Variational Bayes: Unified M1/M3 Analysis

M2-Hierarchical-Linear-Diagnostic.Rmd | Compiled: 2026-02-06 18:26:20

Contents

1 scenario 1: Conditional on Model Type **7**

2 scenario 2: Conditional on Model Type (M3 only) **12**

3 scenario 3: Q=20 (M3 only) **16**

4 scenario 4: Q=50 (M3 only) **17**

5 scenario 5: Q=100 (M3 only) **18**

6 comparison Between Scenarios (M3 only) **20**

7 sample Size Effects on __u (Multi-Configuration Analysis) **20**

7.1 Three-Panel Validation Comparison 36

M2-Hierarchical-Linear-Diagnostic.Rmd | Compiled: 2026-02-06 18:26:21

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

# M3: Hierarchical model with  , u (random effects),  ,
#     - Demonstrates variance component under-dispersion
#     - Multi-configuration: Q = 5, 10, 20, 50, 100 groups

model_type <- "M3"  # Fixed to M3 for hierarchical model only

# SHARED PARAMETERS (all configurations)
n <- 300  # Total observations
p <- 3    # Number of fixed effects (beta_0, beta_1, beta_2) - Dr John's setup

# true Parameter Values (Dr John's exact values)
tau_e_true <- 0.5   # Residual precision (variance = 2)
tau_u_true <- 1  # Random effect precision (variance = 1)
beta_true  <- c(0.5, -2, 3)  # Dr John's beta values

# prior Hyperparameters (Dr John's flat priors - no prior information)
alpha_e <- 0
gamma_e <- 0
alpha_u <- 0
gamma_u <- 0

# 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
# Dr John Holmes' Gibbs sampler for hierarchical models
# EXACT original from Dr John - NO modifications
normalmm.Gibbs<-function(iter,Z,X,y,burnin,taue_0,tauu_0,Kinv,a.u,b.u,a.e,b.e){
  n   <-length(y) #no. observations
  p   <-dim(X)[2] #no of fixed effect predictors.
  q   <-dim(Z)[2] #no of random effect levels
  tauu<-tauu_0
  taue<-taue_0
  beta0<-rnorm(p)
  u0   <-rnorm(q,0,sd=1/sqrt(tauu))

  #Building combined predictor matrix.
  W<-cbind(X,Z)
  WTW <-crossprod(W)
  WTy <-crossprod(W,y)
  library(mvtnorm)

  #storing results.
  par <-matrix(0,iter,p+q+2)
  #Calculating log predictive densities
  lppd<-matrix(0,iter,n)

  #Create modified identity matrix for joint posterior.
  I0  <-diag(p+q)
  diag(I0)[1:p]<-0
  I0[-c(1:p),-c(1:p)]  <-Kinv

  for(i in 1:iter){
    #Conditional posteriors.
    uKinvu <- t(u0)%*%Kinv%*%u0
    uKinvu <-as.numeric(uKinvu)
    tauu <-rgamma(1,a.u+0.5*q,b.u+0.5*uKinvu)
    #Updating component of normal posterior for beta,u
    Prec <-WTW*taue + tauu*I0
    P.var <-solve(Prec)
    P.mean<- P.var%*%WTy*taue
    betau <-rmvnorm(1,mean=P.mean,sigma=P.var)
    betau <-as.numeric(betau)
    err   <- y-W%*%betau
    taue  <-rgamma(1,a.e+0.5*n,b.e+0.5*sum(err^2))
    #storing iterations.
    par[i,]<-c(betau,tauu,taue)
    beta0  <-betau[1:p]
    u0     <-betau[p+1:q]
    lppd[i,]= dnorm(y,mean=as.numeric(W%*%betau),sd=1/sqrt(taue))
  }

  lppd      = lppd[-c(1:burnin),]
  lppdest   = sum(log(colMeans(lppd)))        #Estimating lppd for whole dataset.
  pwaic2    = sum(apply(log(lppd),2,FUN=var)) #Estimating effective number of parameters.
  par <-par[-c(1:burnin),]
  colnames(par)<-c(paste('beta',1:p,sep=''),paste('u',1:q,sep=''),'tau_u','tau_e')
  mresult<-list(par,lppdest,pwaic2)
```

```
  names(mresult)<-c('par','lppd','pwaic')
  return(mresult)
}
```

```
# Dr John Holmes' VB algorithm for hierarchical models
# EXACT original from Dr John - NO modifications
VB.mm<-function(epsilon,iter,Kinv,Z,X,y,taue_0,tauu_0,u0,beta0,a.e,g.e,a.u,g.u){
  n<-dim(X)[1]
  p<-dim(X)[2]
  q<-dim(Z)[2]
  W <-cbind(X,Z)
  WTW<-crossprod(W)
  WTY<-crossprod(W,y)
  Kinvall<-matrix(0,p+q,p+q)
  Kinvall[-c(1:p),-c(1:p)]<-Kinv

  for(i in 1:iter){
    Vub <-solve(taue_0*WTW+tauu_0*Kinvall) #update Var(b,u)
    ub  <-taue_0*Vub%*%WTY                 #update E(b,u)
    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)
    err      <- y - W%*%ub
    TrWTWub  <- sum(diag(WTW%*%Vub))
    taue     <- (a.e+0.5*n)/(g.e+0.5*sum(err^2)+0.5*TrWTWub)
    taue     <- as.numeric(taue)

    if(i > 1){
      diffub  <- sqrt((ub-ub0)^2)/(abs(ub)+0.01)
      diffte <- abs(taue_0-taue)/(taue+0.01)
      difftu <- abs(tauu_0-tauu)/(tauu+0.01)
      diffvub <- sqrt((diag(Vub0) - diag(Vub))^2)/(diag(Vub))
      diff.all<-c(diffub,diffte,difftu,diffvub)
      if(max(diff.all) < epsilon) break
    }
    Vub0 <- Vub;ub0<-ub;taue_0<-taue;tauu_0<-tauu
    #Calculate relative change.
  }

  taue.param<-c((a.e+0.5*n),(g.e+0.5*sum(err^2)+0.5*TrWTWub))
  tauu.param<-c((a.u+0.5*q),(g.u+0.5*uKinvub+0.5*TrKinvub))
  param<-list(ub,Vub,taue.param,tauu.param,i)
  names(param)<-c('betau_mean','betau_var','tau_e','tau_u','iter')
  return(param)
}
```

```
# Dr John Holmes' VB algorithm WITH history tracking
# Modified version that tracks E[ ], E[ ], and ELBO at each iteration
VB.mm_with_history<-function(epsilon,iter,Kinv,Z,X,y,taue_0,tauu_0,u0,beta0,a.e,g.e,a.u,g.u){
  n<-dim(X)[1]
  p<-dim(X)[2]
  q<-dim(Z)[2]
  W <-cbind(X,Z)
```

```r
  WTW<-crossprod(W)
  WTY<-crossprod(W,y)
  Kinvall<-matrix(0,p+q,p+q)
  Kinvall[-c(1:p),-c(1:p)]<-Kinv

  # Initialize history storage
  E_tau_e_history <- numeric(iter)
  E_tau_u_history <- numeric(iter)
  elbo_history <- numeric(iter)

  for(i in 1:iter){
    Vub <-solve(taue_0*WTW+tauu_0*Kinvall) #update Var(b,u)
    ub  <-taue_0*Vub%*%WTY                 #update E(b,u)
    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)
    err      <- y - W%*%ub
    TrWTWub  <- sum(diag(WTW%*%Vub))
    taue     <- (a.e+0.5*n)/(g.e+0.5*sum(err^2)+0.5*TrWTWub)
    taue     <- as.numeric(taue)

    # Store history
    E_tau_e_history[i] <- taue
    E_tau_u_history[i] <- tauu

    # Calculate ELBO (simplified version for monitoring convergence)
    elbo <- -0.5 * n * log(2*pi) + 0.5 * n * (digamma(a.e+0.5*n) - log(g.e+0.5*sum(err^2)+0.5*TrWTWub))
    elbo <- elbo - 0.5 * q * log(2*pi) + 0.5 * q * (digamma(a.u+0.5*q) - log(g.u+0.5*as.numeric(uKinvub
    elbo_history[i] <- elbo

    if(i > 1){
      diffub  <- sqrt((ub-ub0)^2)/(abs(ub)+0.01)
      diffte <- abs(taue_0-taue)/(taue+0.01)
      difftu <- abs(tauu_0-tauu)/(tauu+0.01)
      diffvub <- sqrt((diag(Vub0) - diag(Vub))^2)/(diag(Vub))
      diff.all<-c(diffub,diffte,difftu,diffvub)
      if(max(diff.all) < epsilon) break
    }
    Vub0 <- Vub;ub0<-ub;taue_0<-taue;tauu_0<-tauu
  }

