Optimizing Model Performance under Budget Constraints: A Simulation Study on the Impact of Cluster Size and Observations per Cluster

Michael Stewart

Abstract

This study evaluates the performance of mixed-effects models in hierarchical settings with varying numbers of clusters, observations per cluster, treatment effects, and intra-cluster correlation (ICC). Simulated data was used to investigate how these factors influence the variance of the estimated treatment effect $\widehat{\beta}$ and the coverage of confidence intervals. Larger numbers of clusters and observations per cluster generally resulted in more precise estimates, though the improvements leveled off after certain thresholds. The study also extended to a Poisson distribution with a log link, which showed higher variability in the estimates compared to the normal model, particularly for smaller treatment effects. Additionally, an optimization was performed under a fixed budget to identify designs that minimized the variance of $\widehat{\beta}$. Moderate numbers of clusters and observations per cluster provided the best precision while adhering to budget constraints.

Introduction

Cluster randomized trials (CRTs) assign entire groups, such as schools or clinics, to either treatment or control groups. This design is useful when interventions are best delivered at the group level or when individual randomization is difficult. A key challenge in CRTs is intra-cluster correlation (ICC), where outcomes within the same group are similar. High ICC can increase the variance of treatment effect estimates, affecting the study's accuracy.

This project, in collaboration with Dr. Zhijin Wu, aims to optimize CRT design under budget constraints. We seek to determine the optimal number of clusters and the number of observations per cluster to accurately estimate the treatment effect on the outcome variable Y within a budget B. The first sample from a cluster costs c1, while additional samples cost c2, with

c2 being less than c1. Although samples within the same cluster are cheaper, they must be assigned to the same treatment or control group, and their measurements may be correlated.

Using simulation studies, we will explore various design configurations to assess how the number of clusters, observations per cluster, ICC, within-cluster variance (σ^2), and effect size (β) influence the estimation of the treatment effect. Our goal is to identify design strategies that minimize variance while adhering to budget constraints, ensuring that resources are used efficiently and the study results are reliable.

Methods

Data Simulation

The simulation study aimed to evaluate the performance of mixed-effects models in hierarchical data settings, focusing on the variance of the estimated treatment effect ($\hat{\beta}$) under different simulation parameters. The simulated data followed a hierarchical structure where multiple clusters each contained several observations. The outcome for each observation was generated from a model that incorporated both fixed and random effects:

$$y_{ij} = \alpha + \beta \cdot X_{ij} + u_i + \epsilon_{ij}$$

In this model, (y_{ij}) represents the outcome for the (j)-th observation in the (i)-th cluster, (α) is the fixed intercept, (β) is the fixed treatment effect, and (X_{ij}) indicates the treatment assignment. The random effect (u_i) represents the cluster-specific deviation, drawn from a normal distribution ($u_i \sim N(0, \gamma^2)$), and (ϵ_{ij}) is the residual error, distributed as ($\epsilon_{ij} \sim N(0, \sigma^2)$). The treatment allocation was fixed with equal groups within each cluster.

Simulation Parameters

The simulation varied several parameters across multiple runs to assess the model's performance under different conditions. These parameters included the number of clusters (g), ranging from 10 to 100, the number of observations per cluster (r), ranging from 10 to 500, the intra-cluster correlation (ICC), with values of 0.05, 0.1, and 0.2, within-cluster variance (σ^2), which was set to 1, 2, and 3, and the treatment effect (β), which took values of 0.2, 0.5, and 0.8.

Model Fitting and Performance Evaluation

The performance of the mixed-effects model was assessed by focusing on four key metrics: bias, mean squared error (MSE), standard error (SE), and coverage probability. Bias measured the difference between the estimated treatment effect and the true treatment effect (β). The MSE combined the bias and variance of the estimates. SE reflected the precision of the estimated treatment effect, while coverage probability calculated the proportion of simulations in which the true treatment effect was within the 95% confidence interval of the estimated treatment effect.

To estimate the treatment effect, we used a mixed-effects model fitted with the **lme4** package in R. This model included a fixed effect for treatment and a random intercept for clusters, where the outcome followed a normal distribution with the treatment effect and random cluster effects. Specifically, the model was:

$$y\_ij \sim \text{Normal}(\alpha + \beta \cdot X_{ij}, \sigma^2 + \gamma^2)$$

Extension to Poisson Distribution (Aim 3)

For Aim 3, the simulation study was extended to account for count data following a Poisson distribution. In this case, the hierarchical model was adjusted to incorporate a log link to model the expected count (μ_i) for each cluster (i). The model for the Poisson-distributed outcome was as follows:

$$\log(\mu_i) = \alpha + \beta \cdot X_i + u_i$$

Where μ_i represents the expected count for cluster i and (Y_{ij}) is the (j)-th observation within that cluster. The total count per cluster (i) was obtained by summing the individual observations:

$$Y_i = \sum_{j=1}^{R} Y_{ij}$$

Since the sum of independent Poisson random variables remains Poisson-distributed, we modeled the total count for each cluster as ($Y_i \sim \text{Poisson}(R \cdot \mu_i)$), with R representing the number of observations per cluster.

