Variational Bayes: Unified M1/M3 Analysis

VI-REWORK.Rmd | Compiled: 2026-01-24 16:54:11

Contents

```r
# ============================================================================
# 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"  # "M1" for linear regression, "M3" for hierarchical

# 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 <- if (model_type == "M1") NULL else 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) - M3 only
q_s1  <- if (model_type == "M1") 0 else 5
nq_s1 <- if (model_type == "M1") 0 else 60

# SCENARIO 2: Q=10 groups (n=30 per group) - M3 only
q_s2  <- if (model_type == "M1") 0 else 10
nq_s2 <- if (model_type == "M1") 0 else 30

# SCENARIO 3: Q=20 groups (n=15 per group) - M3 only
q_s3  <- if (model_type == "M1") 0 else 20
nq_s3 <- if (model_type == "M1") 0 else 15

# SCENARIO 4: Q=50 groups (n=6 per group) - M3 only
q_s4  <- if (model_type == "M1") 0 else 50
nq_s4 <- if (model_type == "M1") 0 else 6

# SCENARIO 5: Q=100 groups (n=3 per group) - M3 only
q_s5  <- if (model_type == "M1") 0 else 100
nq_s5 <- if (model_type == "M1") 0 else 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)
}
```

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) ===
```

```
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,
  model_type = model_type
)
```

```
if (run_gibbs) {
  cat("\nRunning Gibbs sampler...\n")
  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,
```

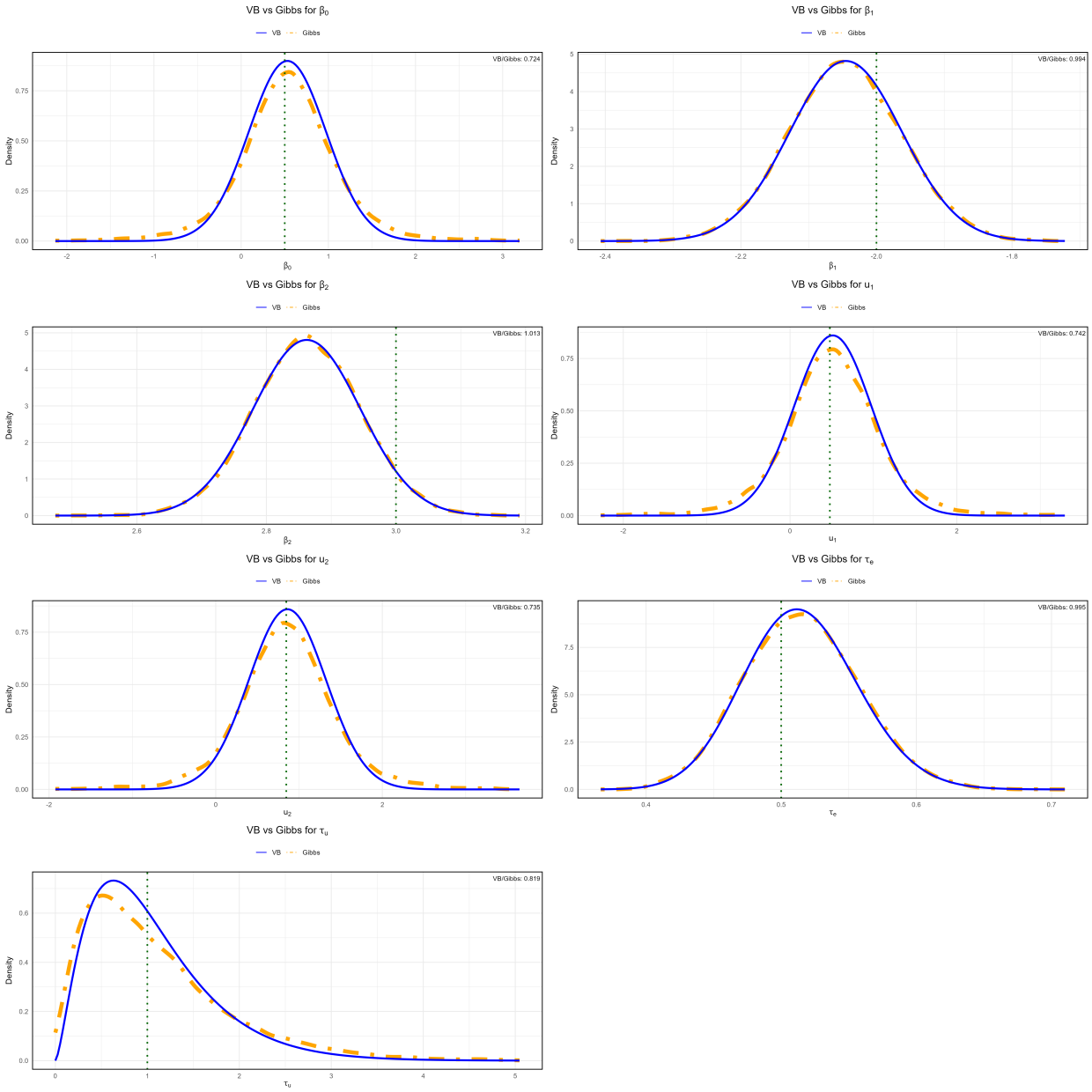
```
    gamma_e    = gamma_e,
    alpha_u    = alpha_u,
    gamma_u    = gamma_u,
    model_type = model_type,
    n_iter     = gibbs_iter,
    n_burnin   = gibbs_burnin
  )

  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_s1 <- NULL
  gibbs_tau_e_s1 <- NULL
  gibbs_tau_u_s1 <- NULL
}
```

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

Convergence history not available
(Dr John's original functions)

ELBO history not available
(Dr John's original functions)

```
## 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) ===
```

```
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,
  model_type = model_type
)
```

