Optimizing Study Designs for Cluster Randomized Trials under Budget Constraints: A Simulation Study

Jizhou Tian

2024-12

Abstract

Cluster randomized trials pose unique challenges in balancing cost and study design, particularly under budget constraints. This study investigates optimal combinations of clusters and within-cluster observations for estimating treatment effects. Using simulations, we evaluate designs with outcomes following normal and Poisson distributions. The model parameters and costs-related parameters are systematically varied. Our results demonstrate that when the data come from normal distribution, the optimal design is likely to take a large value of the number of clusters with fewer observations. If data come from poisson distribution, the optimal number of clusters tends to be evenly distributed in its possible range. However, the interdependence of parameters complicates straightforward recommendations. The simulation results do not clearly support some of the expected impacts of parameters on the optimal design. Future work should address model convergence issues, incorporate broader parameter spaces, and consider practical constraints to refine recommendations for optimal study designs.

Introduction

In genetic and genomic research, hierarchical data structures are common due to the nature of experimental designs. For example, in RNA sequencing (RNA-Seq) studies, multiple technical replicates may be generated from the same biological sample, while biological replicates come from different individuals or conditions. Similarly, in genome-wide association studies (GWAS), data often consists of individuals grouped by family, population, or geographic region, leading to shared genetic backgrounds within groups. These settings highlight the need for cluster-based designs to account for such dependencies.

In these studies, a key challenge lies in balancing technical replicates, which are cheaper but highly correlated, and biological replicates, which are more expensive but provide independent information. For instance, in single-cell RNA-Seq experiments, researchers must decide how many individuals (biological replicates) to sample versus how many cells (technical replicates) to sequence within each individual. The cost of collecting samples from new individuals is typically higher, whereas sequencing additional cells from the same individual is relatively cheaper. These decisions are often constrained by a fixed budget and can significantly impact the power and generalizability of the study.

Randomized controlled trials (RCTs) share similar challenges when hierarchical structures and resource limitations must be considered. In cluster randomized trials, individuals are grouped into clusters such as schools or hospitals that are randomized to treatment or control groups. The intra-cluster correlation may affect our estimation to the treatment effect. Moreover, sampling costs differ between clusters: the initial sample from a cluster is typically more expensive than subsequent samples.

In this project, we conduct a simulation study to identify optimal study designs for estimating treatment effects in cluster randomized trials under budget constraints. We first assume that the outcome follows a normal distribution and then extend our analysis to scenarios where the outcome follows a Poisson distribution. Additionally, we aim to explore the relationships between the underlying data generation mechanism parameters and the relative costs, and their impact on the optimal study design.

Design of simulation

We aim to explore the optimal experimental design under budget constraints through a simulation study. Our focus is on a cluster randomized trial where observations are assigned to either the treatment or control group, with the primary goal of estimating the treatment effect on an outcome variable Y. We conduct simulations to determine the best allocation of the number of clusters (G) and the number of observations per cluster (R), given a fixed budget B.

We design the whole simulation study based on the ADEMP framework.

Aims

1. Under a fixed budget (B), cost structures c1/c2 (c1 is the cost for the first sample from a cluster and c2 is the cost for subsequent samples) and model setting, identify the optimal combinations of the number of clusters (G) and the number of observations per cluster (R).
2. Explore how the underlying data generation mechanism parameters and cost-related parameters impact the optimal study design. For the data generation mechanism, we consider different distributions: Normal and Poisson.

Data-generating mechanisms (Conceptual Model + Operational Details)

Conceptual Model

We consider two different distributions for the outcome variable Y: Normal and Poisson.

For observation j (j=1,..., R repeated observations) in cluster i (i=1,..., G groups), let X_i be a binary indicator of whether or not cluster i is assigned to the treatment group (0 = control, 1 = treatment) and let Y_{ij} be the observed outcome. To estimate the treatment effect, we will assume a hierarchical model for Y_{ij}.