  # Trim history to actual iterations
  E_tau_e_history <- E_tau_e_history[1:i]
  E_tau_u_history <- E_tau_u_history[1:i]
  elbo_history <- elbo_history[1:i]

  taue.param<-c((a.e+0.5*n),(g.e+0.5*sum(err^2)+0.5*TrWTWub))
  tauu.param<-c((a.u+0.5*q),(g.u+0.5*uKinvub+0.5*TrKinvub))
  param<-list(ub,Vub,taue.param,tauu.param,i,E_tau_e_history,E_tau_u_history,elbo_history)
  names(param)<-c('betau_mean','betau_var','tau_e','tau_u','iter','E_tau_e_history','E_tau_u_history','
  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) ===
```

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

if (model_type == "M3") {
  # Dr John's method: generate then standardize
  u_true_s1 <- rnorm(q_s1, 0, 1)
  u_true_s1 <- scale(u_true_s1)
  Z_s1 <- table(1:n, rep(1:q_s1, n/q_s1))  # Dr John's method
  K_s1 <- diag(q_s1)
  linear_predictor_s1 <- X_s1 %*% beta_true + Z_s1 %*% u_true_s1
} else {
  u_true_s1 <- NULL
  Z_s1 <- matrix(0, nrow=n, ncol=1)
  K_s1 <- matrix(1)
  linear_predictor_s1 <- X_s1 %*% beta_true
}

# Dr John's method: hardcoded sd = sqrt(2)
residuals_true_s1 <- rnorm(n, 0, sqrt(2))
y_s1 <- as.vector(linear_predictor_s1 + residuals_true_s1)
```

```
scenario_name <- if (model_type == "M3") glue("SCENARIO 1: {q_s1} levels ({nq_s1} obs each)") else "SCE
cat(glue("\n=== {scenario_name} ===\n"))
```

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

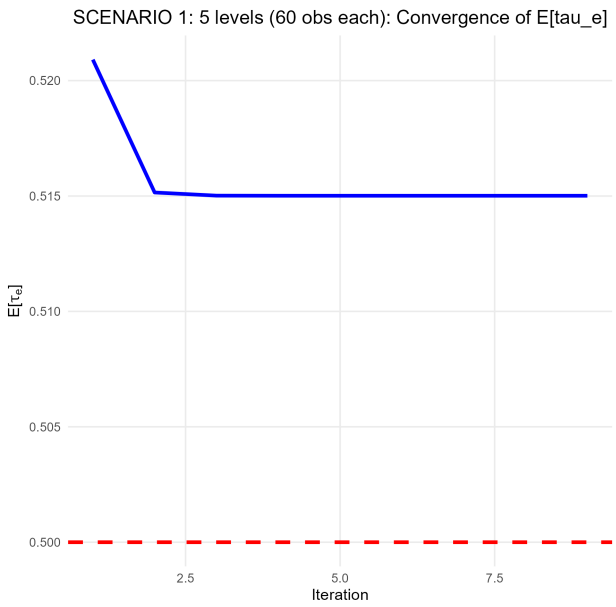
```
vb_time_s1 <- system.time({
  results_s1 <- run_vb_algorithm(
    X          = X_s1,
    Z          = Z_s1,
    y          = y_s1,
    K          = K_s1,
    p          = p,
    q          = q_s1,
    n          = n,
    alpha_e    = alpha_e,
    gamma_e    = gamma_e,
    alpha_u    = alpha_u,
    gamma_u    = gamma_u
  )
})
cat(sprintf("VB elapsed: %.3f seconds\n", vb_time_s1[3]))
```

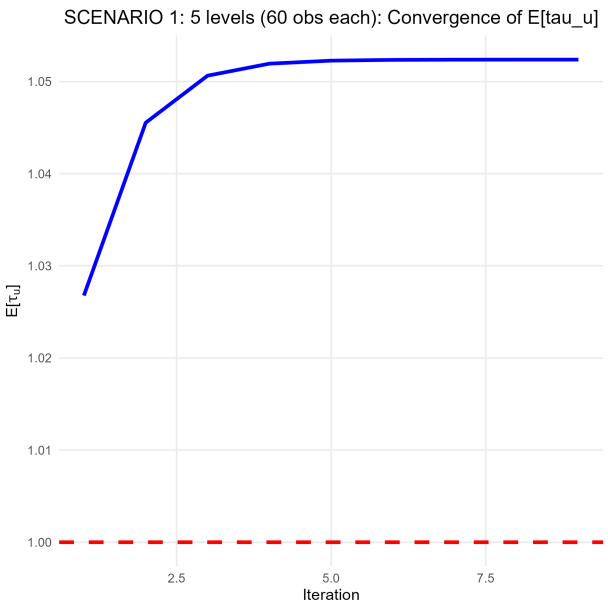
```
## VB elapsed: 0.250 seconds
```

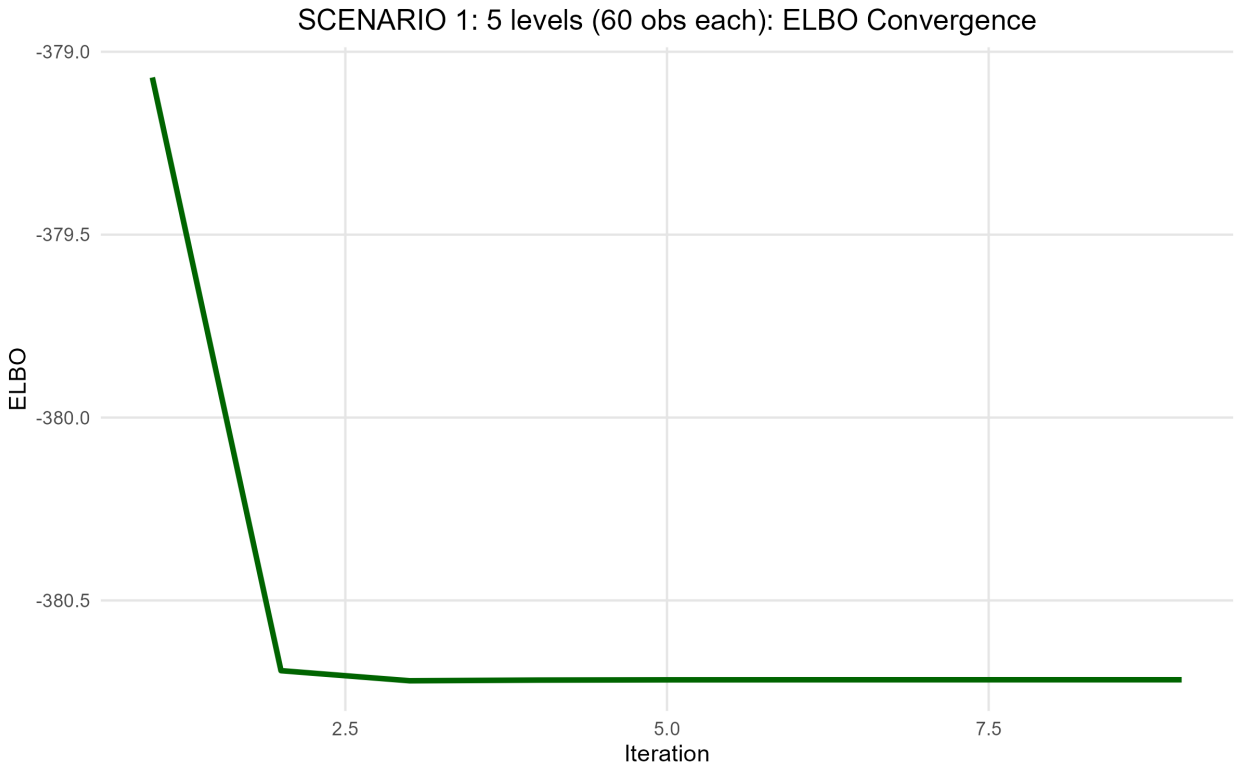
```
if (run_gibbs) {
  cat("\nRunning Gibbs sampler...\n")
  gibbs_time_s1 <- system.time({
    gibbs_s1 <- run_gibbs_sampler(
      X          = X_s1,
      Z          = Z_s1,
```

```
      y          = y_s1,
      p          = p,
      q          = q_s1,
      n          = n,
      alpha_e    = alpha_e,
      gamma_e    = gamma_e,
      alpha_u    = alpha_u,
      gamma_u    = gamma_u,
      n_iter     = gibbs_iter,
      n_burnin   = gibbs_burnin
    )
  })
  cat(sprintf("Gibbs elapsed: %.3f seconds\n", gibbs_time_s1[3]))

  cat("Gibbs posterior means:\n")
  cat("beta:", colMeans(gibbs_s1[, 1:p]), "\n")
  cat("tau_e:", mean(gibbs_s1[, "tau_e"]), "\n")
  if (model_type == "M3") {
    cat("tau_u:", mean(gibbs_s1[, "tau_u"]), "\n")
  }

  gibbs_tau_e_s1 <- gibbs_s1[, "tau_e"]
  gibbs_tau_u_s1 <- if (model_type == "M3") gibbs_s1[, "tau_u"] else NULL
} else {
  gibbs_time_s1 <- NA
  gibbs_s1 <- NULL
  gibbs_tau_e_s1 <- NULL
  gibbs_tau_u_s1 <- NULL
}
```

