Optimal study design for GLS estimator under hierarchical model: A Simulation Study

December 2024

Abstract

In this project, we provide a theatrical insight for the simple hierarchical linear model with binary covariate. We point out that the GLS estimator of β is the same as OLS , no matter the γ^2, σ^2 is known or unknown. Therefore, for unknown scenario, the empirical variance of the estimator will closer to the true variance as the number of simulation increases. And for optimal study design, we derive a closed form for the target function which is a integer non-linear programming. Instead of optimization for minimal variance of the estimator we show it is the same as using grid searching for the lowest empirical variance. We also derive the distribution of $\mathcal{X}$ (inflated), marginal variance of estimatorVar($\widehat{\beta}$) and the lower bound of the variance of the estimator. We also provide a simulation study for the unknown variance case which can further explain the theatrical part of aim1 and aim2. We use ADEMP framework to do the simulation study for non-linear case(poisson).

Introduction and Theory

We consider the setting in which Y is assumed to be normally distributed. 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}, where the fixed effect and Y_{ij} is generated as follows:

$$\begin{aligned} \mu_{i0} &= \alpha + \beta X_i \\ \mu_i &\sim N(\mu_{i0}, \gamma^2) \\ Y_{ij}|\mu_i &\sim N(\mu_i, \sigma^2) \end{aligned}$$

We focus on the marginal distribution of Y:

$$Y_{ij} \sim N(\mu_{i0}, \gamma^2 + \sigma^2)$$

The correlation between two observations within the same cluster is given by the Intra-class Correlation Coefficient (ICC):

$$\rho = \frac{\text{Cov}(Y_{ij}, Y_{ij'})}{\text{Cov}(Y_i)} = \frac{\gamma^2}{\gamma^2 + \sigma^2}$$

As ρ increases, the within-cluster observations become less informative, emphasizing the need for more clusters rather than larger cluster sizes. Also we use the notation $\sigma_Y^2 = \gamma^2 + \sigma^2$ represent the total variance of the outcome.

Back to the regression case:

$$Y = N(\alpha + X\beta, (\gamma^2 + \sigma^2)\Sigma)$$

without loss of generality, we will not consider the offset α in the linear case, set $\alpha = 0$, then the design matrix X is just a vector, where Σ is a block diagonal matrix, for each block $\Sigma_i, (i = 1, ..., G)$, the correlation matrix is 1 in the diagonal and the other elements are ρ (Compound Symmetry or Exchangeable):

$$\Sigma_1 = \Sigma_i = \begin{bmatrix} 1 & \rho & \rho & \cdots & \rho \\ \rho & 1 & \rho & \cdots & \rho \\ \rho & \rho & 1 & \cdots & \rho \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \rho & \rho & \rho & \cdots & 1 \end{bmatrix}_{R \times R}$$

The inverse of the Σ is also a block diagonal matrix, each block is the same:

$$\Sigma_i^{-1} = \frac{1}{(1-\rho)[(R-1)\rho + 1]} \begin{bmatrix} 1 + (R-2)\rho & -\rho & -\rho & \cdots & -\rho \\ -\rho & 1 + (R-2)\rho & -\rho & \cdots & -\rho \\ -\rho & -\rho & 1 + (R-2)\rho & \cdots & -\rho \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -\rho & -\rho & -\rho & \cdots & 1 + (R-2)\rho \end{bmatrix}_{R \times R}$$

Most of the derivation of this part are from the book of Gałecki and Burzykowski (2013) [2] Chapter 10, we simplify the formula in the setting of binary treatment and hierarchical model instead of mixed effect. We first think about the scenario that the variance γ^2, σ^2 is known. The GLS(generalized least squares) estimator of β is given by:

$$\hat{\beta} = \left(X^\top \Sigma^{-1} X\right)^{-1} X^\top \Sigma^{-1} Y$$

We note that the treatment in one cluster is the same either 1 or 0, therefore $X_i = x_i \vec{1}$:

$$X_i^\top \Sigma_i^{-1} X_i = \frac{x_i R}{(R-1)\rho + 1}$$
$$\hat{\beta} = \frac{\sum_{i=1}^{G} x_i \sum_{j=1}^{R} Y_{ij}}{R \sum_{i=1}^{G} x_i}$$

which is the same as the OLS estimator$\hat{\beta} = \left(X^\top X\right)^{-1} X^\top Y$. And we notice that it is not related to γ^2 and σ^2. It is an unbiased estimator and the variance of GLS estimator is given by:

$$\mathrm{Var}(\hat{\beta}|X) = (\sigma^2 + \gamma^2)(X^\top \Sigma^{-1} X)^{-1} = \frac{(\sigma^2 + \gamma^2)[(R-1)\rho + 1]}{R \sum_{i=1}^{G} x_i} = \frac{\sigma^2 + \gamma^2}{\sum_{i=1}^{G} x_i} \times (\rho + \frac{1-\rho}{R})$$

Which is of course the same as we derive it directly by OLS estimator:

$$\mathrm{Var}(\hat{\beta}) = \mathrm{Var}(\left(X^\top X\right)^{-1} X^\top Y) = \left(X^\top X\right)^{-2} X^\top \mathrm{Var}(Y) X = \frac{\sigma^2 + \gamma^2}{(\sum_i x_i R)^2} (\sum_i x_i)(\rho R^2 + (1-\rho)R)$$

From the theatrical insight, the variance of estimator in not related to alpha and beta, and when conditional on X it only depends on γ^2 and σ^2. However, we are interested in how the variance of the estimator changes with the number of clusters G and the number of repeated measurements per cluster R, which means we are interested in the "marginal" variance of the estimator $\mathrm{Var}(\hat{\beta}) = E_X[\mathrm{Var}(\hat{\beta}|X)]$. As we have mentioned, X is a high dimensional binary variable, and the probability of $x_i = 1$ is p(and replicate R times , $X_i = \vec{1}$). However, when X is a vector with all the same elements zero or one, it will not bring any information for the regression coefficient. And there is a point mass probability at this situation. Therefore, we need to consider the distribution of X has a zero probability on that and equally assign the probability to the other values which is similar as the zero-inflated distribution, we noted as $\mathcal{X}$.