If we assume that Y follows a normal distribution, then

$$\mu_{i0} = \alpha + \beta X_i \quad \text{(fixed effect, } \mu_{i0} = \alpha \text{ or } \mu_{i0} = \alpha + \beta)$$

$$\mu_i \mid \epsilon_i = \mu_{i0} + \epsilon_i \quad \text{with } \epsilon_i \sim N(0, \gamma^2), \text{ or in other words } \mu_i \sim N(\mu_{i0}, \gamma^2)$$

$$Y_{ij} \mid \mu_i = \mu_i + e_{ij} \quad \text{with } e_{ij} \sim \text{iid } N(0, \sigma^2), \text{ or in other words } Y_{ij} \mid \mu_i \sim N(\mu_i, \sigma^2)$$

If we assume that Y follows a poisson distribution, then for each cluster i, we have:

$$\log(\mu_i) \sim N(\alpha + \beta X_i, \gamma^2)$$

We may observe conditionally independent units j=1...R within the cluster:

$$Y_{ij} \mid \mu_i \sim \text{Poisson}(\mu_i)$$

Operational Details

The number of total observations ($n_{obs} = R * G$) in each simulated data set is determined by R and G, and both of them are determined by B, c1, and c2. c1 is the cost for the first sample from a cluster and c2 is the cost for subsequent samples. c1 is far larger than c2. Since {B=500, c1=10, c2=1} and {B=5000, c1=100, c2=10} will provide the same possible combination of G and R, then it's more important to know the relative relationship among B, c1 and c2. We consider the ratio between B and c1 (B/c1) and the ratio between c1 and c2 (c1/c2) as the costs-related parameters. Given B/c1 and c1/c2, we can obtain all the possible combinations {G, R}. Note that samples within a cluster must be assigned to the same treatment or control group, we exclude the case when G=1. We choose the largest R for a given G because we want our samples to be as large as possible. Then we choose the largest G for a given R. This situation, where multiple G values correspond to the same R, may occur because both G and R have to be integers.

For normal distribution, we set values for model parameters α, β, γ and σ. We generate the G values of the binary variable X_i (i=1,...,G) under Bernoulli distribution with 50% probability to be 1. For a small value of G, the generated X_is may all equal to 1 or all equal to 0. If this happens, we regenerate X_is until the values of X_is contain both 0 and 1. Next, we generate G values of ϵ_i under the normal distribution $N(0, \gamma^2)$. We then obtain G values of u_i where $u_i = \alpha + \beta * X_i + \epsilon_i$. We generate G*R values of e_{ij} (i=1,...,G, j=1,...,R) from $N(0, \sigma^2)$. We obtain u_{ij} by replicating all those u_is R times. Then $Y_{ij} = u_{ij} + e_{ij}$. A group variable is needed for each Y_{ij} to identify its cluster when storing Y_{ij} into a data frame. This group variable will be helpful in hierarchical modeling.

For poisson distribution, we set values for model parameters α, β, γ. Similarly, we generate the G values of the binary variable X_i (i=1,...,G) under Bernoulli distribution with 50% probability to be 1. We will regenerate X_is until the values of X_is contain both 0 and 1. Next, we generate G values of ϵ_i under the normal distribution $N(0, \gamma^2)$. We then obtain G values of u_i by exponentiating $log(u_i)$ where $log(u_i) = \alpha + \beta * X_i + \epsilon_i$. We obtain u_{ij} by replicating all those u_is R times. We treat each u_{ij} as a parameter of a poisson distribution and generate one value (which is Y_{ij}) from the corresponding poisson distribution for each u_{ij}. A group variable is also needed for each Y_{ij} when storing data.

For normal distribution, we select values of B/c1 as 10 and 50, values of c1/c2 as 10 and 50, value of α as 3, values of β as 1,5 and 10, values of γ as 1, 2, 5, values of σ as 0.1 and 0.5.

For poisson distribution, we select values of B/c1 as 10 and 50, values of c1/c2 as 10 and 50, values of α as 0.1 and 0.5, values of β as 0.5 and 1, values of γ as 0.1 and 0.5.

For each distribution, simulations are conducted for all possible combinations of parameter values.

Estimands

Our estimand β is the coefficient of X, which represents the fixed effect of treatment.

Methods

For normal distribution, we fit a hierarchical model (using lmer() function in R) assuming a random intercept for each cluster. If each cluster only has one observation (R=1), we fit a linear regression instead because lmer() function can only work when the number of levels of each grouping factor is smaller than the number of observations.

For poisson distribution, we fit a hierarchical model (using glmer() function in R) assuming a random intercept for each cluster.

Perfomance measures

We will choose the optimal G and R based on the variance of our estimated coefficient $\hat{\beta}$. This is because $\hat{\beta}$ is unbiased, using MSE would be equal to evaluating the performance by variance.

Given a set of costs-related parameters B/c1 and c1/c2, and a set of model parameters α, β, γ (and σ), we obtain the estimated coefficient $\hat{\beta}$ in each simulated data set, and calculate its variance among all the simulated data set. We choose the pair {G, R} with the least variance. Then this pair of {G, R} would be the optimal value given the set of parameters.

