Bayesian Analysis of Factorial Cluster Randomized Trials

A comprehensive simulation study comparing Bayesian and frequentist approaches for analyzing 2^3 factorial cluster randomized trials, with applications to emergency department interventions.

Overview

This repository contains code for conducting Monte Carlo simulation studies to evaluate statistical methods for factorial cluster randomized trials. The framework is designed for healthcare settings where multiple interventions are tested simultaneously across clustered units (e.g., emergency departments).

Key Features

- **2^3 Factorial Design**: Analyzes trials with three binary interventions (A, B, C) and all interactions
- **Cluster Randomization**: Accounts for clustering within healthcare sites using random effects
- **Bayesian Methods**: Implements hierarchical models with different prior specifications
- **Frequentist Comparison**: Includes hierarchical model selection via likelihood ratio tests
- **Power Analysis**: Determines required sample sizes based on posterior precision criteria
- **Prior Sensitivity**: Evaluates robustness to different prior assumptions

Repository Structure

R Programs

File	Description
`01_data_generation.R`	Core functions for simulating factorial cluster randomized trial data
`02_prior_predictive.R`	Prior predictive checks to validate prior specifications
`03_bayesian_power.R`	Monte Carlo simulation for Bayesian power analysis and sample size determination
`04_comparison_study.R`	Comprehensive comparison of Bayesian and frequentist approaches

Stan Models

File	Description
`model_HEX.stan`	**Hierarchical Exchangeable (HEx) Model**: Main Bayesian model with hierarchical priors for treatment effects
`model_nonX.stan`	**Non-Exchangeable Model**: Alternative specification without hierarchical structure
`model_HEX_ig.stan`	**HEx with Inverse Gamma Priors**: Variant using inverse gamma priors for scale parameters
`prior_predictive_model.stan`	**Prior Predictive Model**: Generates samples from prior distributions for prior checking

Study Design

Treatment Structure

- **Control**: No interventions

- **Single Interventions**: A only, B only, C only
- **Two-way Combinations**: AB, AC, BC
- **Three-way Combination**: ABC

Data Generation Process

1. **Cluster Level**: Random effects and group assignments (4 size categories)
2. **Time Structure**: Multiple quarters per cluster
3. **Individual Level**: Binary outcomes with logistic regression
4. **Effect Structure**: Main effects, two-way interactions, three-way interaction

Model Specifications

The Bayesian models use the following hierarchical structure:

$$logit(P(y_i = 1)) = \alpha_j[i] + \tau_a * A_i + \tau_b * B_i + \tau_c * C_i + \tau_{ab} * A_i * B_i + \tau_{ac} * A_i * C_i + \tau_{bc} * B_i * C_i + \tau_{abc} * A_i * B_i * C_i$$

Where: - $\alpha_j[i]$: Random effect for cluster j - τ_a, τ_b, τ_c: Main effects for treatments A, B, C - τ_{ab}, τ_{ac}, τ_{bc}: Two-way interaction effects
- τ_{abc}: Three-way interaction effect

Getting Started

Prerequisites

```r
# Required R packages
library(simstudy)      # Data simulation
library(data.table)    # Data manipulation
library(cmdstanr)      # Stan interface
library(posterior)     # Posterior analysis
library(glmmTMB)       # Frequentist mixed models
library(ggplot2)       # Visualization
library(ggpubr)        # Plot arrangements
library(parallel)      # Parallel computing
```

Installation

1. Clone this repository:

```
git clone https://github.com/yourusername/factorial-cluster-trials.git
cd factorial-cluster-trials
```

2. Install required R packages:

```r
install.packages(c("simstudy", "data.table", "glmmTMB", "ggplot2", "ggpubr", "parallel"))
```

3. Install cmdstanr and Stan:

```r
install.packages("cmdstanr", repos = c("https://mc-stan.org/r-packages/", getOption("repos")))
cmdstanr::install_cmdstan()
```

Simulation Parameters

Study Design Parameters

- **Number of clusters**: 48-88 emergency departments
- **Intracluster correlation**: 0.015
- **Time periods**: 2 quarters per cluster
- **Cluster sizes**: Variable (60-300 patients per quarter)

Treatment Effect Parameters

- **Baseline log-odds**: -0.40 ($\approx$ 40% control rate)
- **Single intervention OR**: 0.80-0.82
- **Interaction effects**: Small positive or null

Prior Specifications

Three scenarios representing different levels of informativeness: 1. **Informative**: Small prior SDs (σ = 0.3-0.4) 2. **Moderate**: Medium prior SDs (σ = 0.6-0.8)
3. **Weakly informative**: Large prior SDs (σ = 2.5)

Key Analyses

1. Prior Predictive Checking

- Validates prior assumptions by examining implied outcome distributions
- Ensures priors allow for clinically meaningful effect sizes
- Identifies potential prior-data conflicts

2. Power Analysis

- Determines required sample size for adequate posterior precision
- Target: Posterior SD $\leq$ 0.13 for interaction effects
- Monte Carlo estimation with 1000 replications per scenario

3. Method Comparison

- Bayesian hierarchical models vs. frequentist hierarchical model selection
- Type I error rates and power comparison
- Robustness to prior specifications

Computational Requirements

- **Memory**: 8+ GB RAM recommended
- **CPU**: Multi-core processor for parallel processing
- **Time**: Full simulation study requires several hours/days on HPC
- **Dependencies**: Stan installation required

For large-scale simulations, high-performance computing (HPC) is recommended. Contact the authors for HPC setup guidance.

01_data_generation.R

```r
#' ---
#' title: "Generating Data for Simulation Study of Cluster Randomized Trial"
#' author: "Keith Goldfeld"
#' ---

# Load required libraries

library(simstudy)  # For data simulation
library(data.table) # For efficient data manipulation

#' Define Data Generation Structure
#'
#' Creates the data definition objects for the simulation study.
#' This function sets up:
#' - Cluster-level random effects and group assignments
#' - Conditional quarter definitions based on group membership
#' - Period-level observation counts
#' - Binary outcome model with main effects and interactions
#'
#' @return A list containing four data definition objects:
#'   \item{d1}{Cluster-level definitions (random effects, group assignment)}
#'   \item{d2}{Period-level definitions (number of observations per period)}
#'   \item{def_qn}{Conditional definitions for quarter sizes by group}
#'   \item{defY}{Binary outcome model with logistic regression structure}
#'
#' @details The outcome model includes:
#'   - Intercept (t_0)
#'   - Cluster random effect (a)
#'   - Main effects for three factors (t_a, t_b, t_c)
#'   - Two-way interactions (x_ab, x_ac, x_bc)
#'   - Three-way interaction (x_abc)

s_define <- function() {

  # Cluster-level data definitions
  # - Random effect 'a' with variance determined by ICC
  # - Group assignment with specified probabilities (55%, 15%, 15%, 15%)
  d1 <- defData(varname = "a",
                formula = 0,
                variance = "..revar",
                dist = "normal",
                id = "ed") |>
    defData(varname = "s_grp",
            formula = "0.55;0.15;0.15;0.15",
            dist = "categorical")

  # Conditional definitions for quarter sizes by group
  # Each group has different expected number of observations per quarter
  def_qn <- defCondition(condition = "s_grp == 1",
                         formula = "..grp1",
                         dist = "poisson") |>
```

```
    defCondition(condition = "s_grp == 2",
                 formula = "..grp2",
                 dist = "poisson") |>
    defCondition(condition = "s_grp == 3",
                 formula = "..grp3",
                 dist = "poisson") |>
    defCondition(condition = "s_grp == 4",
                 formula = "..grp4",
                 dist = "poisson")

  # Period-level data: number of observations per period
  d2 <- defDataAdd(varname = "nper",
                   formula = "quarter_n",
                   dist = "poisson")

  # Binary outcome model with logistic link
  # Includes main effects, two-way interactions, and three-way interaction
  defY <-
    defDataAdd(varname = "arm",
      formula = "r1 + r2*2 + r3*4",
      dist = "nonrandom") |>
    defDataAdd(varname = "y",
      formula = "..t_0 + a + ..t_a*r1 + ..t_b*r2 + ..t_c*r3 +
      ..x_ab*r1*r2 + ..x_ac*r1*r3 + ..x_bc*r2*r3 +
      ..x_abc*r1*r2*r3",
      dist = "binary",
      link = "logit")

  return(list(d1 = d1,
              d2 = d2,
              def_qn = def_qn,
              defY = defY))
}

#' Generate Simulated Dataset
#'
#' Generates a single simulated dataset based on the provided definitions
#' and parameter values.
#'
#' @param list_of_defs List of data definition objects from s_define()
#' @param argsvec Named vector of parameter values for the simulation
#'
#' @return A data.table containing the simulated dataset with:
#'   - Cluster identifiers (ed)
#'   - Time period identifiers
#'   - Treatment assignments (r1, r2, r3)
#'   - Individual observations (id)
#'   - Binary outcomes (y)
#'
#' @details The function:
#'   1. Generates clusters with random effects and group assignments
#'   2. Assigns quarter sizes based on group membership
#'   3. Performs stratified randomization by cluster size
```

```r
#'   4. Creates longitudinal structure with multiple periods
#'   5. Generates individual-level observations within periods
#'   6. Simulates binary outcomes based on the specified model

s_generate <- function(list_of_defs, argsvec) {

  # Import definitions and parameters into local environment
  list2env(list_of_defs, envir = environment())
  list2env(as.list(argsvec), envir = environment())

  # Calculate random effect variance based on ICC
  revar <- iccRE(icc, dist = "binary")

  # Generate cluster-level data
  dd <- genData(n_ed, d1)

  # Add conditional quarter definitions
  dd <- addCondition(def_qn, dd, "quarter_n")

  # Stratified randomization by cluster size group
  # Ensures balanced treatment assignment within each size stratum
  dd <- rbindlist(
    lapply(c(1:4), function(x) {
      addMultiFac(dd[s_grp == x],
                  nFactors = 3,
                  colNames = c("r1", "r2", "r3"))
    })
  )
  setkey(dd, "ed")

  # Add longitudinal structure (multiple quarters per cluster)
  dd <- addPeriods(dd, nPeriods = n_quarters, idvars = "ed")

  # Add number of observations per period
  dd <- addColumns(d2, dd)

  # Generate individual-level observations within each period
  dd <- genCluster(dd, "timeID", "nper", "id")

  # Generate binary outcomes
  dd <- addColumns(defY, dd)

  # Alternative grouping

  dd[, Grp := arm + 1]
  dd[arm == 3, Grp := 5]
  dd[arm == 4, Grp := 4]
  dd[, Grp := factor(Grp)]

  return(dd)
}

#' Create Parameter Scenarios
```

```r
#'
#' Generates all combinations of specified parameter values for simulation study.
#'
#' @param ... Named arguments specifying parameter values to vary
#'
#' @return List of parameter vectors, each representing one simulation scenario
#'
#' @examples
#' scenarios <- scenario_list(n_clusters = c(20, 40), icc = c(0.01, 0.05))

scenario_list <- function(...) {
  argmat <- expand.grid(...)
  return(asplit(argmat, MARGIN = 1))
}

# ============================================================================
# SIMULATION PARAMETERS
# ============================================================================

# Study design parameters
n_ed <- c(40, 48, 56)             # Number of clusters (sites)
icc <- 0.015                      # Intracluster correlation coefficient
n_quarters <- 2                   # Number of time periods per cluster
n_quarter <- c(60, 90, 190, 300)  # Expected observations per quarter by group

# Model parameters
t_0 <- -0.40                 # Intercept (log-odds scale)
t_a <- t_b <- t_c <- 0       # Main effect parameters (null hypothesis)
x_ab <- x_ac <- x_bc <- 0    # Two-way interaction parameters (null)
x_abc <- 0                   # Three-way interaction parameter (null)

# Generate all parameter combinations for simulation study
argsvec <- scenario_list(
  n_ed = n_ed,
  icc = icc,
  n_quarters = n_quarters,
  grp1 = n_quarter[1],       # Small clusters
  grp2 = n_quarter[2],       # Medium-small clusters
  grp3 = n_quarter[3],       # Medium-large clusters
  grp4 = n_quarter[4],       # Large clusters
  t_0 = t_0,
  t_a = t_a,
  t_b = t_b,
  t_c = t_c,
  x_ab = x_ab,
  x_ac = x_ac,
  x_bc = x_bc,
  x_abc = x_abc
)

# ============================================================================
# GENERATE SIMULATED DATASETS
# ============================================================================
```

```r
# Create data definition structure
list_of_defs <- s_define()

# Generate datasets for all parameter scenarios
# Each element of the list contains one simulated dataset
generated_data <- lapply(argsvec, function(args) {
  s_generate(list_of_defs, args)
})
```

