Bayesian Model for Stroke Outcomes

```
# ----------------------------
# Bayesian stroke model - Final specification with Age sub-model
# ----------------------------

# packages
if (!requireNamespace("R2jags", quietly = TRUE)) install.packages("R2jags")
if (!requireNamespace("coda", quietly = TRUE)) install.packages("coda")
if (!requireNamespace("dplyr", quietly = TRUE)) install.packages("dplyr")
if (!requireNamespace("pROC", quietly = TRUE)) install.packages("pROC")

library(R2jags)
```

```
Loading required package: rjags
```

```
Loading required package: coda
```

```
Linked to JAGS 4.3.2
```

```
Loaded modules: basemod,bugs
```

```

Attaching package: 'R2jags'
```

```
The following object is masked from 'package:coda':

    traceplot
```

```
library(coda)
library(dplyr)
```

```
Attaching package: 'dplyr'

The following objects are masked from 'package:stats':

    filter, lag

The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union
```

```
library(pROC)
```

```
Type 'citation("pROC")' for a citation.


Attaching package: 'pROC'

The following objects are masked from 'package:stats':

    cov, smooth, var
```

```
# ---------- 0. load strokeStudy.RData ----------
obj_names <- load("strokeStudy.RData")
message("Loaded objects: ", paste(obj_names, collapse = ", "))
```

```
Loaded objects: x
```

```
if ("strokeStudy" %in% obj_names) {
  strokeStudy <- get("strokeStudy")
} else {
  possible <- obj_names[sapply(obj_names, function(n) is.data.frame(get(n)))]
  if (length(possible) == 0) stop("No data.frame found in strokeStudy.RData.")
  strokeStudy <- get(possible[1])
  message("Using object '", possible[1], "' as strokeStudy.")
}
```

```
Using object 'x' as strokeStudy.
```

```r
# ---------- 1. Data preparation ----------
df <- strokeStudy

required_cols <- c("homeOrRehab","siteID","Time2","Age","Gender","EMSvsCar","hadTPA","hadThro
missing_cols <- setdiff(required_cols, names(df))
if (length(missing_cols) > 0) stop("Missing required columns: ", paste(missing_cols, collapse

# Outcome: Y_i = 1 if discharged home or to inpatient rehab, 0 otherwise
df$y <- as.integer(
  (is.logical(df$homeOrRehab) & df$homeOrRehab == TRUE) |
  toupper(as.character(df$homeOrRehab)) %in% c("YES","HOME","TRUE","1")
)

# Site index
df$site <- as.integer(factor(df$siteID))
nSite <- length(unique(df$site))

# Time as integer
df$time <- as.integer(factor(df$Time2))

# Program indicator: 1 if Time2 >= 4 (post-implementation), 0 otherwise
df$program <- as.integer(df$time >= 4)

# Age: numeric (can be missing - will be imputed)
df$Age_num <- suppressWarnings(as.numeric(as.character(df$Age)))

# Female indicator
df$female <- as.integer(toupper(as.character(df$Gender)) == "FEMALE")

# EMS indicator
df$EMS_chr <- toupper(as.character(df$EMSvsCar))
df$ems <- as.integer(df$EMS_chr %in% c("EMS","1","TRUE","YES"))

# Treatment indicators
df$tpa <- as.integer(toupper(as.character(df$hadTPA)) %in% c("TRUE","1","YES"))
df$thr <- as.integer(toupper(as.character(df$hadThrombectomy)) %in% c("TRUE","1","YES"))

# For this model: only require non-missing outcome, site, and fully observed covariates
# Age can be missing (will be imputed based on Program, Female, EMS)
vars_essential <- c("y", "site", "program", "female", "ems", "tpa", "thr")
df_model <- df[complete.cases(df[, vars_essential]), ]
```

```
message("\nData summary:")
```

```
Data summary:
```

```
message("Total rows in original data: ", nrow(df))
```

```
Total rows in original data: 1752
```

```
message("Rows with complete essential variables: ", nrow(df_model))
```

```
Rows with complete essential variables: 1694
```

```
message("Number of sites: ", nSite)
```

```
Number of sites: 9
```

```
message("Missing Age values: ", sum(is.na(df_model$Age_num)))
```

```
Missing Age values: 790
```

```
message("\nOutcome distribution:")
```

```
Outcome distribution:
```

```
print(table(df_model$y, useNA = "ifany"))
```

```
   0    1
 484 1210
```

```r
if (nrow(df_model) == 0) stop("No valid cases available for analysis")

# Standardize Age for better MCMC (use observed values only for standardization)
age_mean <- mean(df_model$Age_num, na.rm = TRUE)
age_sd <- sd(df_model$Age_num, na.rm = TRUE)
df_model$age_std <- (df_model$Age_num - age_mean) / age_sd

message("Age standardization: mean = ", round(age_mean, 2), ", sd = ", round(age_sd, 2))
```

```
Age standardization: mean = 68.37, sd = 14.88
```

```r
# Create JAGS data list
data_jags <- list(
  N       = nrow(df_model),
  J       = nSite,
  y       = df_model$y,
  site    = df_model$site,
  program = df_model$program,
  age     = df_model$age_std,  # Can contain NA - will be imputed
  female  = df_model$female,
  ems     = df_model$ems,
  tpa     = df_model$tpa,
  thr     = df_model$thr
)

# ---------- 2. JAGS model matching specification ----------
model_text <- "
model {
  # ==========================================
  # OUTCOME MODEL: Y_i ~ Bernoulli(p_i)
  # ==========================================
  # logit(p_i) =  _j(i) +  _0 +  _1*Program_i +  _2*A_i +
  #               _3*Female_i +  _4*EMS_i +  _5*TPA_i +  _6*Thrombectomy_i

  for (i in 1:N) {
    y[i] ~ dbern(p[i])
    logit(p[i]) <- alpha[site[i]] + beta0 +
                   beta1 * program[i] +
                   beta2 * age[i] +
                   beta3 * female[i] +
                   beta4 * ems[i] +
```

```
                  beta5 * tpa[i] +
                  beta6 * thr[i]

  # ============================================
  # AGE SUB-MODEL: A_i ~ N( _A,i,  _A²)
  # ============================================
  #  _A,i =  _0 +  _1*Female_i +  _2*Program_i +  _3*EMS_i
  # For observed Age, this contributes to likelihood
  # For missing Age, this is the imputation model

  age[i] ~ dnorm(mu_age[i], tau_age)
  mu_age[i] <- gamma0 + gamma1 * female[i] + gamma2 * program[i] + gamma3 * ems[i]
}

# ============================================
# SITE RANDOM INTERCEPTS
# ============================================
#  _j ~ N( _ ,  _ ²) for j = 1,...,J

for (j in 1:J) {
  alpha[j] ~ dnorm(mu_alpha, tau_alpha)
}

# ============================================
# PRIORS: Outcome Model Fixed Effects
# ============================================
#  _k ~ N(0, 10²) for k = 0,...,6

beta0 ~ dnorm(0, 0.01)   # precision = 1/100 = 0.01
beta1 ~ dnorm(0, 0.01)   # Program effect
beta2 ~ dnorm(0, 0.01)   # Age effect
beta3 ~ dnorm(0, 0.01)   # Female effect
beta4 ~ dnorm(0, 0.01)   # EMS effect
beta5 ~ dnorm(0, 0.01)   # TPA effect
beta6 ~ dnorm(0, 0.01)   # Thrombectomy effect

# ============================================
# PRIORS: Age Model Fixed Effects
# ============================================
#  _  ~ N(0, 10²) for   = 0,...,3

gamma0 ~ dnorm(0, 0.01)  # Baseline age (intercept)
```

```
  gamma1 ~ dnorm(0, 0.01)  # Female effect on age
  gamma2 ~ dnorm(0, 0.01)  # Program effect on age
  gamma3 ~ dnorm(0, 0.01)  # EMS effect on age

  # ============================================
  # HYPERPRIORS: Random Effects
  # ============================================
  #  _  ~ N(0, 10²)
  mu_alpha ~ dnorm(0, 0.01)

  #  _  ~ Half-Normal(0, 2.5)
  sigma_alpha ~ dnorm(0, 0.16) T(0,)  # precision = 1/(2.5²) = 0.16
  tau_alpha <- pow(sigma_alpha, -2)

  # ============================================
  # HYPERPRIORS: Age Model Variance
  # ============================================
  #  _A ~ Half-Normal(0, 20)
  sigma_age ~ dnorm(0, 0.0025) T(0,)  # precision = 1/(20²) = 0.0025
  tau_age <- pow(sigma_age, -2)
}
"
cat(model_text, file = "stroke_model_final.jags")
message("Wrote stroke_model_final.jags")
```

```
Wrote stroke_model_final.jags
```

```
# ---------- 3. Initial values function ----------
inits_fn <- function() {
  # CRITICAL: Only provide initial values for MISSING age values
  # Observed age values must remain NA in the inits
  age_init <- rep(NA_real_, length(df_model$age_std))
  missing_idx <- which(is.na(df_model$age_std))

  # Only initialize where age is actually missing
  if (length(missing_idx) > 0) {
    age_init[missing_idx] <- 0  # Start missing ages at 0 (standardized mean)
  }

  list(
    beta0 = rnorm(1, 0, 0.1),
```

```
    beta1 = rnorm(1, 0, 0.1),
    beta2 = rnorm(1, 0, 0.1),
    beta3 = rnorm(1, 0, 0.1),
    beta4 = rnorm(1, 0, 0.1),
    beta5 = rnorm(1, 0, 0.1),
    beta6 = rnorm(1, 0, 0.1),
    gamma0 = 0,
    gamma1 = 0,
    gamma2 = 0,
    gamma3 = 0,
    mu_alpha = rnorm(1, 0, 0.5),
    sigma_alpha = runif(1, 0.1, 2),
    sigma_age = runif(1, 0.5, 2),
    alpha = rnorm(data_jags$J, 0, 0.5),
    age = age_init  # NA for observed, initial values for missing
  )
}

# Parameters to save
params <- c("beta0", "beta1", "beta2", "beta3", "beta4", "beta5", "beta6",
            "gamma0", "gamma1", "gamma2", "gamma3",
            "mu_alpha", "sigma_alpha", "sigma_age", "alpha")

# ---------- 4. Run JAGS ----------
set.seed(440)
message("\nRunning JAGS (4 chains, 20000 iterations, 10000 burn-in)...\n")
```

```
Running JAGS (4 chains, 20000 iterations, 10000 burn-in)...
```

```
jags_out <- jags(
  data = data_jags,
  inits = inits_fn,
  parameters.to.save = params,
  model.file = "stroke_model_final.jags",
  n.chains = 4,
  n.iter = 20000,
  n.burnin = 10000,
  n.thin = 5,
  DIC = TRUE,
  progress.bar = "text"
)
```

```
module glm loaded

Compiling model graph
   Resolving undeclared variables
   Allocating nodes
Graph information:
   Observed stochastic nodes: 2598
   Unobserved stochastic nodes: 813
   Total graph size: 17638

Initializing model
```

```
print(jags_out)
```

```
Inference for Bugs model at "stroke_model_final.jags", fit using jags,
 4 chains, each with 20000 iterations (first 10000 discarded), n.thin = 5
 n.sims = 8000 iterations saved. Running time = 133.272 secs
             mu.vect sd.vect     2.5%      25%      50%      75%    97.5%  Rhat
alpha[1]       0.383   5.613  -11.969   -2.912    0.630    4.474   10.106 1.299
alpha[2]       0.467   5.612  -11.799   -2.807    0.721    4.594   10.251 1.298
alpha[3]      -0.057   5.612  -12.379   -3.348    0.148    4.073    9.692 1.298
alpha[4]      -0.309   5.609  -12.644   -3.603   -0.083    3.845    9.403 1.298
alpha[5]       0.370   5.611  -11.962   -2.918    0.653    4.488   10.061 1.299
alpha[6]      -0.105   5.612  -12.426   -3.389    0.132    4.051    9.601 1.298
alpha[7]       0.316   5.611  -11.993   -2.961    0.583    4.454   10.002 1.298
alpha[8]       0.460   5.612  -11.855   -2.840    0.719    4.586   10.173 1.298
alpha[9]       0.127   5.613  -12.206   -3.156    0.368    4.256    9.756 1.299
beta0          1.499   5.611   -8.229   -2.631    1.253    4.815   13.805 1.297
beta1          0.163   0.136   -0.100    0.071    0.163    0.257    0.425 1.001
beta2         -0.909   0.101   -1.109   -0.976   -0.908   -0.841   -0.717 1.001
beta3         -0.374   0.130   -0.632   -0.462   -0.374   -0.286   -0.117 1.002
beta4         -0.609   0.202   -1.011   -0.743   -0.605   -0.474   -0.211 1.002
beta5          0.534   0.169    0.203    0.420    0.535    0.647    0.869 1.001
beta6         -0.700   0.165   -1.018   -0.812   -0.702   -0.591   -0.374 1.001
gamma0        -0.519   0.102   -0.720   -0.588   -0.518   -0.452   -0.320 1.001
gamma1         0.253   0.064    0.127    0.210    0.252    0.297    0.378 1.001
gamma2         0.155   0.070    0.019    0.108    0.155    0.203    0.294 1.001
gamma3         0.337   0.089    0.163    0.277    0.338    0.398    0.505 1.001
mu_alpha       0.183   5.609  -12.125   -3.099    0.417    4.310    9.927 1.299
sigma_age      0.986   0.023    0.942    0.970    0.986    1.001    1.033 1.001
sigma_alpha    0.391   0.156    0.175    0.287    0.362    0.463    0.761 1.001
deviance    4232.500  26.592 4180.033 4214.622 4233.050 4250.518 4283.084 1.001
```

```
            n.eff
alpha[1]       13
alpha[2]       13
alpha[3]       13
alpha[4]       13
alpha[5]       13
alpha[6]       13
alpha[7]       13
alpha[8]       13
alpha[9]       13
beta0          13
beta1        8000
beta2        5300
beta3        2900
beta4        2200
beta5        8000
beta6        8000
gamma0       6300
gamma1       8000
gamma2       8000
gamma3       5000
mu_alpha       13
sigma_age    8000
sigma_alpha  8000
deviance     6400

For each parameter, n.eff is a crude measure of effective sample size,
and Rhat is the potential scale reduction factor (at convergence, Rhat=1).

DIC info (using the rule: pV = var(deviance)/2)
pV = 353.5 and DIC = 4586.0
DIC is an estimate of expected predictive error (lower deviance is better).
```

```r
# ---------- 5. Convergence diagnostics ----------
sims <- as.mcmc(jags_out)
sims_mat <- as.matrix(sims)

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