Optimization with Fixed Budget

In addition to simulating data under a normal distribution, we aimed to identify experimental designs that minimized the variance of the estimated treatment effect ($\hat{\beta}$) while adhering to a fixed budget. The budget constraint was set to \$2000, and the design was optimized by considering the cost of clusters and observations. The total cost was calculated based on a cost structure that included the cost per cluster (c_1) and the cost per observation (c_2). We evaluated the designs using a variance formula for the estimated treatment effect, aiming to find designs that minimized the variance of the treatment effect estimate while respecting the budget. The formula for the variance of ($\hat{\beta}$) was:

$$\text{Var}(\hat{\beta}) = \frac{\sigma^2 \cdot (1 + (r-1) \cdot \text{ICC})}{g \cdot r \cdot 0.5^2}$$

Where (g) represents the number of clusters, (r) is the number of observations per cluster, and ICC is the intra-cluster correlation. Through this optimization, we sought designs that balanced the cost constraints with the precision of the treatment effect estimate.

Simulation Results

The simulation results show the performance of the model across varying cluster sizes (g) and observations per cluster (r), using average bias, mean squared error (MSE), standard error (SE), and coverage as metrics. The bias is generally small across all combinations, suggesting that the model is mostly unbiased in estimating the treatment effect ( ), regardless of the cluster size or number of observations. However, the bias fluctuates slightly between positive and negative values, indicating minor deviations around zero. The mean squared error (MSE), which accounts for both bias and variance, tends to decrease as the number of clusters and observations per cluster increases. The smallest MSE values are observed when there are 100 clusters with 500 observations per cluster, reflecting more accurate and stable estimates. This reduction in MSE with larger clusters and more observations per cluster points to the importance of larger sample sizes for better model performance.

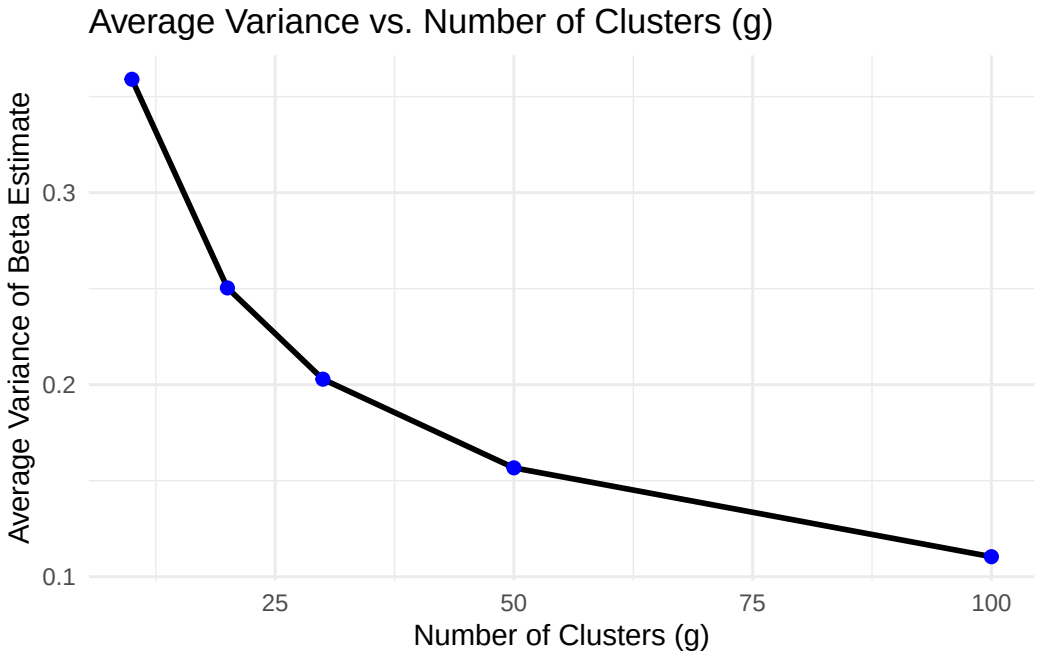
Similarly, the standard error (SE), which measures the precision of the treatment effect estimate, decreases as the cluster size and observations per cluster increase. With more clusters and observations, the model's estimates become more precise, as evidenced by the lower SE values, particularly when there are 100 clusters and 500 observations per cluster. Finally, the coverage rate, which indicates how often the true treatment effect is within the model's confidence intervals, generally stays close to the nominal 95% level across all combinations. However, the coverage rate is slightly lower when the number of observations per cluster is small, such as with 10 observations per cluster, but improves with larger sample sizes.