02_prior_predictive.R

```r
#' ---
#' title: "Bayesian Prior Predictive Analysis for Cluster Randomized Trial"
#' author: "Keith Goldfeld"
#' ---

# ============================================================================
# REQUIRED LIBRARIES
# ============================================================================

library(cmdstanr)     # Bayesian modeling with Stan
library(ggplot2)      # Data visualization
library(ggpubr)       # Publication-ready plots and plot arrangements
library(data.table)   # Efficient data manipulation (from previous code)

#' Convert Data Table to Stan Data List
#'
#' Transforms a data.table containing trial data into a list format
#' suitable for Stan model fitting.
#'
#' @param dx A data.table containing the simulated trial data with columns:
#'   \itemize{
#'     \item{ed: cluster/site identifier}
#'     \item{r1, r2, r3: binary treatment indicators}
#'     \item{y: binary outcome variable}
#'   }
#' @param argsvec Named vector containing prior hyperparameter values:
#'   \itemize{
#'     \item{sigma_tau_m: prior SD for main effect coefficients}
#'     \item{sigma_tau_x: prior SD for interaction coefficients}
#'     \item{ss_m, ss_x, ss_3: hierarchical prior parameters}
#'     \item{ss_ed: prior SD for cluster random effects}
#'   }
#'
#' @return A list containing Stan data components:
#'   \itemize{
#'     \item{N: number of observations}
#'     \item{N_ED: number of clusters}
#'     \item{x_abc: design matrix with main effects and interactions}
#'     \item{ed: cluster identifiers}
#'     \item{y: binary outcomes}
#'     \item{sigma_*: prior hyperparameters}
#'   }
#'
#' @details Creates a full factorial design matrix including:
#'   - Intercept
#'   - Main effects (r1, r2, r3)
#'   - Two-way interactions (r1:r2, r1:r3, r2:r3)
#'   - Three-way interaction (r1:r2:r3)

dt_to_list <- function(dx, rep_scenario) {
```

```r
  # Import hyperparameters into local environment
  list2env(as.list(rep_scenario), envir = environment())

  # Extract data dimensions and components
  N <- nrow(dx)                                  # Total number of observations
  x_abc <- model.matrix(~r1*r2*r3, data = dx) # Full factorial design matrix
  y <- dx[, y]                                   # Binary outcomes
  N_ED <- dx[, length(unique(ed))]               # Number of clusters
  ed <- dx[, ed]                                 # Cluster identifiers

  # Return Stan data list with data and hyperparameters
  list(N_ED = N_ED,
       N = N,
       x_abc = x_abc,
       ed = ed,
       y = y,
       sigma_tau_m = sigma_tau_m,     # Prior SD for main effects
       sigma_tau_x = sigma_tau_x,     # Prior SD for interactions
       sigma_sigma_m = ss_m,          # Hierarchical prior parameter
       sigma_sigma_x = ss_x,          # Hierarchical prior parameter
       sigma_3 = ss_3,                # Hierarchical prior parameter
       sigma_sigma_ed = ss_ed)        # Prior SD for cluster effects
}

#' Generate Prior Predictive Check Plot
#'
#' Runs prior predictive simulation and creates histogram plots
#' showing the distribution of predicted probabilities for each
#' treatment arm combination.
#'
#' @param argsvec Named vector containing hyperparameter values for the prior
#'   distributions. Must include sigma_tau_m for panel labeling.
#'
#' @return A ggplot object showing histograms of predicted probabilities
#'   faceted by treatment arm, with panel title indicating hyperparameter values.
#'
#' @details The function:
#'   1. Samples from prior distributions using Stan
#'   2. Generates predicted outcomes (y_rep) for each observation
#'   3. Calculates mean predicted probability for each treatment arm
#'   4. Creates histograms showing distribution across prior draws
#'
#'   Treatment arms are labeled as:
#'   - None: no interventions
#'   - A, B, C: single interventions
#'   - AB, AC, BC: two-way combinations
#'   - ABC: all three interventions
#'
#' @note Requires a compiled Stan model object 'mod_prior' and
#'   simulated data 'generated_data' to be available in the global environment.

replicate <- function(rep_scenario) {
```

```r
  # Import hyperparameters into local environment
  list2env(as.list(rep_scenario), envir = environment())

  # Create panel labels based on main effect prior SD
  if (sigma_tau_m == 0.4) P <- "Panel 1: "
  if (sigma_tau_m == 0.8) P <- "Panel 2: "
  if (sigma_tau_m == 2.5) P <- "Panel 3: "

  # Run prior predictive sampling
  fit_prior <- mod_prior$sample(
    data = dt_to_list(generated_data, rep_scenario),
    chains = 1,                    # Single chain for prior predictive
    iter_sampling = 10000,         # Number of prior draws
    fixed_param = TRUE,            # Prior predictive mode
    refresh = 1000                 # Progress update frequency
  )

  # Extract posterior predictive draws
  y_rep_draws <- fit_prior$draws(c("y_rep"))

  # Calculate mean predicted probability for each treatment arm
  probs <- rbindlist(lapply(0:7, function(i) {
    vars_of_interest <- which(generated_data$arm == i)
    arm_probs <- apply(y_rep_draws[,,vars_of_interest], 1, mean)
    data.table(arm_probs)
  }), idcol = TRUE)

  # Create treatment arm labels
  labels <- c("None","A", "B", "C", "AB", "AC", "BC", "ABC")
  probs[, arm_label := factor(.id, levels = 1:8, labels = labels)]

  # Reverse factor levels for display (None at top, ABC at bottom)
  probs[, arm_label := factor(arm_label, levels = rev(labels))]

  # Create histogram plot
  ggplot(data = probs, aes(x = arm_probs)) +
    geom_histogram(boundary = 0,
                   color = "black",
                   fill = "tomato",
                   aes(y = after_stat(density)),
                   bins = 30) +
    facet_grid(arm_label ~ .) +
    ggtitle(bquote(bold(.(P)) ~ nu[tau] == .(sigma_tau_m) * "," ~
                     nu[sigma] == .(ss_m))) +
    labs(x = "predicted probability",
         y = "density") +
    theme_minimal(base_size = 9) +
    theme(plot.title = element_text(size = 9),
          axis.text.y = element_blank(),
          axis.ticks.y = element_blank(),
          axis.title.y = element_blank(),
          panel.grid = element_blank())
}
```

```r
# ============================================================================
# COMPILE STAN MODEL
# ============================================================================

# Compile the prior-only model
# Note: Requires "prior_predictive_model.stan" file in working directory
mod_prior <- cmdstan_model("prior_predictive_model.stan")

# ============================================================================
# GENERATE SIMULATED DATASET
# ============================================================================

# Study design parameters
n_ed <- 48                          # Number of clusters (sites)
icc <- 0.015                        # Intracluster correlation coefficient
n_quarters <- 2                     # Number of time periods per cluster
n_quarter <- c(60, 90, 190, 300)  # Expected observations per quarter by group

# Model parameters (under null hypothesis)
t_0 <- -0.85                  # Intercept (log-odds scale)
t_a <- t_b <- t_c <- 0        # Main effect parameters
x_ab <- x_ac <- x_bc <- 0     # Two-way interaction parameters
x_abc <- 0                    # Three-way interaction parameter

# Generate parameter combinations for simulation study
# Note: Requires scenario_list() function from previous code
argsvec <- scenario_list(
  n_ed = n_ed,
  icc = icc,
  n_quarters = n_quarters,
  grp1 = n_quarter[1],        # Small clusters
  grp2 = n_quarter[2],        # Medium-small clusters
  grp3 = n_quarter[3],        # Medium-large clusters
  grp4 = n_quarter[4],        # Large clusters
  t_0 = t_0,
  t_a = t_a,
  t_b = t_b,
  t_c = t_c,
  x_ab = x_ab,
  x_ac = x_ac,
  x_bc = x_bc,
  x_abc = x_abc
)

# Create data definition structure
# Note: Requires s_define() function from previous code
list_of_defs <- s_define()

# Generate single simulated dataset
# Note: Requires s_generate() function from previous code
generated_data <- lapply(argsvec, function(args) {
  s_generate(list_of_defs, args)
})[[1]]
```

```r
# ============================================================================
# PRIOR SENSITIVITY ANALYSIS SCENARIOS
# ============================================================================

#' Prior Hyperparameter Scenarios
#'
#' Three scenarios representing different levels of prior informativeness:
#'
#' Scenario 1 (Informative): Small prior SDs suggesting modest treatment effects
#' Scenario 2 (Moderate): Medium prior SDs allowing moderate treatment effects
#' Scenario 3 (Weakly informative): Large prior SDs allowing substantial effects
#'
#' Each scenario specifies:
#' - sigma_tau_m/x: Prior SDs for main effects and interactions
#' - ss_m/x: Hierarchical prior parameters for effect heterogeneity
#' - ss_3: Additional hierarchical parameter
#' - ss_ed: Prior SD for cluster random effects

scenarios <- list(
  # Scenario 1: Informative priors (small treatment effects expected)
  c(sigma_tau_m = 0.4, sigma_tau_x = 0.4, ss_m = 0.3, ss_x = 0.3,
    ss_3 = 0.3, ss_ed = 0.5),

  # Scenario 2: Moderately informative priors
  c(sigma_tau_m = 0.8, sigma_tau_x = 0.8, ss_m = 0.6, ss_x = 0.6,
    ss_3 = 0.3, ss_ed = 0.5),

  # Scenario 3: Weakly informative priors (large effects possible)
  c(sigma_tau_m = 2.5, sigma_tau_x = 2.5, ss_m = 2.5, ss_x = 2.5,
    ss_3 = 0.3, ss_ed = 0.5)
)

# ============================================================================
# GENERATE PRIOR PREDICTIVE CHECK PLOTS
# ============================================================================

# Generate plots for each prior scenario
plots <- lapply(scenarios, function(x) replicate(x))

# Combine plots into single figure
p1 <- ggarrange(plotlist = plots, nrow = 1)

# Display the combined plot
print(p1)

# ============================================================================
# INTERPRETATION NOTES
# ============================================================================

# The resulting plots show:
#
# 1. How different prior specifications affect predicted outcome probabilities
# 2. Whether priors lead to reasonable predictions across treatment arms
```

```
# 3. The range of plausible effect sizes implied by each prior
#
# Key considerations:
# - Do the predicted probabilities span a reasonable range?
# - Are there concerning patterns (e.g., all arms having identical distributions)?
# - Do the priors allow for clinically meaningful effect sizes?
#
# This analysis helps validate prior choices before fitting to real data.
```