Considering the extensive computational time required, the number of simulation n_{sim} is set to 100 for normal distribution and 50 for poisson distribution.

Results

Normal distribution

Table 1: Simulation Results of Normal Distribution

B/c1	c1/c2	alpha	beta	gamma	sigma	G	R	Var_X
10	10	3	1	1	0.1	10	1	0.548
10	10	3	1	1	0.5	8	3	0.478
10	10	3	1	2	0.1	10	1	1.284
10	10	3	1	2	0.5	9	2	1.787
10	10	3	1	5	0.1	10	1	9.108
10	10	3	1	5	0.5	10	1	10.562
10	10	3	5	1	0.1	10	1	0.420
10	10	3	5	1	0.5	8	3	0.493
10	10	3	5	2	0.1	10	1	1.790
10	10	3	5	2	0.5	9	2	1.860
10	10	3	5	5	0.1	10	1	9.656
10	10	3	5	5	0.5	7	5	11.556
10	10	3	10	1	0.1	8	3	0.387
10	10	3	10	1	0.5	10	1	0.564
10	10	3	10	2	0.1	10	1	2.118
10	10	3	10	2	0.5	10	1	2.124
10	10	3	10	5	0.1	10	1	9.806
10	10	3	10	5	0.5	10	1	12.051
10	50	3	1	1	0.1	8	13	0.465
10	50	3	1	1	0.5	10	1	0.538
10	50	3	1	2	0.1	10	1	1.983
10	50	3	1	2	0.5	10	1	2.020
10	50	3	1	5	0.1	9	6	12.166
10	50	3	1	5	0.5	10	1	10.004
10	50	3	5	1	0.1	8	13	0.449
10	50	3	5	1	0.5	9	6	0.494
10	50	3	5	2	0.1	10	1	1.941
10	50	3	5	2	0.5	10	1	1.978
10	50	3	5	5	0.1	9	6	11.355
10	50	3	5	5	0.5	8	13	10.103

10	50	3	10	1	0.1	8	13	0.403
10	50	3	10	1	0.5	8	13	0.511
10	50	3	10	2	0.1	10	1	1.353
10	50	3	10	2	0.5	9	6	1.583
10	50	3	10	5	0.1	10	1	11.528
10	50	3	10	5	0.5	8	13	11.485
50	10	3	1	1	0.1	50	1	0.096
50	10	3	1	1	0.5	50	1	0.091
50	10	3	1	2	0.1	41	3	0.280
50	10	3	1	2	0.5	45	2	0.318
50	10	3	1	5	0.1	45	2	2.405
50	10	3	1	5	0.5	50	1	2.076
50	10	3	5	1	0.1	50	1	0.093
50	10	3	5	1	0.5	45	2	0.091
50	10	3	5	2	0.1	50	1	0.337
50	10	3	5	2	0.5	50	1	0.349
50	10	3	5	5	0.1	45	2	1.466
50	10	3	5	5	0.5	50	1	1.357
50	10	3	10	1	0.1	50	1	0.074
50	10	3	10	1	0.5	50	1	0.081
50	10	3	10	2	0.1	38	4	0.369
50	10	3	10	2	0.5	50	1	0.329
50	10	3	10	5	0.1	45	2	2.231
50	10	3	10	5	0.5	50	1	1.897
50	50	3	1	1	0.1	49	2	0.080
50	50	3	1	1	0.5	48	3	0.083
50	50	3	1	2	0.1	41	11	0.254
50	50	3	1	2	0.5	42	10	0.342
50	50	3	1	5	0.1	50	1	1.786
50	50	3	1	5	0.5	48	3	1.627
50	50	3	5	1	0.1	47	4	0.066
50	50	3	5	1	0.5	47	4	0.069
50	50	3	5	2	0.1	50	1	0.283
50	50	3	5	2	0.5	49	2	0.320
50	50	3	5	5	0.1	50	1	1.624
50	50	3	5	5	0.5	48	3	1.924
50	50	3	10	1	0.1	47	4	0.074
50	50	3	10	1	0.5	38	16	0.079
50	50	3	10	2	0.1	50	1	0.331
50	50	3	10	2	0.5	47	4	0.302
50	50	3	10	5	0.1	46	5	1.834
50	50	3	10	5	0.5	50	1	1.903

The simulation results for normal distribution are shown in table 1. Since B/c1 can take either 10 or 50, so the maximum clusters for the two scenarios are 10 and 50. We observe that the optimal design is likely to take a large value of G, which means under budget restriction, a large number of clusters with fewer observations in each cluster is recommended. There are many settings where the optimal design is the maximum number of clusters with 1 observation in each cluster.