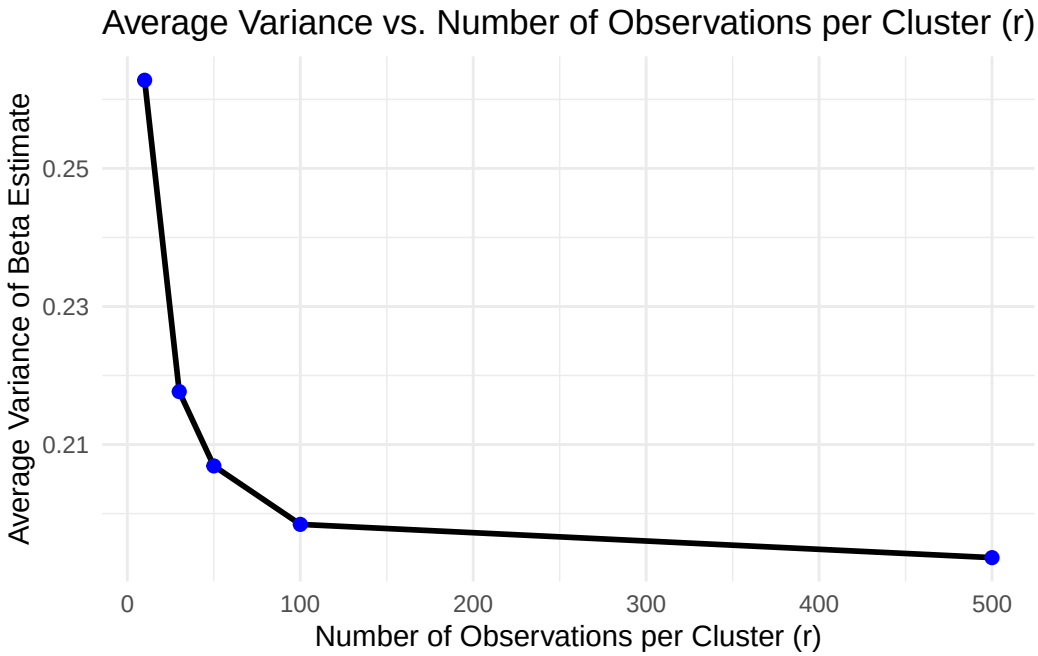
Overall, the results indicate that increasing the number of clusters and observations per cluster improves model performance by reducing bias, MSE, and SE, while maintaining stable coverage

rates. The smaller sample sizes, such as 10 clusters with 10 observations per cluster, tend to result in higher variability in the estimates, leading to higher MSE and SE values. To achieve more accurate, precise, and stable estimates, it is recommended to increase the number of clusters and observations.

Simulation Results Summary by Cluster Size and Observations per Cluster

		Performance Metrics			
Number of Clusters	Observations per Cluster	Average Bias	Average MSE	Average SE	Average Covera
10	10	0.002	0.441	0.437	0.9
10	30	0.045	0.295	0.363	0.9
10	50	−0.028	0.274	0.342	0.9
10	100	−0.010	0.271	0.326	0.9
10	500	0.005	0.255	0.327	0.9
20	10	0.005	0.195	0.306	0.9
20	30	−0.010	0.156	0.251	0.9
20	50	−0.003	0.140	0.241	0.9
20	100	−0.024	0.137	0.234	0.9
20	500	0.009	0.113	0.221	0.9
30	10	0.008	0.137	0.247	0.9
30	30	−0.010	0.094	0.205	0.9
30	50	0.015	0.089	0.194	0.9
30	100	0.008	0.087	0.187	0.9
30	500	0.003	0.068	0.182	0.9
50	10	0.000	0.079	0.190	0.9
50	30	0.000	0.057	0.158	0.9
50	50	0.001	0.050	0.151	0.9
50	100	−0.001	0.045	0.144	0.9
50	500	−0.003	0.043	0.139	0.9
100	10	0.010	0.038	0.133	0.9
100	30	−0.003	0.029	0.111	0.9
100	50	0.004	0.024	0.106	0.9
100	100	0.000	0.023	0.102	0.9
100	500	0.006	0.023	0.100	0.9





