A Simulation Study of Statistical Approaches to Data Analysis in the Stepped Wedge Design

Yuqi Ren

A thesis

submitted in partial fulfillment of the

requirements for the degree of Master

Master of Science

University of Washington

2018

Reading Committee:

Jim Hughes, Chair

Patrick Heagerty

Program Authorized to Offer Degree:

Biostatistics

University of Washington

Abstract

A Simulation Study of Statistical Approaches to Data Analysis in the Stepped Wedge Design

Yuqi Ren

Chair of the Supervisory Committee:
Professor, Jim Hughes
Biostatistics

This paper studies model-based and permutation-based approaches to analyze data in the stepped wedge design under 9 scenarios. We compare robustness, efficiency, Type I error rate under null conditions, and power under alternative conditions for GEE and LMM based approaches. We find that GEE models with exchangeable correlation structures are more efficient than GEE models with independent correlation structures under all scenarios. The model-based GEE Type I error rate can be inflated when applied with a small number of clusters, but this problem is solved under a permutation approach. Correct model specification is more important to LMM as compared to GEE. However, in contrast to the model-based LMM results, the permutation-based Type I error rates for LMM models under scenarios with a random treatment effect has an unexpected inflation even though the models perfectly match the corresponding scenarios.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. Introduction

A stepped wedge cluster randomized trial design is a type of one-way crossover design in which different clusters start as controls and cross over to treatment at different time points (Hussey and Hughes, 2007). Typically, at baseline all clusters are in the control or standard of care condition; as time proceeds, different clusters switch from control to treatment. The time of switching to treatment is randomized for each cluster. Eventually, at the last time point, all clusters receive treatment. In contrast, in a cluster randomized parallel design, half of the clusters are randomly assigned to the intervention and half to the control at the beginning of the trial with no crossover. In a cluster randomized crossover design each cluster receives both the control and treatment interventions in a random order. In both crossover and stepped wedge trials, a washout time period may be included between intervention and control periods in order to make sure one condition does not affect the other or to allow individuals enrolled under one condition to complete their intervention before their cluster changes conditions. Figure 1 illustrates the settings for a traditional crossover design, a parallel design and a stepped wedge design (Hussey and Hughes, 2007).

Crossover

		Time	
		1	2
Cluster	1	X	O
	2	X	O
	3	O	X
	4	O	X

Parallel

		Time
		1
Cluster	1	X
	2	X
	3	O
	4	O

Stepped Wedge

		Time				
		1	2	3	4	5
Cluster	1	O	X	X	X	X
	2	O	O	X	X	X
	3	O	O	O	X	X
	4	O	O	O	O	X

Figure 1: Settings for traditional crossover and stepped wedge designs. “X” represents a treatment; “O” represents control.

Although a stepped wedge cluster randomized trial often takes a longer time to complete due to the multiple time points setting, it has become popular in recent years (Mdege et al., 2011). One of the important reasons is that this type of design is particularly useful in settings where it is logistically or financially impossible to provide the intervention to all participants at once due to resource limitations or geographical constraints (Brown and Lilford, 2006; Mdege et al., 2011; Ji et al., 2017). In this case, the stepped wedge design is feasible because it allows only a small fraction of the clusters to initiate the intervention at each time point, and in the end, all participants will receive the intervention. Because all participants are able to receive the intervention eventually, the stepped wedge design is also useful when it is not ethical or practical to withhold or withdraw treatment but logistical constraints prevent immediate provision of the intervention (Rhoda et al., 2011). For example, if we would like to use mobile surgical hospitals to provide male circumcision for HIV prevention in a country in Africa, it would be ideal to offer all men circumcision immediately since it has been shown that male circumcision helps prevent acquisition of HIV. However, limited resources and other practical issues such as limited number of surgeons might make it hard to offer circumcision to all men at once; one solution is to roll out provision of circumcision community by community using a stepped wedge design. There are additional benefits of the stepped wedge design. For example, the longitudinal nature of the stepped wedge design allows one to study changes in the effectiveness of the intervention over time by modeling the effects of time (Woertman et al., 2013).