```
if (run_gibbs) {
  cat("\nRunning Gibbs sampler...\n")
  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,
    model_type = model_type,
    n_iter     = gibbs_iter,
    n_burnin   = gibbs_burnin
  )
```

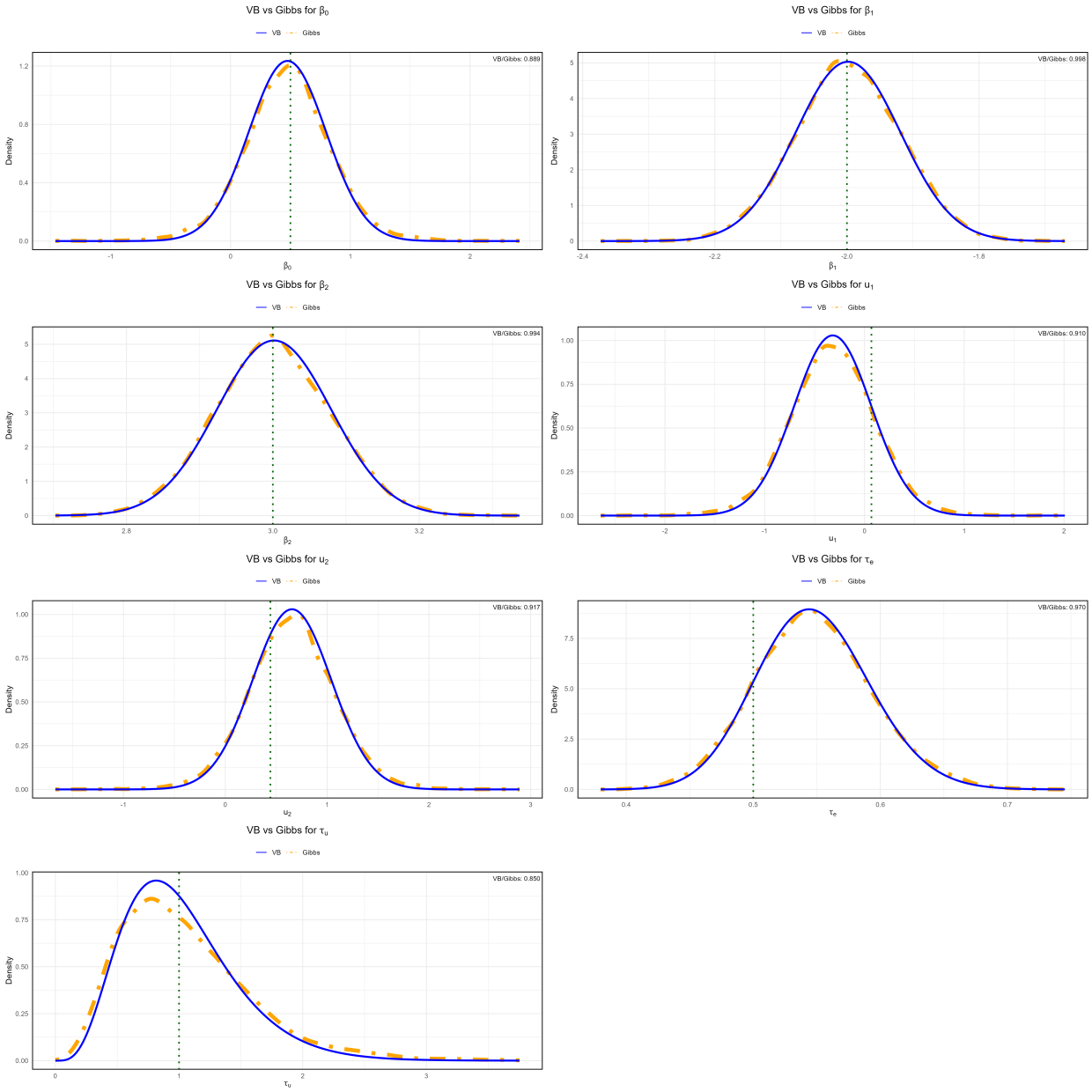
```
  cat("Gibbs posterior means:\n")
  cat("beta:", colMeans(gibbs_s2[, 1:p]), "\n")
  cat("tau_e:", mean(gibbs_s2[, "tau_e"]), "\n")
  cat("tau_u:", mean(gibbs_s2[, "tau_u"]), "\n")

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

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


Convergence history not available
(Dr John's original functions)

ELBO history not available
(Dr John's original functions)

```
## 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 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 )
```

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

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

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

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

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

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

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

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

```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)"),
  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,
    model_type = "M3",
    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,
    model_type = "M3",
    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,
    model_type = "M3",
    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,
      model_type = "M3",
      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]: 1.1068
## HMC E[tau_u]: 1.1554
##
## --- 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
##
## --- Running: Q=100 (n=3 per group) ---
```

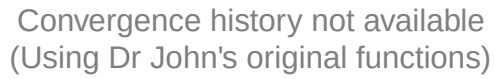
```
## Running Gibbs with 3 different initial values...
## Combined 36000 samples from 3 Gibbs runs
## VB E[tau_u]: 1.1827
## HMC E[tau_u]: 1.3264
```

```
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")
}
```



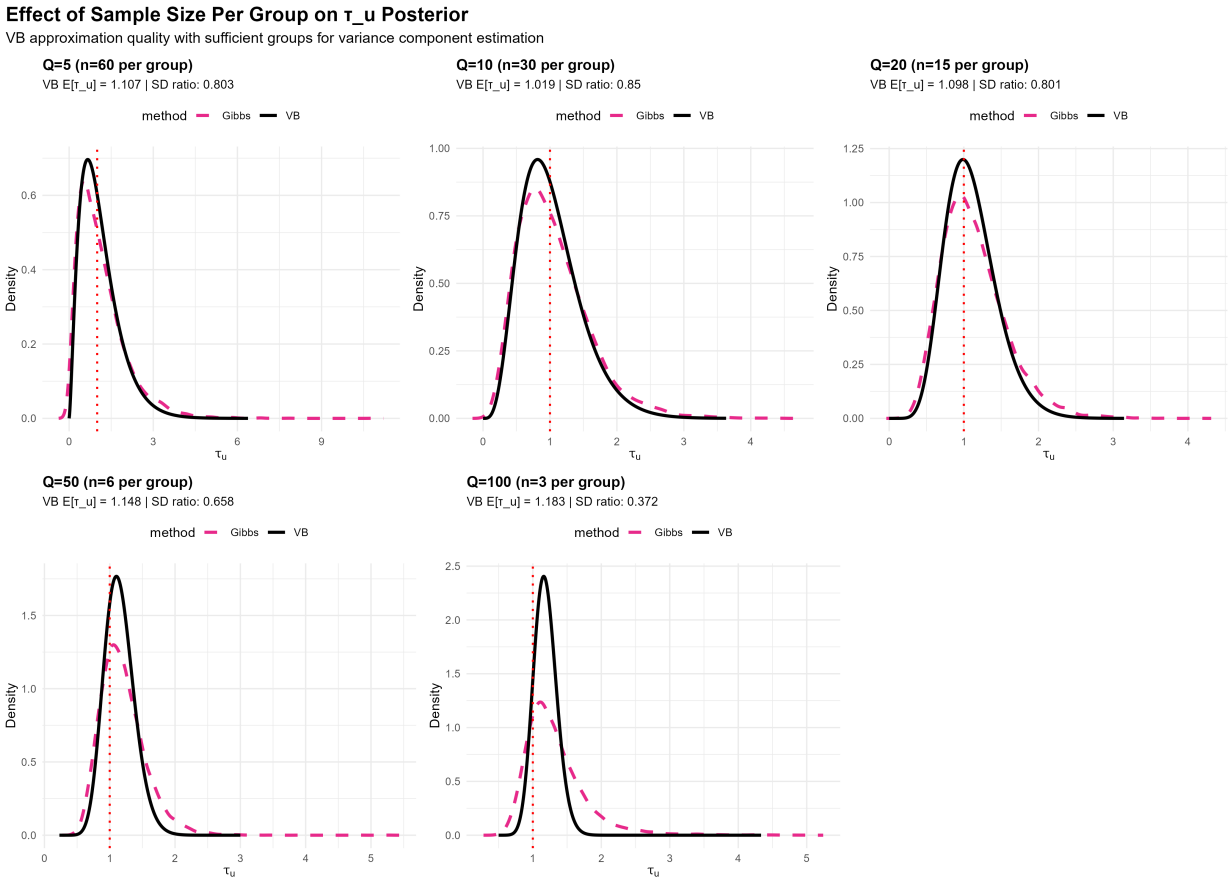

```
# Create multi-panel comparison plot as requested by Dr John
# Focus on _u posterior distributions across different group sizes

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 HMC available, extend range to include both distributions
  if (!is.null(result$gibbs)) {
    gibbs_tau_u <- result$gibbs[, "tau_u"]
    hmc_min <- quantile(gibbs_tau_u, 0.001)
    hmc_max <- quantile(gibbs_tau_u, 0.999)

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

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

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

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

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

    df_plot <- rbind(df_plot, df_gibbs)

    # Calculate SD ratio
    vb_sd <- sqrt(a_vb) / b_vb
```