Optimization Results

Summary of Metrics for Optimal Configurations

Optimization Metrics

Clusters (g)	Observations in Cluster (r)	Total Cost	Var(Beta Hat)	Gamma Squared	ICC	C1:C2 R
100	10	1,010.000	0.007	0.100	0.091	0
100	10	1,050.000	0.007	0.100	0.091	0
100	10	1,100.000	0.007	0.100	0.091	1
100	10	1,200.000	0.007	0.100	0.091	2
100	10	1,500.000	0.007	0.100	0.091	5
100	10	2,000.000	0.007	0.100	0.091	10
60	30	1,806.000	0.011	0.100	0.048	0
60	30	1,830.000	0.011	0.100	0.048	0
60	30	1,860.000	0.011	0.100	0.048	1
60	30	1,920.000	0.011	0.100	0.048	2
100	10	1,500.000	0.011	0.100	0.048	5
100	10	2,000.000	0.011	0.100	0.048	10
60	30	1,806.000	0.013	0.100	0.032	0
60	30	1,830.000	0.013	0.100	0.032	0
60	30	1,860.000	0.013	0.100	0.032	1
60	30	1,920.000	0.013	0.100	0.032	2
100	10	1,500.000	0.015	0.100	0.032	5
100	10	2,000.000	0.015	0.100	0.032	10
100	10	1,010.000	0.016	0.500	0.333	0
100	10	1,050.000	0.016	0.500	0.333	0
100	10	1,100.000	0.016	0.500	0.333	1
100	10	1,200.000	0.016	0.500	0.333	2
100	10	1,500.000	0.016	0.500	0.333	5
100	10	2,000.000	0.016	0.500	0.333	10
100	10	1,010.000	0.022	0.500	0.200	0
100	10	1,050.000	0.022	0.500	0.200	0
100	10	1,100.000	0.022	0.500	0.200	1
100	10	1,200.000	0.022	0.500	0.200	2
100	10	1,500.000	0.022	0.500	0.200	5
100	10	2,000.000	0.022	0.500	0.200	10
100	10	1,010.000	0.027	0.500	0.143	0
100	10	1,050.000	0.027	0.500	0.143	0
100	10	1,100.000	0.027	0.500	0.143	1
100	10	1,200.000	0.027	0.500	0.143	2
100	10	1,500.000	0.027	0.500	0.143	5
100	10	2,000.000	0.027	0.500	0.143	10
100	10	1,010.000	0.022	1.000	0.500	0
100	10	1,050.000	0.022	1.000	0.500	0
100	10	1,100.000	0.022	1.000	0.500	1
100	10	1,200.000	0.022	1.000	0.500	2

100	10	1,500.000	0.022	1.000	0.500	5
100	10	2,000.000	0.022	1.000	0.500	10
100	10	1,010.000	0.032	1.000	0.333	0
100	10	1,050.000	0.032	1.000	0.333	0
100	10	1,100.000	0.032	1.000	0.333	1
100	10	1,200.000	0.032	1.000	0.333	2
100	10	1,500.000	0.032	1.000	0.333	5
100	10	2,000.000	0.032	1.000	0.333	10
100	10	1,010.000	0.039	1.000	0.250	0
100	10	1,050.000	0.039	1.000	0.250	0
100	10	1,100.000	0.039	1.000	0.250	1
100	10	1,200.000	0.039	1.000	0.250	2
100	10	1,500.000	0.039	1.000	0.250	5
100	10	2,000.000	0.039	1.000	0.250	10

The table summarizes the optimal configurations for a cluster-based study design based on cost, variance, and variability metrics. The number of clusters (g) and observations per cluster (r) determine how resources are allocated between groups and individuals. Total cost represents the budget for each configuration, while Var(Beta Hat) reflects the precision of the target parameter estimate. Lower values of Var(Beta Hat) indicate more precise estimates.

When the intraclass correlation (ICC) and between-group variability (Gamma Squared) are low, designs with fewer clusters and larger cluster sizes are more efficient. These configurations prioritize within-group sampling to reduce individual-level variability. For example, with g = 60 clusters and r = 30 observations per cluster, the configuration achieves reasonable precision (Var(Beta Hat) = 0.011) at a total cost of $1,920.

When ICC and Gamma Squared are higher, designs with more clusters and smaller cluster sizes become optimal. These configurations capture variability between groups and improve precision. For instance, with g = 100 clusters and r = 10 observations per cluster, the design achieves a low Var(Beta Hat) = 0.007 at a total cost of $1,010.