We only keep α at 3 without changing it because in normal distribution, α is only location parameter and does not affect the estimate of β.

The increase of β dose not have a clear pattern in affecting G. Its pattern changes when other parameter changes.

When B/c1=10, the increase of γ from 1 to 2 will increase G, while if γ increases from 2 to 5, G will sometimes decrease. When B/c1=10, if c1/c2=10, the increase of γ from 1 to 2 is more likely to decrease G; if c1/c2=50, the increase of γ from 1 to 2 is more likely to increase G.

The increase of σ dose not have a clear pattern in affecting G.

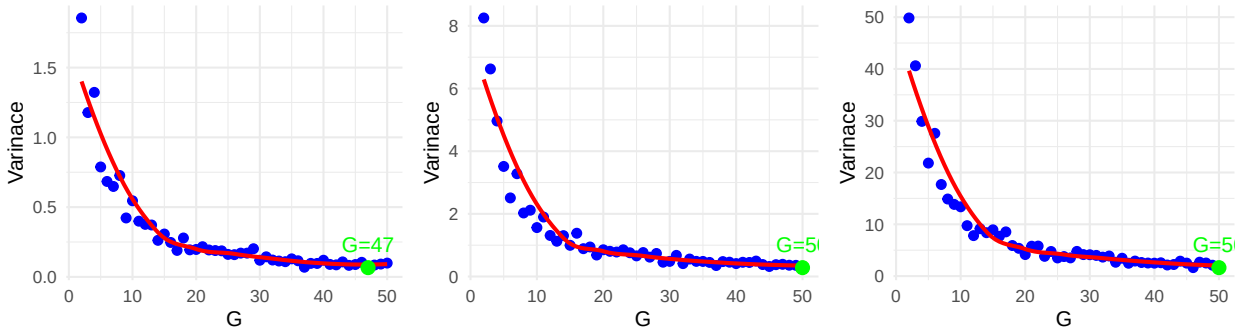

Figure 1: Variance vs G, Normal distribution, Gamma=1,2,5

To have a better understanding of the optimal design, we illustrate an example by fixing B/c1=50, c1/c2=50, α=3, β=5, *sigma*=0.1 and only changes γ from 1 to 2 to 5. We can learn that the variance of β is decreasing as G increases. This again reflects that it's recommended to choose large number of clusters.

Poisson distribution

Table 2: Simulation Results of Poisson Distribution

B_c1	c1_c2	alpha	beta	gamma	G	R	Var_X
10	10	0.1	0.5	0.1	2	41	0.046
10	10	0.1	0.5	0.5	5	11	0.263
10	10	0.1	1.0	0.1	3	24	0.055
10	10	0.1	1.0	0.5	7	5	0.209
10	10	0.5	0.5	0.1	4	16	0.025
10	10	0.5	0.5	0.5	6	7	0.155
10	10	0.5	1.0	0.1	3	24	0.024
10	10	0.5	1.0	0.5	8	3	0.208
10	50	0.1	0.5	0.1	6	34	0.018
10	50	0.1	0.5	0.5	9	6	0.159
10	50	0.1	1.0	0.1	5	51	0.017

10	50	0.1	1.0	0.5	8	13	0.170
10	50	0.5	0.5	0.1	3	117	0.011
10	50	0.5	0.5	0.5	9	6	0.140
10	50	0.5	1.0	0.1	6	34	0.012
10	50	0.5	1.0	0.5	9	6	0.118
50	10	0.1	0.5	0.1	15	24	0.008
50	10	0.1	0.5	0.5	41	3	0.027
50	10	0.1	1.0	0.1	27	9	0.007
50	10	0.1	1.0	0.5	41	3	0.029
50	10	0.5	0.5	0.1	15	24	0.006
50	10	0.5	0.5	0.5	41	3	0.029
50	10	0.5	1.0	0.1	19	17	0.005
50	10	0.5	1.0	0.5	35	5	0.038
50	50	0.1	0.5	0.1	31	31	0.003
50	50	0.1	0.5	0.5	43	9	0.021
50	50	0.1	1.0	0.1	28	40	0.002
50	50	0.1	1.0	0.5	43	9	0.024
50	50	0.5	0.5	0.1	23	59	0.002
50	50	0.5	0.5	0.5	43	9	0.022
50	50	0.5	1.0	0.1	28	40	0.002
50	50	0.5	1.0	0.5	43	9	0.018

The simulation results for poisson distribution are shown in table 1. Since B/c1 can take either 10 or 50, so the maximum clusters for the two scenarios are 10 and 50. We observe that the optimal design may choose G that seems evenly distributed in its possible range, not like normal distribution.