```
=== CONVERGENCE DIAGNOSTICS ===
```

```r
cat("\n--- Gelman-Rubin diagnostics (Rhat, should be < 1.1) ---\n")
```

```
--- Gelman-Rubin diagnostics (Rhat, should be < 1.1) ---
```

```r
gelman_diag <- gelman.diag(sims, multivariate = FALSE)
print(gelman_diag)
```

```
Potential scale reduction factors:

            Point est. Upper C.I.
alpha[1]           1.3        1.8
alpha[2]           1.3        1.8
alpha[3]           1.3        1.8
alpha[4]           1.3        1.8
alpha[5]           1.3        1.8
alpha[6]           1.3        1.8
alpha[7]           1.3        1.8
alpha[8]           1.3        1.8
alpha[9]           1.3        1.8
beta0              1.3        1.8
beta1              1.0        1.0
beta2              1.0        1.0
beta3              1.0        1.0
beta4              1.0        1.0
beta5              1.0        1.0
beta6              1.0        1.0
deviance           1.0        1.0
gamma0             1.0        1.0
gamma1             1.0        1.0
gamma2             1.0        1.0
gamma3             1.0        1.0
mu_alpha           1.3        1.8
sigma_age          1.0        1.0
sigma_alpha        1.0        1.0
```

```r
psrf_values <- gelman_diag$psrf[,1]
if (any(psrf_values > 1.1, na.rm = TRUE)) {
  warning("Some parameters have Rhat > 1.1. Consider longer chains.")
  cat("\nParameters with poor convergence (Rhat > 1.1):\n")
```

```
  print(psrf_values[psrf_values > 1.1])
}
```

```
Warning: Some parameters have Rhat > 1.1. Consider longer chains.
```

```
Parameters with poor convergence (Rhat > 1.1):
alpha[1] alpha[2] alpha[3] alpha[4] alpha[5] alpha[6] alpha[7] alpha[8]
1.298728 1.298492 1.297690 1.298276 1.298792 1.298182 1.298389 1.298183
alpha[9]    beta0 mu_alpha
1.298931 1.297231 1.298679
```

```
cat("\n--- Effective sample sizes ---\n")
```

```
--- Effective sample sizes ---
```

```
ess <- effectiveSize(sims)
print(head(ess, 20))
```

```
  alpha[1]   alpha[2]   alpha[3]   alpha[4]   alpha[5]   alpha[6]   alpha[7]
  23.19935   22.90111   23.45171   23.57551   23.47507   23.49758   23.42305
  alpha[8]   alpha[9]      beta0      beta1      beta2      beta3      beta4
  22.97938   23.34357   22.84851 6562.32438 5447.96292 7920.30643 7142.44481
     beta5      beta6   deviance     gamma0     gamma1     gamma2
8182.67619 7688.25798 6784.78236 1454.86421 5897.66982 3668.27181
```

```
if (any(ess < 400, na.rm = TRUE)) {
  warning("Some parameters have ESS < 400. Consider more iterations.")
}
```

```
Warning: Some parameters have ESS < 400. Consider more iterations.
```

```
# ---------- 6. Posterior summaries ----------
cat("\n=== POSTERIOR SUMMARIES ===\n")
```

```
=== POSTERIOR SUMMARIES ===
```

```r
cat("\n--- OUTCOME MODEL (Fixed Effects) ---\n")
```

```
--- OUTCOME MODEL (Fixed Effects) ---
```

```r
outcome_params <- c("beta0", "beta1", "beta2", "beta3", "beta4", "beta5", "beta6")
outcome_labels <- c("Intercept", "Program", "Age", "Female", "EMS", "TPA", "Thrombectomy")
print(summary(sims[, outcome_params]))
```