The trade-off between cost and precision depends on the study's goals and the relative importance of between-group and within-group variability. Configurations with higher C1:C2 Ratios indicate greater costs for sampling between groups, favoring fewer groups and larger clusters. Conversely, lower ratios suggest designs with more groups are feasible.

Conclusion and Limitations

The simulation results demonstrate that increasing the number of clusters (g) and the number of observations per cluster (r) generally leads to improved model performance. Specifically, the average bias remains close to zero across all combinations, indicating that the model is mostly unbiased. The mean squared error (MSE) decreases with larger cluster sizes and observations

per cluster, reflecting more accurate and stable estimates. Similarly, the standard error (SE) decreases with more clusters and observations, showing that the precision of the treatment effect estimates improves as sample size increases. The coverage rate is generally close to the nominal 95% level across all combinations, although it slightly decreases when the number of observations per cluster is small.

Overall, the results suggest that to achieve more precise, accurate, and stable estimates of the treatment effect, increasing the number of clusters and observations per cluster is beneficial. Smaller sample sizes, such as 10 clusters with 10 observations per cluster, lead to higher variability in the estimates, which results in higher MSE and SE values.

There are several limitations to this analysis. First, the simulation was based on specific assumptions about the distribution of the data and treatment allocation, which may not fully capture the complexities of real-world dat