$$P(\mathcal{X} = x) \propto \begin{cases} 0 & \text{if } \mathcal{X} \text{ is all zero or one,} \\ P(X = x) & \text{otherwise} \end{cases}$$

If we don't do this transformation, but we we will get "NA" values when we calculate the β estimator, and drop off those values when we do the simulation. It will have the same results but just waste the computation or reduce the power. Then the marginal variance of the estimator is given below as if is $\hat{\beta}$ is an unbiased estimator:

$$\begin{aligned}\text{Var}(\hat{\beta}) = E_{\mathcal{X}|G,R}[\text{Var}(\hat{\beta}|X)] &= (\sigma^2+\gamma^2) \times (\rho + \frac{1-\rho}{R}) E_{\mathcal{X}|G,R}[\frac{1}{\sum_{i=1}^{G} x_i}] \\ &= \frac{(\sigma^2+\gamma^2)(\rho+\frac{1-\rho}{R})}{(1-2p^G)} \sum_{k=1}^{G-1} \frac{1}{k} \binom{G}{k} p^k (1-p)^{G-k}\end{aligned}$$

There is also a lower bound given by Jensen's Inequality, notice that $\sum_{i=1}^{G} x_i \propto Binomial(G,p)$ with expectation $E[\sum_{i=1}^{G} x_i] = E[\sum \mathcal{X}/R] = G(p-p^G)/(1-2p^G)$:

$$E_{\mathcal{X}|G,R}[\text{Var}(\hat{\beta}|X)] \geq (\sigma^2+\gamma^2)(\rho+\frac{1-\rho}{R}) \frac{(1-2p^G)}{G(p-p^G)}$$

Then we focus on exploring relationships between the underlying data generation mechanism parameters and the relative costs and how these impact the optimal study design. Optimal Design Considerations: The optimal design (G^*, R^*) balances the following factors are: Cost Efficiency: Minimize the cost of additional clusters (c_1) versus additional observations (c_2). 2.Statistical Efficiency: Maximize the precision of $\hat{\beta}$ by balancing the reduction in variance due to increasing G or R. 3.Intra-class Correlation: Account for the diminishing returns of increasing R in high-correlation settings. In the context of cluster randomized trials, the primary challenge lies in balancing the trade-offs between the number of clusters (G) and the number of repeated measurements per cluster (R). This trade-off is governed by cost constraints and the correlation structure of the data. Let B denote the total available budget. Sampling from a new cluster incurs a fixed cost c_1, while additional observations within the same cluster cost c_2, where $c_2 << c_1$. The budget constraint can be expressed as:$Gc_1 + G(R-1)c_2 \leq B$. The target function is marginal variance of $\hat{\beta}$, full operational research problem is given by:

$$\begin{aligned}\min_{G,R} g(G,R) &= Var(\hat{\beta}) \\ Gc_1 + G(R-1)c_2 &\leq B\end{aligned}$$

without loss of generality, we can do a normalization (reparametrization) on c_1, c_2, B as follows:$r_1 = \frac{c_1}{c_2} >> 1$ and $r_2 = \frac{B}{c_1}$, the the restriction are as follows:

$$Gr_1 + G(R-1) \leq r_1 r_2$$

G and R should be a number, which is a integer nonlinear programming and it is NP-hard. We can also use the grid search to find the optimal solution. We can use the lower bound of the variance to find the optimal solution, when G is large, the lower bound can be approximated as:

$$\min_{G,R} \frac{\sigma^2 + \gamma^2}{(\rho + \frac{1-\rho}{R})Gp}$$
$$Gr_1 + G(R-1) \leq r_1 r_2$$

There are three extreme sceneria: When ρ is closed to 1 and 0, when the $\sum_i x_i$ is closed to 0 or G (each x_i is the same), when c_1 is closer to 1 or extreme large. The design parameters (G, R) must satisfy this constraint. Increasing G improves between-cluster variation estimates, while increasing R reduces within-cluster measurement error. The goal is to identify the combination of (G, R) that provides the most efficient estimate of the treatment effect. This shows that increasing either G or R reduces the variance of $\hat{\beta}$. However, increasing G adds independent information about the treatment effect but is expensive due to c_1 and increasing R reduces within-cluster variability at a lower cost but yields diminishing returns as ρ increases. More theroy can be referred as effective samle size.

Then we consider the scenario that the variance γ^2, σ^2 is unknown. We need to highlight that the GLS(generalized least squares) estimator of β is the same as it doesn't contain the variance term. If the ρ is known, but the $\sigma_Y^2 = \sigma^2 + \gamma^2$ is unknown, we can still get the estimator of $\hat{\sigma_Y^2}$ through OLS formula. When both are unknown, the estimator of variance coefficient $\hat{\sigma^2}, \hat{\gamma^2}$ or equivalently $\hat{\sigma_Y^2}, \hat{\rho^2}$is calculated through Likelihood-Based Estimation or Restricted maximum likelihood method. Which are both provided by the lmer() function or gls() function in R. The difference of these two function is compared in [3], here we use the lmer() because it is faster. And the variance of the estimator is given by:

$$\text{Var}(\hat{\beta}|X) = \frac{\hat{\sigma_Y^2}}{\sum_{i=1}^{G} x_i} \times (\hat{\rho^2} + \frac{1 - \hat{\rho^2}}{R})$$

We point out that when we do a simulation study, we can get four kinds of variance for $\hat{\beta}$. One is the theoretical variance Var_{true}, which is calculated by the true γ^2, σ^2, the second is theoretical marginal variance Var_{marg} which is taking expectation on distribution of $\mathcal{X}$, the third is the empirical variance Var_{mc}, which is calculated through the process of Monte Carlo, and the last is the variance $Var_{estimate}$ calculated by the formula above.

For each configuration, theoretical marginal variance Var_{marg} is calculated before simulation, and the average of Var_{true} across simulation should closer Var_{marg} as iteration increases because of Law of Large Number. When the estimation for $\hat{\sigma_Y^2}$ and $\hat{\rho^2}$ is closed to the true value, the $Var_{estimate}$ will closer to the Var_{true}. More importantly, since the special case here is that

the $\hat{\beta}$ is functional independent with γ^2, σ^2, no matter is known or unknown. Therefore, for unknown scenario, Var_{mc} should closer to the Var_{true} as the number of simulation increases.