03_sample_size.R

```r
#' ---
#' title: "Bayesian Power Analysis for Sample Size Determination"
#' author: "Keith Goldfeld"
#' date: "`r Sys.Date()`"
#' description: "Monte Carlo simulation study to determine required sample size
#'               for adequate precision in Bayesian analysis of factorial cluster
#'               randomized trial based on posterior standard deviation criteria"
#' ---

# ============================================================================
# REQUIRED LIBRARIES
# ============================================================================

library(simstudy)      # Data simulation
library(data.table)    # Efficient data manipulation
library(cmdstanr)      # Bayesian modeling with Stan
library(posterior)     # Working with posterior draws
library(parallel)      # Parallel computing for Monte Carlo simulation
library(ggplot2)       # Data visualization

#' Bayesian Model Fitting and Posterior Summarization
#'
#' Fits a Bayesian model to simulated trial data and extracts key posterior
#' statistics for power/precision analysis.
#'
#' @param generated_data A data.table containing simulated trial data from s_generate()
#' @param mod A compiled Stan model object
#' @param argsvec Named vector containing hyperparameter values for the model
#'
#' @return A data.table with posterior summaries for each parameter:
#'   \itemize{
#'     \item{variable: parameter name (lOR, tau, sigma, delta)}
#'     \item{p.025, p.25, p.50, p.75, p.95, p.975: posterior quantiles}
#'     \item{sd: posterior standard deviation (key metric for precision)}
#'     \item{var: posterior variance}
#'     \item{p_less_zero: probability parameter is negative}
#'     \item{p_meaningful: probability of clinically meaningful effect (< -0.223)}
#'   }
#'
#' @details The function:
#'   1. Converts simulated data to Stan format
#'   2. Fits Bayesian model with robust MCMC settings
#'   3. Extracts posterior draws for treatment effects and model parameters
#'   4. Computes posterior summaries including precision metrics
#'
#'   MCMC Settings optimized for complex hierarchical models:
#'   - 4 chains with 500 warmup + 2500 sampling iterations
#'   - High adapt_delta (0.98) and max_treedepth (20) for difficult geometry
#'   - Parallel chain execution for efficiency

s_bayes <- function(generated_data, mod, argsvec) {
```

```r
#' Convert Data.Table to Stan Data List
#'
#' Internal helper function that transforms simulated trial data
#' into the list format required by Stan models.

dt_to_list <- function(dx) {

  N <- nrow(dx)                                  # Total number of observations
  x_abc <- model.matrix(~r1*r2*r3, data = dx) # Full factorial design matrix
  y <- dx[, y]                                   # Binary outcomes
  N_ED <- dx[, length(unique(ed))]               # Number of clusters
  ed <- dx[, ed]                                 # Cluster identifiers
  svals <- c(s1, s2, s3, s4, s5, s6)             # Prior hyperparameters

  list(N_ED = N_ED, N = N, x_abc = x_abc, ed = ed, y = y, svals = svals)
}

# Import hyperparameters into local environment
list2env(as.list(argsvec), envir = environment())

# Fit Bayesian model with robust MCMC settings
fit <- mod$sample(
  data = dt_to_list(generated_data),
  refresh = 0,                     # Suppress progress output for batch processing
  chains = 4L,                     # Multiple chains for convergence assessment
  parallel_chains = 4L,            # Parallel execution for efficiency
  iter_warmup = 500,               # Warmup iterations for adaptation
  iter_sampling = 2500,            # Sampling iterations per chain
  adapt_delta = 0.98,              # High acceptance rate for difficult geometry
  max_treedepth = 20,              # Allow deep trees for complex posteriors
  show_messages = FALSE            # Suppress Stan messages for batch processing
)

# Convert to posterior draws format
posterior <- as_draws_array(fit$draws())

# Extract and summarize key parameters
# Focus on treatment effects (lOR), coefficients (tau), and scale parameters (sigma)
sumbayes <- as.data.table(posterior)[
  substr(variable, 1, 3) == "lOR" |          # Log-odds ratios (treatment effects)
    substr(variable, 1, 3) == "tau" |        # Regression coefficients
    substr(variable, 1, 5) == "sigma" |      # Scale parameters
    substr(variable, 1, 5) == "delta",       # Additional parameters (if present)
  .(
    p.025 = quantile(value, 0.025),          # 95% credible interval bounds
    p.25 = quantile(value, 0.25),            # IQR bounds
    p.50 = quantile(value, 0.50),            # Posterior median
    p.75 = quantile(value, 0.75),            # IQR bounds
    p.95 = quantile(value, 0.95),            # 90% credible interval
    p.975 = quantile(value, 0.975),          # 95% credible interval bounds
    sd = sd(value),                          # PRECISION METRIC: posterior SD
    var = var(value),                        # Posterior variance
    p_less_zero = mean(value < 0),           # Probability of negative effect
```

```r
      p_meaningful = mean(value < -0.223)      # Probability of meaningful effect
      # (log(0.8) = -0.223 corresponds to OR < 0.8)
    ), keyby = variable]

  return(sumbayes)  # Return posterior summary table
}

#' Single Replication of Power Analysis
#'
#' Performs one complete replication of the power analysis: generates data,
#' fits model, and extracts results.
#'
#' @param argsvec Named vector of simulation parameters including:
#'   \itemize{
#'     \item{n_ed: number of clusters}
#'     \item{icc: intracluster correlation}
#'     \item{grp1-grp4: cluster size parameters}
#'     \item{t_0, t_a, t_b, t_c: effect parameters}
#'     \item{s1-s6: prior hyperparameters}
#'   }
#' @param mod Compiled Stan model object
#'
#' @return List containing:
#'   \itemize{
#'     \item{args: input simulation parameters}
#'     \item{model_res: posterior summary table from s_bayes()}
#'   }
#'
#' @details This function represents one Monte Carlo iteration:
#'   1. Generate simulated dataset under null hypothesis
#'   2. Fit Bayesian model to simulated data
#'   3. Extract posterior summaries
#'   4. Return results with parameter tracking

s_replicate <- function(argsvec, mod) {

  # Generate simulated dataset
  # Note: Requires s_define() and s_generate() functions from previous code
  list_of_defs <- s_define()                      # Create data definitions
  generated_data <- s_generate(list_of_defs, argsvec)  # Generate one dataset

  # Fit Bayesian model and extract posterior summaries
  model_bayes_1 <- s_bayes(generated_data, mod, argsvec)

  # Package results with input parameters for tracking
  summary_stats <- c(
    list(args = argsvec,              # Input parameters
         model_res = model_bayes_1)   # Posterior summaries
  )

  return(summary_stats)
}
```

```r
#' Create Parameter Scenarios for Power Analysis
#'
#' Generates all combinations of specified parameter values.
#' Identical to previous definition but included for completeness.

scenario_list <- function(...) {
  argmat <- expand.grid(...)
  return(asplit(argmat, MARGIN = 1))
}

# ============================================================================
# POWER ANALYSIS PARAMETERS
# ============================================================================

# Sample size range for evaluation
# Testing 6 different cluster counts: 48, 56, 64, 72, 80, 88 clusters
n_ed <- 8 * c(6, 7, 8, 9, 10, 11)

# Fixed study design parameters
icc <- 0.015                          # Intracluster correlation coefficient
n_quarters <- 2                       # Time periods per cluster

# Base cluster sizes (before scaling)
grp1 <- 60                           # Small clusters
grp2 <- 90                           # Medium-small clusters
grp3 <- 190                          # Medium-large clusters
grp4 <- 300                          # Large clusters

# Cluster size scaling based on expected patient volume
# Testing scenario with 40 patients/month average
mean <- 40                           # Expected patients per month per cluster
# Alternative values: 25, 30, 35, 40
ratio <- mean/40                     # Scaling factor from base scenario

# Apply scaling to all cluster size groups
grp1 <- grp1 * ratio
grp2 <- grp2 * ratio
grp3 <- grp3 * ratio
grp4 <- grp4 * ratio

# Treatment effect parameters (NULL HYPOTHESIS scenario)
# All effects set to zero to evaluate Type I error control and precision
t_0 <- -0.40                         # Baseline log-odds (approximately 40% control rate)
t_a <- 0                            # Main effect A (null)
t_b <- 0                            # Main effect B (null)
t_c <- 0                            # Main effect C (null)
x_ab <- 0                           # Interaction A×B (null)
x_ac <- 0                           # Interaction A×C (null)
x_bc <- 0                           # Interaction B×C (null)
x_abc <- 0                          # Three-way interaction A×B×C (null)

# Prior hyperparameter scenario
# These values determine the informativeness of priors
```

```r
sds <- c(0.3, 0.3, 0.4, 0.4, 0.5, 0.4)  # Scenario 1 hyperparameters
# [s1, s2, s3, s4, s5, s6]

# Generate all parameter combinations
scenarios <- scenario_list(
  n_ed = n_ed, icc = icc, n_quarters = n_quarters,
  grp1 = grp1, grp2 = grp2, grp3 = grp3, grp4 = grp4,
  t_0 = t_0, t_a = t_a, t_b = t_b, t_c = t_c,
  x_ab = x_ab, x_ac = x_ac, x_bc = x_bc, x_abc = x_abc,
  s1 = sds[1], s2 = sds[2], s3 = sds[3], s4 = sds[4], s5 = sds[5], s6 = sds[6]
)

# Replicate each scenario for Monte Carlo simulation
# 1000 replications per parameter combination for stable estimates
# but be forewarned that this was actually run on an HPC. If you
# want code to set that up, pleast contact me.
scenarios <- rep(scenarios, each = 1000)

# ============================================================================
# MONTE CARLO SIMULATION
# ============================================================================

# Compile Stan model (HEx model variant)
# Note: Requires "model_HEX.stan" file in working directory
mod <- cmdstan_model("model_HEX.stan")

# Run Monte Carlo simulation in parallel
# Using mclapply for parallel processing across CPU cores
# Note: Adjust mc.cores based on available hardware
res <- mclapply(scenarios, function(a) s_replicate(a, mod), mc.cores = 4)

# ============================================================================
# RESULTS ANALYSIS AND VISUALIZATION
# ============================================================================

#' Precision Analysis for Multiple Intervention Effects
#'
#' Focuses on the precision (posterior standard deviation) of interaction
#' effects, which are typically the most difficult to estimate precisely
#' and represent the key scientific questions in factorial designs.

# Define interaction effects of interest
# These represent two-way and three-way treatment combinations
x_effects <- c("lOR[4]",    # AB vs None
               "lOR[5]",    # AC vs None
               "lOR[6]",    # BC vs None
               "lOR[7]")    # ABC vs None

# Extract sample sizes from results
n_ed <- sapply(res, function(x) x[["args"]]["n_ed"])

# Extract posterior standard deviations for interaction effects
res_sd_x <- lapply(res, function(x) {
```

```
  x[["model_res"]][variable %in% c(x_effects), .(variable, sd)]
})

# Combine results with sample size information
res_sd_x <- rbindlist(
  Map(function(dt, val) {
    dt[, n_ed := val]
    return(dt)
  }, res_sd_x, n_ed)
)

# Compute average precision across Monte Carlo replications
sum_sd_x <- res_sd_x[, .(sd = mean(sd)), keyby = n_ed]

#' Create Precision vs Sample Size Plot
#'
#' Visualizes the relationship between number of clusters and precision
#' of interaction effect estimates. The horizontal line at 0.13 represents
#' a precision target (somewhat arbitrary but represents "adequate precision").
#'

p40 <- ggplot() +

  # Reference line for precision target
  geom_hline(yintercept = 0.13, size = .5, color = "grey80") +

  # Individual Monte Carlo results (density and intervals)
  stat_halfeye(data = res_sd_x,
               aes(group = n_ed, y = sd, x = n_ed,
                   color = after_stat(as.character(.width))),
               width = 2, fill = "grey72",
               point_color = "black", .width = c(.5, .90)) +

  # Formatting
  scale_color_manual(values = c("#9a0000", "black"),
                     name = "Interval") +
  scale_x_continuous(limits = c(45, 91), breaks = seq(48, 88, by = 8), name = "number of EDs") +
  scale_y_continuous(limits = c(0.08, 0.20), breaks = c(.13, seq(0.10, .20, by = .05)),
                     name = "sd of posterior distribution") +
  theme(panel.grid = element_blank(),
        plot.title = element_text(size = 12, face = "bold"),
        axis.title = element_text(size = 11),
        axis.text = element_text(size = 12),
        legend.position = "none"
  )

# Display results
print(p40)

# ============================================================================
# INTERPRETATION AND SAMPLE SIZE RECOMMENDATION
# ============================================================================
```

```
#' INTERPRETATION OF RESULTS:
#'
#' The plot shows how posterior precision (standard deviation) decreases
#' as the number of clusters increases. Key considerations:
#'
#' 1. PRECISION TARGET: The horizontal line at SD = 0.13 represents a
#'    somewhat arbitrary precision target - speficic to our appli aiton.
#'    You may want to adjust this based on clinically meaningful effect sizes.
#'
#' 2. DIMINISHING RETURNS: The curve shows diminishing returns - initial
#'    increases in sample size provide large precision gains, but
#'    additional clusters provide smaller improvements.
#'
#' 3. INTERACTION EFFECTS: This analysis focuses on interaction effects
#'    (lOR[4]-lOR[7]) which are typically hardest to estimate precisely.
#'    Main effects will generally have better precision.
#'
#' 4. NULL SCENARIO: Results are based on null hypothesis (no true effects).
#'    Precision under alternative hypotheses may differ slightly.
#'
#' SAMPLE SIZE RECOMMENDATION:
#' Based on the precision target of SD = 0.13, approximately [X] clusters
#' would be needed. However, consider:
#'
#' - Clinical meaningfulness of the precision target
#' - Cost/feasibility constraints
#' - Balance between Type I error, power, and precision
#' - Sensitivity to prior specifications
#'
#' EXTENSIONS:
#' - Test different precision targets
#' - Analyze main effects separately
#' - Include power analysis for non-null scenarios
#' - Sensitivity analysis across different prior specifications
```