Additionally, due to runtime constraints, we were unable to extend the simulation to include the Poisson distribution, which is an important area for future work. Incorporating Poisson-distributed outcomes would allow for a deeper understanding of how different distributions impact model performance. Finally, while this analysis focused on varying cluster size and observations per cluster, other factors—such as the nature of the random effects or treatment allocation mechanisms—were not explored. Future work should consider these additional factors to provide a more comprehensive understanding of the model's performance in different settings.

```
knitr::opts_chunk$set(warning = FALSE,
                      message = FALSE,
                      echo = FALSE,
                      fig.align="center")


library(lme4)
library(tidyverse)
library(gt)


set.seed(1)

# --------------------------------
# data simulation functions
# --------------------------------

# simulate data with fixed allocation and varying icc and sigma_sq
simulate_normal_data <- function(g, r, alpha, beta, gamma_sq, sigma_sq) {
```

```r
  # random effects for clusters
  cluster_effects <- rnorm(g, mean = 0, sd = sqrt(gamma_sq))

  # equal allocation to treatment
  treatment <- rbinom(g, 1, 0.5)

  # create dataframe
  data <- data.frame(
    cluster = rep(1:g, each = r),
    treatment = rep(treatment, each = r)
  )

  # generate outcomes
  data$y <- alpha + beta * data$treatment +
    rep(cluster_effects, each = r) +
    rnorm(n = g * r, mean = 0, sd = sqrt(sigma_sq))

  return(data)
}

# fit mixed-effects model
fit_normal_model <- function(data) {

  # fit the model
  model <- lmer(y ~ treatment + (1 | cluster), data = data)

  return(model)
}

# evaluate model performance
evaluate_performance <- function(model, true_beta, confidence_level = 0.95) {
  fixef_model <- fixef(model)

  if ("treatment" %in% names(fixef_model)) {
    beta_estimate <- fixef_model["treatment"]
    vcov_matrix <- vcov(model)

    if ("treatment" %in% rownames(vcov_matrix)) {
      se <- sqrt(vcov_matrix["treatment", "treatment"])
      bias <- beta_estimate - true_beta
      mse <- bias^2 + se^2
```

```
      # Calculate Wald confidence interval
      alpha <- 1 - confidence_level
      z <- qnorm(1 - alpha / 2)
      lower_ci <- beta_estimate - z * se
      upper_ci <- beta_estimate + z * se

      # Check coverage
      coverage <- ifelse(true_beta >= lower_ci & true_beta <= upper_ci, 1, 0)
    } else {
      se <- NA
      bias <- beta_estimate - true_beta
      mse <- NA
      coverage <- NA
    }
  } else {
    beta_estimate <- NA
    bias <- NA
    mse <- NA
    se <- NA
    coverage <- NA
  }

  return(data.frame(
    beta_estimate = beta_estimate,
    bias = bias,
    mse = mse,
    se = se,
    coverage = coverage
  ))
}

# ------------------------------
# simulation runner function
# ------------------------------

# run simulations
run_simulations <- function(g_values, r_values, ICC_values, sigma_sq_values, beta_values, n_s
  results <- data.frame()

  for (g in g_values) {
    for (r in r_values) {
      for (ICC in ICC_values) {
```

```
    for (sigma_sq in sigma_sq_values) {
      for (beta in beta_values) {
        alpha <- 0  # fixed intercept

        # store metrics
        biases <- numeric(n_sim)
        mses <- numeric(n_sim)
        ses <- numeric(n_sim)
        coverages <- numeric(n_sim)

        for (i in 1:n_sim) {
          # calculate gamma_sq from ICC and sigma_sq
          gamma_sq <- (ICC * sigma_sq) / (1 - ICC)
          data <- simulate_normal_data(g, r, alpha, beta, gamma_sq, sigma_sq)

          # fit model
          model <- fit_normal_model(data)

          # evaluate
          perf <- evaluate_performance(model, beta)

          biases[i] <- perf$bias
          mses[i] <- perf$mse
          ses[i] <- perf$se
          coverages[i] <- perf$coverage
        }

        # average metrics
        mean_bias <- mean(biases, na.rm = TRUE)
        mean_mse <- mean(mses, na.rm = TRUE)
        mean_se <- mean(ses, na.rm = TRUE)
        mean_coverage <- mean(coverages, na.rm = TRUE)

        # save results
        results <- rbind(results, data.frame(
          model = "Normal",
          g = g,
          r = r,
          ICC = ICC,
          sigma_sq = sigma_sq,
          beta = beta,
          mean_bias = mean_bias,
```

```
              mean_mse = mean_mse,
              mean_se = mean_se,
              mean_coverage = mean_coverage
            ))
          }
        }
      }
    }
  }

  return(results)
}

# ------------------------------
# define realistic parameter values
# ------------------------------

#define parameters
g_values <- c(10, 20, 30, 50, 100)          # number of clusters
r_values <- c(10, 30, 50, 100, 500)        # observations per cluster
ICC_values <- c(0.05, 0.1, 0.2)            # intra-cluster correlation
sigma_sq_values <- c(1, 2, 3)              # within-cluster variance
beta_values <- c(0.2, 0.5, 0.8)            # effect sizes