For optimal study design, if we are implementing a integer non-linear programming, the object function should be $Var_{estimate}$. But here we use grid searching for the optimal solution, the metrics we compare should be Var_{mc}. As the number of simulation increases, the results should be the same. More details about simulation will provided in the next section.

Extension to Non-Linear: in some cases, Y_{ij} follows a non-normal distribution, such as a Poisson distribution:

$$\begin{aligned} \log(\mu_i) &\sim N(\alpha + \beta X_i, \gamma^2) \\ Y_{ij}|\mu_i &\sim \text{Poisson}(\mu_i) \\ Y_i|\mu_i = \sum_{j=1}^{R} Y_{ij}|\mu_i &\sim \text{Poisson}(R\mu_i) \end{aligned}$$

The hierarchical model introduces over dispersion, which increases with γ^2. This overdispersion affects the precision of $\hat{\beta}$, requiring adjustments in the design to account for cluster-level heterogeneity. The sum of R observations within a cluster is not a Poisson distribution:

$$\begin{aligned} E[Y_i] &= E[\mu_i] \\ Var[Y_i] &= E[\mu_i] + Var[\mu_i] \end{aligned}$$

We can use normal to approximate the poisson regression when ν_{i0} is small and R is big enough, as follows:

$$\begin{aligned} \nu_{i0} &= \alpha + \beta X_i \\ \nu_i &\sim N(\nu_{i0}, \gamma^2) \\ \mu_i = \exp(\nu_i) = 1 + \nu_i + O(\nu_i^2) &\approx N(\nu_{i0} + 1, \gamma^2) \\ Y_i|\mu_i &\sim \text{Poisson}(R\mu_i) \approx N(R\mu_i, R\mu_i) \\ Y_i &\approx N(R\nu_{i0} + R, R\nu_{i0} + R + R^2\gamma^2) \end{aligned}$$

And also, without approximation, the variance and correlation within cluster is given by:

$$\begin{aligned}
\nu_{i0} &= \alpha + \beta X_i \\
\nu_i &\sim N(\nu_{i0}, \gamma^2) \\
\mu_i = \exp(\nu_i) &\sim LogNormal(\nu_{i0}, \gamma^2) \\
E[\mu_i] = e^{\nu_{i0}+\frac{\gamma^2}{2}}, &Var[\mu_i] = e^{\nu_{i0}} \\
Y_{ij}|\mu_i &\sim \text{Poisson}(\mu_i) \\
E[Y_{ij}] = E(E[Y_{ij} \mid \mu_i]) &= E[\mu_i] = e^{\nu_{i0}+\frac{\gamma^2}{2}} \\
\text{Var}(Y_{ij}) = E[\text{Var}(Y_{ij} \mid \mu_i)] + \text{Var}(E[Y_{ij} \mid \mu_i]) &= E[\mu_i] + \text{Var}[\mu_i] = e^{\nu_{i0}+\frac{\gamma^2}{2}} + e^{2\nu_{i0}+\gamma^2}\left(e^{\gamma^2}-1\right) \\
\text{Cov}(Y_{ij}, Y_{ik}) = E[\text{Cov}(Y_{ij}, Y_{ik} \mid \mu_i)] &+ \text{Cov}(E[Y_{ij} \mid \mu_i], E[Y_{ik} \mid \mu_i]) \\
= \text{Cov}(\mu_i, \mu_i) &= e^{2\nu_{i0}+\gamma^2}(e^{\gamma^2}-1)
\end{aligned}$$

Therefore, correlation between two observations within the same cluster is given by the intra-class correlation coefficient (ICC):

$$\rho = \frac{\text{Cov}(Y_{ij}, Y_{ij'})}{\text{Cov}(Y_i)} = \frac{e^{2\nu_{i0}+\gamma^2}(e^{\gamma^2}-1)}{e^{\nu_{i0}+\frac{\gamma^2}{2}} + e^{2\nu_{i0}+\gamma^2}\left(e^{\gamma^2}-1\right)}$$

The theocratic is unclear here, we will use the ADEMP framework(Define the aims, data-generating mechanisms, estimands, methods, and performance metrics for the simulation study) to do the simulation study.[4]

Simulation study for linear case

Since the sceneria for known variance is trivial, we will focus on the unknown variance case by doing simulation study. The aim is to see how R and G affect the variance of the estimator $\hat{\beta}$ under different factor: $\alpha, \beta, \gamma^2, \sigma^2, p_treatment$. Since from the theoretical part, α, β is not related to the variance of the estimator which means it will not influence the variance without considering computational error, we will fix them as 0 and 1.

We start from a very simple case: $G = 10, R = 20, \gamma^2 = 5, \sigma^2 = 1, p_treatment = 0.5$, and simulation 100 times:

```
  Beta Gamma2 Sigma2  G  R var_true       Bias var_est   Bias_SE   var_mc
1    1      5      1 10 20 1.207912 0.07842232 1.12061 0.1041221 1.084141
    var_mc_se      MSE    MSE_SE
1 0.005475458 1.079449 0.1385245
```

The bias is not close to zero, and more important, the three kind of variance are not close to each other. Therefore, we change the number of simulation to 1000 to see the results.