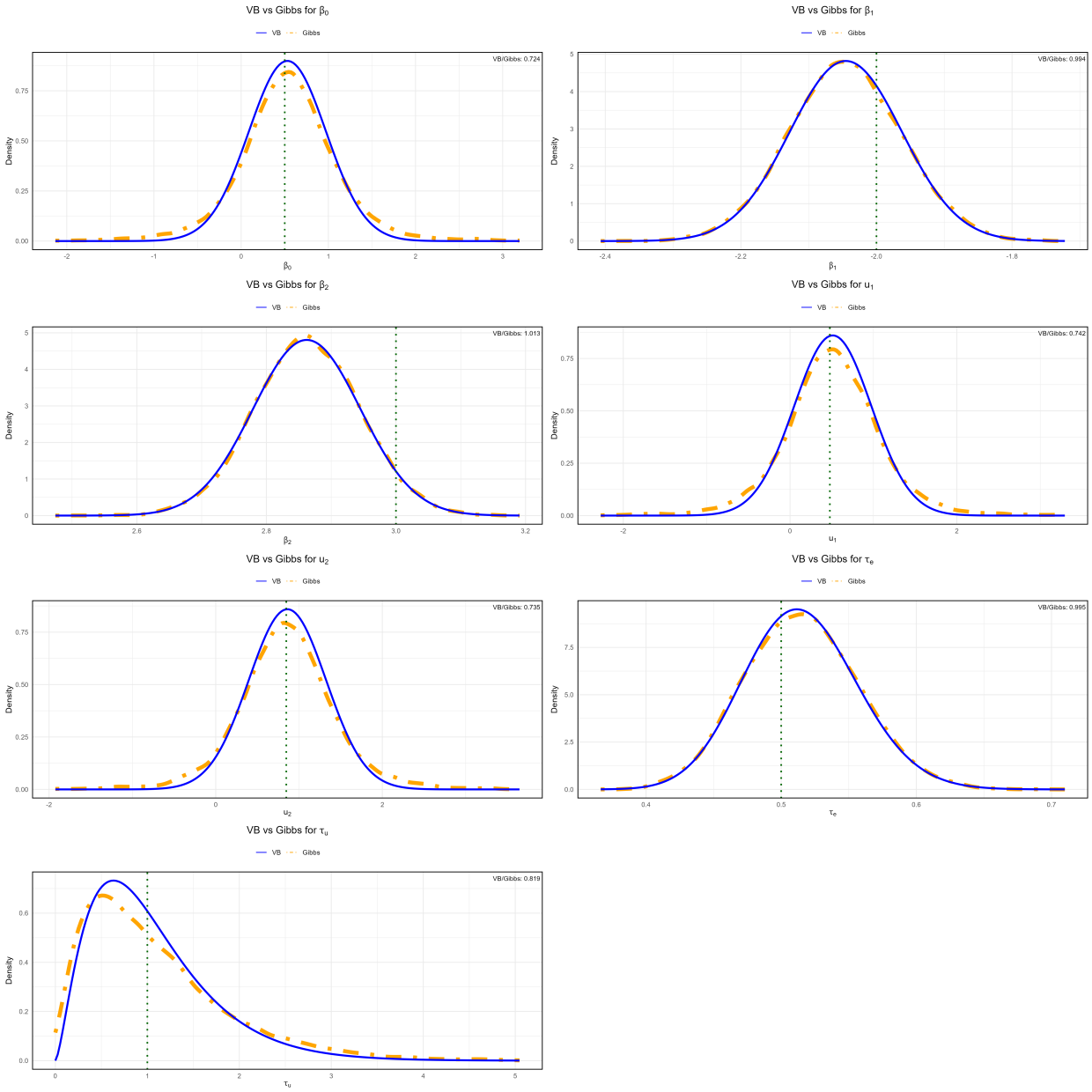
```
##
## Running Gibbs sampler...
## Gibbs elapsed: 13.680 seconds
## Gibbs posterior means:
## beta: 0.5402562 -2.045634 2.862959
## tau_e: 0.5152183
## tau_u: 1.096369
```


SCENARIO 1: 5 levels (60 obs each): Convergence of E[tau_e]
0.520
0.515
0.510
0.505
0.500
E[τe]
2.5
5.0
7.5
Iteration


SCENARIO 1: 5 levels (60 obs each): Convergence of E[tau_u]
1.05
1.04
1.03
1.02
1.01
1.00
E[τu]
2.5
5.0
7.5
Iteration




```
## M3: 7-panel ggplot saved for Scenario 1
```


VB vs Gibbs for β_0
VB
Gibbs
VB/Gibbs: 0.724
Density
β_0
VB vs Gibbs for β_1
VB
Gibbs
VB/Gibbs: 0.994
Density
β_1
VB vs Gibbs for β_2
VB
Gibbs
VB/Gibbs: 1.013
Density
β_2
VB vs Gibbs for u_1
VB
Gibbs
VB/Gibbs: 0.742
Density
u_1
VB vs Gibbs for u_2
VB
Gibbs
VB/Gibbs: 0.735
Density
u_2
VB vs Gibbs for τ_e
VB
Gibbs
VB/Gibbs: 0.995
Density
τ_e
VB vs Gibbs for τ_u
VB
Gibbs
VB/Gibbs: 0.819
Density
τ_u

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

```
if (model_type == "M3") {
  cat(glue("\n=== Scenario 2: Q={q_s2} groups (n={nq_s2} per group) ===\n"))

  # Dr John's method: generate then standardize
  u_true_s2 <- rnorm(q_s2, 0, 1)
  u_true_s2 <- scale(u_true_s2)

  X_s2 <- cbind(1, matrix(rnorm(n*(p-1)), nrow=n, ncol=p-1))
  Z_s2 <- table(1:n, rep(1:q_s2, n/q_s2))  # Dr John's method
  K_s2 <- diag(q_s2)

  linear_predictor_s2 <- X_s2 %*% beta_true + Z_s2 %*% u_true_s2
  residuals_true_s2   <- rnorm(n, 0, sqrt(2))  # Dr John's method: hardcoded
  y_s2 <- as.vector(linear_predictor_s2 + residuals_true_s2)
} else {
  cat("\nScenario 2 skipped (M3 only)\n")
}
```

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

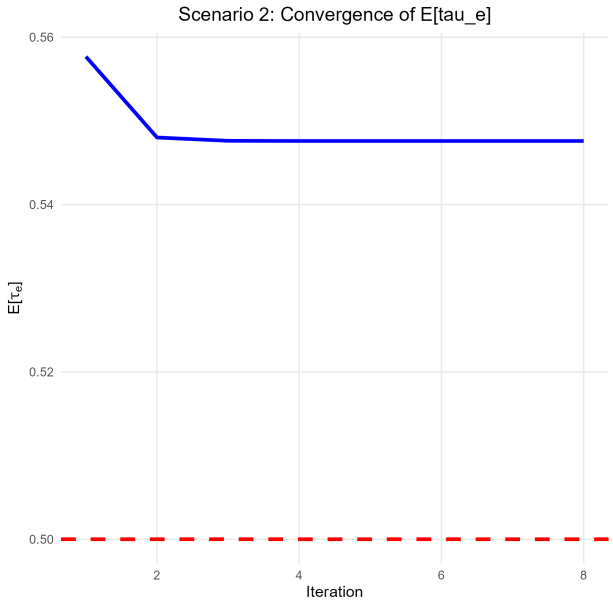
```
vb_time_s2 <- system.time({
  results_s2 <- run_vb_algorithm(
    X          = X_s2,
    Z          = Z_s2,
    y          = y_s2,
    K          = K_s2,
    p          = p,
    q          = q_s2,
    n          = n,
    alpha_e    = alpha_e,
    gamma_e    = gamma_e,
    alpha_u    = alpha_u,
    gamma_u    = gamma_u
  )
})
cat(sprintf("VB elapsed: %.3f seconds\n", vb_time_s2[3]))
```