In this paper we contribute to the literature by comparing the performance of marginal and random effect models from both a model-based and permutation-based perspective for evaluating the treatment effect in the stepped wedge design. Ji et al. (2017) conducted both model-based simulations and 10,000 Monte Carlo simulations based on linear mixed models

specification, such as failing to account for cluster-by-time interactions in the data. Therefore, the model-based inference may provide wrong standard error and leads to wrong Type I error rates. They then used permutation tests and the permutation test gave correct Type I error rates under the scenarios they investigate. In the current study, we conduct both model-based (asymptotic) analysis and permutation-based analysis for linear mixed model (LMM) and generalized estimating equations (GEE) under both null and alternative conditions for a variety of data generating scenarios and varying numbers of clusters. We estimate the treatment effect bias, standard error, Type I error rate under null conditions, and power under alternative conditions for the different analysis methods. In Section 2, we describe the simulation scenarios and models we use for analysis. In Section 3, we present results of comparing efficiency, robustness and power among the analysis methods. In section 4, we summarize our findings and discuss future steps.

2. Methods

2.1 Data Generating Model

We generated normally distributed data with identity link from a stepped wedged design with 5 time points *(T = 5)* and either twenty *(I = 20)* or forty clusters *(I = 40)*. One hundred observations *(n = 100)* were generated for each cluster at each time point for a total sample size of N = 20000. Let Y_{ijt} be the response for individual j in cluster i at time t (i in 1…40; j in 1…100; t in 1…5). We generate data from the model

$$Y_{ijt} = \mu + \alpha_i + \beta_t + X_{it}(\theta + \gamma_i) + e_{ijt} \quad [1]$$

where μ is the overall mean, α_i is a random effect for cluster i where $\alpha_i \sim N(0, \tau^2)$, β_t is the fixed effect of time point t, X_{it} is the treatment indicator (0 = control; 1 = treatment) for

$N(0, \nu^2)$, and e_{ijt} is a random error where $e_{ijt} \sim N(0, \sigma^2)$. We assume the $\alpha_{i,} \gamma_i$ and e_{ijt} are all mutually independent unless indicated otherwise.

We can obtain the cluster mean Y_{it} by summing all individuals in a cluster at each time point. The model would be similar to [1]:

$$Y_{it} = \mu + \alpha_i + \beta_t + X_{it}(\theta + \gamma_i) + e_{it} \quad [2]$$

where $e_{it} \sim N(0, \sigma_{cluster}^2)$, and $\sigma_{cluster}^2 = \sigma^2/n_{it}$. Again, we assume the random effects are all mutually independent.

2.2 Simulation scenarios

Table 1 shows 9 scenarios used for the simulation studies. We investigated scenarios with different number of clusters (20 vs. 40) under the null condition to understand the effect of number of clusters on Type I error rate (Leyrat et al., 2017). All scenarios contain a fixed treatment effect ($\theta = 0$ under null condition vs. $\theta = 0.08$, 0.8 or 1 under alternative conditions), a time effect ($\beta = 0$, 0.2, 0.3, 0.4, 0.5 for $t = 1,\ldots, 5$ under all conditions) and a random cluster effect ($\tau^2 = 4$). A random treatment effect ($\nu^2 = 4$) is included in some scenarios. When a random treatment effect is included, we allow it to be uncorrelated (corr = 0) or correlated (corr = 0.3) with the random cluster effect. The error variance is (σ^2) is equal to 1 in all simulations.

2.3 Approaches to analysis

We fit each simulated dataset using three GEE models and four LMM models based on individual level observations for asymptotic analysis, and fit three GEE models and five LMM models based on individual level data for permutation analysis. We estimate bias,

variance, type I error rate under null conditions, and power under alternative conditions. Under the null condition, an average treatment effect of 0 implies unbiasedness. Smaller values of the estimated treatment variance indicate higher efficiency. Under the alternative conditions we use the same logic to perform the simulation and compare the estimated treatment effect with the true treatment effect value to evaluate the bias, and also compare the efficiency and power of each model.