It is much closer, but still not close enough. When number of simulation is 5000, it is almost the same with error 0.001. (This can show the size of simulation we want also we can calculate the coverage to determine shown in [4]) As the converge rate will influence by the factor setting here, we will not explore this question.

```
  Beta Gamma2 Sigma2  G  R var_true       Bias  var_est    Bias_SE   var_mc
1    1      5      1 10 20 1.158526 0.03981759 1.155457 0.03269946 1.069254
     var_mc_se      MSE     MSE_SE
1 0.0005351624 1.069771 0.05007768
```

And we will consider some special case for the other factor$\gamma^2, \sigma^2, p_treatment$. In the real world problem, it is always $\gamma^2 >> \sigma^2$, we consider the reparametrization as before: $1 \geq \rho \sim 1$. Review the formula, if $\rho = 1$ which means $\sigma^2 = 0$, all the y_{ij} in cluster i should be the same as μ_i. The variance of the estimator is $\text{Var}(\hat{\beta}|X) = \frac{\gamma^2}{\sum_{i=1}^{G} x_i}$ and the marginal variance of the estimator is:

$$E_{X|G,R}[\text{Var}(\hat{\beta}|X)] = \frac{\gamma^2}{(1-2p^G)} \sum_{k=1}^{G-1} \frac{1}{k} \binom{G}{k} p^k (1-p)^{G-k}$$

We set the parameters as follows: $\alpha = 0, \beta = 1, \gamma^2 = 1, \sigma^2 = 0.01, p_treatment = 0.5$. With the R=10 and G=20, the results are as follows in 1000 simulation:

```
  Beta Gamma2 Sigma2  G  R var_true       Bias   var_est    Bias_SE    var_mc
1    1      1   0.01 10 20  0.22945 0.01790554 0.2273238 0.01571125 0.2468432
     var_mc_se      MSE     MSE_SE
1 0.0001235452 0.246917 0.01320346
```

We change the setting with the R=10 and G=20. It will have a lower variance. It is clear when inner-cluster correlation is closer to one, more clusters are needed to achieve less variance of the estimator.

```
  Beta Gamma2 Sigma2  G  R  var_true        Bias   var_est    Bias_SE    var_mc
1    1      1   0.01 20 10 0.1075337 -0.01818763 0.1071413 0.01018397 0.1037132
     var_mc_se       MSE      MSE_SE
1 5.190852e-05 0.1039403 0.004760926
```

The other extreme situation will not consider here is $R = 1$, it will becomes a linear regression because all the correlation is zero, We will not consider this situation here, the same as $G = 1$. Now, we focus on the optimal design for the unknown variance case. We use grid searching to find the optimal solution.

Instead of change the value for each parameter it self, we consider to change the ratio. Because as mentioned before, σ_Y will be proportional to the variance of estimator and can be obtained through GLS when ρ is known. The key point for the variance of estimator is ρ when we consider how G, R affect the variance. We change the ρ from (0.05, 0.1, 0.5,0.99), r_1, r_2 is change from (5,10,20,50,100), also $\alpha = 0, \beta = 2, p = 0.5$. The results are as follows:

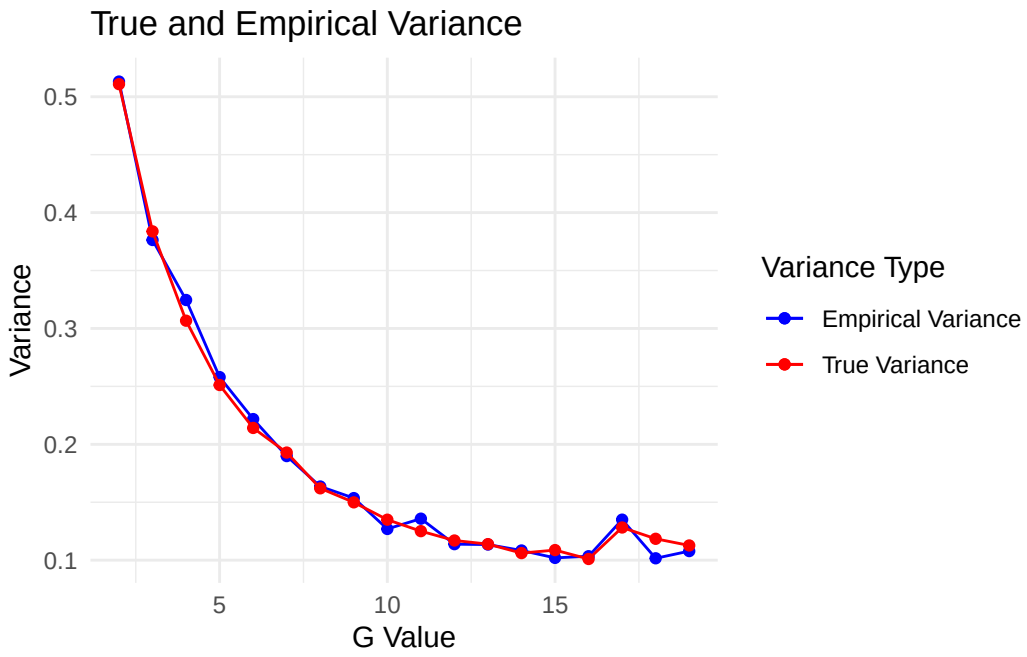
Table 1: Simulation results: optimal design

rho	**c_1**	**c_2**	**G**	**R**	**var_true**	**rank**	**diff_var_true**	**var_mc**
0.10	5	5	2	8	0.2125000	1/3	0.0000000	0.2149845
0.10	5	20	7	10	0.0643562	7/18	0.0072136	0.0576142
0.10	5	100	41	8	0.0106267	7/98	0.0004027	0.0096185
0.10	20	5	3	14	0.1242821	1/3	0.0000000	0.1207955
0.10	20	20	11	17	0.0316752	4/18	0.0011225	0.0295684
0.10	20	100	64	12	0.0055632	2/98	0.0000017	0.0051515
0.10	100	5	4	26	0.0786154	1/3	0.0000000	0.0760599
0.10	100	20	16	26	0.0182177	2/18	0.0000227	0.0177986
0.10	100	100	72	39	0.0034683	11/98	0.0000910	0.0031935
0.50	5	5	4	2	0.4377500	1/3	0.0000000	0.4620271
0.50	5	20	18	1	0.1184620	7/18	0.0174342	0.1016282
0.50	5	100	68	3	0.0199993	11/98	0.0017507	0.0173012
0.50	20	5	4	6	0.3424167	1/3	0.0000000	0.3320513
0.50	20	20	15	7	0.0834946	5/18	0.0045235	0.0741225
0.50	20	100	73	8	0.0156611	18/98	0.0009980	0.0140578
0.50	100	5	4	26	0.3183750	1/3	0.0000000	0.2876239
0.50	100	20	19	6	0.0648521	2/18	0.0014336	0.0593912
0.50	100	100	95	6	0.0123499	8/98	0.0001362	0.0114235
0.99	5	5	4	2	0.5963367	1/3	0.0000000	0.5867705
0.99	5	20	19	1	0.1125741	1/18	0.0000000	0.1099297
0.99	5	100	97	1	0.0207953	3/98	0.0004336	0.0198660
0.99	20	5	4	6	0.5799597	1/3	0.0000000	0.5430351
0.99	20	20	19	2	0.1122267	1/18	0.0000000	0.1122365
0.99	20	100	99	1	0.0204786	1/98	0.0000000	0.0191450
0.99	100	5	4	26	0.5970369	1/3	0.0000000	0.5931226
0.99	100	20	19	6	0.1101915	1/18	0.0000000	0.1099939

