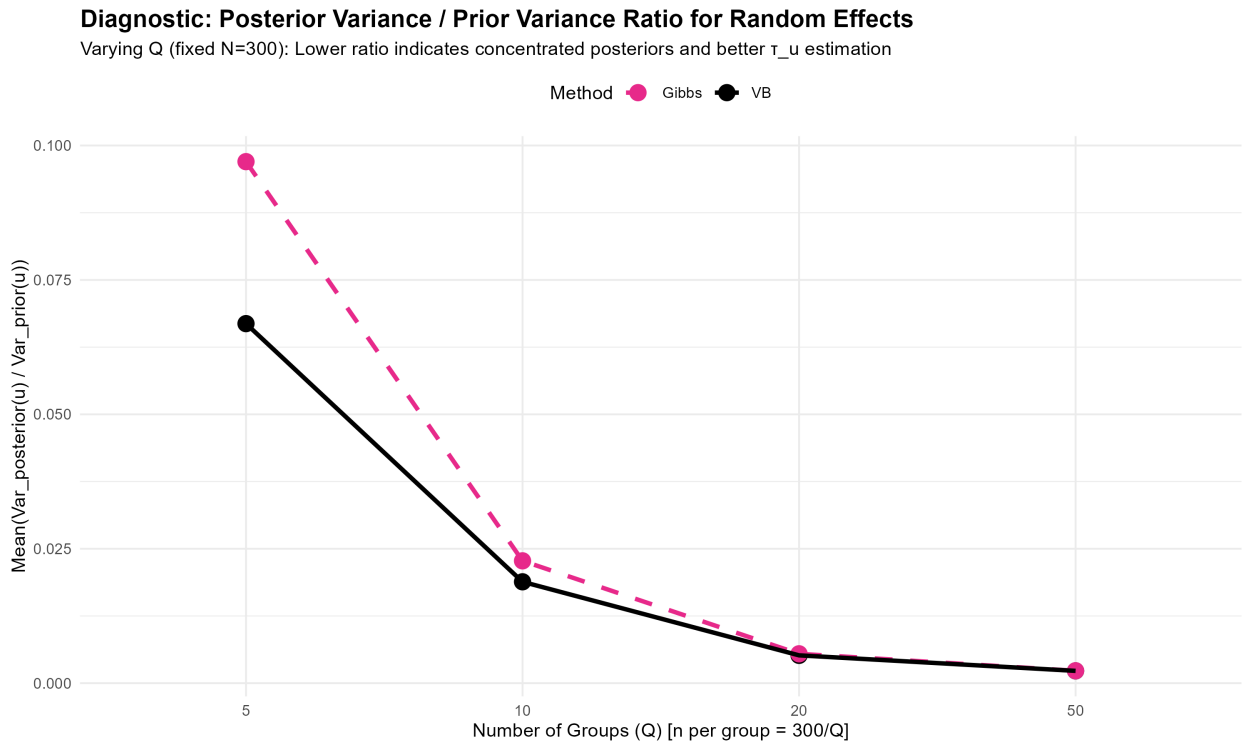
```
## - Narrow posteriors for u → better tau_u estimation in VB
```

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

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

```r
# 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/diagnostic_variance_ratio.png",
  plot = p_diagnostic,
  width = 10,
  height = 6,
  dpi = 300
)
```

```
img_diagnostic <- readPNG("../figs/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")
```

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