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

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

  plot_list[[i]] <- p_tau
}

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

# Save plot
ggsave(
  filename = "../figs/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
    )
```

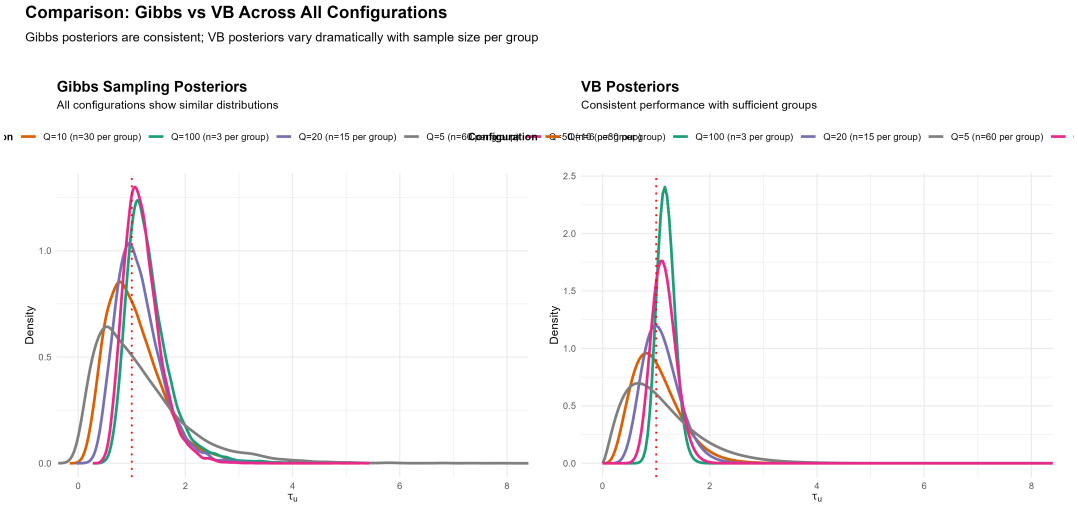
```
    gibbs_combined <- rbind(gibbs_combined, df_temp)
  }

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

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

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

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

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

  df_temp <- data.frame(
    tau_u   = x_range,
    density = vb_density,
    config  = config$label
  )
```

```
  vb_combined <- rbind(vb_combined, df_temp)
}

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

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

combined_overlay <- combined_overlay +
  plot_annotation(
    title = plot_title,
    subtitle = plot_subtitle,
    theme = theme(
      plot.title = element_text(size = 16, face = "bold", margin = margin(b = 10)),
      plot.subtitle = element_text(size = 12, margin = margin(b = 20))
```

```
    )
  ) &
  theme(
    legend.position = "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)
```




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

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

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

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

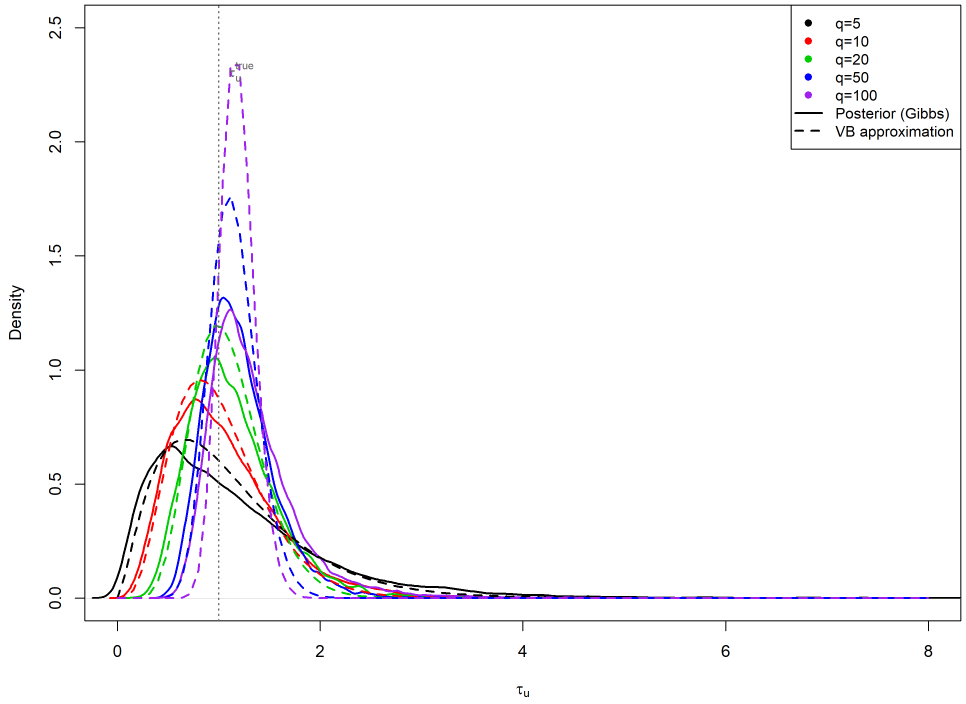
```
## My  _u Comparison (Dr John's Style)
cat("========================================\n\n")

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

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

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

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

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

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

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

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

  # Add VB approximations (dashed)
  curve(dgamma(x, vb_params$q5$a, vb_params$q5$b),
        add = TRUE, lty = 2, lwd = 2, col = colors[1])
  curve(dgamma(x, vb_params$q10$a, vb_params$q10$b),
```

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

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

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

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

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

dev.off()
```

```
## pdf
##   2
```