Scenarios	Hypothesis	No. Clusters	F	C	R	T	Corr
S1	Null	20	√	√		√	NA
S2	Null	20	√	√	√	√	0
S3	Null	20	√	√	√	√	0.3
S4	Null	40	√	√		√	NA
S5	Null	40	√	√	√	√	0
S6	Null	40	√	√	√	√	0.3
S7	Alt	40	√	√		√	NA
S8	Alt	40	√	√	√	√	0
S9	Alt	40	√	√	√	√	0.3

Table 1: Scenarios for simulation. All scenarios represent variations of the mixed effects model shown in equation 2.
Note: Null: null condition ($\theta = 0$)
Alt: alternative condition ($\theta = 0.08$, 0.8 or 1)
F: fixed treatment effect
R: random treatment effect ($\nu^2 = 4$)
T: time effect $\beta = (0, 0.2, 0.3, 0.4, 0.5)$
C: random cluster effect
Corr = Correlation between random intercept and random treatment effect (NA = not applicable)

2.3.1 GEE approaches

For GEE, an exchangeable working correlation matrix, in which the correlation between observations within a cluster are the same (Leyrat, et al., 2017), is often chosen in the analysis of stepped wedge trials. Table 2 shows three GEE models we fit under each simulation scenario. Note that the model without time effect (G1) is mis-specified since a

permutation results. In GEE we investigate models with independent (G2) and exchangeable (G3) working correlation structures.

Models	Time effect	Independent correlation structure	Exchangeable correlation structure
G1		√	
G2	√	√	
G3	√		√

Table 2: Characteristics of GEE models fit to S1-S9, checks indicate key characteristics of each GEE model.

For the asymptotic analysis we compare the z-score from the GEE analysis to the standard normal distribution. GEE tends to inflate type I error rates when the number of clusters is small (Sharples and Breslow, 1992), so we expect that performance may be better for I = 40 clusters compared to I = 20.

For the permutation analysis we reject when the observed z-score from the observed dataset is smaller than the 2.5 percentile or larger than the 97.5 percentile of the permutation distribution. We expect that the asymptotic results for GEE will be robust to mis-specification of the working variance structure because GEE uses sandwich type variance estimates (Diggle et al., 2002).

2.3.2 LMM approaches

Table 3 shows four LMM model fit to simulation scenarios S1 to S9. As previously noted, the model without a time effect (L1) is mis-specified. Significance levels are evaluated using both model-based (asymptotic) results and permutation tests by comparing the observed z-score to the permutation distribution. We compare the asymptotic result to the standard normal distribution. LMM is less robust to model mis-specification of the variance structure

effect model structure is mis-specified, then the model-based variance in LMM will be wrong and an inflated Type I error rate may result. Similar to GEE approaches, for the permutation test we reject when the observed z-score is smaller than the 2.5 percentile or larger than the 97.5 percentile of the permutation distribution.

Models	Time effect	Random cluster effect	Random treatment effect	Cluster effect correlated with treatment effect
L0				
L1		√		
L2	√	√		
L3	√	√	√	
L4	√	√	√	√

Table 3: Characteristics of LMM models fit to S1-S9, checks indicate key characteristics of each LMM model.

3. Results

3.1 GEE Asymptotic

Based on Table 4, G1 always gives biased estimates of θ. The bias depends on the strength and direction of the time effect. We value unbiasedness over efficiency, i.e., if an analysis gives a biased treatment effect coefficient, its efficiency will not be so important. The treatment effect for G1 is biased upwards by approximately 0.24 in all cases and G2 and G3 are unbiased. As expected the Type I error rate is also far away from 0.05 for analysis G1. GEE models with a time effect but with different correlation structures (G2 and G3) both give unbiased estimates of the treatment effect; however, GEE with exchangeable correlation structure (G3) leads to smaller variance than G2, i.e. higher efficiency since the exchangeable structure corresponds more closely to the true correlation structure. This is true even in scenarios with a random treatment effect, which is not an exchangeable correlation structure. For 20 clusters, the type I error rate is inflated to approximately 0.10. As we change the

number of clusters from 20 to 40, the variance becomes smaller, more closely approximates the sampling variance, and the Type I error rate approaches 0.05. The presence of correlation between the intercept and slope (e.g. S3 and S4) does not meaningfully affect the results.