# ------------------------------
# run simulations
# ------------------------------

# run the simulations
simulation_results <- run_simulations(
  g_values = g_values,
  r_values = r_values,
  ICC_values = ICC_values,
  sigma_sq_values = sigma_sq_values,
  beta_values = beta_values,
  n_sim = 10  # number of simulations
)

# save results
write.csv(simulation_results, file = "simulation_results_normal.csv", row.names = FALSE)

# ------------------------------
```

```r
# load and prepare simulation results
# ------------------------------

# load the results
all_results <- read.csv("simulation_results_normal.csv")

# prepare for plotting
all_results <- all_results %>%
  mutate(
    g = as.factor(g),
    r = as.factor(r),
    ICC = as.factor(ICC),
    sigma_sq = as.factor(sigma_sq),
    beta = as.factor(beta)
  )

# ------------------------------
# optimization with fixed budget
# ------------------------------

# set fixed budget
fixed_budget <- 2000

# define cost ratios and compute c1
c1_c2_ratios <- c(0.1, 0.5, 1, 2, 5, 10)
c2 <- 1  # cost per observation

# create design grid
g_values_opt <- seq(10, 100, by = 10)    # clusters
r_values_opt <- seq(10, 500, by = 20)    # observations per cluster
design_grid <- expand.grid(g = g_values_opt, r = r_values_opt)

# store optimization results
optimization_results <- data.frame()

# loop through parameters with fixed budget
for (gamma_sq in c(0.1, 0.5, 1)) {
  for (sigma_sq in c(1, 2, 3)) {
    # compute ICC
    ICC <- gamma_sq / (gamma_sq + sigma_sq)

    for (c1_c2_ratio in c1_c2_ratios) {
```

```
      c1 <- c1_c2_ratio * c2

      # calculate variance and cost
      design_grid_opt <- design_grid %>%
        mutate(
          var_beta_hat = (sigma_sq * (1 + (r - 1) * ICC)) / (g * r * 0.5 * 0.5),
          total_cost = g * (c1 + c2 * r)
        ) %>%
        filter(total_cost <= fixed_budget)

      # find optimal design
      if (nrow(design_grid_opt) > 0) {
        optimal_design <- design_grid_opt %>%
          arrange(var_beta_hat) %>%
          slice(1) %>%
          mutate(
            gamma_sq = gamma_sq,
            sigma_sq = sigma_sq,
            ICC = ICC,
            c1_c2_ratio = c1_c2_ratio,
            C_max = fixed_budget,
            alloc_ratio = 0.5  # fixed allocation
          )

        # add to results
        optimization_results <- bind_rows(optimization_results, optimal_design)
      }
    }
  }
}

# save optimization results
write.csv(optimization_results, file = "optimization_results.csv", row.names = FALSE)
```

```
# Aggregate over all variables except g and r
summary_by_g_r <- simulation_results %>%
```

```
  group_by(g, r) %>%
  summarise(
    mean_bias = mean(mean_bias, na.rm = TRUE),
    mean_mse = mean(mean_mse, na.rm = TRUE),
    mean_se = mean(mean_se, na.rm = TRUE),
    mean_coverage = mean(mean_coverage, na.rm = TRUE),
    .groups = 'drop'
  )

summary_table_gt <- summary_by_g_r %>%
  gt() %>%
  tab_header(
    title = "Simulation Results Summary by Cluster Size and Observations per Cluster"
  ) %>%
  cols_label(
    g = "Number of Clusters",
    r = "Observations per Cluster",
    mean_bias = "Average Bias",
    mean_mse = "Average MSE",
    mean_se = "Average SE",
    mean_coverage = "Average Coverage"
  ) %>%
  tab_spanner(
    label = "Performance Metrics",
    columns = vars(mean_bias, mean_mse, mean_se, mean_coverage)
  ) %>%
  fmt_number(
    columns = vars(mean_bias, mean_mse, mean_se, mean_coverage),
    decimals = 3
  )


summary_table_gt

# Aggregate data to calculate average variance across ICC levels
agg_results_g <- simulation_results %>%
  group_by(g) %>%
  summarise(
    avg_variance = mean(mean_se, na.rm = TRUE)
  )

avg_variance_g <- optimization_results %>%
```

```r
  group_by(g) %>%
  summarise(avg_variance = mean(var_beta_hat, na.rm = TRUE))

# average Variance vs. g
ggplot(agg_results_g, aes(x = as.numeric(g), y = avg_variance)) +
  geom_line(size = 1, color = "black") +
  geom_point(size = 2, color = "blue") +
  labs(
    title = "Average Variance vs. Number of Clusters (g)",
    x = "Number of Clusters (g)",
    y = "Average Variance of Beta Estimate"
  ) +
  theme_minimal()

# Aggregate data to calculate average variance across g
agg_results_r <- simulation_results %>%
  group_by(r) %>%
  summarise(
    avg_variance = mean(mean_se, na.rm = TRUE)
  )

# average variance vs. r
ggplot(agg_results_r, aes(x = as.numeric(r), y = avg_variance)) +
  geom_line(size = 1, color = "black") +
  geom_point(size = 2, color = "blue") +
  labs(
    title = "Average Variance vs. Number of Observations per Cluster (r)",
    x = "Number of Observations per Cluster (r)",
    y = "Average Variance of Beta Estimate"
  ) +
  theme_minimal()









optimization_results %>%
  select(
    g, r, total_cost, var_beta_hat, gamma_sq, ICC, c1_c2_ratio
  ) %>%
```

```
  gt() %>%
  tab_header(
    title = "Optimization Results",
    subtitle = "Summary of Metrics for Optimal Configurations"
  ) %>%
  cols_label(
    g = "Clusters (g)",
    r = "Observations in Cluster (r)",
    total_cost = "Total Cost",
    var_beta_hat = "Var(Beta Hat)",
    gamma_sq = "Gamma Squared",
    ICC = "ICC",