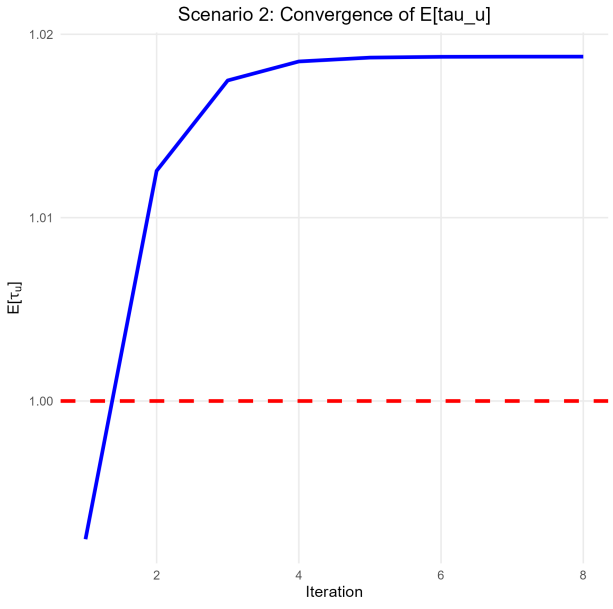
```
## VB elapsed: 0.000 seconds
```

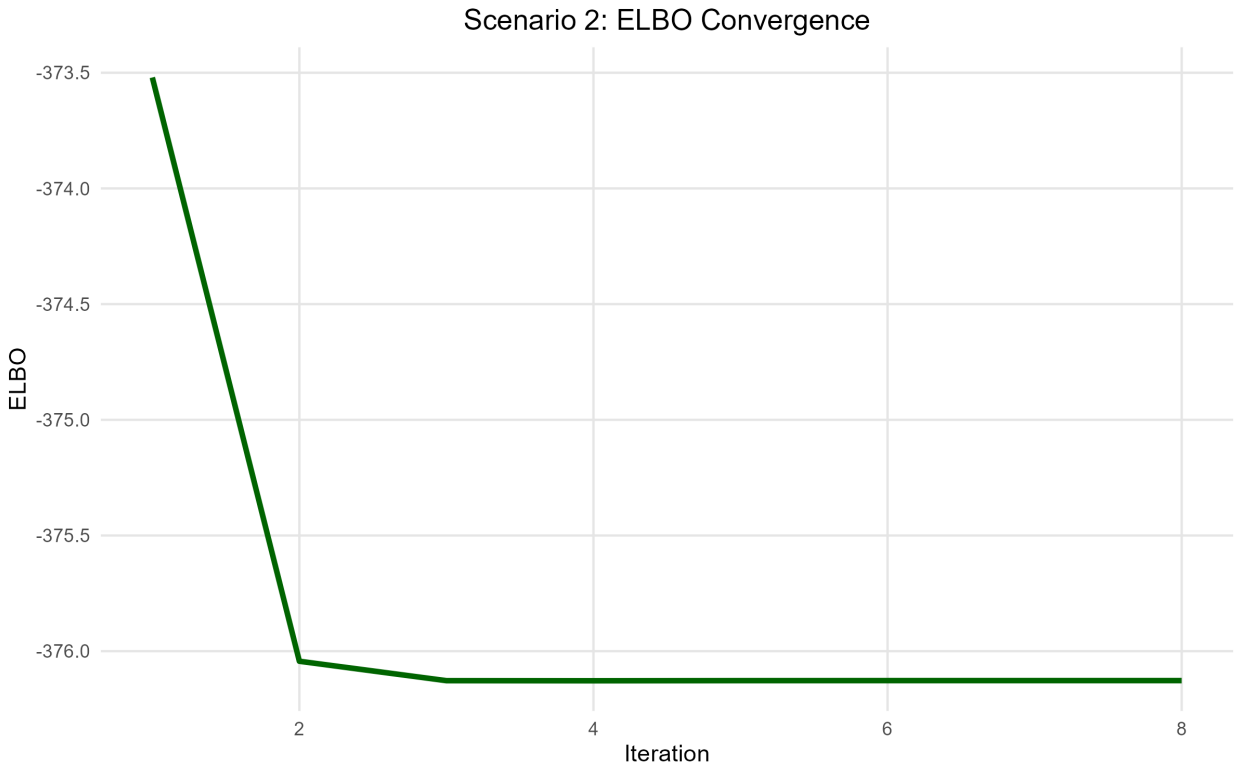
```
if (run_gibbs) {
  cat("\nRunning Gibbs sampler...\n")
  gibbs_time_s2 <- system.time({
    gibbs_s2 <- run_gibbs_sampler(
      X          = X_s2,
      Z          = Z_s2,
      y          = y_s2,
      p          = p,
      q          = q_s2,
      n          = n,
      alpha_e    = alpha_e,
      gamma_e    = gamma_e,
      alpha_u    = alpha_u,
      gamma_u    = gamma_u,
```

```
      n_iter     = gibbs_iter,
      n_burnin   = gibbs_burnin
    )
  })
  cat(sprintf("Gibbs elapsed: %.3f seconds\n", gibbs_time_s2[3]))

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

  gibbs_tau_e_s2 <- gibbs_s2[, "tau_e"]
  gibbs_tau_u_s2 <- gibbs_s2[, "tau_u"]
} else {
  gibbs_time_s2 <- NA
  gibbs_s2 <- NULL
  gibbs_tau_e_s2 <- NULL
  gibbs_tau_u_s2 <- NULL
}
```

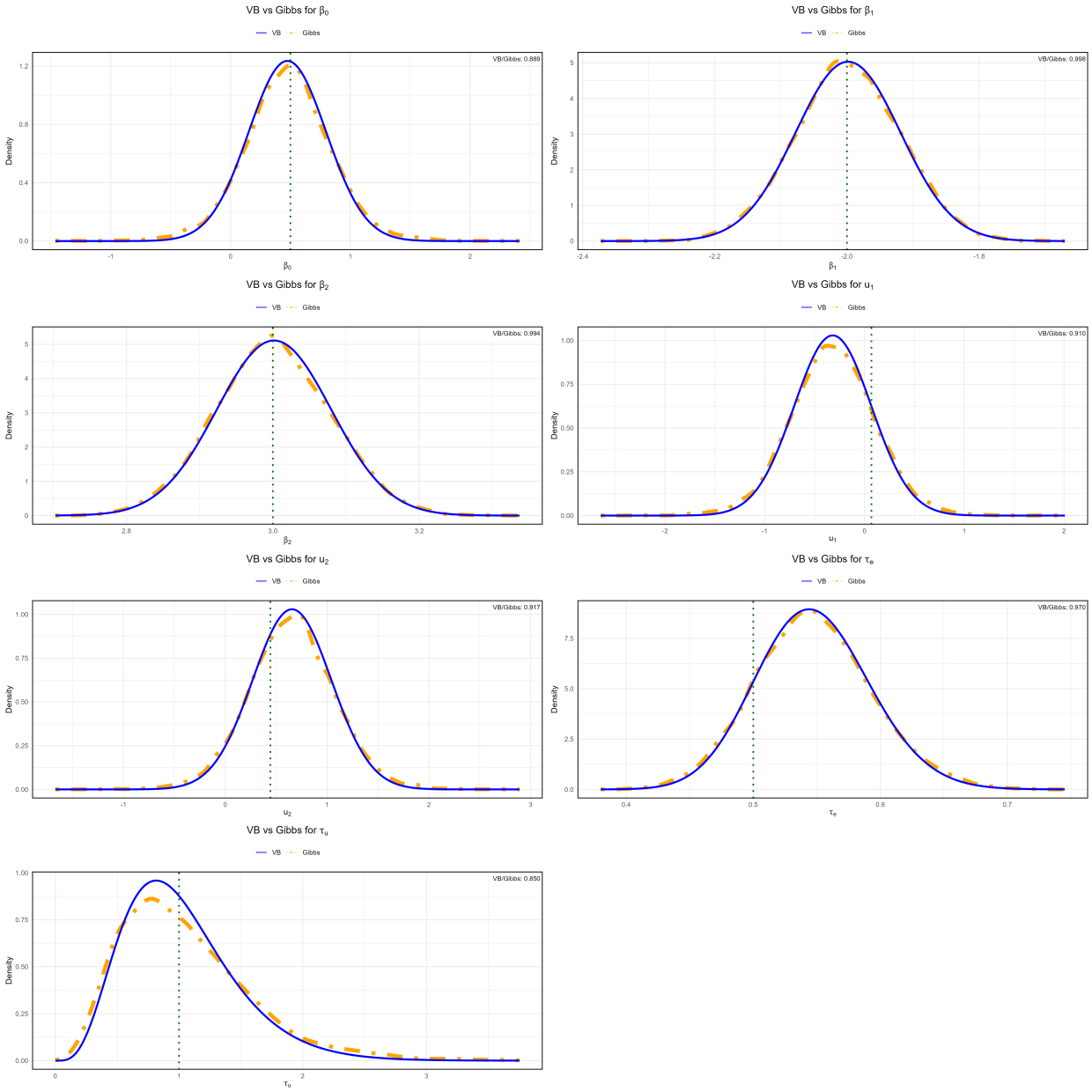
```
##
## Running Gibbs sampler...
## Gibbs elapsed: 14.120 seconds
## Gibbs posterior means:
## beta: 0.47785 -1.998485 3.000389
## tau_e: 0.5473838
## tau_u: 1.054129
```