As the budget ratio B/c1 increases (from 10 to 50), the number of clusters G generally increases. When c1/c2 is higher, the average value of G seems increasing for both B/c1=10 and B/c1=50 settings.

Changes in α do not directly impact the choice of G and the the pattern of G when increasing β or increasing γ are also unclear through the table.

We expect that, as c1/c2 increases, the number of G may decrease considering the budget restriction. As β increase, fewer clusters G are required since stronger treatment effects are easier to detect. Also, higher γ (representing the variability between clusters) requires a larger G to capture the overall population effect accurately. However, the findings through our table does not provide enough evidence for these statements (or expectations).

Discussion

Our simulation study sheds light on the complexities of identifying optimal study designs for cluster randomized trials under budget constraints. However, several methodological considerations and limitations arise from our analysis:

1. Re-generating X Until Balanced Treatment Groups

In our simulation, we chose to regenerate the binary treatment variable X until it included both 0 (control) and 1 (treatment). While this ensures treatment allocation across clusters, it eliminates the possibility of

all clusters being assigned to a single group. This approach may introduce potential biases into parameter estimation and does not reflect real-world randomization scenarios. Future studies could explore limiting the minimum number of clusters G and randomly assigning treatment and control status.

2. Model Convergence Challenges

During the fitting process, we encountered issues with model convergence, particularly when extreme values of G and R were used. This is likely due to the hierarchical random effects structure combined with the variability in data generation. Addressing these challenges might require alternative estimation methods or adjustments to the random effects model.

3.Interdependence of Parameters

While our initial expectations suggested clear relationships (e.g., c1/c2 values impacting G or larger β requiring fewer clusters), the results do not consistently support these patterns. This inconsistency indicates potential interactions between parameters that complicate straightforward interpretations. A more exhaustive exploration of parameter combinations or the use of sensitivity analysis may provide deeper insights.

4. Computational Constraints

Due to computational limitations, we restricted the number of simulations and the range of parameters. While this approach provided preliminary insights, a broader simulation framework with more iterations and parameter values might help uncover patterns obscured by variability.

Conclusion

This study investigates optimal study designs for cluster randomized trials under budget constraints, focusing on both normal and Poisson-distributed outcomes. Our simulations highlight the critical role of costs-related parameters in determining the number of clusters and within-cluster observations. We found that when the data come from normal distribution, the optimal design is likely to take a large value of the number of clusters with fewer observations. If data come from poisson distribution, the optimal number of clusters tends to be evenly distributed in its possible range. However, the interplay between model parameters and costs-related parameters introduces complexities that need further exploration. The simulation results do not clearly support some of the expected impacts of parameters on the optimal design.