Under alternative conditions (S7-S9), we choose an effect size of 0.08 for S7 and an effect size of 1.00 for S8 & S9 to investigate power. The GEE model with a time effect and exchangeable correlation structure (G3) is much more efficient than the GEE model with independent correlation structure (G2) due to the large ICC ($\frac{4}{1+4} = 0.8$) in these data. The estimate of treatment effect of G1 is still biased; therefore, its power is also less important in this case.

									G1	**G2**	**G3**
Scenario /model									GEE independent without time effect	GEE independent with time effect	GEE exchangeable with time effect
			F	C	R	T	C	Corr			
S1	Null	20	√	√		√	√	NA	0.24; 0.68; 0.11	0.00; 1.05; 0.12	-0.01; 0.47; 0.09
S2	Null	20	√	√	√	√	√	0	0.24; 0.68; 0.11	0.00; 1.05; 0.12	-0.01; 0.47; 0.09
S3	Null	20	√	√	√	√	√	0.3	0.24; 0.71; 0.12	0.001; 1.14; 0.12	-0.01; 0.45; 0.09
S4	Null	40	√	√		√	√	NA	0.24; 0.28; 0.16	0.002; 0.57; 0.07	0.000; 0.02; 0.05
S5	Null	40	√	√	√	√	√	0	0.26; 0.44; 0.12	0.04; 0.68; 0.07	-0.003; 0.32; 0.06
S6	Null	40	√	√	√	√	√	0.3	0.26; 0.46; 0.12	0.04; 0.77; 0.07	-0.003; 0.30; 0.06
S7 (0.08)	Alt	40	√	√		√	√	NA	0.32; 0.29; 0.24	0.08; 0.57; 0.08	0.08; 0.02; 0.89
S8 (1)	Alt	40	√	√	√	√	√	0	1.26; 0.44; 0.81	1.04; 0.68; 0.33	1.00; 0.32; 0.87
S9 (1)	Alt	40	√	√	√	√	√	0.3	1.26; 0.46; 0.78	1.04; 0.77; 0.31	1.00; 0.30; 0.90

Table 4: GEE results based on 500 simulations, asymptotic results for θ, the treatment effect
Results are presented as: Mean; Sd across simulations (standard error of the parameter value); Pr(reject null H_0: θ = 0)
Note: Null: null condition (θ = 0)

20 vs. 40: 20 clusters vs. 40 clusters
F: fixed treatment effect
R: random treatment effect
T: time effect (0, 0.2, 0.3, 0.4, 0.5)
C: random cluster effect
Corr = NA: the correlation between intercept and slope is not applicable
Corr = 0.3: the correlation between intercept and slope is 0.3

3.2 LMM Asymptotic

For LMM, we use four different models for analysis (table 3). The model with no time effect is mis-specified under all scenarios. The other three models all include time effects; however, there is a difference based on whether a random treatment effect is included and, if included, whether the random cluster effect and random treatment effect are correlated. As seen in Table 5, the LMM model without a time effect (L1) leads to biased treatment effect estimates (by about 0.30) and high Type I error rate. The other three models with a time effect (L2-L4) all give unbiased treatment effect estimates under all scenarios. Also, the Type I error rates for models with a random treatment effect are all close to the nominal threshold 0.05. For analysis without a random treatment effect, the Type I error rates are far above the nominal level under simulation scenarios that include a random treatment effect (e.g. S2, S3). Interestingly, whether the random cluster effect and random treatment effect are modeled as correlated or not does not seem to have a significant effect on the statistical Type I error rates. In addition, the actual variance of the intervention effect estimate is not noticeably different between models L2 – L4, suggesting that it would be better to over-fit than to under-fit a model. Based on the results, correct model specification is more important for LMM compared to GEE.