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


VB vs Gibbs for β_0
VB
Gibbs
VB/Gibbs: 0.889
Density
β_0
VB vs Gibbs for β_1
VB
Gibbs
VB/Gibbs: 0.998
Density
β_1
VB vs Gibbs for β_2
VB
Gibbs
VB/Gibbs: 0.994
Density
β_2
VB vs Gibbs for u_1
VB
Gibbs
VB/Gibbs: 0.910
Density
u_1
VB vs Gibbs for u_2
VB
Gibbs
VB/Gibbs: 0.917
Density
u_2
VB vs Gibbs for τ_e
VB
Gibbs
VB/Gibbs: 0.970
Density
τ_e
VB vs Gibbs for τ_u
VB
Gibbs
VB/Gibbs: 0.850
Density
τ_u

3 scenario 3: Q=20 (M3 only)

```
if (model_type == "M3") {
  cat(glue("\n=== Scenario 3: Q={q_s3} groups (n={nq_s3} per group) ===\n"))

  u_true_s3 <- rnorm(q_s3, 0, 1)
  u_true_s3 <- scale(u_true_s3)

  X_s3 <- cbind(1, matrix(rnorm(n*(p-1)), nrow=n, ncol=p-1))
  Z_s3 <- table(1:n, rep(1:q_s3, n/q_s3))
  K_s3 <- diag(q_s3)

  linear_predictor_s3 <- X_s3 %*% beta_true + Z_s3 %*% u_true_s3
  residuals_true_s3   <- rnorm(n, 0, sqrt(2))
  y_s3 <- as.vector(linear_predictor_s3 + residuals_true_s3)
} else {
  cat("\nScenario 3 skipped (M3 only)\n")
}
```

```
## === Scenario 3: Q=20 groups (n=15 per group) ===
```

```
if (model_type == "M3") {
  vb_time_s3 <- system.time({
    results_s3 <- run_vb_algorithm(
      X          = X_s3,
      Z          = Z_s3,
      y          = y_s3,
      K          = K_s3,
      p          = p,
      q          = q_s3,
      n          = n,
      alpha_e    = alpha_e,
      gamma_e    = gamma_e,
      alpha_u    = alpha_u,
      gamma_u    = gamma_u
    )
  })
  cat(sprintf("VB elapsed: %.3f seconds\n", vb_time_s3[3]))
}
```

```
## VB elapsed: 0.020 seconds
```

```
if (model_type == "M3" && run_gibbs) {
  cat("\nRunning Gibbs sampler for Scenario 3...\n")
  gibbs_time_s3 <- system.time({
    gibbs_s3 <- run_gibbs_sampler(
      X          = X_s3,
      Z          = Z_s3,
      y          = y_s3,
      p          = p,
      q          = q_s3,
      n          = n,
      alpha_e    = alpha_e,
      gamma_e    = gamma_e,
      alpha_u    = alpha_u,
```

```
      gamma_u    = gamma_u,
      n_iter     = gibbs_iter,
      n_burnin   = gibbs_burnin
    )
  })
  cat(sprintf("Gibbs elapsed: %.3f seconds\n", gibbs_time_s3[3]))
} else {
  gibbs_time_s3 <- NA
}
```

```
##
## Running Gibbs sampler for Scenario 3...
## Gibbs elapsed: 14.190 seconds
```

4 scenario 4: Q=50 (M3 only)

```
if (model_type == "M3") {
  cat(glue("\n=== Scenario 4: Q={q_s4} groups (n={nq_s4} per group) ===\n"))

  u_true_s4 <- rnorm(q_s4, 0, 1)
  u_true_s4 <- scale(u_true_s4)

  X_s4 <- cbind(1, matrix(rnorm(n*(p-1)), nrow=n, ncol=p-1))
  Z_s4 <- table(1:n, rep(1:q_s4, n/q_s4))
  K_s4 <- diag(q_s4)

  linear_predictor_s4 <- X_s4 %*% beta_true + Z_s4 %*% u_true_s4
  residuals_true_s4   <- rnorm(n, 0, sqrt(2))
  y_s4 <- as.vector(linear_predictor_s4 + residuals_true_s4)
} else {
  cat("\nScenario 4 skipped (M3 only)\n")
}
```

```
## === Scenario 4: Q=50 groups (n=6 per group) ===
```

```
if (model_type == "M3") {
  vb_time_s4 <- system.time({
    results_s4 <- run_vb_algorithm(
      X          = X_s4,
      Z          = Z_s4,
      y          = y_s4,
      K          = K_s4,
      p          = p,
      q          = q_s4,
      n          = n,
      alpha_e    = alpha_e,
      gamma_e    = gamma_e,
      alpha_u    = alpha_u,
      gamma_u    = gamma_u
    )
  })
  cat(sprintf("VB elapsed: %.3f seconds\n", vb_time_s4[3]))
}
```

```
## VB elapsed: 0.020 seconds
```

```
if (model_type == "M3" && run_gibbs) {
  cat("\nRunning Gibbs sampler for Scenario 4...\n")
  gibbs_time_s4 <- system.time({
    gibbs_s4 <- run_gibbs_sampler(
      X          = X_s4,
      Z          = Z_s4,
      y          = y_s4,
      p          = p,
      q          = q_s4,
      n          = n,
      alpha_e    = alpha_e,
      gamma_e    = gamma_e,
      alpha_u    = alpha_u,
      gamma_u    = gamma_u,
      n_iter     = gibbs_iter,
      n_burnin   = gibbs_burnin
    )
  })
  cat(sprintf("Gibbs elapsed: %.3f seconds\n", gibbs_time_s4[3]))
} else {
  gibbs_time_s4 <- NA
}
```

```
##
## Running Gibbs sampler for Scenario 4...
## Gibbs elapsed: 26.670 seconds
```