04_comparison_study.R

```r
#' ---
#' title: "Factorial Intervention Simulation Study"
#' author: "Keith Goldfeld"
#' date: "`r Sys.Date()`"
#' description: "Monte Carlo simulation study comparing Bayesian and frequentist
#'               approaches for analyzing 2³ factorial cluster randomized trials"
#' ---

# ============================================================================
# REQUIRED LIBRARIES
# ============================================================================

library(cmdstanr)      # Stan interface for Bayesian modeling
library(simstudy)      # Data simulation tools
library(data.table)    # Enhanced data manipulation
library(posterior)     # Posterior analysis tools
library(glmmTMB)       # Generalized linear mixed models

#' Bayesian Model Fitting and Posterior Summarization
#'
#' Fits a Bayesian model to simulated trial data using Stan and extracts
#' posterior summaries for key parameters.
#'
#' @param generated_data A data.table containing simulated trial data
#' @param mod A compiled Stan model object (from cmdstan_model)
#' @param argsvec Named vector containing simulation parameters and hyperparameters
#'
#' @return A data.table with posterior summaries for each parameter:
#'   \itemize{
#'     \item{variable: parameter name (lOR, tau, sigma, delta)}
#'     \item{p.025, p.25, p.50, p.75, p.95, p.975: posterior quantiles}
#'     \item{sd: posterior standard deviation}
#'     \item{var: posterior variance}
#'     \item{p_less_zero: probability parameter is negative}
#'     \item{p_meaningful: probability of meaningful effect (< -0.223, OR < 0.8)}
#'   }
#'
#' @details The function:
#'   \enumerate{
#'     \item Converts data to Stan format using internal dt_to_list helper
#'     \item Fits Bayesian model with robust MCMC settings
#'     \item Extracts posterior draws for treatment effects and model parameters
#'     \item Computes comprehensive posterior summaries
#'   }
#'
#'   MCMC Settings:
#'   \itemize{
#'     \item 4 chains with 500 warmup + 2500 sampling iterations
#'     \item High adapt_delta (0.98) and max_treedepth (20) for complex models
#'     \item Parallel chain execution for efficiency
#'   }
```

```r
#'
#' @examples
#' \dontrun{
#' posterior_summary <- s_bayes(generated_data, mod, argsvec)
#' }

s_bayes <- function(generated_data, mod, argsvec) {

  #' Convert Data.Table to Stan Data List
  #'
  #' Internal helper function that transforms simulated trial data
  #' into the list format required by Stan models.

  dt_to_list <- function(dx) {

    N <- nrow(dx)                                ## number of observations
    x_abc <- model.matrix(~r1*r2*r3, data = dx) # Full factorial design matrix

    y <- dx[, y]                                 # Binary outcomes

    N_ED <- dx[, length(unique(ed))]             # Number of clusters
    ed <- dx[, ed]                               # Cluster identifiers
    svals <- c(s1, s2, s3, s4, s5, s6)           # Prior hyperparameters

    list(N_ED = N_ED, N = N, x_abc = x_abc, ed = ed, y = y, svals = svals)
  }

  # Import simulation parameters into local environment
  list2env(as.list(argsvec), envir = environment())

  # Fit Bayesian model with robust MCMC settings
  fit <- mod$sample(
    data = dt_to_list(generated_data),
    refresh = 0,                    # Suppress progress output
    chains = 4L,                    # Multiple chains for convergence
    parallel_chains = 4L,           # Parallel execution
    iter_warmup = 500,              # Warmup iterations
    iter_sampling = 2500,           # Sampling iterations per chain
    adapt_delta = 0.98,             # High acceptance rate for difficult geometry
    max_treedepth = 20,             # Allow deep trees for complex posteriors
    show_messages = FALSE           # Suppress Stan messages
  )

  # Convert to posterior draws format
  posterior <- as_draws_array(fit$draws())

  # Extract and summarize key parameters
  sumbayes <- as.data.table(posterior)[
    substr(variable, 1, 3) == "lOR" |          # Log-odds ratios (treatment effects)
      substr(variable, 1, 3) == "tau" |        # Regression coefficients
      substr(variable, 1, 5) == "sigma" |      # Scale parameters
      substr(variable, 1, 5) == "delta",       # Additional parameters
    .(
```

```
      p.025 = quantile(value, 0.025),          # 95% credible interval bounds
      p.25 = quantile(value, 0.25),            # IQR bounds
      p.50 = quantile(value, 0.50),            # Posterior median
      p.75 = quantile(value, 0.75),            # IQR bounds
      p.95 = quantile(value, 0.95),            # 90% credible interval
      p.975 = quantile(value, 0.975),          # 95% credible interval bounds
      sd = sd(value),                          # Posterior standard deviation
      var = var(value),                        # Posterior variance
      p_less_zero = mean(value < 0),           # Probability of negative effect
      p_meaningful = mean(value < -0.223)      # Probability of meaningful effect
    ), keyby = variable]

  return(sumbayes) # model_results is a data.table

}

#' Frequentist Analysis Using Generalized Linear Mixed Models
#'
#' Performs hierarchical model selection using likelihood ratio tests to
#' determine the appropriate level of complexity for the factorial design.
#'
#' @param dd A data.table containing the trial data for analysis
#'
#' @return List containing:
#'   \itemize{
#'     \item{glm0: Null model coefficients (intercept only + random effects)}
#'     \item{glm1: Main effects model coefficients}
#'     \item{glm2: Two-way interactions model coefficients}
#'     \item{glm3: Three-way interaction model coefficients}
#'     \item{glmg: Group model coefficients (factor(Grp) + random effects)}
#'     \item{h_conclusion: Hierarchical model selection conclusion}
#'   }
#'
#' @details The function fits five nested models and uses sequential likelihood
#'   ratio tests (LRT) with $\alpha$ = 0.05 to determine model complexity:
#'   \enumerate{
#'     \item Test main effects vs null model
#'     \item If significant, test 2-way interactions vs main effects
#'     \item If significant, test 3-way interaction vs 2-way model
#'   }
#'
#'   Models fitted:
#'   \itemize{
#'     \item{Null: y ~ 1 + (1|ed)}
#'     \item{Main effects: y ~ r1 + r2 + r3 + (1|ed)}
#'     \item{Two-way: y ~ r1*r2 + r1*r3 + r2*r3 + (1|ed)}
#'     \item{Three-way: y ~ r1*r2*r3 + (1|ed)}
#'     \item{Group: y ~ factor(Grp) + (1|ed)}
#'   }
#'
#' @examples
#' \dontrun{
#' freq_results <- s_freq(generated_data)
```

```r
#' print(freq_results$h_conclusion)
#' }

s_freq <- function(dd) {

  # Fit nested models for hierarchical testing

  glmfit_3 <- glmmTMB(y ~ r1*r2*r3 + (1|ed), family=binomial, data = dd)
  glmfit_2 <- glmmTMB(y ~ r1*r2 + r1*r3 + r2*r3 + (1|ed), family=binomial, data = dd)
  glmfit_1 <- glmmTMB(y ~ r1 + r2 + r3 + (1|ed), family=binomial, data = dd)
  glmfit_0 <- glmmTMB(y ~ 1 + (1|ed), family=binomial, data = dd)
  glmfit_g <- glmmTMB(y ~ factor(Grp) + (1|ed), family=binomial, data = dd)

  # Sequential likelihood ratio tests for model selection

  LRT_p1 <- anova(glmfit_1, glmfit_0)$`Pr(>Chisq)`[2]  # Main effects test
  h_conclusion <- NA

  if (LRT_p1 <= 0.05) { # main effects present

    LRT_p2 <- anova(glmfit_2, glmfit_1)$`Pr(>Chisq)`[2]  # 2-way interactions test

    if (LRT_p2 <= 0.05) { # 2-way effects present

      LRT_p3 <- anova(glmfit_3, glmfit_2)$`Pr(>Chisq)`[2]  # 3-way interaction test
      if (LRT_p3 <= 0.05) {
        h_conclusion = "3-way interaction"
      } else {
        h_conclusion = "2-way interaction"
      }
    } else {
      h_conclusion = "main effects only"
    }
  } else {
    h_conclusion = "no effects"
  }

  # Return model summaries and hierarchical conclusion
  return(list(
    glm0 = as.data.table(summary(glmfit_0)$coefficients$cond, keep.rownames = "param"),
    glm1 = as.data.table(summary(glmfit_1)$coefficients$cond, keep.rownames = "param"),
    glm2 = as.data.table(summary(glmfit_2)$coefficients$cond, keep.rownames = "param"),
    glm3 = as.data.table(summary(glmfit_3)$coefficients$cond, keep.rownames = "param"),
    glmg = as.data.table(summary(glmfit_g)$coefficients$cond, keep.rownames = "param"),
    h_conclusion = h_conclusion
  ))

}

#' Complete Simulation Replication
#'
#' Executes one complete simulation replication including data generation,
#' Bayesian model fitting, and frequentist analysis.
```

```r
#'
#' @param argsvec Named vector containing all simulation parameters:
#'   \itemize{
#'     \item{n_ed: number of clusters}
#'     \item{icc: intracluster correlation}
#'     \item{n_quarters: number of time periods}
#'     \item{grp1-grp4: cluster size parameters}
#'     \item{t_0, t_a, t_b, t_c: treatment effect parameters}
#'     \item{x_ab, x_ac, x_bc, x_abc: interaction parameters}
#'     \item{s1-s6: prior hyperparameters}
#'   }
#' @param mod List containing two compiled Stan model objects [mod_HEX, mod_nonX]
#'
#' @return List containing:
#'   \itemize{
#'     \item{args: input simulation parameters}
#'     \item{model_Q: posterior summaries from first Bayesian model}
#'     \item{model_S: posterior summaries from second Bayesian model}
#'     \item{model_glm: frequentist analysis results}
#'   }
#'
#' @details This function represents one Monte Carlo iteration:
#'   \enumerate{
#'     \item Generate simulated dataset using s_define() and s_generate()
#'     \item Fit two different Bayesian models
#'     \item Perform frequentist hierarchical model selection
#'     \item Package all results with input parameters
#'   }
#'
#' @examples
#' \dontrun{
#' replication_result <- s_replicate(argsvec, list(mod_HEX, mod_nonX))
#' }

s_replicate <- function(argsvec, mod) {

  # Generate simulated dataset
  list_of_defs <- s_define()                          # Define data structure
  generated_data <- s_generate(list_of_defs, argsvec)  # Generate one dataset

  # Fit Bayesian models
  model_bayes_1 <- s_bayes(generated_data, mod[[1]], argsvec)  # First model
  model_bayes_2 <- s_bayes(generated_data, mod[[2]], argsvec)  # Second model

  # Fit frequentist models
  model_freq <- s_freq(generated_data)

  #--- Package summary statistics ---#

  summary_stats <- c(
    list(args = argsvec,              # Input parameters
         model_Q =  model_bayes_1,    # First Bayesian model results
         model_S = model_bayes_2,     # Second Bayesian model results
```

```
        model_glm = model_freq)      # Frequentist results
  )

  return(summary_stats) # summary_stats is a list
}

#' Create Parameter Scenarios for Simulation Study
#'
#' Generates all combinations of specified parameter values for the
#' Monte Carlo simulation study.
#'
#' @param ... Named arguments representing parameter values to combine
#'
#' @return List of parameter vectors, each representing one simulation scenario
#'
#' @details Uses expand.grid() to create all parameter combinations and splits
#'   into list format suitable for parallel processing with mclapply().
#'
#' @examples
#' \dontrun{
#' scenarios <- scenario_list(n_ed = c(48, 64, 80), icc = 0.015)
#' }

scenario_list <- function(...) {
  argmat <- expand.grid(...)
  return(asplit(argmat, MARGIN = 1))
}

# ============================================================================
# SIMULATION PARAMETERS
# ============================================================================

#--- Study Design Parameters ---#

n_ed <- 8 * 10                       # Number of emergency departments (80)
icc <- 0.015                         # Intracluster correlation coefficient
n_quarters <- 2                      # Number of time periods per cluster

#--- Cluster Size Parameters ---#
# Base values: mean 40 (distribution: .55, .15, .15, .15)

grp1 <- 60                           # Small clusters baseline
grp2 <- 90                           # Medium-small clusters baseline
grp3 <- 190                          # Medium-large clusters baseline
grp4 <- 300                          # Large clusters baseline

### Choose a mean <-------------------

mean <- 40        # possible values = 25, 30, 35, 40
ratio <- mean/40  # Scaling factor from base scenario

# Apply scaling to all cluster size groups
grp1 <- grp1 * ratio
```

```r
grp2 <- grp2 * ratio
grp3 <- grp3 * ratio
grp4 <- grp4 * ratio

#--- Treatment Effect Parameters ---#
# OR for single intervention: 0.80, OR for 2 interventions: 0.70

t_0 <- -0.40      # Baseline log-odds (approximately 40% control rate)
t_a <- -0.20      # Main effect A (OR = 0.82)
t_b <- 0          # Main effect B (null)
t_c <- -0.20      # Main effect C (OR = 0.82)
x_ab <- 0         # A×B interaction (null)
x_ac <- 0.05      # A×C interaction (small positive)
x_bc <- 0         # B×C interaction (null)
x_abc <- 0.0      # A×B×C three-way interaction (null)

# Alternative null scenario (commented out):
# t_0 <- -0.40
# t_a <- 0
# t_b <- 0
# t_c <- 0
# x_ab <- 0
# x_ac <- 0
# x_bc <- 0
# x_abc <- 0

#--- Prior Hyperparameters ---#

sds <- c(0.3, 0.3, 0.4, 0.4, 0.5, 0.4) # Scenario 1: moderate priors
# sds <- c(0.6, 0.6, 0.8, 0.8, 0.5, 0.8) # Scenario 2: wide priors
# sds <- c(3.0, 0.15, 0.40, 0.40, 3.0, 0.30)  # Scenario 3: inverse gamma

#--- Generate All Parameter Combinations ---#

scenarios <- scenario_list(n_ed = n_ed, icc = icc, n_quarters = n_quarters,
                           grp1 = grp1, grp2 = grp2, grp3 = grp3, grp4 = grp4,
                           t_0 = t_0, t_a = t_a, t_b = t_b, t_c = t_c,
                           x_ab = x_ab, x_ac = x_ac, x_bc = x_bc, x_abc = x_abc,
                           s1 = sds[1], s2 = sds[2], s3 = sds[3],
                           s4 = sds[4], s5 = sds[5], s6 = sds[6])

# Replicate each scenario for Monte Carlo simulation
scenarios <- rep(scenarios, each = 1000) # 1000 replications per scenario

# ============================================================================
# MODEL COMPILATION AND SIMULATION EXECUTION
# ============================================================================

#### Compile Stan Models ####

mod_HEX <- cmdstan_model("model_HEX.stan")       # Primary Bayesian model
mod_nonX <- cmdstan_model("model_nonX.stan")     # Alternative Bayesian model
# mod_HEX <- cmdstan_model("model_HEX_ig.stan") # Inverse gamma variant
```

```r
#### Run Monte Carlo Simulation ####

# Execute simulation in parallel
# Note: Fix typo in original code (mode_nonX should be mod_nonX)
res <- mclapply(scenarios, function(a) s_replicate(a, list(mod_HEX, mod_nonX)), mc.cores = 4)

# ============================================================================
# SIMULATION STUDY NOTES
# ============================================================================

#' STUDY DESIGN NOTES:
#'
#' This simulation study compares Bayesian and frequentist approaches for
#' analyzing 2³ factorial cluster randomized trials with the following features:
#'
#' 1. DESIGN: 2³ factorial (8 treatment combinations) with 80 clusters
#' 2. OUTCOME: Binary outcome with logistic regression
#' 3. CLUSTERING: Random intercepts for emergency departments
#' 4. BAYESIAN MODELS: Two different Stan implementations
#' 5. FREQUENTIST: Hierarchical model selection via LRT
#'
#' KEY RESEARCH QUESTIONS:
#' - Type I error rates and power comparison
#' - Robustness to prior specifications
#' - Model selection performance
#' - Estimation accuracy for interaction effects
#'
#' COMPUTATIONAL REQUIREMENTS:
#' - 1000 replications × 2 Bayesian models = 2000 MCMC runs
#' - Substantial memory and time requirements
#' - Parallel processing recommended (mc.cores = 4)
#'
#' DEPENDENCIES:
#' - Requires s_define() and s_generate() functions (not included)
#' - Requires model_HEX.stan and model_nonX.stan files
#' - Original code has typo: "mode_nonX" should be "mod_nonX"
```