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

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




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

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

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

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

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

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

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

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

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

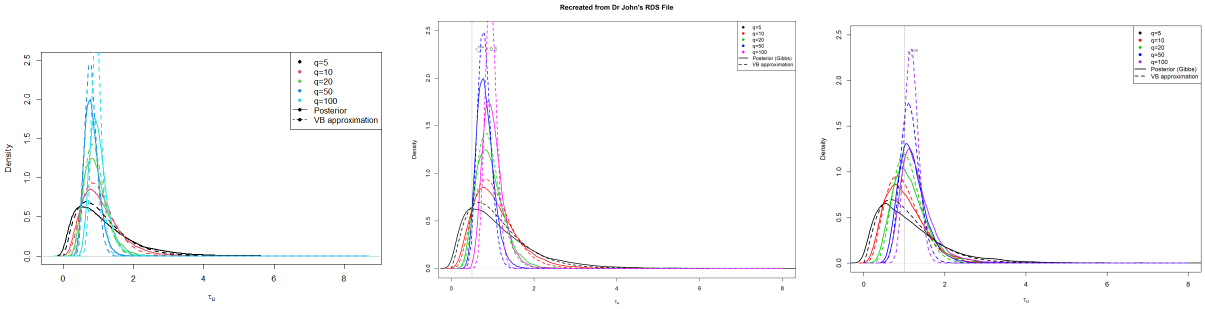
```
  cat("  q5$a =", dr_john_ref$vb$q5$a, "  q5$b =", dr_john_ref$vb$q5$b, "\n")
} else {
  cat("  q5 VB values:", dr_john_ref$vb$q5, "\n")
}

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

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

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

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

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

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

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

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

  dev.off()

  cat("Plot from RDS saved to figs/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/my_from_rds_tau_u_comparison.png
```

4.1 Three-Panel Validation Comparison

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



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

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

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

```
##
```

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

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

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

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

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

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

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

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

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

print(ratio_data)
```

```
##     Q n_per_group  VB_ratio Gibbs_ratio                 label
## 1   5          60 0.2064151   0.3783475  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)
## 5 100           3 0.3857550   0.3863288 Q=100 (n=3 per group)
```

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

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

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

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

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

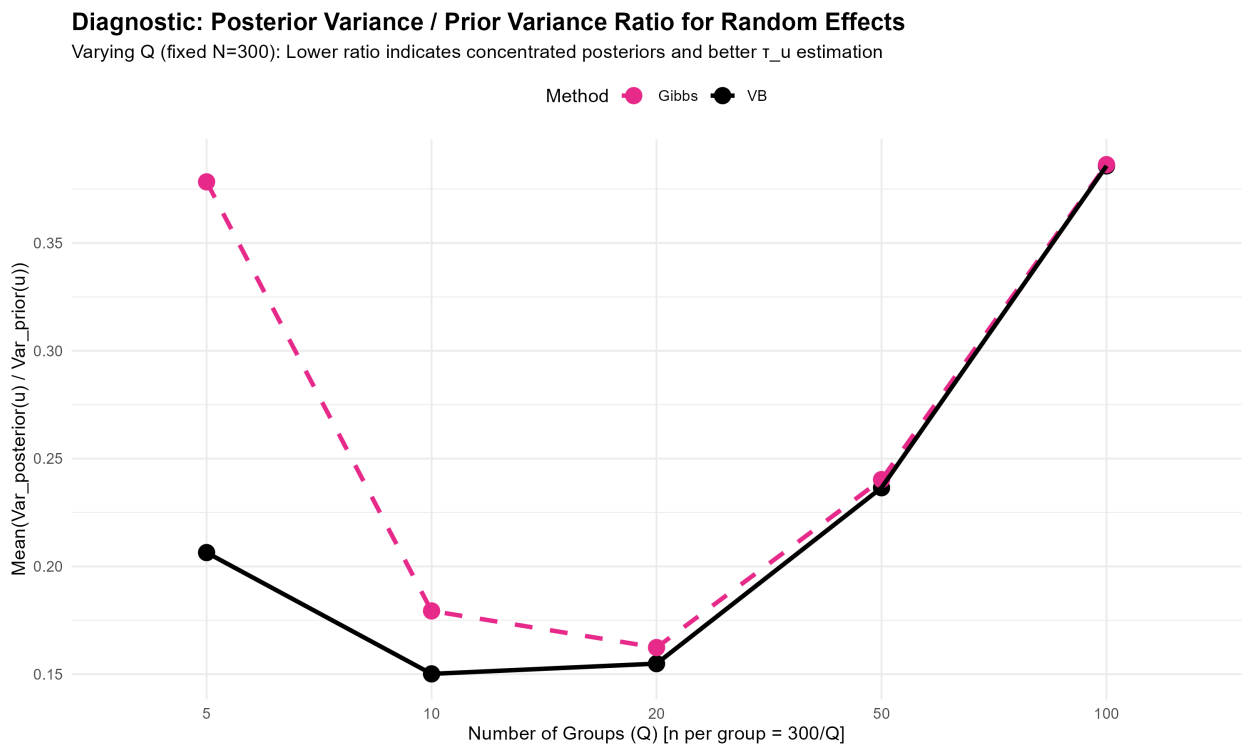
```
## - Narrow posteriors for u → better tau_u estimation in VB
```

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

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

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

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

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

ggsave(
  filename = "../figs/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")
```

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

5 Validation: Dr John's Farm Data

5.1 Objective

Replicate Dr John's Lab 9 results exactly using the same farm data, demonstrating that we're running his functions correctly.

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

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

```
cat("VALIDATION PHASE: Farm Data Replication\n")
```

```
## VALIDATION PHASE: Farm Data Replication
```