5 scenario 5: Q=100 (M3 only)

```
if (model_type == "M3") {
  cat(glue("\n=== Scenario 5: Q={q_s5} groups (n={nq_s5} per group) ===\n"))

  u_true_s5 <- rnorm(q_s5, 0, 1)
  u_true_s5 <- scale(u_true_s5)

  X_s5 <- cbind(1, matrix(rnorm(n*(p-1)), nrow=n, ncol=p-1))
  Z_s5 <- table(1:n, rep(1:q_s5, n/q_s5))
  K_s5 <- diag(q_s5)

  linear_predictor_s5 <- X_s5 %*% beta_true + Z_s5 %*% u_true_s5
  residuals_true_s5   <- rnorm(n, 0, sqrt(2))
  y_s5 <- as.vector(linear_predictor_s5 + residuals_true_s5)
} else {
  cat("\nScenario 5 skipped (M3 only)\n")
}
```

```
## === Scenario 5: Q=100 groups (n=3 per group) ===
```

```
if (model_type == "M3") {
  vb_time_s5 <- system.time({
    results_s5 <- run_vb_algorithm(
      X          = X_s5,
```

```
      Z          = Z_s5,
      y          = y_s5,
      K          = K_s5,
      p          = p,
      q          = q_s5,
      n          = n,
      alpha_e    = alpha_e,
      gamma_e    = gamma_e,
      alpha_u    = alpha_u,
      gamma_u    = gamma_u
    )
  })
  cat(sprintf("VB elapsed: %.3f seconds\n", vb_time_s5[3]))
}
```

```
## VB elapsed: 0.090 seconds
```

```
if (model_type == "M3" && run_gibbs) {
  cat("\nRunning Gibbs sampler for Scenario 5...\n")
  gibbs_time_s5 <- system.time({
    gibbs_s5 <- run_gibbs_sampler(
      X          = X_s5,
      Z          = Z_s5,
      y          = y_s5,
      p          = p,
      q          = q_s5,
      n          = n,
      alpha_e    = alpha_e,
      gamma_e    = gamma_e,
      alpha_u    = alpha_u,
      gamma_u    = gamma_u,
      n_iter     = gibbs_iter,
      n_burnin   = gibbs_burnin
    )
  })
  cat(sprintf("Gibbs elapsed: %.3f seconds\n", gibbs_time_s5[3]))

  cat("\n=== SCENARIO 5 (Q=100) TIMING SUMMARY ===\n")
  cat(sprintf("VB time:    %.4f seconds\n", vb_time_s5[3]))
  cat(sprintf("Gibbs time: %.4f seconds\n", gibbs_time_s5[3]))
  cat(sprintf("Speedup:    %.1fx\n", gibbs_time_s5[3]/vb_time_s5[3]))
  cat("==========================================\n")
} else {
  gibbs_time_s5 <- NA
}
```

```
##
## Running Gibbs sampler for Scenario 5...
## Gibbs elapsed: 99.190 seconds
##
## === SCENARIO 5 (Q=100) TIMING SUMMARY ===
## VB time:    0.0900 seconds
## Gibbs time: 99.1900 seconds
## Speedup:    1102.1x
## ==========================================
```

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_Value  = c(tau_e_true, tau_u_true, 1/tau_e_true, 1/tau_u_true),
  Scenario_30 = c(results_s1$E_tau_e, results_s1$E_tau_u,
                  1/results_s1$E_tau_e, 1/results_s1$E_tau_u),
  Scenario_6  = c(results_s2$E_tau_e, results_s2$E_tau_u,
                  1/results_s2$E_tau_e, 1/results_s2$E_tau_u)
)

print(comparison_df)
```

```
##   Parameter True_Value Scenario_30 Scenario_6
## 1  E[tau_e]        0.5   0.5150117  0.5475981
## 2  E[tau_u]        1.0   1.0523839  1.0187825
## 3 sigma^2_e        2.0   1.9417035  1.8261569
## 4 sigma^2_u        1.0   0.9502236  0.9815638
```

```
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 = 1.0524 vs True = 1 (ratio: 1.0524 )
```

```
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 = 1.0188 vs True = 1 (ratio: 1.0188 )
```

7 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")
```

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

```r
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)")
  # Q=100 commented out due to memory constraints (6GB+ usage)
  # 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 residual sd = sqrt(2)
  eta_temp <- X_temp %*% beta_true + Z_temp %*% u_true_temp
  y_temp <- as.vector(eta_temp + rnorm(n, 0, sqrt(2)))

  # 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_e = alpha_e,
    gamma_e = gamma_e,
    alpha_u = alpha_u,
    gamma_u = gamma_u,
    tol     = 1e-4,
    max_iter = 500
  )
```

```r
# Run Gibbs 3 times with different inits (Dr John's convergence check)
if (run_gibbs) {
  cat("Running Gibbs with 3 different initial values...\n")

  # Run 1: taue_0=3, tauu_0=0.5
  gibbs_run1 <- run_gibbs_sampler(
    X        = X_temp,
    Z        = Z_temp,
    y        = y_temp,
    p        = p,
    q        = q_temp,
    n        = n,
    alpha_e  = alpha_e,
    gamma_e  = gamma_e,
    alpha_u  = alpha_u,
    gamma_u  = gamma_u,
    n_iter   = gibbs_iter,
    n_burnin = gibbs_burnin,
    init_taue = 3,
    init_tauu = 0.5
  )

  # Run 2: taue_0=0.5, tauu_0=3
  gibbs_run2 <- run_gibbs_sampler(
    X        = X_temp,
    Z        = Z_temp,
    y        = y_temp,
    p        = p,
    q        = q_temp,
    n        = n,
    alpha_e  = alpha_e,
    gamma_e  = gamma_e,
    alpha_u  = alpha_u,
    gamma_u  = gamma_u,
    n_iter   = gibbs_iter,
    n_burnin = gibbs_burnin,
    init_taue = 0.5,
    init_tauu = 3
  )

  # Run 3: taue_0=5, tauu_0=5
  gibbs_run3 <- run_gibbs_sampler(
    X        = X_temp,
    Z        = Z_temp,
    y        = y_temp,
    p        = p,
    q        = q_temp,
    n        = n,
    alpha_e  = alpha_e,
    gamma_e  = gamma_e,
    alpha_u  = alpha_u,
    gamma_u  = gamma_u,
    n_iter   = gibbs_iter,
```

```
      n_burnin = gibbs_burnin,
      init_taue = 5,
      init_tauu = 5
    )

    # Combine all 3 runs (Dr John's method)
    gibbs_result <- rbind(gibbs_run1, gibbs_run2, gibbs_run3)
    cat("Combined", nrow(gibbs_result), "samples from 3 Gibbs runs\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 with 3 different initial values...
## Combined 36000 samples from 3 Gibbs runs
## VB E[tau_u]: 1.0509
## HMC E[tau_u]: 1.0916
##
## --- Running: Q=10 (n=30 per group) ---
## Running Gibbs with 3 different initial values...
## Combined 36000 samples from 3 Gibbs runs
## VB E[tau_u]: 1.0188
## HMC E[tau_u]: 1.0541
##
## --- Running: Q=20 (n=15 per group) ---
## Running Gibbs with 3 different initial values...
## Combined 36000 samples from 3 Gibbs runs
## VB E[tau_u]: 1.0984
## HMC E[tau_u]: 1.1357
##
## --- Running: Q=50 (n=6 per group) ---
## Running Gibbs with 3 different initial values...
## Combined 36000 samples from 3 Gibbs runs
## VB E[tau_u]: 1.1481
## HMC E[tau_u]: 1.2017
```

```
# Save results for plotting
results_file <- "../results/M2_results_multi.rds"
if (!dir.exists("../results")) {
  dir.create("../results", recursive = TRUE)
```

```
}
saveRDS(results_multi, results_file)
cat("Saved results_multi to", results_file, "\n")
```

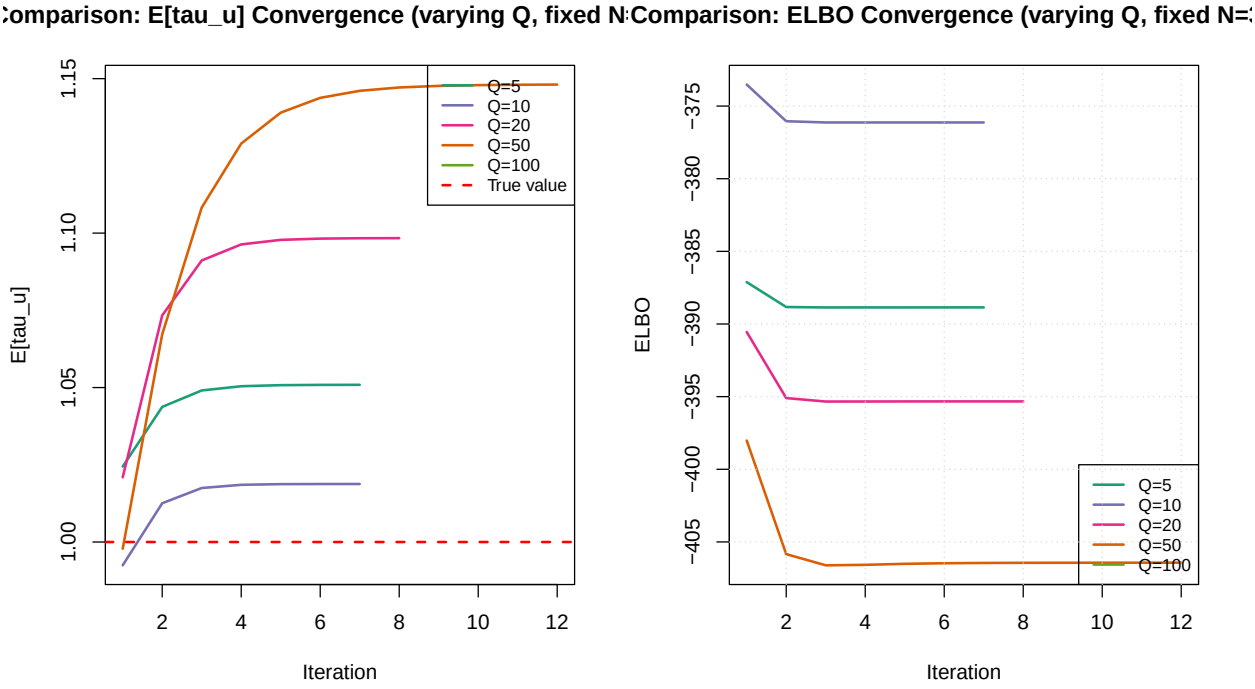
```
## Saved results_multi to ../results/M2_results_multi.rds
```

```
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")
}
```