model_HEX.stan

```
/*
 * Bayesian Hierarchical Exchangeable (HEx) Model for
 * Multi-Factorial Cluster Randomized Trial
 *
 * This Stan model implements a Bayesian analysis for a 2^3 factorial design
 * within a cluster randomized trial framework. The model accounts for:
 * - Three binary treatment factors (A, B, C) and their interactions
 * - Cluster-level random effects for healthcare sites/EDs
 * - Hierarchical priors with different assumptions for main effects vs interactions
 * - Binary outcomes with logistic regression
 *
 * Model Structure:
 * logit(P(y_i = 1)) = alpha_j[i] + tau_a*A_i + tau_b*B_i + tau_c*C_i +
 *                     tau_ab*A_i*B_i + tau_ac*A_i*C_i + tau_bc*B_i*C_i + tau_abc*A_i*B_i*C_i
 *
 * where:
 * - alpha_j[i] is the random effect for cluster j containing individual i
 * - tau_a, tau_b, tau_c are main effects for treatments A, B, C
 * - tau_ab, tau_ac, tau_bc are two-way interaction effects
 * - tau_abc is the three-way interaction effect
 */

data {

  // Sample size information
  int<lower=0> N_ED;                  // Number of clusters (emergency departments)
  int<lower=0> N;                     // Total number of patients across all clusters

  // Design matrix and outcomes
  matrix[N, 8] x_abc;                 // Design matrix: [1, A, B, C, AB, AC, BC, ABC]

  // Cluster and outcome assignments
  array[N] int<lower=1,upper=N_ED> ed;     // Cluster ID for each individual
  array[N] int<lower=0,upper=1> y;         // Binary outcome (0=failure, 1=success)

  // Prior hyperparameters (passed from R)
  array[6] real svals;                // Vector of prior scale parameters:
                                      // [1] sigma_sigma_m: scale for simga_m prior
                                      // [2] sigma_sigma_x: scale for simga_x prior
                                      // [3] sigma_tau_m: scale for tau_m prior
                                      // [4] sigma_tau_x: scale for tau_x prior
                                      // [5] sigma_sigma_ed: scale for sigma_alpha prior
                                      // [6] sigma_3: scale for 3-way interaction (unused)

}

parameters {

  // Non-centered parameterization for computational efficiency
  vector[8] z;                        // Standard normal draws for all coefficients
```

```
  // Hierarchical parameters for main effects (treatments A, B, C)
  real tau_m;                       // Common mean for main effects
  real<lower=1e-6> sigma_m;         // Standard deviation of main effects around tau_m

  // Hierarchical parameters for interaction effects (AB, AC, BC)
  real tau_x;                       // Common mean for 2-way interactions
  real<lower=1e-6> sigma_x;         // Standard deviation of 2-way interactions around tau_x

  // Cluster-level random effects
  vector[N_ED] ed_effect;           // Random intercepts for each cluster (sigma_j)
  real<lower=1e-6> sigma_ed;        // Standard deviation of cluster effects

}

transformed parameters {

  /*
   * Non-centered parameterization transformation
   *
   * Converts standard normal draws (z) into the actual regression coefficients.
   * This approach improves MCMC sampling efficiency by reducing correlation
   * between parameters and their scale parameters.
   *
   * Coefficient interpretation:
   * tau[1] = tau_0    : Intercept (reference: no treatments)
   * tau[2] = tau_a    : Main effect of treatment A
   * tau[3] = tau_b    : Main effect of treatment B
   * tau[4] = tau_c    : Main effect of treatment C
   * tau[5] = tau_ab   : Interaction effect A×B
   * tau[6] = tau_ac   : Interaction effect A×C
   * tau[7] = tau_bc   : Interaction effect B×C
   * tau[8] = tau_abc  : Three-way interaction A×B×C
   */

  vector[8] tau;                    // Final regression coefficients

  // Intercept: no hierarchical structure, just standard normal
  tau[1] = z[1];

  // Main effects (A, B, C): hierarchical structure with common mean tau_m
  for (i in 2:4) {
    tau[i] = sigma_m * z[i] + tau_m;
  }

  // Two-way interactions (AB, AC, BC): hierarchical structure with common mean tau_x
  for (i in 5:7) {
    tau[i] = sigma_x * z[i] + tau_x;
  }

  // Three-way interaction (ABC): fixed small scale, centered at zero
  // Note: 0.3 is a fixed scale parameter, could be made data-driven
  tau[8] = 0.3 * z[8];
```

```
}

model {

  /*
   * Prior Specifications
   *
   * The model uses a hierarchical prior structure that allows for:
   * 1. Different prior assumptions for main effects vs interactions
   * 2. Borrowing strength across similar effect types
   * 3. Robustness through Student-t distributions for scale parameters
   */

  // Priors for hierarchical scale parameters (Student-t for heavy tails)
  sigma_m ~ student_t(3, 0, svals[1]);   // Scale for main effect variation
  sigma_x ~ student_t(3, 0, svals[2]);   // Scale for interaction variation
  sigma_ed ~ student_t(3, 0, svals[5]);  // Scale for cluster variation

  // Priors for hierarchical location parameters
  tau_m ~ normal(0, svals[3]);           // Common mean for main effects
  tau_x ~ normal(0, svals[4]);           // Common mean for interactions

  // Non-centered parameterization: all z parameters are standard normal
  z ~ std_normal();

  // Cluster random effects
  ed_effect ~ normal(0, sigma_ed);

  /*
   * Likelihood
   *
   * Logistic regression with cluster random effects:
   * logit(P(y_i = 1)) = sigma_j[i] + X_i * tau
   *
   * where sigma_j[i] is the random effect for the cluster containing individual i
   * and X_i * tau represents the linear combination of treatment effects
   */
  y ~ bernoulli_logit(ed_effect[ed] + x_abc * tau);

}

generated quantities {

  /*
   * Treatment Effect Contrasts
   *
   * Computes log-odds ratios for each treatment combination relative to control.
   * These quantities facilitate interpretation and comparison of treatment effects.
   *
   * Each lOR represents the log-odds ratio comparing a specific treatment
   * combination to the control condition (no treatments).
   */
```

```
  array[7] real lOR;               // Log-odds ratios for treatment combinations

  // Single treatment effects (vs. control)
  lOR[1] = tau[2];                                          // A vs None
  lOR[2] = tau[3];                                          // B vs None
  lOR[3] = tau[4];                                          // C vs None

  // Two-way treatment combinations (vs. control)
  lOR[4] = tau[2] + tau[3] + tau[5];                        // AB vs None
  lOR[5] = tau[2] + tau[4] + tau[6];                        // AC vs None
  lOR[6] = tau[3] + tau[4] + tau[7];                        // BC vs None

  // Three-way treatment combination (vs. control)
  lOR[7] = tau[2] + tau[3] + tau[4] + tau[5] + tau[6] + tau[7] + tau[8];  // ABC vs None

  /*
   * Additional quantities that could be computed:
   *
   * // Convert to odds ratios
   * array[7] real OR;
   * for (i in 1:7) OR[i] = exp(lOR[i]);
   *
   * // Posterior predictive checks
   * array[N] int y_rep;
   * for (i in 1:N) {
   *   y_rep[i] = bernoulli_logit_rng(ed_effect[ed[i]] + x_abc[i] * tau);
   * }
   *
   * // Treatment effect comparisons (e.g., A vs B)
   * real lOR_A_vs_B = lOR[1] - lOR[2];
   */

}
```