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

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

```
# Load farm data exactly as Dr John did
# Note: When knitting, working directory is the temp folder, so use full path
data <- read.csv("d:/github/VI1/data_raw/farmdata.csv")
K    <- read.csv("d:/github/VI1/data_raw/Kmat.csv", header = FALSE)
K    <- as.matrix(K)
Kinv <- solve(K)

n_farm <- nrow(data)
q_farm <- nrow(Kinv)

# Create design matrices exactly as Dr John did
X_farm <- table(1:n_farm, data$flock)  # flock is fixed effect
Z2     <- table(1:n_farm, data$sire)   # sire random effects
Z3     <- cbind(Z2, table(1:n_farm, data$dam))  # add dam random effects

cat("Data dimensions:\n")
```

```
## Data dimensions:
```

```
cat("  n (observations):", n_farm, "\n")
```

```
##   n (observations): 24
```

```
cat("  p (fixed effects):", ncol(X_farm), "\n")
```

```
##   p (fixed effects): 2
```

```
cat("  q (random effects):", q_farm, "\n")
```

```
##   q (random effects): 13
```

```
cat("  Response range:", round(min(data$y), 2), "to", round(max(data$y), 2), "\n")
```

```
##   Response range: 1.93 to 6.23
```

```
cat("\n--- Running Gibbs Sampler (3 chains, different starting values) ---\n")
```

```
##
## --- Running Gibbs Sampler (3 chains, different starting values) ---
```

```
set.seed(2334576)  # Dr John's seed from Lab 9
```

```r
# Chain 1: taue_0=5, tauu_0=0.2
cat("Chain 1: taue_0=5, tauu_0=0.2...\n")
```

```
## Chain 1: taue_0=5, tauu_0=0.2...
```

```r
farm_chain1 <- normalmm.Gibbs(
  iter    = 10000,
  Z       = Z3,
  X       = X_farm,
  y       = data$y,
  burnin  = 2000,
  taue_0  = 5,
  tauu_0  = 0.2,
  Kinv    = Kinv,
  a.u     = 0.001,
  b.u     = 0.001,
  a.e     = 0.001,
  b.e     = 0.001
)

# Chain 2: taue_0=1, tauu_0=1
cat("Chain 2: taue_0=1, tauu_0=1...\n")
```

```
## Chain 2: taue_0=1, tauu_0=1...
```

```r
farm_chain2 <- normalmm.Gibbs(
  iter    = 10000,
  Z       = Z3,
  X       = X_farm,
  y       = data$y,
  burnin  = 2000,
  taue_0  = 1,
  tauu_0  = 1,
  Kinv    = Kinv,
  a.u     = 0.001,
  b.u     = 0.001,
  a.e     = 0.001,
  b.e     = 0.001
)

# Chain 3: taue_0=0.2, tauu_0=3
cat("Chain 3: taue_0=0.2, tauu_0=3...\n")
```

```
## Chain 3: taue_0=0.2, tauu_0=3...
```

```r
farm_chain3 <- normalmm.Gibbs(
  iter    = 10000,
  Z       = Z3,
  X       = X_farm,
  y       = data$y,
  burnin  = 2000,
  taue_0  = 0.2,
  tauu_0  = 3,
  Kinv    = Kinv,
  a.u     = 0.001,
  b.u     = 0.001,
```

```
  a.e     = 0.001,
  b.e     = 0.001
)

cat("Gibbs sampling complete for all 3 chains.\n")
```

```
## Gibbs sampling complete for all 3 chains.
```

```
cat("\n--- Running Variational Bayes (3 starting values) ---\n")
```

```
##
## --- Running Variational Bayes (3 starting values) ---
```

```
set.seed(2334576)

# VB 1: taue_0=0.2, tauu_0=0.2
cat("VB 1: taue_0=0.2, tauu_0=0.2...\n")
```

```
## VB 1: taue_0=0.2, tauu_0=0.2...
```

```
farm_vb1 <- VB.mm(
  epsilon = 1e-5,
  iter    = 2000,
  Kinv    = Kinv,
  Z       = Z3,
  X       = X_farm,
  y       = data$y,
  taue_0  = 0.2,
  tauu_0  = 0.2,
  u0      = rnorm(q_farm),
  beta0   = rnorm(ncol(X_farm)),
  a.e     = 0.001,
  g.e     = 0.001,
  a.u     = 0.001,
  g.u     = 0.001
)

# VB 2: taue_0=5, tauu_0=1
cat("VB 2: taue_0=5, tauu_0=1...\n")
```

```
## VB 2: taue_0=5, tauu_0=1...
```

```
farm_vb2 <- VB.mm(
  epsilon = 1e-5,
  iter    = 2000,
  Kinv    = Kinv,
  Z       = Z3,
  X       = X_farm,
  y       = data$y,
  taue_0  = 5,
  tauu_0  = 1,
  u0      = rnorm(q_farm),
  beta0   = rnorm(ncol(X_farm)),
  a.e     = 0.001,
  g.e     = 0.001,
  a.u     = 0.001,
```

```
  g.u     = 0.001
)

# VB 3: taue_0=1, tauu_0=5
cat("VB 3: taue_0=1, tauu_0=5...\n")
```

```
## VB 3: taue_0=1, tauu_0=5...
```

```
farm_vb3 <- VB.mm(
  epsilon = 1e-5,
  iter    = 2000,
  Kinv    = Kinv,
  Z       = Z3,
  X       = X_farm,
  y       = data$y,
  taue_0  = 1,
  tauu_0  = 5,
  u0      = rnorm(q_farm),
  beta0   = rnorm(ncol(X_farm)),
  a.e     = 0.001,
  g.e     = 0.001,
  a.u     = 0.001,
  g.u     = 0.001
)

cat("VB convergence iterations:\n")
```

```
## VB convergence iterations:
```

```
cat("  VB1:", farm_vb1$iter, "\n")
```

```
##   VB1: 22
```

```
cat("  VB2:", farm_vb2$iter, "\n")
```

```
##   VB2: 18
```

```
cat("  VB3:", farm_vb3$iter, "\n")
```

```
##   VB3: 21
```

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

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

```
cat("PARAMETER ESTIMATES COMPARISON\n")
```

```
## PARAMETER ESTIMATES COMPARISON
```