0.99	100	100	99	2	0.0202888	1/98	0.0000000	0.0198791

From the table above, if we fixed $\rho = 0.1$, the optimal design of G will be higher (more closer to c_2 which is the max value) as c_1 increase. And it seems that when c_1 fixed, the optimal design of G will be proportional to c_2. In the other fixed ρ, it has the same pattern. For fixed, c_1, G will increase as ρ increase. Futher results can be obtained by doing a factor analysis e.g. using anova.

We choose the optimal design based on lowest empirical variance, we also rank the true variance at the optimal design in the column "rank". When $\rho = 0.99$, the rank of optimal design are almost the same based on two kind of variance. But when $\rho = 0.5$, the rank of optimal design are very different based on two kind of variance. We focus on $c_1 = 5, c_2 = 20$, since the difference of minimal true variance and true variance of the optimal we choose is very large: 0.0174342. We draw a plot to see this case:



The true value has a more smooth curve compared with the empirical value. But both of them show a pattern that at $G = 17$, there is a increase of variance, which might happen as the closed form of variance is non-linear.

Simulation study for Non-Linear case(Poisson)

We use ADEMP framework to do the simulation study as follows:

Define Aims: The aim is to evaluate the performance of estimators under a hierarchical Poisson model with varying settings.

Data-Generating Mechanism (DGM):

$$\log(\mu_i) \sim N(\alpha + \beta X_i, \gamma^2)$$
$$Y_{ij} \sim \text{Poisson}(\mu_i)$$

Estimands: The estimand of interest is the treatment effect β which is estimated using a generalized linear mixed-effects model (GLMM) with a log-link function.

Performance Measures: We will evaluate the following: Bias,Empirical SE, MSE, Coverage, Rejection Percentage.

Simulation Study: We run the simulation for different combinations of $G, R, \gamma^2, \alpha, \beta, p$. In detail, we change G from (10, 20, 50), R from (5, 10, 20) $\gamma 2$ from (0.1, 0.5, 1) α from (0.1, 0.2, 0.5) β from (0.1, 0.2, 0.5) p from (0.3, 0.5,0.7), along with 100 simulation. The reason we choose to change these parameter is because the sources of error should depend on: γ^2 and $\nu_{i0} = \alpha + \beta X_i$ which is shown before. The probability of treatment will change the marginal distribution of X, therefore change the ν_{i0} and marginal variance of estimator. Because the poisson case, mean is related with variance, α, β is a key for the variance. Also, the change of G and R will related to the "effect of sample size". The final results is a 729 rows file, named "results_df_pois".

Results and Visualization: Summarize and visualize the results. We show the table for a simple configuration with all parameter equal to 0.1, $p = 0.5$ as follows:

Table 2: Simulation results: optimal design

G	**R**	**Bias**	**Empirical_SE**	**MSE**	**Coverage**	**SE/Bias**
10	5	-0.1083544	0.3560566	0.1372492	0.93	-3.2860364
20	5	-0.0929779	0.2037242	0.0497334	0.93	-2.1911031
50	5	-0.0957453	0.1421545	0.0291730	0.88	-1.4847144
10	10	-0.0863925	0.2044080	0.0488285	0.94	-2.3660387
20	10	-0.1005962	0.1423183	0.0301716	0.89	-1.4147485
50	10	-0.1082531	0.0937125	0.0204129	0.75	-0.8656790
10	20	-0.1042711	0.1149336	0.0239501	0.92	-1.1022572
20	20	-0.0753241	0.0943318	0.0144832	0.89	-1.2523465
50	20	-0.1073642	0.0655773	0.0157845	0.56	-0.6107934

It is clear that, as G and R increase, the bias and MSE decrease, and the empirical SE also decreases. The ratio of empirical SE to bias will be decrease as effect sample size increase. For the other situation, more work can be done through factor anaylsis.

Further Work:

For future work(in poster), we plan to extend the linear model setting by incorporating offsets and transitioning from simple linear regression to multi-linear models. Additionally, we aim to compare the four kind of variance estimation ways under Maximum Likelihood (ML) and Restricted Maximum Likelihood (REML). Intuitively, REML is expected to provide a better estimator for $\widehat{\beta}$ as it accounts for γ^2 and σ^2 as nuisance parameters, potentially resulting in a lower variance and making Var_{MC} closer to Var_{true}. Further investigation will explore how the convergence rate of Var_{MC} is influenced by other factors, along with the use of the Monte Carlo (MC) variance estimator. We will also evaluate whether `lmer()` or `gls()` performs better under the given setting. Moreover, we intend to derive the form of $\hat{\sigma}$ and $\hat{\gamma}$ estimators under this framework and analyze their behavior across different configurations. The theoretical optimal design will be examined by summing X when generating data. And, we will derive and compare the marginal variance of the estimator $\widehat{\beta}$ with Var_{MC} through simulation studies to validate these findings. Finally, we will explore more about the poisson case.

References

1. Bates D. Fitting linear mixed-effects models using lme4[J]. arXiv preprint arXiv:1406.5823, 2014.
2. Gałecki A, Burzykowski T, Gałecki A, et al. Linear mixed-effects model[M]. Springer New York, 2013.
3. Bates D. Penalized least squares versus generalized least squares representations of linear mixed models[J]. R package version, 2014: 1.1-8.
4. Morris T P, White I R, Crowther M J. Using simulation studies to evaluate statistical methods[J]. Statistics in medicine, 2019, 38(11): 2074-2102.