model_HEX_ig.stan

```
/*
 * HEx Inverse Gamma Prior Model for Multi-Factorial Cluster Randomized Trial
 *
 * This Stan model implements an alternative hierarchical approach to the HEx model,
 * using inverse gamma priors on variance parameters instead of Student-t priors
 * on standard deviation parameters. This represents a different approach to
 * specifying prior beliefs about the scale of treatment effect heterogeneity.
 *
 * Key Differences from Standard HEx Model:
 * - Inverse gamma priors on variance parameters (sigma²) instead of Student-t on SD (sigma)
 * - Non-centered parameterization with variance-based transformation
 * - Direct variance estimation with derived standard deviations
 *
 * Model Structure (identical to HEx):
 * logit(P(y_i = 1)) = alpha_j[i] + tau_a*A_i + tau_b*B_i + tau_c*C_i +
 *                     tau_ab*A_i*B_i + tau_ac*A_i*C_i + tau_bc*B_i*C_i + tau_abc*A_i*B_i*C_i
 *
 * Prior Structure:
 * sigma²_main ~ InverseGamma(alpha_1, beta_1)     [Variance for main effects]
 * sigma²_int ~ InverseGamma(alpha_2, beta_2)      [Variance for interactions]
 * sigma²_cluster ~ InverseGamma(alpha_3, beta_3)  [Variance for cluster effects]
 *
 * Use Cases:
 * - When you have conjugate prior beliefs about variance parameters
 * - Sensitivity analysis with different prior families
 * - When inverse gamma priors better reflect domain knowledge
 * - Comparison with Student-t based approaches
 */

data {

  // Sample size information
  int<lower=0> N_ED;                  // Number of clusters (emergency departments)
  int<lower=0> N;                     // Total number of patients across all clusters

  // Design matrix and outcomes
  matrix[N, 8] x_abc;                 // Design matrix: [1, A, B, C, AB, AC, BC, ABC]

  // Cluster and outcome assignments
  array[N] int<lower=1,upper=N_ED> ed;     // Cluster ID for each individual
  array[N] int<lower=0,upper=1> y;         // Binary outcome (0=failure, 1=success)

  // Prior hyperparameters for inverse gamma distributions
  array[6] real svals;              // Vector of inverse gamma parameters:
                                    // [1] alpha_1: shape parameter for main effect variance
                                    // [2] beta_1: scale parameter for main effect variance
                                    // [3] sigma_taum: scale for main effect location prior
                                    // [4] sigma_taux: scale for interaction location prior
                                    // [5] alpha_2: shape parameter for cluster variance
                                    // [6] beta_2: scale parameter for cluster variance
                                    // Note: Uses same shape/scale for main & interaction variances
```

```
}

parameters {

  /*
   * Non-Centered Parameterization with Variance Parameters
   *
   * This approach parameterizes the model in terms of variances (sigma²) rather
   * than standard deviations (sigma), then derives SDs in transformed parameters.
   * This can have different computational properties and prior implications.
   */

  // Standard normal draws for non-centered parameterization
  vector[8] z;                      // Standard normal draws for all coefficients

  // Hierarchical location parameters (identical to HEx model)
  real tau_m;                       // Common mean for main effects
  real tau_x;                       // Common mean for interaction effects

  // Variance parameters (key difference from HEx model)
  real<lower=0> var_m;              // Variance of main effects around tau_m
  real<lower=0> var_x;              // Variance of interactions around tau_x
  real<lower=0> var_ed;             // Variance of cluster random effects

  // Cluster-level random effects
  vector[N_ED] ed_effect;           // Random intercepts for each cluster (alpha_j)

}

transformed parameters {

  /*
   * Variance-to-Standard Deviation Transformation
   *
   * Convert variance parameters to standard deviations for use in
   * the non-centered parameterization. This ensures computational
   * compatibility with the coefficient transformations.
   */

  // Derived standard deviations from variance parameters
  real<lower=0> sigma_m = sqrt(var_m);    // SD for main effects
  real<lower=0> sigma_x = sqrt(var_x);    // SD for interactions
  real<lower=0> sigma_ed = sqrt(var_ed);  // SD for cluster effects

  /*
   * Non-Centered Parameterization Transformation
   *
   * Identical to HEx model: converts standard normal draws into
   * properly scaled regression coefficients using hierarchical structure.
   */

  vector[8] tau;                    // Final regression coefficients
```

```
  // Intercept: just the standard normal draw
  tau[1] = z[1];

  // Main effects (A, B, C): hierarchical structure with variance-derived SD
  for (i in 2:4) {
    tau[i] = sigma_m * z[i] + tau_m;
  }

  // Two-way interactions (AB, AC, BC): hierarchical structure
  for (i in 5:7) {
    tau[i] = sigma_x * z[i] + tau_x;
  }

  // Three-way interaction (ABC): fixed scale (same as HEx model)
  tau[8] = 0.3 * z[8];

}

model {

  /*
   * Inverse Gamma Prior Specifications
   *
   * Key difference from HEx model: uses inverse gamma priors on variance
   * parameters rather than Student-t priors on standard deviations.
   *
   * Inverse Gamma Properties:
   * - Conjugate for normal likelihood variance parameters
   * - Right-skewed distribution (mass concentrated near zero)
   * - Mean = beta/(alpha-1) for alpha > 1
   * - Mode = beta/(alpha+1)
   * - Can be more/less informative than Student-t depending on parameters
   */

  // Inverse gamma priors on variance parameters
  // Note: Using target += for efficiency with log probability density
  target += inv_gamma_lpdf(var_m | svals[1], svals[2]);   // Main effect variance
  target += inv_gamma_lpdf(var_x | svals[1], svals[2]);   // Interaction variance
  target += inv_gamma_lpdf(var_ed | svals[5], svals[6]);  // Cluster variance

  // Normal priors on hierarchical location parameters (same as HEx)
  tau_m ~ normal(0, svals[3]);      // Common mean for main effects
  tau_x ~ normal(0, svals[4]);      // Common mean for interactions

  // Standard components (identical to HEx model)
  ed_effect ~ normal(0, sigma_ed);  // Cluster random effects
  z ~ std_normal();                 // Non-centered parameterization

  /*
   * Likelihood (identical to HEx model)
   */
  y ~ bernoulli_logit(ed_effect[ed] + x_abc * tau);
```

```
}

generated quantities {

  /*
   * Treatment Effect Contrasts (identical to other models)
   *
   * Despite different prior specification, the interpretation of
   * treatment effects remains the same across all model variants.
   */

  array[7] real lOR;               // Log-odds ratios for treatment combinations

  // Single treatment effects (vs. control)
  lOR[1] = tau[2];                                          // A vs None
  lOR[2] = tau[3];                                          // B vs None
  lOR[3] = tau[4];                                          // C vs None

  // Two-way treatment combinations (vs. control)
  lOR[4] = tau[2] + tau[3] + tau[5];                        // AB vs None
  lOR[5] = tau[2] + tau[4] + tau[6];                        // AC vs None
  lOR[6] = tau[3] + tau[4] + tau[7];                        // BC vs None

  // Three-way treatment combination (vs. control)
  lOR[7] = tau[2] + tau[3] + tau[4] + tau[5] + tau[6] + tau[7] + tau[8];  // ABC vs None

  /*
   * COMPARISON WITH OTHER MODELS:
   *
   * vs HEx Model (Student-t priors):
   * --------------------------------
   * Inverse Gamma Model:
   * + Conjugate priors may improve computational efficiency
   * + Natural for variance parameters
   * + Can encode strong beliefs about small variances
   * - Less robust to prior misspecification
   * - Right-skewed may not reflect uncertainty well
   *
   * HEx Model:
   * + More robust, heavy-tailed priors
   * + Student-t more commonly used in practice
   * + Better for weakly informative priors
   * - Non-conjugate (though rarely matters with MCMC)
   *
   * vs Simple Model:
   * ----------------
   * Inverse Gamma Model:
   * + Hierarchical structure borrows strength
   * + Separate priors for main effects vs interactions
   * + Better regularization with many parameters
   * - More complex, more hyperparameters
   * - Requires careful prior elicitation
   *
```

```
   * PRACTICAL CONSIDERATIONS:
   *
   * Choose Inverse Gamma Model when:
   * - You have strong prior beliefs about variance magnitudes
   * - You want conjugate prior structure
   * - Your domain suggests right-skewed variance priors
   * - You're comparing different prior families
   *
   * Prior Elicitation for Inverse Gamma:
   * - Shape alpha: controls concentration around mode
   * - Scale beta: controls location of distribution
   * - Higher alpha = more concentrated around mode
   * - Consider prior predictive checks to validate choices
   *
   * SENSITIVITY ANALYSIS:
   * Compare results across Student-t, inverse gamma, and simple models
   * to assess robustness of conclusions to prior specification.
   */

}
```