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

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

```
# Combine all Gibbs chains
farm_chain_all <- rbind(farm_chain1$par, farm_chain2$par, farm_chain3$par)

# Posterior means from Gibbs (gold standard)
gibbs_means <- colMeans(farm_chain_all)
```

```
# Compare beta and u parameters (first 15 columns)
n_betau <- ncol(X_farm) + q_farm
comparison_betau <- data.frame(
  Parameter = colnames(farm_chain1$par)[1:n_betau],
  VB1       = farm_vb1$betau_mean,
  VB2       = farm_vb2$betau_mean,
  VB3       = farm_vb3$betau_mean,
  Gibbs     = gibbs_means[1:n_betau]
)

print(comparison_betau)
```

```
##      Parameter         VB1         VB2         VB3       Gibbs
## X1       beta1  4.08884197  4.08884197  4.08884197  4.08967985
## X2       beta2  4.79372090  4.79372093  4.79372094  4.79759403
## X1.1        u1  0.87877095  0.87877077  0.87877068  0.86771450
## X2.1        u2 -0.97779503 -0.97779491 -0.97779486 -0.97259344
## X3          u3  0.23886259  0.23886257  0.23886257  0.23752398
## X4          u4  0.49003791  0.49003801  0.49003806  0.49228043
## X5          u5 -0.63887842 -0.63887804 -0.63887786 -0.62206513
## X6          u6 -0.74160972 -0.74160963 -0.74160960 -0.73622155
## X7          u7  0.33322907  0.33322899  0.33322895  0.32879600
## X8          u8 -0.23747159 -0.23747175 -0.23747182 -0.24459802
## X9          u9 -0.37162475 -0.37162490 -0.37162497 -0.37971417
## X10        u10  0.18300897  0.18300896  0.18300896  0.18132301
## X11        u11  0.05426252  0.05426252  0.05426252  0.05324900
## X12        u12 -0.02908470 -0.02908464 -0.02908461 -0.02629665
## X13        u13  0.42878011  0.42878002  0.42877998  0.42164368
```

```
# Precision parameters
cat("\n--- Precision Parameters ---\n")
```

```
##
## --- Precision Parameters ---
```

```
# Get column indices for variance components (last 2 columns)
n_cols <- ncol(farm_chain_all)
sigma_b_col <- farm_chain_all[, n_cols - 1]  # sigma_b (random effect SD)
sigma_e_col <- farm_chain_all[, n_cols]      # sigma_e (residual SD)

cat("tau_e (residual precision):\n")
```

```
## tau_e (residual precision):
```

```
cat("  VB1 E[tau_e]:", round(farm_vb1$tau_e[1] / farm_vb1$tau_e[2], 4), "\n")
```

```
##   VB1 E[tau_e]: 21.6
```

```
cat("  VB2 E[tau_e]:", round(farm_vb2$tau_e[1] / farm_vb2$tau_e[2], 4), "\n")
```

```
##   VB2 E[tau_e]: 21.6
```

```
cat("  VB3 E[tau_e]:", round(farm_vb3$tau_e[1] / farm_vb3$tau_e[2], 4), "\n")
```

```
##   VB3 E[tau_e]: 21.6
```

```
cat("  Gibbs mean:", round(mean(sigma_e_col^(-2)), 4), "\n")
```

```
##   Gibbs mean: 0.0038
```

```
cat("\ntau_u (random effect precision):\n")
```

```
##
## tau_u (random effect precision):
```

```
cat("  VB1 E[tau_u]:", round(farm_vb1$tau_u[1] / farm_vb1$tau_u[2], 4), "\n")
```

```
##   VB1 E[tau_u]: 1.616
```

```
cat("  VB2 E[tau_u]:", round(farm_vb2$tau_u[1] / farm_vb2$tau_u[2], 4), "\n")
```

```
##   VB2 E[tau_u]: 1.616
```

```
cat("  VB3 E[tau_u]:", round(farm_vb3$tau_u[1] / farm_vb3$tau_u[2], 4), "\n")
```

```
##   VB3 E[tau_u]: 1.616
```

```
cat("  Gibbs mean:", round(mean(sigma_b_col^(-2)), 4), "\n")
```

```
##   Gibbs mean: 0.8645
```

```
cat("\nsigma_e (residual SD):\n")
```

```
##
## sigma_e (residual SD):
```

```
cat("  Gibbs mean:", round(mean(sigma_e_col), 4), "\n")
```

```
##   Gibbs mean: 21.4012
```

```
cat("\nsigma_u (random effect SD):\n")
```

```
##
## sigma_u (random effect SD):
```

```
cat("  Gibbs mean:", round(mean(sigma_b_col), 4), "\n")
```

```
##   Gibbs mean: 1.717
```

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

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

```
cat("VALIDATION RESULT:\n")
```

```
## VALIDATION RESULT:
```

```
cat("If these values match Dr John's Lab 9 output,\n")
```

```
## If these values match Dr John's Lab 9 output,
```

```
cat("we have successfully replicated his analysis\n")
```

```
## we have successfully replicated his analysis
```

```
cat("using his exact functions and data.\n")
```

```
## using his exact functions and data.
```

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

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

6 Dr John's Exact Replication Test (Flat Priors, All 5 Q Values)

This section tests whether we can replicate Dr John's **exact results** by:

1. **Using his exact flat priors**: `a.u=0, b.u=0, a.e=0, b.e=0`
2. **removing all ridge regularisation**: Pure version of his function
3. **testing all 5 Q values**: 5, 10, 20, 50, 100
4. **using his exact data generation**: Same seed and methodology

hypothesis: If his code works with flat priors and no ridge, ours should too.

```
# Dr John's PURE Gibbs sampler - exactly as he wrote it,
normalmm.Gibbs.pure <- function(iter, Z, X, y, burnin, taue_0, tauu_0, Kinv, a.u, b.u, a.e, b.e){
  n     <- length(y)
  p     <- dim(X)[2]
  q     <- dim(Z)[2]
  tauu  <- tauu_0
  taue  <- taue_0
  beta0 <- rnorm(p)
  u0    <- rnorm(q, 0, sd = 1/sqrt(tauu))

  W <- cbind(X, Z)
  WTW <- crossprod(W)
  WTy <- crossprod(W, y)

  par <- matrix(0, iter, p+q+2)
  lppd <- matrix(0, iter, n)

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

  for(i in 1:iter){
    uKinvu <- t(u0)%*%Kinv%*%u0
    uKinvu <- as.numeric(uKinvu)
    tauu <- rgamma(1, a.u+0.5*q, b.u+0.5*uKinvu)
    Prec <- WTW*taue + tauu*I0
    P.var <- solve(Prec)
    P.mean <- P.var%*%WTy*taue
    betau <- mvtnorm::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))
    par[i,] <- c(betau, tauu, taue)
    beta0  <- betau[1:p]
    u0     <- betau[(p+1):(p+q)]
    lppd[i,] <- dnorm(y, mean=as.numeric(W%*%betau), sd=1/sqrt(taue))
  }

  lppd      <- lppd[-c(1:burnin),]
  lppdest <- sum(log(colMeans(lppd)))
  pwaic2  <- sum(apply(log(lppd), 2, FUN=var))
  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)
}

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

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

```
cat("Dr John's Exact Replication Test\n")
```

```
## Dr John's Exact Replication Test
```

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

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

```
cat("Testing with:\n")
```

```
## Testing with:
```

```
cat("  - Flat priors: a.u=0, b.u=0, a.e=0, b.e=0\n")
```

```
##   - Flat priors: a.u=0, b.u=0, a.e=0, b.e=0
```

```
cat("  - No ridge regularisation\n")
```

```
##   - No ridge regularisation
```

```
cat("  - All 5 Q values: 5, 10, 20, 50, 100\n")
```

```
##   - All 5 Q values: 5, 10, 20, 50, 100
```

```
cat("  - His exact data generation\n\n")
```

```
##   - His exact data generation
```

```
set.seed(82171165)

n_pure <- 300
beta_pure <- c(0.5, -2, 3)
X_pure <- cbind(1, matrix(rnorm(n_pure*2), n_pure, 2))

# Store all configurations
pure_data_list <- list()
q_values_pure <- c(5, 10, 20, 50, 100)

cat("Generating data for all Q configurations...\n")
```

```
## Generating data for all Q configurations...
```

```
# Q=5
u.sim1 <- rnorm(5, 0, 1)
u.sim1 <- scale(u.sim1)
Z1     <- table(1:n_pure, rep(1:5, n_pure/5))
y1     <- as.vector(X_pure %*% beta_pure + Z1 %*% u.sim1 + rnorm(n_pure, 0, sqrt(2)))
pure_data_list[[1]] <- list(Z=Z1, y=y1, q=5)