```
Iterations = 10005:20000
Thinning interval = 5
Number of chains = 4
Sample size per chain = 2000

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

         Mean     SD Naive SE Time-series SE
beta0  1.4985 5.6109 0.062731       1.214767
beta1  0.1629 0.1355 0.001515       0.001701
beta2 -0.9094 0.1014 0.001133       0.001376
beta3 -0.3740 0.1301 0.001455       0.001463
beta4 -0.6087 0.2021 0.002259       0.002396
beta5  0.5344 0.1689 0.001888       0.001868
beta6 -0.7004 0.1647 0.001841       0.001881

2. Quantiles for each variable:

          2.5%      25%     50%     75%   97.5%
beta0 -8.22919 -2.63150  1.2532  4.8147 13.8046
beta1 -0.09993  0.07116  0.1633  0.2569  0.4254
beta2 -1.10890 -0.97572 -0.9081 -0.8411 -0.7171
beta3 -0.63227 -0.46211 -0.3740 -0.2862 -0.1171
beta4 -1.01086 -0.74310 -0.6049 -0.4743 -0.2107
beta5  0.20268  0.42017  0.5354  0.6469  0.8694
beta6 -1.01802 -0.81227 -0.7024 -0.5910 -0.3743
```

```r
cat("\n--- AGE MODEL (Fixed Effects) ---\n")
```

```
--- AGE MODEL (Fixed Effects) ---
```

```
age_model_params <- c("gamma0", "gamma1", "gamma2", "gamma3")
age_labels <- c("Intercept", "Female", "Program", "EMS")
print(summary(sims[, age_model_params]))
```

```
Iterations = 10005:20000
Thinning interval = 5
Number of chains = 4
Sample size per chain = 2000

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

          Mean      SD  Naive SE Time-series SE
gamma0 -0.5195 0.10188 0.0011390      0.0026763
gamma1  0.2527 0.06450 0.0007211      0.0008463
gamma2  0.1554 0.07010 0.0007838      0.0011618
gamma3  0.3375 0.08872 0.0009919      0.0021152

2. Quantiles for each variable:

           2.5%     25%     50%     75%   97.5%
gamma0 -0.72006 -0.5884 -0.5179 -0.4517 -0.3196
gamma1  0.12716  0.2098  0.2522  0.2968  0.3775
gamma2  0.01876  0.1078  0.1554  0.2030  0.2937
gamma3  0.16343  0.2771  0.3376  0.3976  0.5050
```

```
cat("\n--- RANDOM EFFECTS (Hyperparameters) ---\n")
```

```
--- RANDOM EFFECTS (Hyperparameters) ---
```

```
print(summary(sims[, c("mu_alpha", "sigma_alpha")]))
```

```
Iterations = 10005:20000
Thinning interval = 5
```

```
Number of chains = 4
Sample size per chain = 2000

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

              Mean     SD Naive SE Time-series SE
mu_alpha    0.1833 5.6093 0.062713       1.228798
sigma_alpha 0.3910 0.1555 0.001739       0.002174

2. Quantiles for each variable:

                2.5%    25%    50%    75%  97.5%
mu_alpha    -12.1251 -3.099 0.4167 4.3096 9.9268
sigma_alpha   0.1751  0.287 0.3620 0.4629 0.7609
```

```
cat("\n--- AGE MODEL VARIANCE ---\n")
```

```

--- AGE MODEL VARIANCE ---
```

```
print(summary(sims[, "sigma_age"]))
```

```

Iterations = 10005:20000
Thinning interval = 5
Number of chains = 4
Sample size per chain = 2000

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

          Mean             SD       Naive SE Time-series SE
     0.9857513      0.0233432      0.0002610      0.0002895

2. Quantiles for each variable:

  2.5%    25%    50%    75%  97.5%
0.9417 0.9696 0.9857 1.0012 1.0328
```

```r
cat("\n=== 95% CREDIBLE INTERVALS ===\n")
```

```
=== 95% CREDIBLE INTERVALS ===
```

```r
cat("\n--- Outcome Model ---\n")
```

```
--- Outcome Model ---
```

```r
for (i in 1:length(outcome_params)) {
  param <- outcome_params[i]
  ci <- quantile(sims_mat[, param], probs = c(0.025, 0.975))
  pm <- mean(sims_mat[, param])
  sig <- if(ci[1] > 0 || ci[2] < 0) " *" else ""
  cat(sprintf("%15s: %7.3f  [%7.3f, %7.3f]%s\n", outcome_labels[i], pm, ci[1], ci[2], sig))
}
```

```
      Intercept:   1.499  [ -8.229,  13.805]
        Program:   0.163  [ -0.100,   0.425]
            Age:  -0.909  [ -1.109,  -0.717] *
         Female:  -0.374  [ -0.632,  -0.117] *
            EMS:  -0.609  [ -1.011,  -0.211] *
            TPA:   0.534  [  0.203,   0.869] *
   Thrombectomy:  -0.700  [ -1.018,  -0.374] *
```

```r
cat("\n--- Age Model ---\n")
```

```
--- Age Model ---
```

```r
for (i in 1:length(age_model_params)) {
  param <- age_model_params[i]
  ci <- quantile(sims_mat[, param], probs = c(0.025, 0.975))
  pm <- mean(sims_mat[, param])
  sig <- if(ci[1] > 0 || ci[2] < 0) " *" else ""
  cat(sprintf("%15s: %7.3f  [%7.3f, %7.3f]%s\n", age_labels[i], pm, ci[1], ci[2], sig))
}
```

```
      Intercept:  -0.519  [ -0.720,  -0.320] *
         Female:   0.253  [  0.127,   0.378] *
        Program:   0.155  [  0.019,   0.294] *
            EMS:   0.337  [  0.163,   0.505] *
```

```
# ---------- 7. Interpret Age Model ----------
cat("\n=== AGE MODEL INTERPRETATION ===\n")
```

```

=== AGE MODEL INTERPRETATION ===
```

```
gamma1_pm <- mean(sims_mat[, "gamma1"])
gamma2_pm <- mean(sims_mat[, "gamma2"])
gamma3_pm <- mean(sims_mat[, "gamma3"])

cat("Baseline age ( 0): ", round(mean(sims_mat[, "gamma0"]), 3), " SD units\n")
```

```
Baseline age ( 0):  -0.519  SD units
```

```
cat("  On original scale: ", round(mean(sims_mat[, "gamma0"]) * age_sd + age_mean, 1), " yea
```

```
  On original scale:  60.6  years
```

```
cat("Female effect ( 1): ", round(gamma1_pm, 3), " SD units\n")
```

```
Female effect ( 1):  0.253  SD units
```

```
cat("  Interpretation: Females are on average ",
    abs(round(gamma1_pm * age_sd, 1)), " years ",
    ifelse(gamma1_pm > 0, "OLDER", "YOUNGER"), " than males\n")
```

```
  Interpretation: Females are on average  3.8  years  OLDER  than males
```

```
cat("  95% CI: [", round(quantile(sims_mat[, "gamma1"], 0.025), 3), ", ",
    round(quantile(sims_mat[, "gamma1"], 0.975), 3), "]\n\n")
```

```
  95% CI: [ 0.127 ,  0.378 ]
```

```
cat("Program effect ( 2): ", round(gamma2_pm, 3), " SD units\n")
```

```
Program effect ( 2):  0.155  SD units
```

```
cat("  Interpretation: Post-implementation patients are on average ",
    abs(round(gamma2_pm * age_sd, 1)), " years ",
    ifelse(gamma2_pm > 0, "OLDER", "YOUNGER"), " than baseline\n")
```

```
  Interpretation: Post-implementation patients are on average  2.3  years  OLDER  than basel
```

```
cat("  95% CI: [", round(quantile(sims_mat[, "gamma2"], 0.025), 3), ", ",
    round(quantile(sims_mat[, "gamma2"], 0.975), 3), "]\n\n")
```

```
  95% CI: [ 0.019 ,  0.294 ]
```

```
cat("EMS effect ( 3): ", round(gamma3_pm, 3), " SD units\n")
```

```
EMS effect ( 3):  0.337  SD units
```

```
cat("  Interpretation: EMS-transported patients are on average ",
    abs(round(gamma3_pm * age_sd, 1)), " years ",
    ifelse(gamma3_pm > 0, "OLDER", "YOUNGER"), " than car arrivals\n")
```

```
  Interpretation: EMS-transported patients are on average  5  years  OLDER  than car arrivals
```

```
cat("  95% CI: [", round(quantile(sims_mat[, "gamma3"], 0.025), 3), ", ",
    round(quantile(sims_mat[, "gamma3"], 0.975), 3), "]\n\n")
```

```
  95% CI: [ 0.163 ,  0.505 ]
```

```
cat("Age variability ( _A): ", round(mean(sims_mat[, "sigma_age"]), 3),
    " SD units (", round(mean(sims_mat[, "sigma_age"]) * age_sd, 1), " years)\n")
```