Appendix

```r
knitr::opts_chunk$set(echo = FALSE, warning = FALSE, message = FALSE)
setwd("D:/Brown/2024fall/Course/PHP2550 Practical Data Analysis/Project/Project3")
getwd()
library(tidyverse)
library(lme4)
library(knitr)
library(kableExtra)
library(gridExtra)
# Buget variables: B, c1, c2
# Data generation variables: alpha, beta, gamma, sigma
# Goal: find best G (number of clusters) and R (number of obs in a cluster)
#       to minimize variance

# Given B, c1, and c2, this function can find all the combinations of G and R
# (For a certain G, we store the corresponding maximum R)
# parameter B_c1 ratio between B and c1, i.e. B/c1
# parameter c1_c2 ratio between c1 and c2, i.e. c1/c2

comb_GR = function(B_c1, c1_c2) {
  # Check input
  if (B_c1 < 1) {
    stop("B should be greater than or equal to c1")
  }
  if (c1_c2 < 1) {
    stop("c1 should be greater than or equal to c2")
  }

  # Store all the combinations of (G, R)
  # (For a certain G, we store the corresponding maximum R)
  # G iterates from 1 to the maximum possible number of clusters
  results = data.frame(G = 1:floor(B_c1)) %>%
    # Calculate the maximum value of R given a certain G
    mutate(R = floor(B_c1 * c1_c2 / G - c1_c2 + 1)) %>%
    # Different values of G may correspond to same value of R,
    # and we will choose the maximum value of G if this happens
    arrange(R, desc(G)) %>%
    distinct(R, .keep_all = TRUE) %>%
    # We have to drop the row where G=1 because a cluster is assigned to
    # the same treatment or control group.
    filter(G!=1)

  return(results)
}
# Generating data from normal distibution
m_normal_data_gen = function(G, R, alpha, beta, gamma, sigma){
  # Generate variable X (0 = control, 1 = treatment) for each cluster
  X = rbinom(G, 1, 0.5)
  # To avoid X for each cluster all have the same value (all 0 or all 1),
  # we keep regenerating X until X have different values
  while (length(unique(X)) == 1) {
    X = rbinom(G, 1, prob = 0.5)
```

```
  }
  # Generate epsilon for each cluster
  epsilon = rnorm(G, 0, gamma)
  # Obtain mu for each cluster
  mu = alpha + beta*X + epsilon
  # Generate e for each observation
  e = rnorm(G*R, 0, sigma)
  # Obtain Y for each observation
  mu_for_all_obs = rep(mu, each = R)
  Y = mu_for_all_obs + e
  # Store data, note the hierarchical data structure
  X_for_all_obs = rep(X, each = R)
  group = rep(1:G, each = R)
  gen_data = data.frame(group = group, Y = Y, X = X_for_all_obs)
  return(gen_data)
}
# Hierarchical modeling
m_normal_get_var = function(data){
  # If there is only one observation for each group, then lmer does not work,
  # we will then fit a linear regression model instead
  if(any(duplicated(data$group))){
    control = lmerControl(check.conv.singular = "ignore")
    m1 = suppressWarnings(lmer(Y ~ X + (1 | group), data = data,
                               control=control))
    var_X = vcov(m1)["X","X"]
    beta_hat = fixef(m1)["X"]
  } else{
    m1 = lm(Y ~ X, data = data)
    se_X = summary(m1)$coefficients["X", "Std. Error"]
    var_X = se_X^2
    beta_hat = coef(m1)["X"]
  }

  result = data.frame(beta_hat = beta_hat,
                      var_X = var_X)
  return(result)
}

# Obtain variance of X for all the possible combinations of G and R
varX_normal_GR_nsim = function(B_c1, c1_c2, alpha, beta, gamma, sigma, nsim) {
  GR_values = comb_GR(B_c1, c1_c2)
  GR_values$Var_beta_hat = NA
  GR_values$Var_mean = NA
  for (i in 1:nrow(GR_values)) {
    G = GR_values[i, "G"]
    R = GR_values[i, "R"]

    # Store the result of var(X) for a certain pair of (G,R)
    beta_hat_list = numeric(nsim)
    var_list = numeric(nsim)

    #Calculate the mean of var(X) in nsim simulations
    for (j in 1:nsim) {
```

```
      data = m_normal_data_gen(G, R, alpha, beta, gamma, sigma)

      # Use tryCatch to skip iterations with errors
      # Common errors are shown below
      # Error in eval_f(x, ...) : Downdated VtV is not positive definite
      # This may due to the random sample composite
      res = tryCatch({
        m_normal_get_var(data)
      }, error = function(e) {
        data.frame(beta_hat = NA, var_X = NA) # Return NA if there's an error
      })

      beta_hat_list[j] = res[["beta_hat"]]
      var_list[j] = res[["var_X"]]
    }

    var_beta_hat = var(beta_hat_list, na.rm=TRUE)