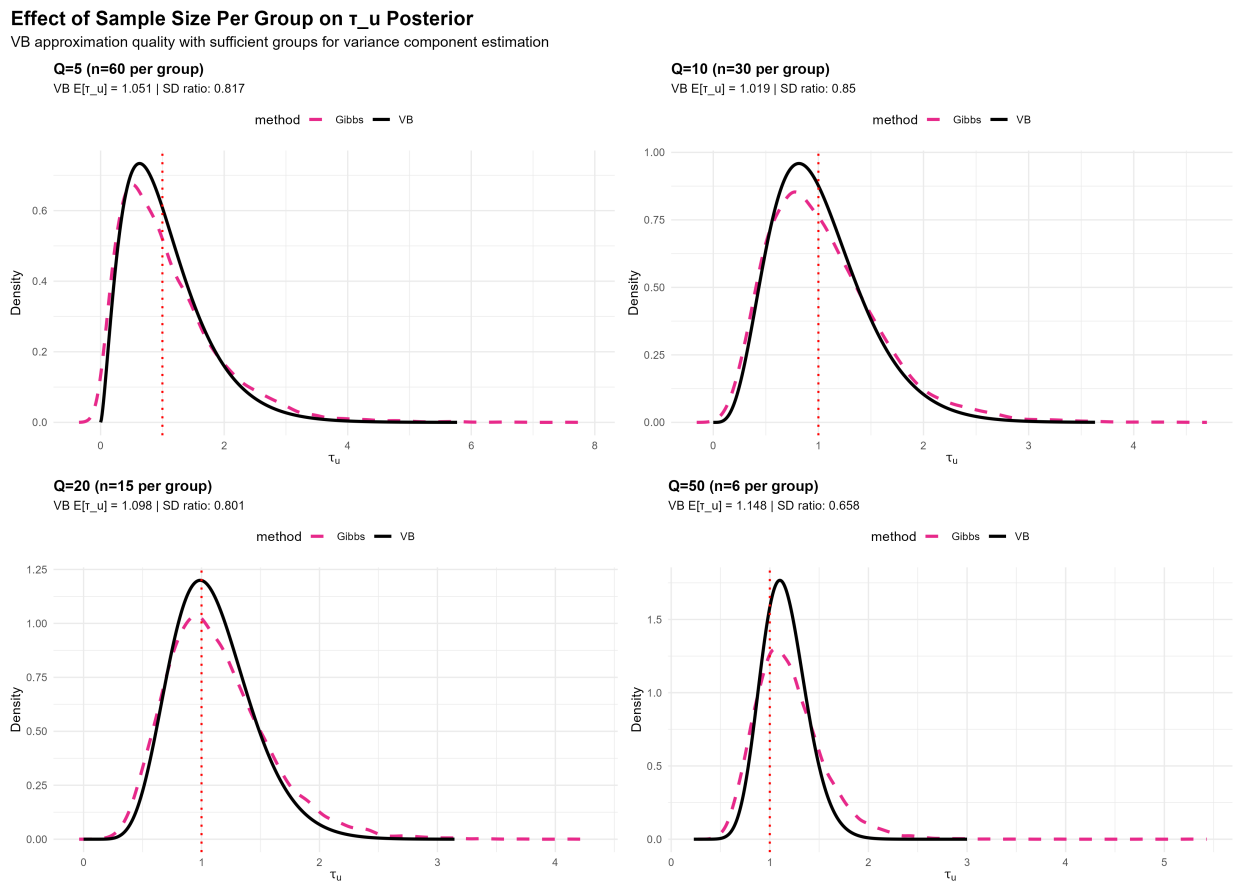

```
# Create multi-panel comparison plot as requested by Dr John
# Focus on _u posterior distributions across different group sizes

plot_list <- list()
```

```r
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 4 plots into a grid (2 rows, 2 columns)
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 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/M2_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/M2_tau_u_sample_size_comparison.png")
grid.newpage()
grid.raster(img_tau_u)
```




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

# Check that results_multi exists and has data
if (!exists("results_multi") || length(results_multi) == 0) {
  cat("ERROR: results_multi not available. Cannot create overlay plot.\n")
  plot.new()
  text(0.5, 0.5, "Error: results_multi not found", cex = 1.5, col = "red")
} else {
  cat("Creating overlay plot with", length(results_multi), "configurations...\n")

# 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"]
```

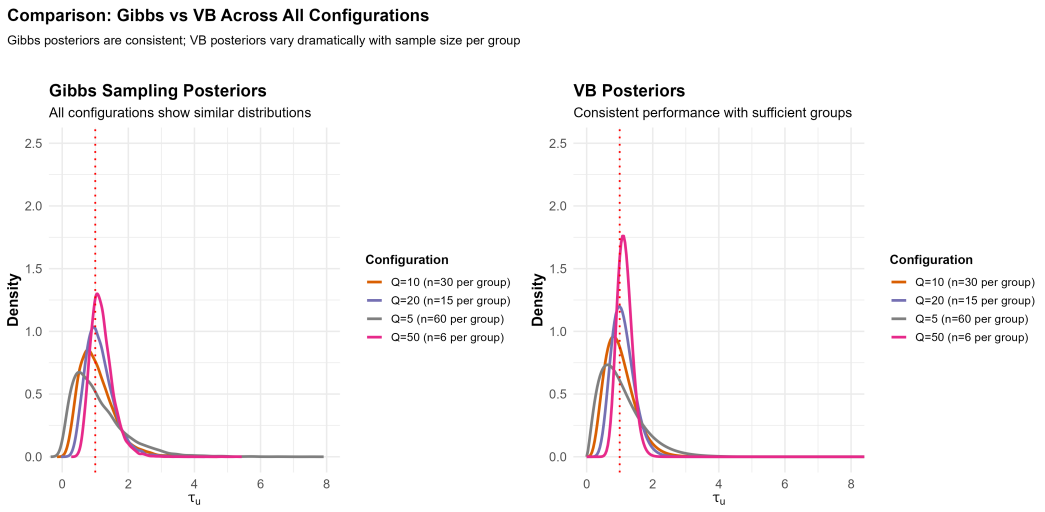
```r
    dens_gibbs <- density(gibbs_tau_u, adjust = 1.5)

    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"
      )
    ) +
    coord_cartesian(xlim = c(0, 8), ylim = c(0, 2.5)) +
    labs(
      title = "Gibbs Sampling Posteriors",
      subtitle = "All configurations show similar distributions",
      x = expression(tau[u]),
      y = "Density",
      color = "Configuration"
    ) +
    theme_minimal(base_size = 14) +
    theme(
      legend.position = "right",
      legend.text = element_text(size = 11),
      legend.title = element_text(size = 12, face = "bold"),
      plot.title = element_text(size = 16, face = "bold"),
      plot.subtitle = element_text(size = 13),
      axis.text = element_text(size = 12),
      axis.title = element_text(size = 14, face = "bold")
    )
}

# 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
```

```r
  # 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"
    )
  ) +
  coord_cartesian(xlim = c(0, 8), ylim = c(0, 2.5)) +
  labs(
    title = "VB Posteriors",
    subtitle = "Consistent performance with sufficient groups",
    x = expression(tau[u]),
    y = "Density",
    color = "Configuration"
  ) +
  theme_minimal(base_size = 14) +
  theme(
    legend.position = "right",
    legend.text = element_text(size = 11),
    legend.title = element_text(size = 12, face = "bold"),
    plot.title = element_text(size = 16, face = "bold"),
    plot.subtitle = element_text(size = 13),
    axis.text = element_text(size = 12),
    axis.title = element_text(size = 14, face = "bold")
  )