```
Age variability ( _A):  0.986  SD units ( 14.7  years)
```

```r
# ---------- 8. Model Performance ----------
cat("\n=== MODEL PERFORMANCE ===\n")
```

```
=== MODEL PERFORMANCE ===
```

```r
# Extract posterior mean ages (including imputed values)
age_cols <- grep("^age\\[", colnames(sims_mat), value = TRUE)
# Note: age posterior draws are not saved by default
# Use posterior mean from age model for missing values
alpha_names <- grep("^alpha\\[", colnames(sims_mat), value = TRUE)
alpha_pm <- colMeans(sims_mat[, alpha_names, drop = FALSE])

beta0_pm <- mean(sims_mat[, "beta0"])
beta1_pm <- mean(sims_mat[, "beta1"])
beta2_pm <- mean(sims_mat[, "beta2"])
beta3_pm <- mean(sims_mat[, "beta3"])
beta4_pm <- mean(sims_mat[, "beta4"])
beta5_pm <- mean(sims_mat[, "beta5"])
beta6_pm <- mean(sims_mat[, "beta6"])

gamma0_pm <- mean(sims_mat[, "gamma0"])
gamma1_pm <- mean(sims_mat[, "gamma1"])
gamma2_pm <- mean(sims_mat[, "gamma2"])
gamma3_pm <- mean(sims_mat[, "gamma3"])

# Impute missing ages using posterior mean
age_pm <- df_model$age_std
missing_age <- is.na(age_pm)
if (any(missing_age)) {
  age_pm[missing_age] <- gamma0_pm +
                         gamma1_pm * df_model$female[missing_age] +
                         gamma2_pm * df_model$program[missing_age] +
                         gamma3_pm * df_model$ems[missing_age]
  cat("Imputed ", sum(missing_age), " missing age values using posterior means\n")
}
```

```
Imputed  790  missing age values using posterior means
```

```r
# Compute predictions
linpred_pm <- beta0_pm + alpha_pm[data_jags$site] +
              beta1_pm * data_jags$program +
              beta2_pm * age_pm +
              beta3_pm * data_jags$female +
              beta4_pm * data_jags$ems +
              beta5_pm * data_jags$tpa +
              beta6_pm * data_jags$thr

p_pm <- plogis(linpred_pm)
df_model$pred_prob_postmean <- p_pm

cat("\nPredicted probability summary:\n")
```

```
Predicted probability summary:
```

```r
print(summary(p_pm))
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.1386  0.6232  0.7663  0.7269  0.8640  0.9895
```

```r
# AUC
roc_obj <- roc(df_model$y, df_model$pred_prob_postmean, quiet = TRUE)
auc_val <- auc(roc_obj)
cat("\nAUC (posterior mean predictions): ", round(auc_val, 4), "\n")
```

```
AUC (posterior mean predictions):  0.7296
```

```r
# ---------- 9. Save results ----------
saveRDS(jags_out, file = "stroke_jags_final.rds")
saveRDS(df_model, file = "stroke_with_preds_final.rds")
saveRDS(list(
  posterior_means = list(
    outcome_model = setNames(c(beta0_pm, beta1_pm, beta2_pm, beta3_pm, beta4_pm, beta5_pm, be
                               outcome_labels),
    age_model = setNames(c(gamma0_pm, gamma1_pm, gamma2_pm, gamma3_pm), age_labels),
    random_effects = list(
      mu_alpha = mean(sims_mat[, "mu_alpha"]),
```

```
      sigma_alpha = mean(sims_mat[, "sigma_alpha"])
    ),
    age_variance = mean(sims_mat[, "sigma_age"])
  ),
  credible_intervals_outcome = sapply(outcome_params, function(p) {
    quantile(sims_mat[, p], probs = c(0.025, 0.975))
  }),
  credible_intervals_age = sapply(age_model_params, function(p) {
    quantile(sims_mat[, p], probs = c(0.025, 0.975))
  }),
  auc = auc_val,
  convergence = gelman_diag,
  ess = ess,
  age_standardization = list(mean = age_mean, sd = age_sd)
), file = "model_results_final.rds")

message("\n=== ANALYSIS COMPLETE ===")
```

```
=== ANALYSIS COMPLETE ===
```

```
message("Files saved:")
```

```
Files saved:
```

```
message("  stroke_jags_final.rds")
```

```
  stroke_jags_final.rds
```

```
message("  stroke_with_preds_final.rds")
```

```
  stroke_with_preds_final.rds
```

```
message("  model_results_final.rds")
```

```
  model_results_final.rds
```

```r
message("\nNote: * indicates 95% CI excludes 0 (significant effect)")
```

```
Note: * indicates 95% CI excludes 0 (significant effect)
```

```r
# ============================================================================
# Streamlined JAGS Model Diagnostics - New Age Sub-Model
# ============================================================================

library(R2jags)
library(coda)
library(pROC)
library(dplyr)

# Load results
jags_out <- readRDS("stroke_jags_final.rds")
df_model <- readRDS("stroke_with_preds_final.rds")
results <- readRDS("model_results_final.rds")

sims <- as.mcmc(jags_out)
sims_mat <- as.matrix(sims)

age_mean <- results$age_standardization$mean
age_sd <- results$age_standardization$sd

# ============================================================================
# CONVERGENCE DIAGNOSTICS
# ============================================================================

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

```
=== CONVERGENCE DIAGNOSTICS ===
```

```r
outcome_params <- c("beta0", "beta1", "beta2", "beta3", "beta4", "beta5", "beta6")
age_params <- c("gamma0", "gamma1", "gamma2", "gamma3")
hyper_params <- c("mu_alpha", "sigma_alpha", "sigma_age")
all_params <- c(outcome_params, age_params, hyper_params)

# Gelman-Rubin
```

```r
gelman_diag <- gelman.diag(sims[, all_params], multivariate = FALSE)
print(gelman_diag)
```

```
Potential scale reduction factors:

            Point est. Upper C.I.
beta0              1.3        1.8
beta1              1.0        1.0
beta2              1.0        1.0
beta3              1.0        1.0
beta4              1.0        1.0
beta5              1.0        1.0
beta6              1.0        1.0
gamma0             1.0        1.0
gamma1             1.0        1.0
gamma2             1.0        1.0
gamma3             1.0        1.0
mu_alpha           1.3        1.8
sigma_alpha        1.0        1.0
sigma_age          1.0        1.0
```

```r
# Effective Sample Sizes
ess <- effectiveSize(sims[, all_params])
cat("\nEffective Sample Sizes:\n")
```

```
Effective Sample Sizes:
```

```r
print(ess)
```

```
      beta0       beta1       beta2       beta3       beta4       beta5
   22.84851  6562.32438  5447.96292  7920.30643  7142.44481  8182.67619
      beta6      gamma0      gamma1      gamma2      gamma3    mu_alpha
 7688.25798  1454.86421  5897.66982  3668.27181  1764.12871    22.73689
sigma_alpha   sigma_age
 5198.06574  6545.56712
```

```r
# Flag issues
bad_rhat <- gelman_diag$psrf[,1] > 1.1
bad_ess <- ess < 400
if (any(bad_rhat)) cat("\nWARNING: Rhat > 1.1 for:", names(which(bad_rhat)), "\n")
```

```
WARNING: Rhat > 1.1 for: beta0 mu_alpha
```

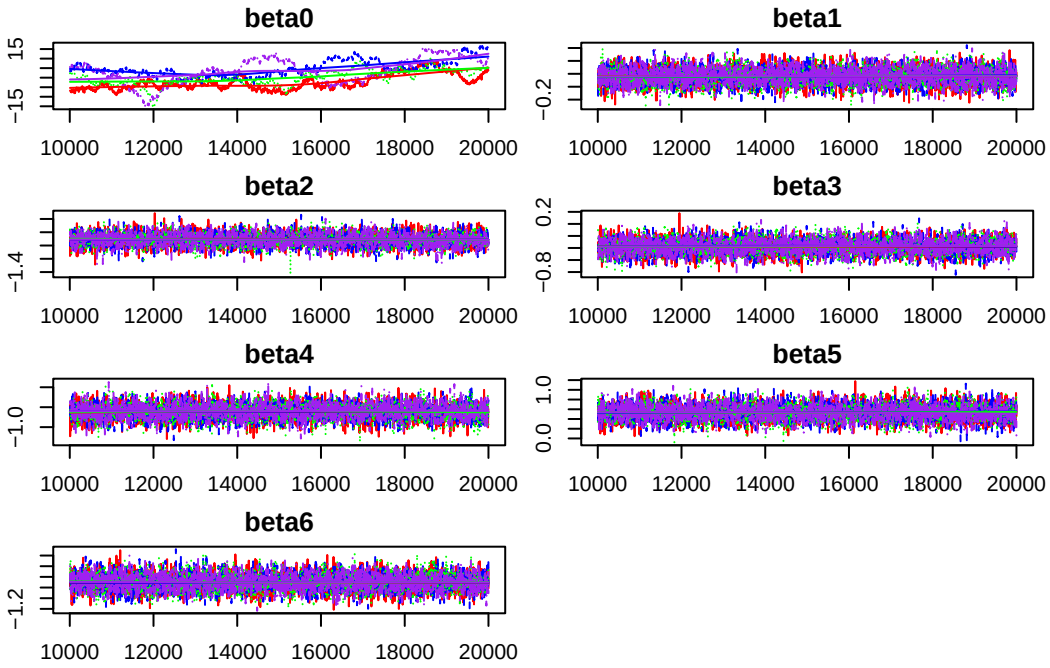
```
if (any(bad_ess)) cat("WARNING: ESS < 400 for:", names(which(bad_ess)), "\n")
```

```
WARNING: ESS < 400 for: beta0 mu_alpha
```