    var_mean = mean(var_list, na.rm=TRUE)
    GR_values[i, "Var_beta_hat"] = var_beta_hat
    GR_values[i,"Var_mean"] = var_mean
    cat("we have finished", i, "\r")
  }

  return(GR_values)
}

# Based on the variance of the nsim coef estimates in nsim simulations
find_best_GR = function(df){
  result = df %>%
    filter(Var_beta_hat == min(Var_beta_hat))

  return(result)
}
# Based on the mean of the nsim variance(X) in nsim simulations
find_best_GR_2 = function(df){
  result = df %>%
    filter(Var_mean == min(Var_mean))

  return(result)
}
# Obtain the optimal G and R and its corresponding variance of X
best_normal_GR = function(B_c1, c1_c2, alpha, beta, gamma, sigma, nsim){
  res_all_GR = varX_normal_GR_nsim(B_c1, c1_c2, alpha, beta, gamma, sigma, nsim)
  # Store simulation results
  filename = paste0("normal results/",
                    "normal_Bc1_", B_c1,
                    "_c1c2_", c1_c2,
                    "_alpha_", alpha,
                    "_beta_", beta,
                    "_gamma_", gamma,
                    "_sigma_", sigma,
                    "_nsim_", nsim,
                    ".rds")
```

```
  saveRDS(res_all_GR, file = filename)
  result = find_best_GR(res_all_GR)

  return(result)
}
# Running simulations
B_c1 = c(10, 50)
c1_c2 = c(10, 50)
alpha = 3
beta = c(1, 5, 10)
gamma = c(1, 2, 5)
sigma = c(0.1, 0.5)
nsim = 100

param_grid_normal = expand.grid(B_c1=B_c1, c1_c2=c1_c2,
                                alpha=alpha, beta=beta,
                                gamma=gamma, sigma=sigma)

start = Sys.time()
set.seed(1)
results_list = lapply(1:nrow(param_grid_normal), function(i) {
  params = param_grid_normal[i, ]
  best_normal_GR(B_c1 = params$B_c1,
          c1_c2 = params$c1_c2,
          alpha = params$alpha,
          beta = params$beta,
          gamma = params$gamma,
          sigma = params$sigma,
          nsim = nsim)
})
end = Sys.time()
end-start
# Combine all results into a single data frame
part_results_df = do.call(rbind, results_list)
results_df = cbind(param_grid_normal, part_results_df)
results_df = results_df %>%
  mutate(Var_X = round(Var_beta_hat, 3)) %>%
  #mutate(Var_X = round(Var_mean, 3)) %>%
  select(-c(Var_beta_hat,Var_mean))
results_df

results_normal_table = results_df

# saveRDS(results_normal_table, file = "results_normal_table.rds")
df_results_normal = readRDS("results_normal_table.rds")

df_results_normal %>%
  arrange(B_c1, c1_c2, alpha, beta, gamma, sigma) %>%
  knitr::kable(
    caption = "Simulation Results of Normal Distribution",
    col.names = c("B/c1", "c1/c2", "alpha", "beta", "gamma", "sigma", "G", "R", "Var_X"),
    align = "c"
  ) %>%
```

```
  kable_styling(full_width = F, position = "center", font_size = 12)
plot_single_normal <- function(data) {
  min_row <- data[which.min(data$Var_beta_hat), ]

  ggplot(data, aes(x = G, y = Var_beta_hat)) +
    geom_point(color = "blue", size = 2) +
    geom_smooth(method = "loess", color = "red", se = FALSE) +
    geom_point(data = min_row, aes(x = G, y = Var_beta_hat), color = "green", size = 3) +
    geom_text(data = min_row, aes(x = G, y = Var_beta_hat, label = paste0("G=", G)),
              vjust = -1, color = "green") +
    labs(
      x = "G",
      y = "Varinace"
    ) +
    theme_minimal()
}
normal_p1 = readRDS("normal results/normal_Bc1_50_c1c2_50_alpha_3_beta_5_gamma_1_sigma_0.1_nsim_100.rds
normal_p2 = readRDS("normal results/normal_Bc1_50_c1c2_50_alpha_3_beta_5_gamma_2_sigma_0.1_nsim_100.rds
normal_p3 = readRDS("normal results/normal_Bc1_50_c1c2_50_alpha_3_beta_5_gamma_5_sigma_0.1_nsim_100.rds
normal.p1 = plot_single_normal(normal_p1)
normal.p2 = plot_single_normal(normal_p2)
normal.p3 = plot_single_normal(normal_p3)

grid.arrange(grobs=list(normal.p1, normal.p2, normal.p3), ncol = 3)
m_poisson_data_gen = function(G, R, alpha, beta, gamma){
  # Generate variable X (0 = control, 1 = treatment) for each cluster
  X = rbinom(G, 1, 0.5)
  # To avoid X for each cluster all have the same value (all 0 or all 1),
  # we keep regenerating X until X has different values
  while (length(unique(X)) == 1) {
    X = rbinom(G, 1, prob = 0.5)
  }
  # Generate epsilon for each cluster
  epsilon = rnorm(G, 0, gamma)
  # Obtain mu for each cluster (log scale)
  log_mu = alpha + beta * X + epsilon