Appendix

Code Appendix

```
library(lme4)
library(dplyr)
library(ggplot2)
library(nlme)
library(knitr)
library(kableExtra)
# Set global simulation parameters
set.seed(2550)
# The data-generating mechanism for the hierarchical normal outcome model.
# We are sampling the $\mathcal{X}$ instead of $X$.
generate_data_linear <- function(G, R, gamma2, sigma2, p_treatment, alpha =0, beta = 1) {
  # Treatment assignment at cluster level
  X <- rbinom(G, 1, p_treatment)
  while (sum(X) == 0 | sum(X) == G) {
    X <- rbinom(G, 1, p_treatment)
  }

  # Random effects for clusters
  epsilon <- rnorm(G, mean = 0, sd = sqrt(gamma2))
  mu_i <- alpha + beta * X + epsilon

  # Generate outcomes for each observation in the cluster
  data <- data.frame(
    cluster = rep(1:G, each = R),
    X = rep(X, each = R),
    epsilon = rep(epsilon, each = R),
    mu_i = rep(mu_i, each = R),
    Y = rep(mu_i, each = R) + rnorm(G * R, mean = 0, sd = sqrt(sigma2))
  )

  return(data)
}
cal_var_marg <- function(G, R, gamma2, sigma2, p) {

  sigma_Y2 <- sigma2 + gamma2
  rho <- gamma2 / sigma_Y2
  term <- sigma_Y2 * (rho + (1 - rho) / R)

  # Pre compute 1 / (1 - 2p^G)
  denominator <- 1 / (1 - 2 * p^G)
```

```r
  # Compute the summation term
  summation <- 0
  for (k in 1:(G - 1)) {
    binom_coeff <- choose(G, k)          # Binomial coefficient
    summation <- summation + (1 / k) * binom_coeff * p^k * (1 - p)^(G - k)
  }

  # Final result
  result <- term * denominator * summation
  return(result)
}

cal_var_true <- function(G, R, gamma2, sigma2, x_values) {
  # Calculate the sum of x_values
  sum_x <- sum(x_values)
  sigma_Y2 <- sigma2 + gamma2
  rho <- gamma2 / sigma_Y2

  return(sigma_Y2 / sum_x * ((R-1)* rho +1) ) #* calculate_formula(G, 0.5)
}
# Function to run simulations and evaluate performance of beta
run_simulation_linear <- function(G, R, gamma2, sigma2, p_treatment, n_sim, alpha = 0, beta =
  # Store results for each simulation
  results <- list()

  for (i in 1:n_sim) {
    # Generate data
    data <- generate_data_linear(G, R, gamma2, sigma2, p_treatment)

    var_true <- cal_var_true(G, R, gamma2, sigma2, data$X)
    # se_true <- sqrt(var_true)

    # Fit hierarchical model and estimate beta
    # model <- lm(Y ~ X - 1, data = data)
    # beta_est <- summary(model)$coefficients[1,1]
    # beta_var <- summary(model)$coefficients[1,2]^2
    model <- lmer(Y ~ X - 1  + (1 | cluster), data = data)
    # gls <- gls(Y ~ X, data = data,correlation = corCompSymm(form = ~1|cluster))

    # Extract beta estimate and standard error
    beta_est <- fixef(model) # ["X"]
    beta_var <- vcov(model)["X", "X"]
```

```r
  # beta_se <- coef(summary(lmer))[2,2]

  # Check if confidence interval covers the true beta4,
  # ci_lower <- beta_est - 1.96 * beta_se
  # ci_upper <- beta_est + 1.96 * beta_se
  # coverage <- (beta >= ci_lower) & (beta <= ci_upper)
  #
  # confidence_intercal <- confint(model)
  # ci_lower <- confidence_intercal[4,1]
  # ci_upper <- confidence_intercal[4,2]

  # Store results
  results[[i]] <- data.frame(
    var_true = var_true,
    beta_est = beta_est,
    beta_var = beta_var
    # beta_se = beta_se
    # coverage = coverage,
  )
}

# Combine results into a single data frame
results_df <- do.call(rbind, results)

# Calculate performance metrics
var_true = mean(results_df$var_true)
bias_mc <- mean(results_df$beta_est) - beta
beta_var <- mean(results_df$beta_var)
# beta_se <- mean(results_df$beta_se)
# empirical_se <- sd(results_df$beta_est)
var_mc <- var(results_df$beta_est)
mse_mc <- mean((results_df$beta_est - beta)^2)
# coverage <- mean(results_df$coverage)

# Monte Carlo standard errors
bias_se <- sqrt(sum((results_df$beta_est - mean(results_df$beta_est))^2) / (n_sim * (n_sim
var_mc_se <- var_mc / (2 * (n_sim - 1))
mse_se <- sqrt(sum(((results_df$beta_est - beta)^2 - mse_mc)^2) / (n_sim * (n_sim - 1)))
# coverage_se <- sqrt((coverage * (1 - coverage)) / n_sim)

# Return performance metrics
return(data.frame(
```

```
    Beta = beta,
    Gamma2 = gamma2,
    Sigma2 = sigma2,
    G = G,
    R = R,
    var_true = var_true,
    Bias = bias_mc,
    var_est = beta_var,
    Bias_SE = bias_se,
    var_mc = var_mc,
    var_mc_se = var_mc_se,
    MSE = mse_mc,
    MSE_SE = mse_se
    # Coverage = paste0(round(coverage * 100, 1), "%"),
    # Coverage_SE = coverage_se,
  ))
}
run_simulation_linear(10, 20, 5, 1, p_treatment = 0.5, n_sim = 100, alpha = 0, beta =1)
run_simulation_linear(10, 20, 5, 1, p_treatment = 0.5, n_sim = 1000, alpha = 0, beta =1)
run_simulation_linear(10, 20, 1, 0.01, p_treatment = 0.5, n_sim = 1000, alpha = 0, beta =1)
run_simulation_linear(20, 10, 1, 0.01, p_treatment = 0.5, n_sim = 1000, alpha = 0, beta =1)
search_GR <- function(r_1,r_2) {
  results <- data.frame()
  for (G in 2:(r_2-1)) {
    R = floor(r_1*r_2 / G) - r_1 + 1
    results <- rbind(results, data.frame(G = G, R = R))
  }
  return(results)
}
# search_GR(5,5)
# Function to run simulations and evaluate performance of beta
run_simulation_fast <- function(G, R, gamma2, sigma2, p_treatment, n_sim, alpha = 0, beta =1