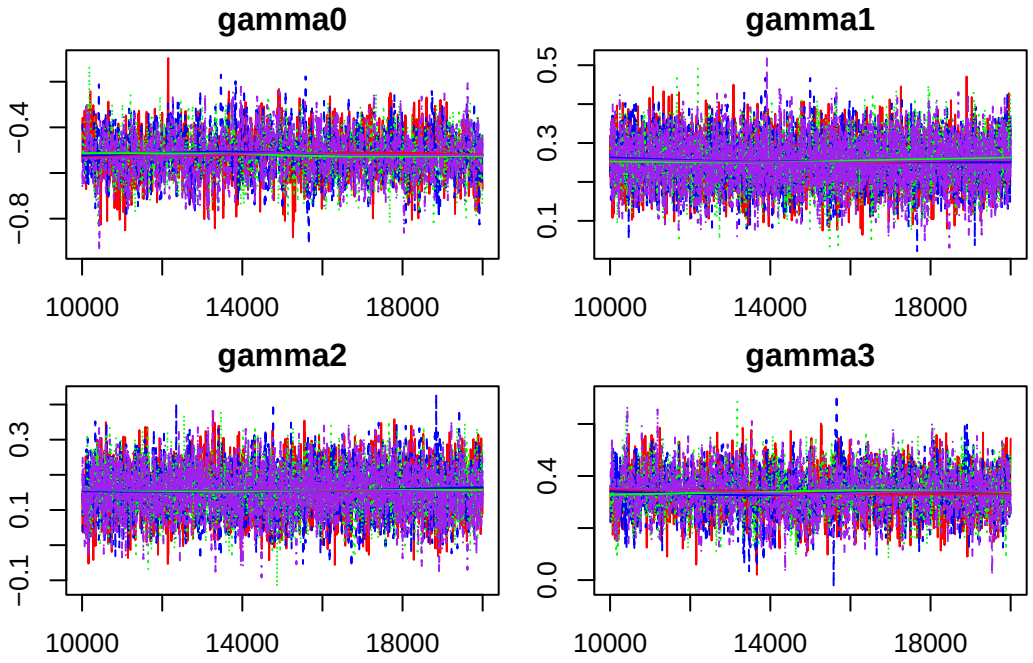
```
# ============================================================================
# TRACE PLOTS
# ============================================================================

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

```
=== TRACE PLOTS ===
```

```
# Outcome model
par(mfrow = c(4, 2), mar = c(2, 2, 2, 1))
for (p in outcome_params) {
  traceplot(sims[, p], main = p, col = c("red", "blue", "green", "purple"))
}

# Age model
par(mfrow = c(2, 2), mar = c(2, 2, 2, 1))
```



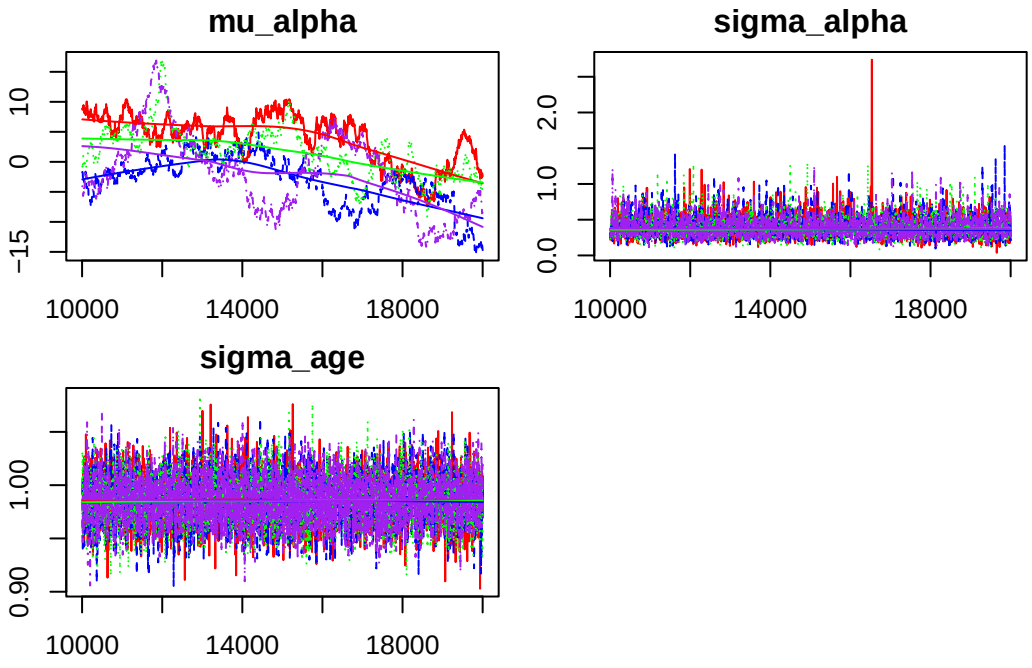

```
for (p in age_params) {
  traceplot(sims[, p], main = p, col = c("red", "blue", "green", "purple"))
}
```



```r
# Hyperparameters
par(mfrow = c(2, 2), mar = c(2, 2, 2, 1))
for (p in hyper_params) {
  traceplot(sims[, p], main = p, col = c("red", "blue", "green", "purple"))
}

# ============================================================================
# POSTERIOR DENSITIES
# ============================================================================

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

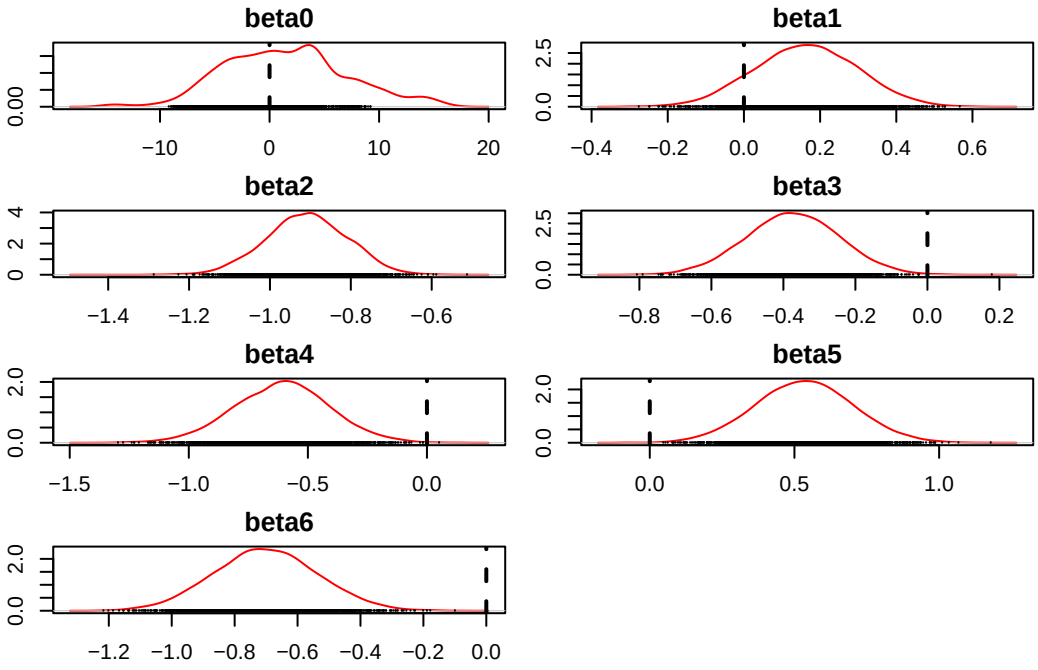
```
=== POSTERIOR DENSITIES ===
```

```r
# Outcome model
par(mfrow = c(4, 2), mar = c(2, 2, 2, 1))
```