# 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/M2_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")

# Verify plot was created
if (file.exists("../figs/M2_tau_u_overlay_comparison.png")) {
  cat("  Plot file verified\n")
} else {
  cat("  WARNING: Plot file not created!\n")
}

# Display
img_overlay <- readPNG("../figs/M2_tau_u_overlay_comparison.png")
grid.newpage()
grid.raster(img_overlay)
}  # End of results_multi existence check
```

```
## Creating overlay plot with 4 configurations...
```

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




```
# 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
```

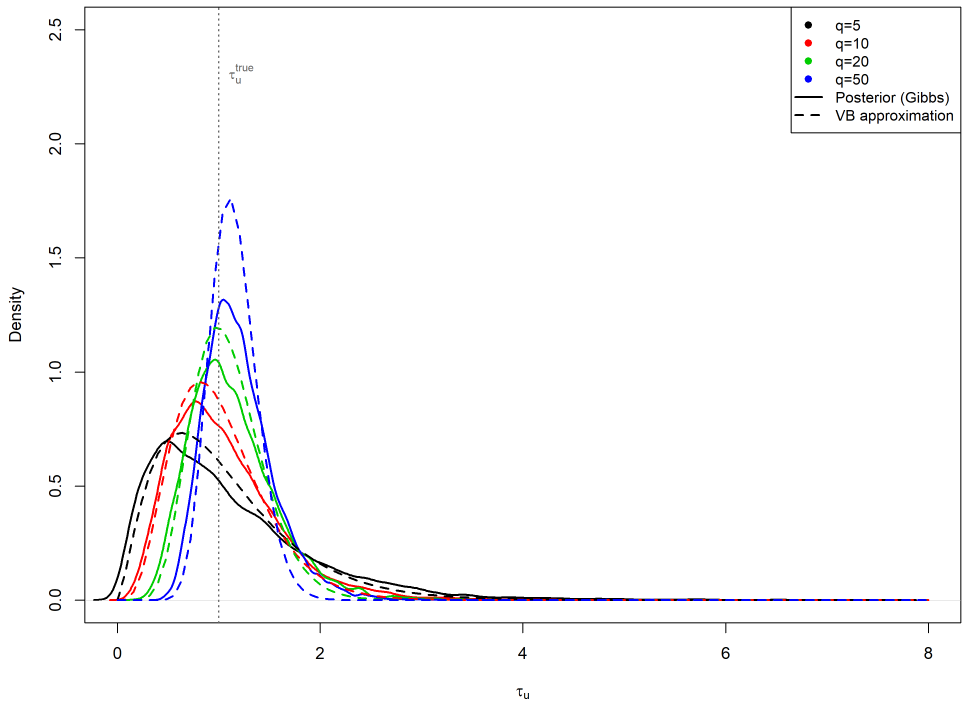
```r
png(filename = "../figs/M2_my_tau_u_comparison.png", width = 10, height = 8, units = "in", res = 300)

# Colors for all 4 Q values
colors <- c("black", "red", "green3", "blue")
q_values <- c(5, 10, 20, 50)

# 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)

  # 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])

  # Legend
  legend('topright',
         col = c(colors, "black", "black"),
         lty = c(0, 0, 0, 0, 1, 2),
         lwd = c(NA, NA, NA, NA, 2, 2),
         pch = c(19, 19, 19, 19, NA, NA),
         legend = c('q=5', 'q=10', 'q=20', 'q=50', '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])

  legend('topright',
         col = colors,
         lty = 1,
         lwd = 2,
         legend = c('q=5', 'q=10', 'q=20', 'q=50'),
         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")
```

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

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

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

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

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

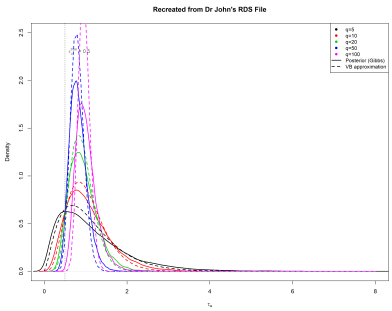
```r
# Use absolute path to ensure it works during knitting
project_root <- "d:/github/VI1"
rds_path <- file.path(project_root, "results", "dr_john_reference_tau_u.rds")

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
```

7.1 Three-Panel Validation Comparison

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



Left: My recreation from his RDS data

```
# 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 0.2169099   0.4000269  Q=5 (n=60 per group)
## 2 10          30 0.1501606   0.1793505 Q=10 (n=30 per group)
## 3 20          15 0.1549048   0.1623693 Q=20 (n=15 per group)
## 4 50           6 0.2364795   0.2402577  Q=50 (n=6 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")
```

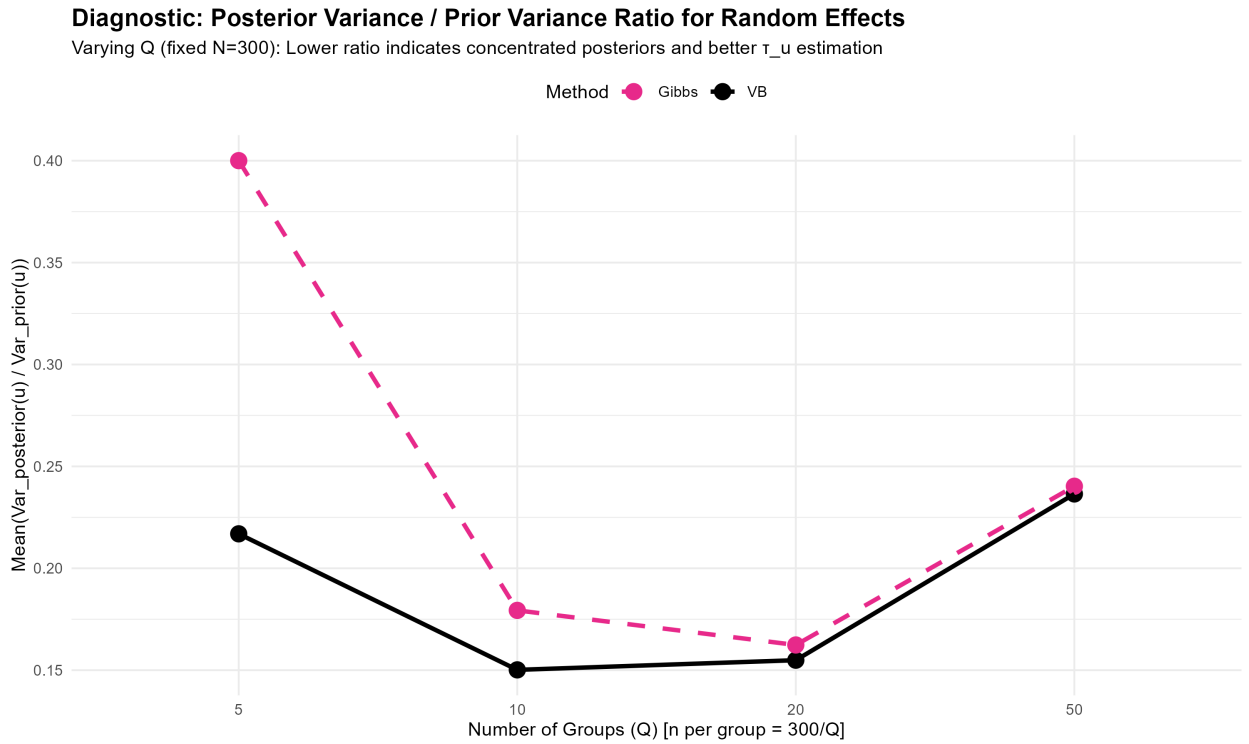
```
## - Narrow posteriors for u → better tau_u estimation in VB
```

```
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/M2_diagnostic_variance_ratio.png",
  plot = p_diagnostic,
  width = 10,
  height = 6,
  dpi = 300
)

img_diagnostic <- readPNG("../figs/M2_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 M2 SD ratios for 4 configurations
```

```
##
## Saved M2 timing data for 5 configurations
```

```
##   Q=5:   VB=0.250s, Gibbs=13.680s, Speedup=54.7x
##   Q=10:  VB=0.000s, Gibbs=14.120s, Speedup=Infx
##   Q=20:  VB=0.020s, Gibbs=14.190s, Speedup=709.5x
##   Q=50:  VB=0.020s, Gibbs=26.670s, Speedup=1333.5x
##   Q=100: VB=0.090s, Gibbs=99.190s, Speedup=1102.1x
```