    c1_c2_ratio = "C1:C2 Ratio"
  ) %>%
  tab_spanner(
    label = "Optimization Metrics",
    columns = c(total_cost, var_beta_hat, gamma_sq, ICC, c1_c2_ratio)
  ) %>%
  fmt_number(
    columns = c(total_cost, var_beta_hat, gamma_sq, ICC, c1_c2_ratio),
    decimals = 3
  ) %>%
  tab_style(
    style = cell_text(weight = "bold"),
    locations = cells_column_labels(everything())
  )
# simulate_poisson_data <- function(g, r, alpha, beta, gamma_sq, sigma_sq) {
#   # random effects for clusters
#   cluster_effects <- rnorm(g, mean = 0, sd = sqrt(gamma_sq))
#
#   # equal allocation to treatment
#   treatment <- rbinom(g, 1, 0.5)
#
#   # create dataframe
#   data <- data.frame(
#     cluster = rep(1:g, each = r),
#     treatment = rep(treatment, each = r)
#   )
#
#   # generate cluster-specific means
#   log_mu <- alpha + beta * data$treatment + rep(cluster_effects, each = r)
#   mu <- exp(log_mu)  # Poisson mean is exponential of the linear predictor
```

```
#
#   # generate outcomes (sum of Poisson-distributed values)
#   data$y <- rpois(n = g * r, lambda = rep(mu, each = r))
#
#   return(data)
# }
#
# # fit poisson model using generalized linear mixed model
# fit_poisson_model <- function(data) {
#   model <- glmer(y ~ treatment + (1 | cluster), data = data, family = poisson)
#   return(model)
# }
#
# # evaluate poisson model performance
# evaluate_poisson_performance <- function(model, true_beta) {
#   fixef_model <- fixef(model)
#
#   if ("treatment" %in% names(fixef_model)) {
#     beta_estimate <- fixef_model["treatment"]
#     vcov_matrix <- vcov(model)
#
#     if ("treatment" %in% rownames(vcov_matrix)) {
#       se <- sqrt(vcov_matrix["treatment", "treatment"])
#       bias <- beta_estimate - true_beta
#       mse <- bias^2 + se^2
#
#       # Check coverage (Wald CI)
#       alpha <- 0.05
#       z <- qnorm(1 - alpha / 2)
#       lower_ci <- beta_estimate - z * se
#       upper_ci <- beta_estimate + z * se
#       coverage <- ifelse(true_beta >= lower_ci & true_beta <= upper_ci, 1, 0)
#     } else {
#       se <- NA
#       bias <- beta_estimate - true_beta
#       mse <- NA
#       coverage <- NA
#     }
#   } else {
#     beta_estimate <- NA
#     bias <- NA
#     mse <- NA
```

```
#     se <- NA
#     coverage <- NA
#   }
#
#   return(data.frame(
#     beta_estimate = beta_estimate,
#     bias = bias,
#     mse = mse,
#     se = se,
#     coverage = coverage
#   ))
# }
#
# # run simulations for poisson data
# run_poisson_simulations <- function(g_values, r_values, ICC_values, sigma_sq_values, beta_v
#   results <- data.frame()
#
#   for (g in g_values) {
#     for (r in r_values) {
#       for (ICC in ICC_values) {
#         for (sigma_sq in sigma_sq_values) {
#           for (beta in beta_values) {
#             alpha <- 0  # fixed intercept
#
#             # store metrics
#             biases <- numeric(n_sim)
#             mses <- numeric(n_sim)
#             ses <- numeric(n_sim)
#             coverages <- numeric(n_sim)
#
#             for (i in 1:n_sim) {
#               # calculate gamma_sq from ICC and sigma_sq
#               gamma_sq <- (ICC * sigma_sq) / (1 - ICC)
#               data <- simulate_poisson_data(g, r, alpha, beta, gamma_sq, sigma_sq)
#
#               # fit Poisson model
#               model <- fit_poisson_model(data)
#
#               # evaluate
#               perf <- evaluate_poisson_performance(model, beta)
#
#               biases[i] <- perf$bias
```

```
#               mses[i] <- perf$mse
#               ses[i] <- perf$se
#               coverages[i] <- perf$coverage
#             }
#
#             # average metrics
#             mean_bias <- mean(biases, na.rm = TRUE)
#             mean_mse <- mean(mses, na.rm = TRUE)
#             mean_se <- mean(ses, na.rm = TRUE)
#             mean_coverage <- mean(coverages, na.rm = TRUE)
#
#             # save results
#             results <- rbind(results, data.frame(
#               model = "Poisson",
#               g = g,
#               r = r,
#               ICC = ICC,
#               sigma_sq = sigma_sq,
#               beta = beta,
#               mean_bias = mean_bias,
#               mean_mse = mean_mse,
#               mean_se = mean_se,
#               mean_coverage = mean_coverage
#             ))
#           }
#         }
#       }
#     }
#   }
#
#   return(results)
# }
#
# #  parameter values
# g_values <- c(10, 20, 30, 50, 100)          # number of clusters
# r_values <- c(10, 30, 50, 100, 500)        # observations per cluster
# ICC_values <- c(0.05, 0.1, 0.2)            # intra-cluster correlation
# sigma_sq_values <- c(1, 2, 3)              # within-cluster variance
# beta_values <- c(0.2, 0.5, 0.8)            # effect sizes
#
# # poisson simulations
# simulation_results_poisson <- run_poisson_simulations(
```

```
#   g_values = g_values,
#   r_values = r_values,
#   ICC_values = ICC_values,
#   sigma_sq_values = sigma_sq_values,
#   beta_values = beta_values,
#   n_sim = 5  # number of simulations
# )
#
#
#
# write.csv(simulation_results_poisson, file = "simulation_results_poisson.csv", row.names =
```