  # Store results for each simulation
  results <- list()

  for (i in 1:n_sim) {
    # Generate data
    data <- generate_data_linear(G, R, gamma2, sigma2, p_treatment)

    var_true <- cal_var_true(G, R, gamma2, sigma2, data$X)
    # se_true <- sqrt(var_true)
```

```
    # Fit hierarchical model and estimate beta
    model <- lm(Y ~ X - 1, data = data)
    beta_est <- summary(model)$coefficients[1,1]

    # Store results
    results[[i]] <- data.frame(
      var_true = var_true,
      beta_est = beta_est
    )
  }

  # Combine results into a single data frame
  results_df <- do.call(rbind, results)

  # Calculate performance metrics
  var_true = mean(results_df$var_true)
  bias_mc <- mean(results_df$beta_est) - beta
  var_mc <- var(results_df$beta_est)
  mse_mc <- mean((results_df$beta_est - beta)^2)

  # Return performance metrics
  return(data.frame(
    G = G,
    R = R,
    var_true = var_true,
    Bias = bias_mc,
    var_mc = var_mc,
    MSE = mse_mc
  ))
}
simulate_optimal_design <- function(rho,c_1,c_2,n_sims,p_treatment) {

  results <- data.frame()


  GR_list <- search_GR(c_1,c_2)

  n_row <- nrow(GR_list)
  gamma2 = rho
  sigma2 = 1 - rho
  factor <- data.frame(
    rho = rep(rho, n_row),     # Repeat rho for 5 rows
```

```
    c_1 = rep(c_1, n_row),     # Repeat c_1 for 5 rows
    c_2 = rep(c_2, n_row)   # Repeat c_2 for 5 rows
    )

  for (list_i in 1:n_row) {

    G = GR_list[list_i,"G"]
    R = GR_list[list_i,"R"]

    sim_results <- run_simulation_fast(G, R, gamma2, sigma2, p_treatment, n_sim, alpha = 0, l

    results <- rbind(results, sim_results)
  }

  results <- cbind(factor,results)

  return(results)
}

# simulate_optimal_design(0.5, 5, 5, 100, 0.5)

set.seed(2550)

# Define parameters
rho_values <- c(0.1, 0.5,0.99)
r_1_values <- c(5,20,100)
r_2_values <- c(5,20,100)
n_sim <- 1000  # Number of simulations
p_treatment = 0.5

# Create all combinations of beta_1 and sigma_x
param_grid <- expand.grid(rho = rho_values, r_1 = r_1_values, r_2 = r_2_values)

# Run simulations for all parameter combinations using lapply
# results_list_optimal <- lapply(1:nrow(param_grid), function(i) {
#   params <- param_grid[i, ]  # Extract current combination of beta_1 and sigma_x
#
#   simulate_optimal_design(
#     rho = params$rho,
#     c_1 = params$r_1,
#     c_2 = params$r_2,
#     n_sims = n_sim,
```

```r
#     p_treatment = p_treatment
#   )
# })
#
# Combine all results into a single data frame
# results_df_optimal <- do.call(rbind, results_list_optimal)
# write.csv(results_df_optimal,"./results_df_optimal.csv", row.names = FALSE)
#
results_df_optimal <- read.csv("./results_df_optimal.csv")

# Process the data
optimal_results <- results_df_optimal %>%
  # Group by rho, c_1, c_2 and calculate ranks, differences, and max G
  group_by(rho, c_1, c_2) %>%
  mutate(
    rank_var_true = rank(var_true, ties.method = "min"),
    total_in_group = n(),
    rank = paste(rank_var_true, "/", total_in_group, sep = ""),
    diff_var_true = var_true - min(var_true)
  ) %>%
  # Filter for rows with the minimum var_mc
  filter(var_mc == min(var_mc)) %>%
  # Select relevant columns
  select(rho, c_1, c_2, G, R, var_true, rank, diff_var_true,
         var_mc) %>%
  ungroup() %>%
  # Sort by rho
  arrange(rho, c_1, c_2)

# View the results
# print(optimal_results)
# write.csv(optimal_results,"./optimal_results.csv", row.names = FALSE)
optimal_results %>%
  kable(
    caption = "Simulation results: optimal design",
    align = "lccccccc"
  ) %>%
  kable_styling(
    bootstrap_options = c("striped", "hover", "condensed"),
    latex_options = "scale_down",
    position = "center"
  ) %>%
```

```
  row_spec(0, bold = TRUE)  # Make the header row bold
optimal_results_choose <- subset(results_df_optimal, rho == 0.5 & c_1 == 5 & c_2 == 20)


ggplot(data = optimal_results_choose) +
  geom_point(aes(x = G, y = var_mc, color = "Empirical Variance")) +
  geom_line(aes(x = G, y = var_mc, color = "Empirical Variance")) +
  geom_point(aes(x = G, y = var_true, color = "True Variance")) +
  geom_line(aes(x = G, y = var_true, color = "True Variance")) +
  scale_color_manual(values = c("Empirical Variance" = "blue", "True Variance" = "red")) +
  labs(title = "True and Empirical Variance",
       x = "G Value",
       y = "Variance",
       color = "Variance Type") +
  theme_minimal()

# Function to generate hierarchical Poisson data
generate_data_poisson <- function(G, R, alpha, beta, gamma2, p_treatment) {
  # Treatment assignment at cluster level
  X <- rbinom(G, 1, p_treatment)

  while (sum(X) == 0 | sum(X) == G) {
    X <- rbinom(G, 1, p_treatment)
  }

  # Random effects for clusters
  log_mu <- rnorm(G, mean = alpha + beta * X, sd = sqrt(gamma2))
  mu <- exp(log_mu)

  # Generate Poisson outcomes
  Y <- sapply(1:G, function(i) {
    rpois(R, mu[i])
  })

  # Aggregate data by cluster
  data <- data.frame(
    cluster = rep(1:G, each = R),
    X = rep(X, each = R),
    mu = rep(mu, each = R),
    Y = as.vector(t(Y))
  )
```

```r
  return(data)
}
# Function to estimate beta using GLMM
estimate_beta_poisson <- function(data) {