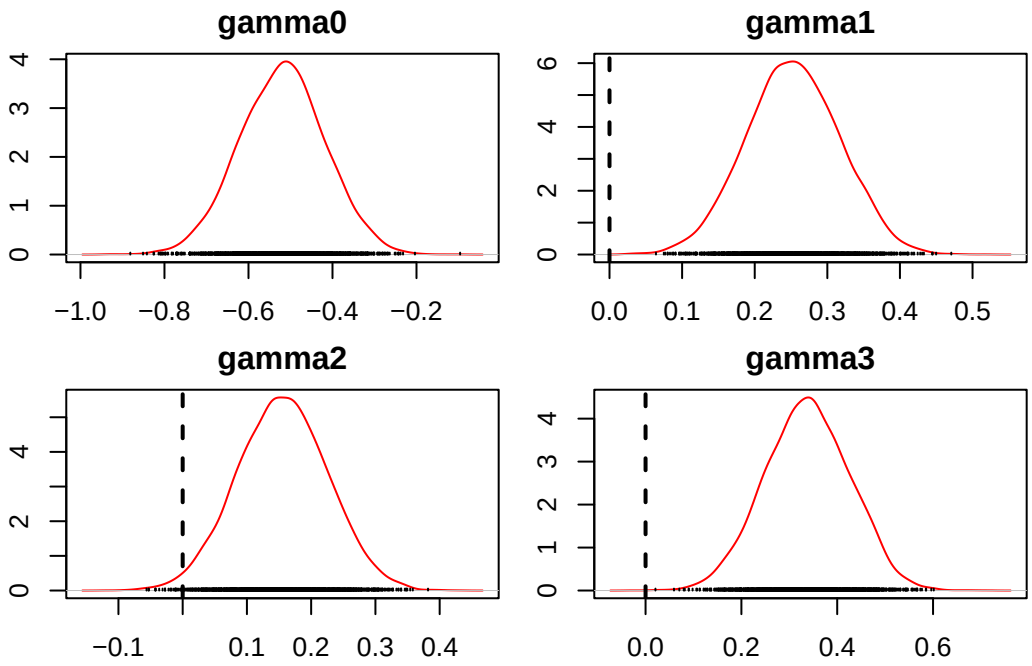



```r
for (p in outcome_params) {
  densplot(sims[, p], main = p, col = c("red", "blue", "green", "purple"))
  abline(v = 0, lty = 2, col = "black", lwd = 2)
```

```
}

# Age model
par(mfrow = c(2, 2), mar = c(2, 2, 2, 1))
```




```
for (p in age_params) {
  densplot(sims[, p], main = p, col = c("red", "blue", "green", "purple"))
  abline(v = 0, lty = 2, col = "black", lwd = 2)
}
```




```
# ============================================================================
# RESIDUAL DIAGNOSTICS
# ============================================================================

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

```
=== RESIDUAL DIAGNOSTICS ===
```

```
# Extract posterior means
alpha_names <- grep("^alpha\\[", colnames(sims_mat), value = TRUE)
alpha_pm <- colMeans(sims_mat[, alpha_names, drop = FALSE])

beta_pm <- colMeans(sims_mat[, outcome_params])
gamma_pm <- colMeans(sims_mat[, age_params])

# Impute missing ages
age_for_pred <- df_model$age_std
missing_age <- is.na(age_for_pred)
if (any(missing_age)) {
  age_for_pred[missing_age] <- gamma_pm[1] +
                               gamma_pm[2] * df_model$female[missing_age] +
```

```
                                  gamma_pm[3] * df_model$program[missing_age] +
                                  gamma_pm[4] * df_model$ems[missing_age]
  cat("Imputed", sum(missing_age), "missing ages\n")
}
```

```
Imputed 790 missing ages
```

```
# Compute fitted values
linpred <- beta_pm[1] + alpha_pm[df_model$site] +
           beta_pm[2] * df_model$program +
           beta_pm[3] * age_for_pred +
           beta_pm[4] * df_model$female +
           beta_pm[5] * df_model$ems +
           beta_pm[6] * df_model$tpa +
           beta_pm[7] * df_model$thr

fitted_prob <- plogis(linpred)

# Residuals
deviance_resid <- sign(df_model$y - fitted_prob) *
                  sqrt(-2 * (df_model$y * log(fitted_prob + 1e-10) +
                             (1 - df_model$y) * log(1 - fitted_prob + 1e-10)))

pearson_resid <- (df_model$y - fitted_prob) / sqrt(fitted_prob * (1 - fitted_prob) + 1e-10)

cat("\nDeviance Residuals Summary:\n")
```

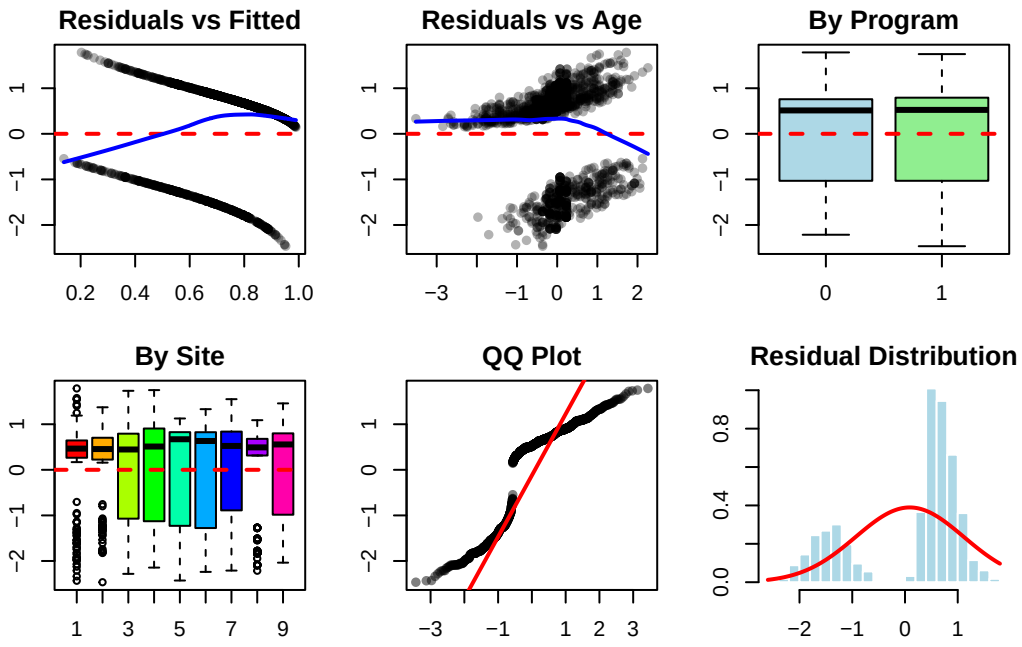
```
Deviance Residuals Summary:
```

```
print(summary(deviance_resid))
```

```
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
-2.46601 -1.03139  0.52334  0.09045  0.78625  1.78567
```

```
# Residual plots
par(mfrow = c(2, 3), mar = c(3, 3, 2, 1))

plot(fitted_prob, deviance_resid, pch = 16, col = rgb(0,0,0,0.3),
     xlab = "Fitted Probability", ylab = "Deviance Residuals",
```

```
     main = "Residuals vs Fitted")
abline(h = 0, col = "red", lwd = 2, lty = 2)
lines(lowess(fitted_prob, deviance_resid), col = "blue", lwd = 2)

plot(age_for_pred, deviance_resid, pch = 16, col = rgb(0,0,0,0.3),
     xlab = "Age (std)", ylab = "Deviance Residuals",
     main = "Residuals vs Age")
abline(h = 0, col = "red", lwd = 2, lty = 2)
lines(lowess(age_for_pred, deviance_resid), col = "blue", lwd = 2)

boxplot(deviance_resid ~ df_model$program, col = c("lightblue", "lightgreen"),
        xlab = "Program", ylab = "Deviance Residuals", main = "By Program")
abline(h = 0, col = "red", lwd = 2, lty = 2)

boxplot(deviance_resid ~ df_model$site, col = rainbow(9),
        xlab = "Site", ylab = "Deviance Residuals", main = "By Site")
abline(h = 0, col = "red", lwd = 2, lty = 2)

qqnorm(deviance_resid, pch = 16, col = rgb(0,0,0,0.5), main = "QQ Plot")
qqline(deviance_resid, col = "red", lwd = 2)

hist(deviance_resid, breaks = 30, col = "lightblue", border = "white",
     main = "Residual Distribution", xlab = "Deviance Residuals", freq = FALSE)
curve(dnorm(x, mean(deviance_resid), sd(deviance_resid)), add = TRUE, col = "red", lwd = 2)
```




```
# ============================================================================
# MODEL PERFORMANCE
# ============================================================================

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

```
=== MODEL PERFORMANCE ===
```

```
# ROC/AUC
roc_obj <- roc(df_model$y, fitted_prob, quiet = TRUE)
auc_val <- auc(roc_obj)

# Confusion matrix
pred_class <- ifelse(fitted_prob > 0.5, 1, 0)
conf_mat <- table(Predicted = pred_class, Actual = df_model$y)

accuracy <- sum(diag(conf_mat)) / sum(conf_mat)
sensitivity <- conf_mat[2, 2] / sum(conf_mat[, 2])
specificity <- conf_mat[1, 1] / sum(conf_mat[, 1])

cat("\nAUC:", round(auc_val, 4))
```

```
AUC: 0.7296
```

```
cat("\nAccuracy:", round(accuracy, 4))
```

```
Accuracy: 0.7361
```

```
cat("\nSensitivity:", round(sensitivity, 4))
```

```
Sensitivity: 0.938
```

```
cat("\nSpecificity:", round(specificity, 4))
```

```
Specificity: 0.2314
```

```
cat("\n\nConfusion Matrix:\n")
```