  mu = exp(log_mu)  # Convert to lambda for Poisson distribution
  # Generate Y for each observation
  mu_for_all_obs = rep(mu, each = R)
  Y = rpois(G * R, lambda = mu_for_all_obs)
  # Store data, note the hierarchical data structure
  X_for_all_obs = rep(X, each = R)
  group = rep(1:G, each = R)
  gen_data = data.frame(group = group, Y = Y, X = X_for_all_obs)
  return(gen_data)
}

m_poisson_get_var = function(data){
  # If there is only one observation for each group, then lmer does not work,
  # we will then fit a linear regression model instead
  control = lmerControl(check.conv.singular = "ignore")
  m2 = suppressWarnings(glmer(Y ~ X + (1 | group),
                              family = poisson(link = "log"),
```

```
                              data = data,
                              control=control))
  var_X = suppressWarnings(vcov(m2)["X","X"])
  beta_hat = fixef(m2)["X"]


  result = data.frame(beta_hat = beta_hat,
                      var_X = var_X)
  return(result)
}

varX_poisson_GR_nsim = function(B_c1, c1_c2, alpha, beta, gamma, nsim) {
  GR_values = comb_GR(B_c1, c1_c2)
  GR_values$Var_beta_hat = NA
  GR_values$Var_mean = NA
  for (i in 1:nrow(GR_values)) {
    G = GR_values[i, "G"]
    R = GR_values[i, "R"]

    # Store the result of var(X) for a certain pair of (G,R)
    beta_hat_list = numeric(nsim)
    var_list = numeric(nsim)

    #Calculate the mean of var(X) in nsim simulations
    for (j in 1:nsim) {
      data = m_poisson_data_gen(G, R, alpha, beta, gamma)

      # Use tryCatch to skip iterations with errors
      # Common errors are shown below
      # Error in eval_f(x, ...) : Downdated VtV is not positive definite
      # This may due to the random sample composite
      res = tryCatch({
        m_poisson_get_var(data)
      }, error = function(e) {
        data.frame(beta_hat = NA, var_X = NA) # Return NA if there's an error
      })

      beta_hat_list[j] = res[["beta_hat"]]
      var_list[j] = res[["var_X"]]
    }

    var_beta_hat = var(beta_hat_list, na.rm=TRUE)
    var_mean = mean(var_list, na.rm=TRUE)
    GR_values[i, "Var_beta_hat"] = var_beta_hat
    GR_values[i,"Var_mean"] = var_mean
    cat("we have finished", i, "\r")
  }

  return(GR_values)
}

# Obtain the optimal G and R and its corresponding variance of X
best_poisson_GR = function(B_c1, c1_c2, alpha, beta, gamma, nsim){
```

```
  res_all_GR = varX_poisson_GR_nsim(B_c1, c1_c2, alpha, beta, gamma, nsim)
  # Store simulation results
  filename = paste0("poisson results/",
                    "poisson_Bc1_", B_c1,
                    "_c1c2_", c1_c2,
                    "_alpha_", alpha,
                    "_beta_", beta,
                    "_gamma_", gamma,
                    "_nsim_", nsim,
                    ".rds")
  saveRDS(res_all_GR, file = filename)
  result = find_best_GR(res_all_GR)
  return(result)
}
# Run simulations
B_c1 = c(10, 50)
c1_c2 = c(10, 50)
alpha = c(0.1, 0.5)
beta = c(0.5, 1)
gamma = c(0.1, 0.5)
nsim = 50

param_grid_poisson = expand.grid(B_c1=B_c1, c1_c2=c1_c2,
                                 alpha=alpha, beta=beta,
                                 gamma=gamma)

start = Sys.time()
set.seed(1)
results_list = lapply(1:nrow(param_grid_poisson), function(i) {
  params = param_grid_poisson[i, ]
  best_poisson_GR(B_c1 = params$B_c1,
          c1_c2 = params$c1_c2,
          alpha = params$alpha,
          beta = params$beta,
          gamma = params$gamma,
          nsim = nsim)
})
end = Sys.time()
end-start
# Combine all results into a single data frame
part_results_df = do.call(rbind, results_list)
results_df = cbind(param_grid_poisson, part_results_df)
results_df = results_df %>%
  mutate(Var_X = round(Var_beta_hat, 3)) %>%
  #mutate(Var_X = round(Var_mean, 3)) %>%
  select(-c(Var_beta_hat, Var_mean))
results_df

results_poisson_table = results_df

# saveRDS(results_poisson_table, file = "results_poisson_table.rds")
df_results_poisson = readRDS("results_poisson_table.rds")
```

```r
df_results_poisson %>%
  arrange(B_c1, c1_c2, alpha, beta, gamma) %>%
  knitr::kable(
    caption = "Simulation Results of Poisson Distribution",
    col.names = c("B_c1", "c1_c2", "alpha", "beta", "gamma", "G", "R", "Var_X"),
    align = "c"
  ) %>%
  kable_styling(full_width = F, position = "center", font_size = 12)
```