model_nonX.stan

```
/*
 * Simple (Non-exchangeable) Model for Multi-Factorial Cluster Randomized Trial
 *
 * This Stan model provides a simpler alternative to the hierarchical exchange (HEx)
 * model for analyzing a 2^3 factorial cluster randomized trial. Unlike the HEx model,
 * this approach treats all treatment effects as exchangeable with identical priors,
 * without distinguishing between main effects and interactions.
 *
 * Key Differences from HEx Model:
 * - No hierarchical structure for treatment effects
 * - All coefficients (main effects + interactions) have identical priors
 * - Simpler parameterization with fewer hyperparameters
 * - No non-centered parameterization needed
 *
 * Model Structure:
 * logit(P(y_i = 1)) = alpha_j[i] + tau_a*A_i + tau_b*B_i + tau_c*C_i +
 *                     tau_ab*A_i*B_i + tau_ac*A_i*C_i + tau_bc*B_i*C_i + tau_abc*A_i*B_i*C_i
 *
 *
 * Prior Structure:
 * tau_k ~ Normal(0, simga_tau) for k = a,...,abc, simga_tau is user-specified
 * alpha_j ~ Normal(0, simga_alpha) for j = 1,...,J , simga_alpha is user-specified
 *
 * Use Cases:
 * - When you want to treat all treatment effects as equally likely a priori
 * - Computational simplicity and faster sampling
 * - Sensitivity analysis comparing hierarchical vs non-hierarchical approaches
 * - When domain knowledge doesn't suggest different priors for main effects vs interactions
 */

data {

  // Sample size information
  int<lower=0> N_ED;                  // Number of clusters (emergency departments)
  int<lower=0> N;                     // Total number of patients across all clusters

  // Design matrix and outcomes
  matrix[N, 8] x_abc;                 // Design matrix: [1, A, B, C, AB, AC, BC, ABC]

  // Cluster and outcome assignments
  array[N] int<lower=1,upper=N_ED> ed;     // Cluster ID for each individual
  array[N] int<lower=0,upper=1> y;         // Binary outcome (0=failure, 1=success)

  // Prior hyperparameters (passed from R)
  array[6] real svals;               // Vector of prior scale parameters:
                                     // [1-4] unused (for compatibility with HEx model)
                                     // [5] sigma_sigma_ed: scale for simga_alpha prior (cluster effects)
                                     // [6] sigma_tau: scale for treatment effect priors

}
```

```
parameters {

  /*
   * Direct Parameterization (vs Non-Centered in HEx Model)
   *
   * This model uses direct parameterization which is simpler but may be
   * less computationally efficient than the non-centered approach in the HEx model.
   * However, for this relatively simple structure, the efficiency difference
   * is likely minimal.
   */

  // Treatment effect coefficients (all treated identically)
  vector[8] tau;                    // [tau_0, tau_1, tau_2, tau_3, tau_4, tau_5, tau_6, tau_7]
                                    // tau_0: Intercept
                                    // tau_1-tau_3: Main effects (A, B, C)
                                    // tau_4-tau_6: Two-way interactions (AB, AC, BC)
                                    // tau_7: Three-way interaction (ABC)

  // Cluster-level random effects (identical to HEx model)
  vector[N_ED] ed_effect;           // Random intercepts for each cluster (alpha_j)
  real<lower=0> sigma_ed;           // Standard deviation of cluster effects

}

model {

  /*
   * Prior Specifications
   *
   * This model uses a much simpler prior structure compared to the HEx model:
   * - All treatment effects have identical Normal(0, simga_tau) priors
   * - No distinction between main effects and interactions
   * - No hierarchical borrowing of strength across effect types
   */

  // Cluster random effects (same as HEx model)
  ed_effect ~ normal(0, sigma_ed);
  sigma_ed ~ student_t(3, 0, svals[5]);  // Scale for cluster variation

  // Treatment effects: all coefficients treated identically
  // Note: This includes the intercept tau[1], which gets Normal(0, simga_tau) prior
  tau ~ normal(0, svals[6]);             // Common prior for all treatment effects

  /*
   * Likelihood (identical to HEx model)
   *
   * Logistic regression with cluster random effects:
   * logit(P(y_i = 1)) = alpha_j[i] + X_i * tau
   */
  y ~ bernoulli_logit(ed_effect[ed] + x_abc * tau);

}
```

```
generated quantities {

  /*
   * Treatment Effect Contrasts (identical to HEx model)
   *
   * Computes log-odds ratios for each treatment combination relative to control.
   * The interpretation is identical to the HEx model, but the underlying
   * coefficients come from different prior structures.
   */

  array[7] real lOR;               // Log-odds ratios for treatment combinations

  // Single treatment effects (vs. control)
  lOR[1] = tau[2];                                          // A vs None
  lOR[2] = tau[3];                                          // B vs None
  lOR[3] = tau[4];                                          // C vs None

  // Two-way treatment combinations (vs. control)
  lOR[4] = tau[2] + tau[3] + tau[5];                        // AB vs None
  lOR[5] = tau[2] + tau[4] + tau[6];                        // AC vs None
  lOR[6] = tau[3] + tau[4] + tau[7];                        // BC vs None

  // Three-way treatment combination (vs. control)
  lOR[7] = tau[2] + tau[3] + tau[4] + tau[5] + tau[6] + tau[7] + tau[8];  // ABC vs None

  /*
   * COMPARISON WITH HEx MODEL:
   *
   * Advantages of Simple Model:
   * - Fewer hyperparameters to specify
   * - Simpler to understand and explain
   * - Faster computation
   * - Less risk of prior specification issues
   *
   * Advantages of HEx Model:
   * - Can incorporate domain knowledge about effect hierarchies
   * - Borrows strength across similar effect types
   * - More flexible prior structure
   * - Better computational properties (non-centered parameterization)
   *
   * PRACTICAL CONSIDERATIONS:
   *
   * Choose Simple Model when:
   * - You have no strong prior beliefs about effect hierarchies
   * - You want computational simplicity
   * - You're doing sensitivity analysis
   * - Sample size is adequate for estimating all effects
   *
```