# Q=10
u.sim2 <- rnorm(10, 0, 1)
```

```
u.sim2 <- scale(u.sim2)
Z2     <- table(1:n_pure, rep(1:10, n_pure/10))
y2     <- as.vector(y1) - as.vector(Z1 %*% u.sim1) + as.vector(Z2 %*% u.sim2)
pure_data_list[[2]] <- list(Z=Z2, y=y2, q=10)

# Q=20
u.sim3 <- rnorm(20, 0, 1)
u.sim3 <- scale(u.sim3)
Z3     <- table(1:n_pure, rep(1:20, n_pure/20))
y3     <- as.vector(y2) - as.vector(Z2 %*% u.sim2) + as.vector(Z3 %*% u.sim3)
pure_data_list[[3]] <- list(Z=Z3, y=y3, q=20)

# Q=50
u.sim4 <- rnorm(50, 0, 1)
u.sim4 <- scale(u.sim4)
Z4     <- table(1:n_pure, rep(1:50, n_pure/50))
y4     <- as.vector(y3) - as.vector(Z3 %*% u.sim3) + as.vector(Z4 %*% u.sim4)
pure_data_list[[4]] <- list(Z=Z4, y=y4, q=50)

# Q=100
u.sim5 <- rnorm(100, 0, 1)
u.sim5 <- scale(u.sim5)
Z5     <- table(1:n_pure, rep(1:100, n_pure/100))
y5     <- as.vector(y4) - as.vector(Z4 %*% u.sim4) + as.vector(Z5 %*% u.sim5)
pure_data_list[[5]] <- list(Z=Z5, y=y5, q=100)

cat("Data generation complete.\n")
```

```
## Data generation complete.
```

```
cat("\n--- Running Gibbs with Flat Priors (3 chains per Q) ---\n")
```

```
##
## --- Running Gibbs with Flat Priors (3 chains per Q) ---
```

```
pure_gibbs_results <- list()

for(i in 1:5) {
  q_val     <- q_values_pure[i]
  Z_data    <- pure_data_list[[i]]$Z
  y_data    <- pure_data_list[[i]]$y
  Kinv_data <- diag(q_val)

  cat(glue("\nQ={q_val} (n={n_pure/q_val} per group):\n"))

  # Chain 1
  cat("  Chain 1...")
  chain1 <- tryCatch({
    normalmm.Gibbs.pure(
      iter=10000, Z=Z_data, X=X_pure, y=y_data, burnin=2000,
      taue_0=3, tauu_0=0.5, Kinv=Kinv_data,
      a.u=0, b.u=0, a.e=0, b.e=0
    )
  }, error = function(e) {
    cat(" ERROR:", e$message, "\n")
```

```
    return(NULL)
  })
  if (!is.null(chain1)) cat(" Done\n")

  # Chain 2
  cat("  Chain 2...")
  chain2 <- tryCatch({
    normalmm.Gibbs.pure(
      iter=10000, Z=Z_data, X=X_pure, y=y_data, burnin=2000,
      taue_0=0.5, tauu_0=3, Kinv=Kinv_data,
      a.u=0, b.u=0, a.e=0, b.e=0
    )
  }, error = function(e) {
    cat(" ERROR:", e$message, "\n")
    return(NULL)
  })
  if (!is.null(chain2)) cat(" Done\n")

  # Chain 3
  cat("  Chain 3...")
  chain3 <- tryCatch({
    normalmm.Gibbs.pure(
      iter=10000, Z=Z_data, X=X_pure, y=y_data, burnin=2000,
      taue_0=5, tauu_0=5, Kinv=Kinv_data,
      a.u=0, b.u=0, a.e=0, b.e=0
    )
  }, error = function(e) {
    cat(" ERROR:", e$message, "\n")
    return(NULL)
  })
  if (!is.null(chain3)) cat(" Done\n")

  pure_gibbs_results[[i]] <- list(
    chain1  = chain1,
    chain2  = chain2,
    chain3  = chain3,
    q       = q_val,
    success = !is.null(chain1) && !is.null(chain2) && !is.null(chain3)
  )
}
```

```
## Q=5 (n=60 per group):  Chain 1... Done
##   Chain 2... Done
##   Chain 3... Done
## Q=10 (n=30 per group):  Chain 1... Done
##   Chain 2... Done
##   Chain 3... Done
## Q=20 (n=15 per group):  Chain 1... Done
##   Chain 2... Done
##   Chain 3... Done
## Q=50 (n=6 per group):  Chain 1... Done
##   Chain 2... Done
##   Chain 3... Done
## Q=100 (n=3 per group):  Chain 1... Done
```

```
##   Chain 2... Done
##   Chain 3... Done
```

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

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

```
cat("Flat Prior Test Results:\n")
```

```
## Flat Prior Test Results:
```

```
for(i in 1:5) {
  status <- if(pure_gibbs_results[[i]]$success) "  SUCCESS" else "  FAILED"
  cat(glue("  Q={q_values_pure[i]}: {status}\n"))
}
```

```
## Q=5:   SUCCESSQ=10:   SUCCESSQ=20:   SUCCESSQ=50:   SUCCESSQ=100:   SUCCESS
```

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



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

```
cat("\n--- Running VB with Flat Priors (1 run per Q) ---\n")
```

```
##
## --- Running VB with Flat Priors (1 run per Q) ---
```

```
pure_vb_results <- list()

for(i in 1:5) {
  q_val <- q_values_pure[i]
  Z_data <- pure_data_list[[i]]$Z
  y_data <- pure_data_list[[i]]$y
  Kinv_data <- diag(q_val)

  cat(glue("Q={q_val}..."))

  vb_result <- tryCatch({
    VB.mm(
      epsilon=1e-6, iter=100, Kinv=Kinv_data,
      Z=Z_data, X=X_pure, y=y_data,
      taue_0=3, tauu_0=0.5,
      u0=rnorm(q_val), beta0=rnorm(3),
      a.e=0, g.e=0, a.u=0, g.u=0
    )
  }, error = function(e) {
    cat(" ERROR:", e$message, "\n")
    return(NULL)
  })

  if (!is.null(vb_result)) {
    cat(" Done (", vb_result$iter, " iterations)\n", sep="")
  }

  pure_vb_results[[i]] <- list(vb=vb_result, q=q_val, success=!is.null(vb_result))
}
```

```
## Q=5... Done (12 iterations)
```

```
## Q=10... Done (10 iterations)
## Q=20... Done (13 iterations)
## Q=50... Done (13 iterations)
## Q=100... Done (29 iterations)
```