  # Fit a Poisson mixed-effects model
  model <- glmer(Y ~ X + (1 | cluster), data = data, family = poisson(link = "log"),
                 control = glmerControl(optimizer = "bobyqa", optCtrl = list(maxfun = 1e6)))

  # Extract beta estimate and standard error
  beta_est <- fixef(model)["X"]
  beta_se <- sqrt(vcov(model)["X", "X"])

  # Calculate p-value
  z_stat <- beta_est / beta_se
  p_value <- 2 * (1 - pnorm(abs(z_stat)))

  return(list(beta_est = beta_est, beta_se = beta_se, p_value = p_value))
}
# Function to evaluate performance metrics
evaluate_performance_poisson <- function(results, beta_true, alpha = 0.05) {
  bias <- mean(results$beta_est) - beta_true
  empirical_se <- sd(results$beta_est)
  mse <- mean((results$beta_est - beta_true)^2)

  # Coverage probability
  coverage <- mean(results$beta_est - 1.96 * results$beta_se <= beta_true &
                   results$beta_est + 1.96 * results$beta_se >= beta_true)

  # Rejection percentage
  rejection <- mean(results$p_value <= alpha)

  return(data.frame(
    Bias = bias,
    Empirical_SE = empirical_se,
    MSE = mse,
    Coverage = coverage,
    Rejection_Percentage = rejection
  ))
}
# Simulation function
run_simulation_poisson <- function(G, R, gamma2, alpha, beta, p_treatment, n_sim) {
```

```r
  results <- data.frame(beta_est = numeric(), beta_se = numeric(), p_value = numeric())

  for (i in 1:n_sim) {
    # Generate data
    data <- generate_data_poisson(G, R, alpha, beta, gamma2, p_treatment)

    # write.csv(data, "./data_problem.csv", row.names = FALSE)

    # Estimate beta
    est <- estimate_beta_poisson(data)

    # Store results
    results <- rbind(results, data.frame(
      beta_est = est$beta_est,
      beta_se = est$beta_se,
      p_value = est$p_value
    ))
  }

  return(results)
}

set.seed(2550)

# Parameters
G_vals <- c(10, 20, 50)               # Number of clusters
R_vals <- c(5, 10, 20)                # Observations per cluster
gamma2_vals <- c(0.1, 0.5, 1)         # Random effects variance
alpha_vals <- c(0.1, 0.2, 0.5)            # Intercept values
beta_vals <- c(0.1, 0.2, 0.5)             # Treatment effects
p_treatment_vals <- c(0.3, 0.5,0.7)  # Proportion of treated clusters
n_sim <- 100                         # Number of simulations

# Create a parameter grid
param_grid_pois <- expand.grid(
  G = G_vals,
  R = R_vals,
  gamma2 = gamma2_vals,
  alpha = alpha_vals,
  beta = beta_vals,
  p_treatment = p_treatment_vals
)
```

```r
results_list <- list()

# Run simulations for all parameter combinations using lapply
# results_list_pois <- lapply(1:nrow(param_grid_pois), function(i) {
#   params <- param_grid_pois[i, ]  # Extract current combination of beta_1 and sigma_x
#
#   # Run the simulation for the current parameter set
#   sim_results <- run_simulation_poisson(
#     G = params$G,
#     R = params$R,
#     gamma2 = params$gamma2,
#     alpha = params$alpha,
#     beta = params$beta,
#     p_treatment = params$p_treatment,
#     n_sim = n_sim
#   )
#
#   # Evaluate performance metrics
#   performance <- evaluate_performance_poisson(sim_results, params$beta)
#
#   # Combine parameters and performance metrics into a single row
#   results_list <- rbind(
#     results_list,
#     cbind(params, performance)
#   )
# })
#
# # Convert results to a data frame
# results_df_pois <- do.call(rbind, results_list_pois)
#
# # Save results to a CSV file
# write.csv(results_df_pois, "./results_df_pois.csv", row.names = FALSE)

results_df_pois <- read.csv("./results_df_pois.csv")


# View the results
# print(results_df_pois)

# Filter the dataset for the specified conditions
pois_results_choose <- subset(
  results_df_pois,
```

```
  gamma2 == 0.1 & alpha == 0.1 & beta == 0.1 & p_treatment == 0.5
)

# Add a new column: Empirical_SE / Bias
pois_results_choose$`SE/Bias` <- with(pois_results_choose, Empirical_SE / Bias)

# Select columns to display (excluding gamma2, alpha, beta, p_treatment)
pois_results_display <- pois_results_choose[, !(colnames(pois_results_choose) %in% c("gamma2"


# Display the filtered results in a styled table
pois_results_display %>%
  kable(
    caption = "Simulation results: optimal design",
    align = "lccccccc",
    row.names = FALSE  # Suppress row indices
  ) %>%
  kable_styling(
    bootstrap_options = c("striped", "hover", "condensed"),
    latex_options = "scale_down",
    position = "center"
  ) %>%
  row_spec(0, bold = TRUE)  # Make the header row bold

# system.time(for (i in 1:100){
#   data <- generate_data_linear(50, 50, 0.1, 0.5, 0.5)
#   lmer <- lmer(Y ~ X + (1 | cluster), data = data)
#   # gls <- gls(Y ~ X, data = data, corCompSymm(form = ~1|cluster))
# })
#
# results_1 <- rep(NA,1000)
# results_2 <- rep(NA,1000)
# results_3 <- rep(NA,1000)
# p = 0.3
# for (i in 1:1000){
#   data <- generate_data_linear(2,5,0.5,0.2,0.5)
#   results_1[i] <- summary(lm(Y ~ X, data = data))$coefficients[2,1]
#   results_2[i] <- cal_var_true(100,1,0.5,0,data$X)
#   results_3[i] <- summary(lm(Y ~ X, data = data))$coefficients[2,2]^2
# }
#
# var(results_1)
```

```
# mean(results_2)
# mean(results_3)
```