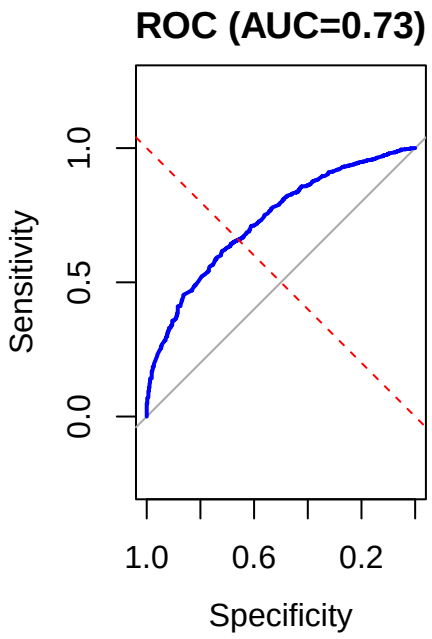
```
Confusion Matrix:
```

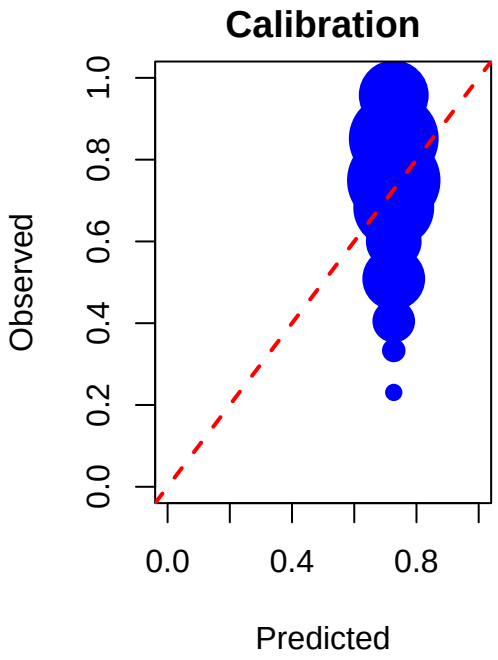
```
print(conf_mat)
```

```
         Actual
Predicted    0    1
        0  112   75
        1  372 1135
```

```
# Plots
par(mfrow = c(1, 2), mar = c(4, 4, 2, 1))

plot(roc_obj, col = "blue", lwd = 2, main = paste0("ROC (AUC=", round(auc_val, 3), ")"))
abline(a = 0, b = 1, lty = 2, col = "red")

# Calibration
```

```r
df_model$prob_bin <- cut(fitted_prob, breaks = 10, include.lowest = TRUE)
cal_data <- df_model %>%
  group_by(prob_bin) %>%
  summarise(observed = mean(y), predicted = mean(fitted_prob), n = n())

plot(cal_data$predicted, cal_data$observed, xlim = c(0,1), ylim = c(0,1),
     pch = 16, cex = sqrt(cal_data$n)/3, col = "blue",
     xlab = "Predicted", ylab = "Observed", main = "Calibration")
abline(a = 0, b = 1, col = "red", lwd = 2, lty = 2)
```






```r
# ============================================================================
# POSTERIOR PREDICTIVE CHECK
# ============================================================================

cat("\n=== POSTERIOR PREDICTIVE CHECK ===\n")
```

```
=== POSTERIOR PREDICTIVE CHECK ===
```

```r
n_sim <- 100
n_obs <- nrow(df_model)
```

```r
y_rep <- matrix(NA, n_sim, n_obs)

for (i in 1:n_sim) {
  idx <- sample(nrow(sims_mat), 1)
  beta_s <- sims_mat[idx, outcome_params]
  alpha_s <- sims_mat[idx, alpha_names]

  linpred_s <- beta_s[1] + alpha_s[df_model$site] +
               beta_s[2] * df_model$program +
               beta_s[3] * age_for_pred +
               beta_s[4] * df_model$female +
               beta_s[5] * df_model$ems +
               beta_s[6] * df_model$tpa +
               beta_s[7] * df_model$thr

  y_rep[i, ] <- rbinom(n_obs, 1, plogis(linpred_s))
}

mean_obs <- mean(df_model$y)
mean_rep <- rowMeans(y_rep)
ppp_mean <- mean(mean_rep >= mean_obs)

cat("Observed mean:", round(mean_obs, 3))
```

```
Observed mean: 0.714
```

```r
cat("\nReplicated mean:", round(mean(mean_rep), 3))
```

```

Replicated mean: 0.727
```

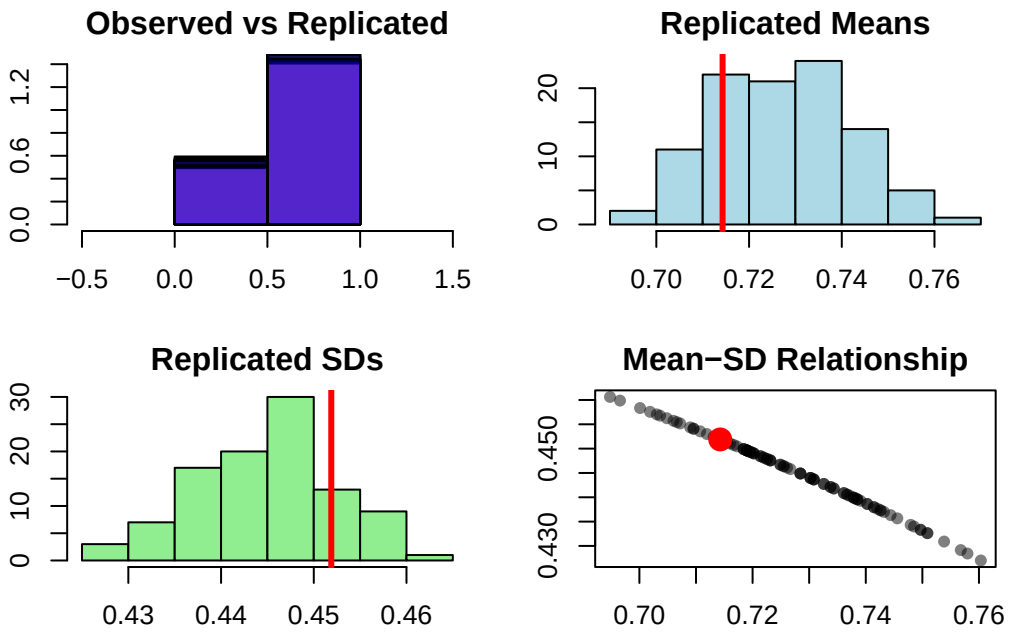
```r
cat("\nPosterior predictive p-value:", round(ppp_mean, 3))
```

```

Posterior predictive p-value: 0.84
```

```r
cat(" (should be ~0.5)\n")
```

```
 (should be ~0.5)
```

```r
par(mfrow = c(2, 2), mar = c(3, 3, 2, 1))

hist(df_model$y, breaks = 2, col = rgb(1,0,0,0.5), main = "Observed vs Replicated",
     xlab = "Outcome", xlim = c(-0.5,1.5), freq = FALSE)
for (i in 1:20) hist(y_rep[i,], breaks = 2, add = TRUE, col = rgb(0,0,1,0.05), freq = FALSE)

hist(mean_rep, col = "lightblue", main = "Replicated Means", xlab = "Mean")
abline(v = mean_obs, col = "red", lwd = 3)

sd_rep <- apply(y_rep, 1, sd)
hist(sd_rep, col = "lightgreen", main = "Replicated SDs", xlab = "SD")
abline(v = sd(df_model$y), col = "red", lwd = 3)

plot(mean_rep, sd_rep, pch = 16, col = rgb(0,0,0,0.5),
     xlab = "Mean", ylab = "SD", main = "Mean-SD Relationship")
points(mean_obs, sd(df_model$y), pch = 16, col = "red", cex = 2)
```




```r
# ============================================================================
# AGE MODEL DIAGNOSTICS
# ============================================================================

cat("\n=== AGE MODEL DIAGNOSTICS ===\n")
```

```
=== AGE MODEL DIAGNOSTICS ===
```

```
cat("\nAge Effects (on standardized scale):\n")
```

```
Age Effects (on standardized scale):
```

```
for (i in 1:length(age_params)) {
  ci <- quantile(sims_mat[, age_params[i]], probs = c(0.025, 0.975))
  pm <- mean(sims_mat[, age_params[i]])
  sig <- if(ci[1] > 0 || ci[2] < 0) " *" else ""
  cat(sprintf("  %s: %.3f [%.3f, %.3f]%s\n",
              c("Intercept","Female","Program","EMS")[i], pm, ci[1], ci[2], sig))
}
```

```
  Intercept: -0.519 [-0.720, -0.320] *
  Female: 0.253 [0.127, 0.378] *
  Program: 0.155 [0.019, 0.294] *
  EMS: 0.337 [0.163, 0.505] *
```

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

```
Interpretation (in years):
```

```
cat("  Female effect:", round(gamma_pm[2] * age_sd, 1), "years\n")
```

```
  Female effect: 3.8 years
```