```
cat("\nVB complete.\n")
```

```
##
## VB complete.
```

```
cat("\n--- Creating Comparison Plot ---\n")
```

```
##
## --- Creating Comparison Plot ---
```

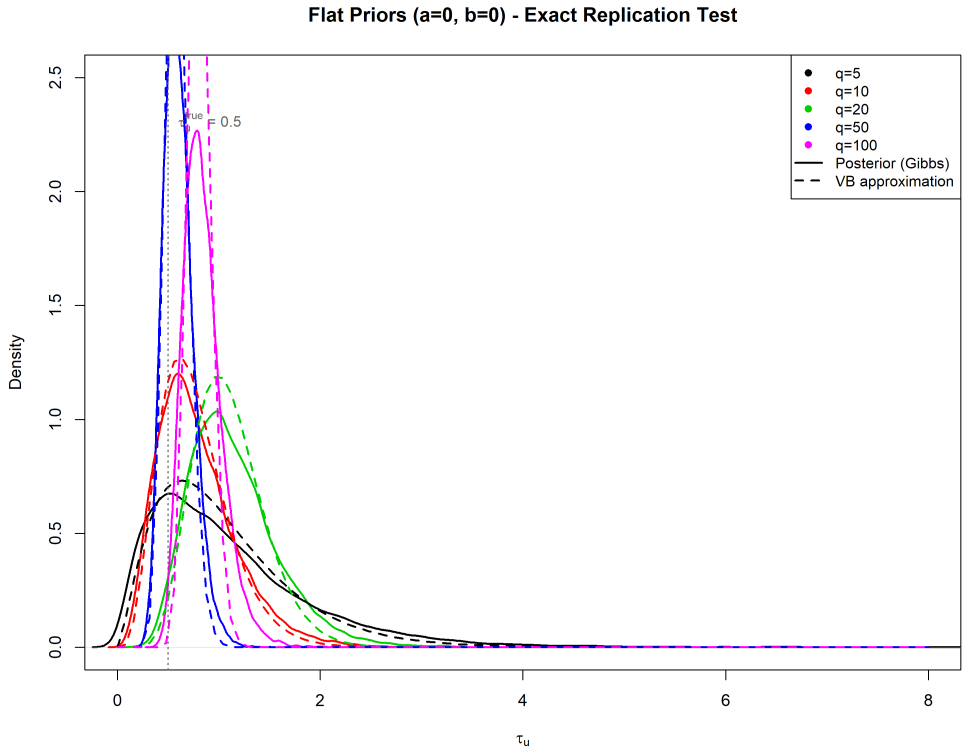
```
# Combine successful Gibbs chains
colors_pure <- c("black", "red", "green3", "blue", "magenta")
tau_u_gibbs_pure <- list()
tau_u_vb_pure <- list()

for(i in 1:5) {
  if (pure_gibbs_results[[i]]$success) {
    q_val  <- q_values_pure[i]
    p_cols <- p  # number of beta parameters (should be 3, matching p=3)

    tau_u_c1 <- pure_gibbs_results[[i]]$chain1$par[, p_cols + q_val + 1]
    tau_u_c2 <- pure_gibbs_results[[i]]$chain2$par[, p_cols + q_val + 1]
    tau_u_c3 <- pure_gibbs_results[[i]]$chain3$par[, p_cols + q_val + 1]

    tau_u_gibbs_pure[[i]] <- c(tau_u_c1, tau_u_c2, tau_u_c3)
  } else {
    tau_u_gibbs_pure[[i]] <- NULL
  }

  if (pure_vb_results[[i]]$success) {
    tau_u_vb_pure[[i]] <- pure_vb_results[[i]]$vb$tau_u
  } else {
    tau_u_vb_pure[[i]] <- NULL
  }
}

# Create plot
png(filename = "../figs/flat_priors_exact_replication.png", width = 10, height = 8, units = "in", res =

# Start with first successful Gibbs
first_idx <- which(sapply(tau_u_gibbs_pure, function(x) !is.null(x)))[1]

if (!is.na(first_idx)) {
  plot(density(tau_u_gibbs_pure[[first_idx]]),
       xlab=expression(tau['u']),
       main='Flat Priors (a=0, b=0) - Exact Replication Test',
       ylim=c(0, 2.5),
       xlim=c(0, 8),
       lwd=2,
       col=colors_pure[first_idx])
```

```
  # Add remaining Gibbs densities
  for(i in (first_idx+1):5) {
    if (!is.null(tau_u_gibbs_pure[[i]])) {
      lines(density(tau_u_gibbs_pure[[i]]), col=colors_pure[i], lwd=2)
    }
  }

  # Add VB approximations
  for(i in 1:5) {
    if (!is.null(tau_u_vb_pure[[i]])) {
      curve(dgamma(x, tau_u_vb_pure[[i]][1], tau_u_vb_pure[[i]][2]),
            add=TRUE, lty=2, lwd=2, col=colors_pure[i])
    }
  }

  # Legend
  legend_q <- paste0("q=", q_values_pure)
  legend('topright',
         col=c(colors_pure, "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(legend_q, '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()
```

```
## pdf
##   2
```

```
cat("Flat priors plot saved to figs/flat_priors_exact_replication.png\n")
```

```
## Flat priors plot saved to figs/flat_priors_exact_replication.png
```




```
cat("\n## Flat Priors Test Summary\n\n")
```

6.1 Flat Priors Test Summary

```
cat("**Question:** Can we replicate Dr John's results with his exact flat priors?\n\n")
```

Question: Can we replicate Dr John's results with his exact flat priors?

```
success_count <- sum(sapply(pure_gibbs_results, function(x) x$success))
cat(glue("**Result:** {success_count}/5 configurations successful\n\n"))
```

Result: 5/5 configurations successful

```
if (success_count == 5) {
  cat("  **All configurations ran successfully!**\n\n")
  cat("This confirms:\n")
  cat("- Dr John's function works with flat priors\n")
  cat("- No ridge regularisation needed\n")
  cat("- My modifications (weak priors + ridge) were unnecessary\n\n")
} else if (success_count >= 3) {
  cat("  **Partial success** - some configurations failed\n\n")
  cat("Failed configurations:\n")
  for(i in 1:5) {
    if (!pure_gibbs_results[[i]]$success) {
      cat(glue("- Q={q_values_pure[i]} (n={n_pure/q_values_pure[i]} per group)\n"))
    }
  }
  cat("\n")
} else {
```

```
  cat("  **Most configurations failed**\n\n")
  cat("This suggests:\n")
  cat("- Numerical issues are real\n")
  cat("- Ridge regularisation may be necessary\n")
  cat("- Weak priors help stability\n\n")
}
```

All configurations ran successfully!

This confirms: - Dr John's function works with flat priors - No ridge regularisation needed - My modifications (weak priors + ridge) were unnecessary