prior_predictive_model.stan

```
/*
 * Prior Predictive Model for Multi-Factorial Cluster Randomized Trial
 *
 * This Stan model generates samples from the prior predictive distribution
 * for a 2^3 factorial cluster randomized trial. It's used for prior checking
 * to ensure that prior specifications lead to reasonable predicted outcomes
 * before fitting the model to real data.
 *
 * Purpose:
 * - Validate prior assumptions by examining implied outcome distributions
 * - Check if priors allow for clinically plausible treatment effects
 * - Identify potential issues with prior specifications
 * - Guide prior elicitation through visualization of implied predictions
 *
 * Workflow:
 * 1. Sample all model parameters from their prior distributions
 * 2. Generate predicted outcomes (y_rep) for each observation
 * 3. Analyze distributions of predicted probabilities by treatment arm
 *
 * This model mirrors the main analysis model but operates in "prior-only" mode.
 */

data {

  // Sample size information (same as main model)
  int<lower=0> N_ED;                  // Number of clusters (emergency departments)
  int<lower=0> N;                     // Total number of patients

  // Design structure (same as main model)
  matrix[N, 8] x_abc;                 // Design matrix: [1, A, B, C, AB, AC, BC, ABC]
  array[N] int<lower=1,upper=N_ED> ed;     // Cluster assignment for each individual
  array[N] int<lower=0,upper=1> y;         // Observed outcomes (unused in prior predictive)

  // Prior hyperparameters (explicitly named for clarity)
  real sigma_tau_m;                   // Prior SD for main effect location parameters
  real sigma_tau_x;                   // Prior SD for interaction location parameters
  real sigma_sigma_m;                 // Prior scale for main effect variation
  real sigma_sigma_x;                 // Prior scale for interaction variation
  real sigma_3;                       // Prior scale for three-way interaction
  real sigma_sigma_ed;                // Prior scale for cluster effect variation

}

generated quantities {

  /*
   * STEP 1: Sample Hierarchical Location Parameters
   *
   * These represent the "typical" or "average" treatment effects
   * around which individual effects vary.
   */
```

```
// Common mean for main effects (A, B, C)
real tau_m = normal_rng(0, sigma_tau_m);

// Common mean for two-way interactions (AB, AC, BC)
real tau_x = normal_rng(0, sigma_tau_x);

/*
 * STEP 2: Sample Hierarchical Scale Parameters
 *
 * These control how much individual effects vary around their common means.
 * Using while loops ensures positive values since Student-t can be negative.
 * This is equivalent to half-Student-t priors but implemented via rejection sampling.
 */

// Scale parameter for main effect variation
real sigma_m = -1;
while (sigma_m < 0) {
  sigma_m = student_t_rng(3, 0, sigma_sigma_m);
}

// Scale parameter for interaction effect variation
real sigma_x = -1;
while (sigma_x < 0) {
  sigma_x = student_t_rng(3, 0, sigma_sigma_x);
}

// Scale parameter for cluster random effects
real sigma_ed = -1;
while (sigma_ed < 0) {
  sigma_ed = student_t_rng(3, 0, sigma_sigma_ed);
}

/*
 * STEP 3: Sample Cluster Random Effects
 *
 * Each cluster gets its own random intercept representing
 * cluster-specific factors affecting baseline risk.
 */

array[N_ED] real ed_effect;
for (i in 1:N_ED) {
  ed_effect[i] = normal_rng(0, sigma_ed);
}

/*
 * STEP 4: Sample Standard Normal Draws for Non-Centered Parameterization
 *
 * These will be transformed into the actual regression coefficients
 * using the hierarchical structure.
 */

vector[8] z;
for (i in 1:8) {
```

```
  z[i] = normal_rng(0, 1);
}

/*
 * STEP 5: Transform to Final Regression Coefficients
 *
 * This mirrors the transformed parameters block in the main model,
 * converting standard normal draws into properly scaled coefficients.
 */

vector[8] tau;                    // Final regression coefficients

// Intercept: just the standard normal draw
tau[1] = z[1];

// Main effects (A, B, C): hierarchical structure
for (i in 2:4) {
  tau[i] = sigma_m * z[i] + tau_m;
}

// Two-way interactions (AB, AC, BC): hierarchical structure
for (i in 5:7) {
  tau[i] = sigma_x * z[i] + tau_x;
}

// Three-way interaction (ABC): directly scaled (no hierarchical mean)
tau[8] = sigma_3 * z[8];

/*
 * STEP 6: Compute Treatment Effect Contrasts
 *
 * Calculate log-odds ratios for each treatment combination
 * relative to the control condition. These are the key quantities
 * of interest for understanding treatment effects.
 */

array[7] real lOR;                // Log-odds ratios vs control

// Single treatments vs control
lOR[1] = tau[2];                                          // A vs None
lOR[2] = tau[3];                                          // B vs None
lOR[3] = tau[4];                                          // C vs None

// Two-way combinations vs control
lOR[4] = tau[2] + tau[3] + tau[5];                        // AB vs None
lOR[5] = tau[2] + tau[4] + tau[6];                        // AC vs None
lOR[6] = tau[3] + tau[4] + tau[7];                        // BC vs None

// Three-way combination vs control
lOR[7] = tau[2] + tau[3] + tau[4] + tau[5] + tau[6] + tau[7] + tau[8];  // ABC vs None

/*
 * STEP 7: Generate Prior Predictive Outcomes
```

```
   *
   * For each observation, compute the linear predictor and generate
   * a binary outcome from the implied Bernoulli distribution.
   * These y_rep values show what outcomes the prior expects to see.
   */

  array[N] int y_rep;               // Prior predictive outcomes
  array[N] real linpred;            // Linear predictors (for debugging/analysis)

  for (n in 1:N) {
    // Linear predictor: cluster effect + treatment effects
    linpred[n] = ed_effect[ed[n]] + dot_product(row(x_abc, n), tau);

    // Generate binary outcome from logistic model
    y_rep[n] = bernoulli_logit_rng(linpred[n]);
  }

  /*
   * INTERPRETATION OF OUTPUTS:
   *
   * y_rep: Shows what outcomes we expect to see under the prior
   *        - Can compute treatment arm-specific success rates
   *        - Should span reasonable clinical ranges
   *        - Extreme values may indicate problematic priors
   *
   * lOR: Shows the range of treatment effects implied by priors
   *      - Values near 0 suggest small effects
   *      - Large absolute values suggest strong effects
   *      - Distribution should match clinical expectations
   *
   * linpred: Linear predictors on log-odds scale
   *          - Can help diagnose numerical issues
   *          - Extreme values may cause computational problems
   *
   * PRIOR CHECKING WORKFLOW:
   * 1. Run this model with candidate prior specifications
   * 2. Examine distributions of y_rep by treatment arm
   * 3. Check if implied success rates are clinically reasonable
   * 4. Adjust priors if predictions are unrealistic
   * 5. Iterate until satisfied with prior implications
   */

}
```