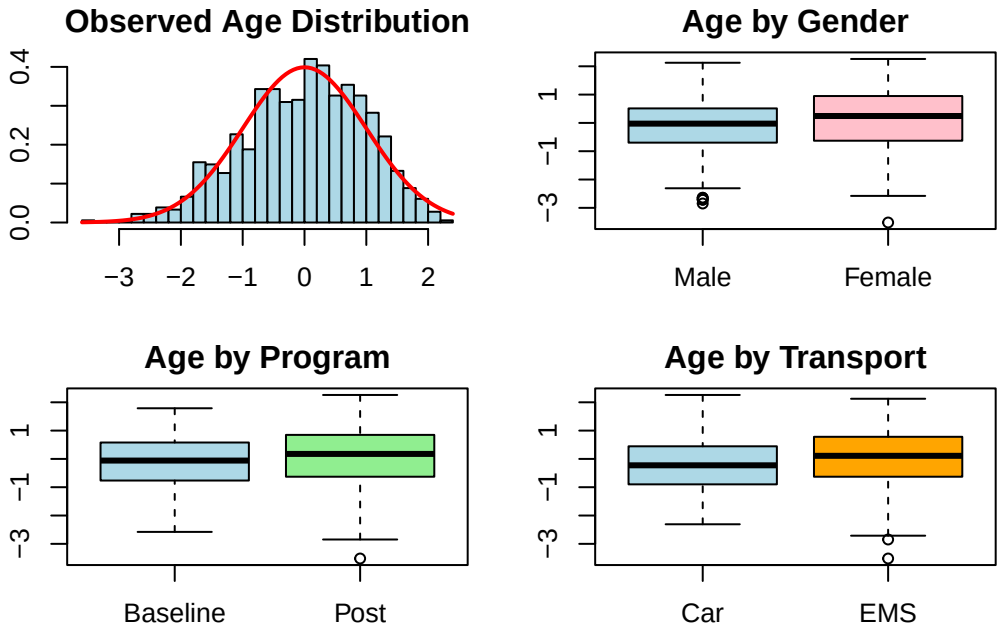
```
cat("  Program effect:", round(gamma_pm[3] * age_sd, 1), "years\n")
```

```
  Program effect: 2.3 years
```

```
cat("  EMS effect:", round(gamma_pm[4] * age_sd, 1), "years\n")
```

```
  EMS effect: 5 years
```

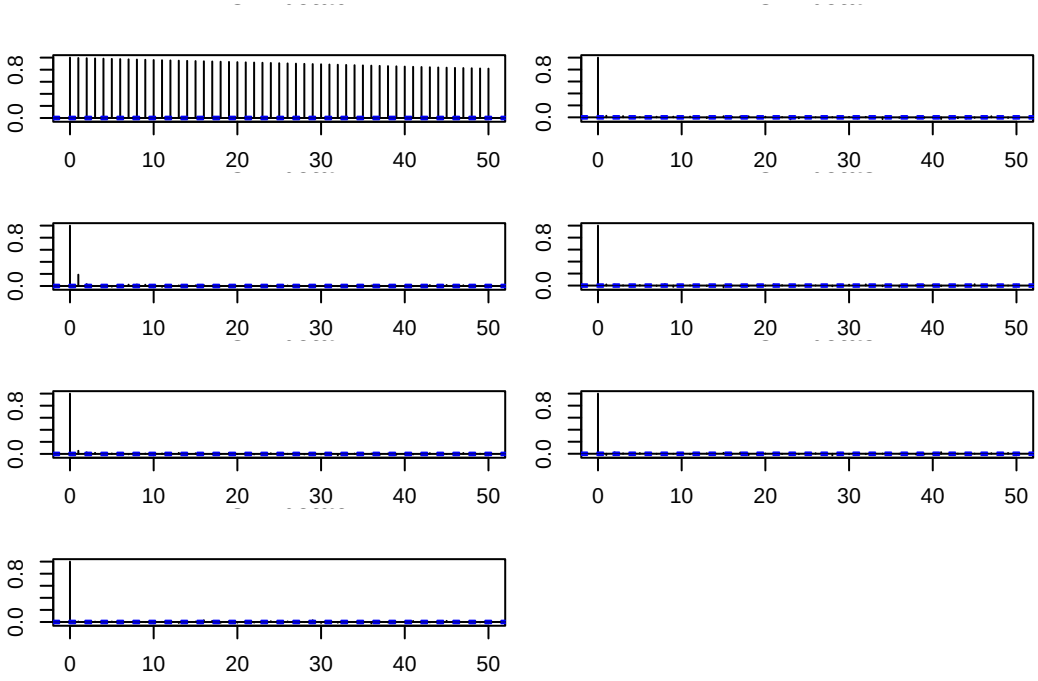
```r
par(mfrow = c(2, 2), mar = c(3, 3, 2, 1))

# Age by covariates
age_obs <- df_model$age_std[!is.na(df_model$age_std)]
hist(age_obs, breaks = 30, col = "lightblue", main = "Observed Age Distribution",
     xlab = "Age (std)", freq = FALSE)
curve(dnorm(x, mean(age_obs), sd(age_obs)), add = TRUE, col = "red", lwd = 2)

boxplot(age_std ~ female, data = df_model, col = c("lightblue", "pink"),
        names = c("Male", "Female"), ylab = "Age (std)", main = "Age by Gender")

boxplot(age_std ~ program, data = df_model, col = c("lightblue", "lightgreen"),
        names = c("Baseline", "Post"), ylab = "Age (std)", main = "Age by Program")

boxplot(age_std ~ ems, data = df_model, col = c("lightblue", "orange"),
        names = c("Car", "EMS"), ylab = "Age (std)", main = "Age by Transport")
```



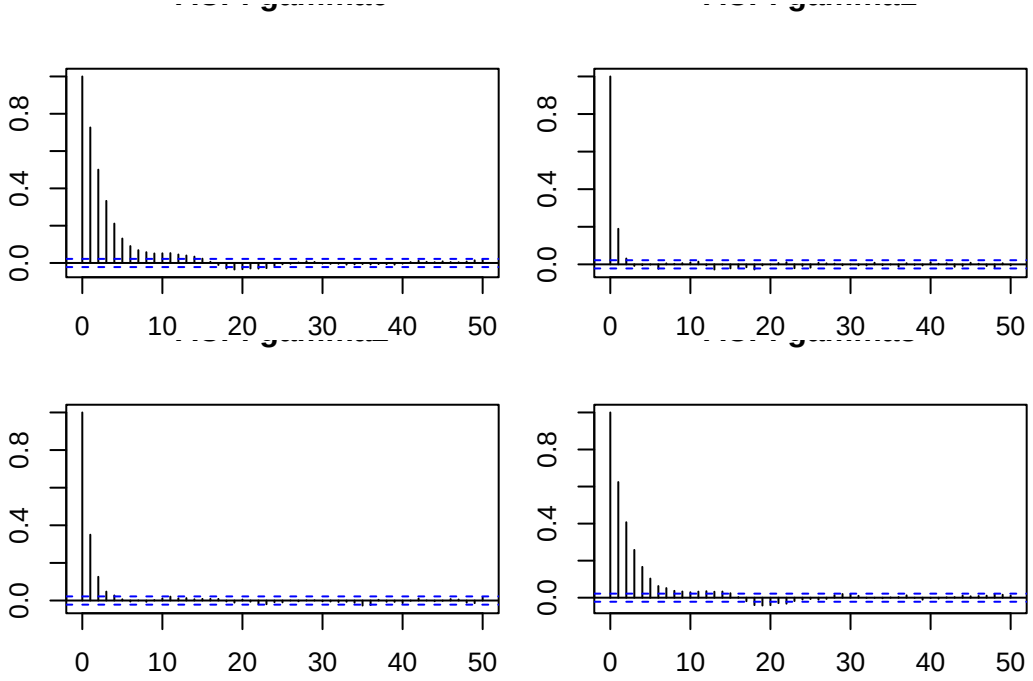

```r
# ============================================================================
# ACF PLOTS
# ============================================================================

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

```
=== ACF PLOTS ===
```

```r
# Outcome model
par(mfrow = c(4, 2), mar = c(2, 2, 2, 1))
for (p in outcome_params) {
  acf(sims_mat[, p], main = paste("ACF:", p), lag.max = 50)
}

# Age model
par(mfrow = c(2, 2), mar = c(2, 2, 2, 1))
```




```r
for (p in age_params) {
  acf(sims_mat[, p], main = paste("ACF:", p), lag.max = 50)
}
```




```
# Hyperparameters
par(mfrow = c(2, 2), mar = c(2, 2, 2, 1))
for (p in hyper_params) {
  acf(sims_mat[, p], main = paste("ACF:", p), lag.max = 50)
}

# ============================================================================
# SUMMARY
# ============================================================================

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

```
=== SUMMARY ===
```

```
cat("AUC:", round(auc_val, 3), "\n")
```

```
AUC: 0.73
```

```
cat("Convergence:", ifelse(all(!bad_rhat), "Good", "Issues detected"), "\n")
```

```
Convergence: Issues detected
```

```
cat("Model fit (PPP):", ifelse(abs(ppp_mean - 0.5) < 0.3, "Good", "Check fit"), "\n")
```

```
Model fit (PPP): Check fit
```

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

```
Diagnostics complete.
```