Under alternative conditions, we choose effect size of 0.08 for S7 and effect size of 0.80 for S8 & S9 to evaluate power. Since L1 & L2 produce biased results and their Type I error rates are mostly off under null conditions, we do not take their efficiency into account.

The power for L3 and L4 are the same; therefore, there is not much difference between their efficiency.

								L1	**L2**	**L3**	**L4**
Scenario /model								No time effect No random treatment	Time effect No random treatment	Time effect Random treatment Slope and intercept are NOT correlated	Time effect Random treatment Slope and intercept are correlated
			F	C	R	T	Corr				
S1	Null	20	√	√		√	NA	0.30; 0.02; 1	0.001; 0.04; 0.04	0.001; 0.04; 0.04	0.001; 0.04; 0.03
S2	Null	20	√	√	√	√	0	0.29; 0.47; 0.91	-0.01; 0.47; 0.85	-0.006; 0.46; 0.06	-0.006; 0.46; 0.06
S3	Null	20	√	√	√	√	0.3	0.29; 0.45; 0.91	-0.01; 0.45; 0.85	-0.006; 0.44; 0.06	-0.006; 0.44; 0.06
S4	Null	40	√	√		√	NA	0.30; 0.02; 1	0.000; 0.02; 0.04	0.000; 0.02; 0.04	0.000; 0.02; 0.04
S5	Null	40	√	√	√	√	0	0.30; 0.32; 0.92	-0.003; 0.32; 0.84	-0.005; 0.31; 0.05	-0.005; 0.31; 0.05
S6	Null	40	√	√	√	√	0.3	0.30; 0.30; 0.93	-0.003; 0.30; 0.83	-0.004; 0.30; 0.05	-0.004; 0.30; 0.05
S7 (0.08)	Alt	40	√	√		√	NA	0.38; 0.02; 1	0.08; 0.02; 0.88	0.08; 0.02; 0.88	0.08; 0.02; 0.88
S8 (0.8)	Alt	40	√	√	√	√	0	1.10; 0.32; 1	0.80; 0.32; 0.99	0.80; 0.31; 0.71	0.80; 0.31; 0.71
S9 (0.8)	Alt	40	√	√	√	√	0.3	1.10; 0.30; 1	0.80; 0.30; 1	0.80; 0.30; 0.75	0.80; 0.30; 0.75

Table 5: LMM results based on 500 simulations, asymptotic results for θ, the treatment effect

Results are presented as: Mean; Sd across simulations (standard error of the parameter value); alpha-level

Note: Null: null condition (treatment effect = 0)
Alt: alternative condition (treatment effect = 0.08 or 0.8)
20 vs. 40: 20 clusters vs. 40 clusters
F: fixed treatment effect
R: random treatment effect
T: time effect (0, 0.2, 0.3, 0.4, 0.5)
C: random cluster effect
Corr = NA: the correlation between intercept and slope is not applicable
Corr = 0.3: the correlation between intercept and slope is 0.3

3.3 GEE Permutation Test

In addition to GEE model based (asymptotic) analysis shown in Table 4, we also conduct GEE permutation based analyses shown in Table 6. Results are based on the permutation distribution of the z-statistics (see section 3.5 for results based on the permutation distribution of the estimated treatment effect parameter). For GEE with time effects (G2 and G3), the mean of the permutation distribution is 0.00 in all cases. Under the null condition, the mean of the permutation distribution for GEE models that do not include a time effect (G1) is not centered at 0; however, the Type I error rate is close to the nominal threshold (0.05). The Type I error rates for the three GEE models are in general a little different but are all close to 0.05. The Type I error rate is also similar for GEE models with exchangeable correlation structure (G3). Under the scenarios with and without a random treatment effect, GEE exchangeable models all give reasonable Type I error rate, which is close to 0.05. Furthermore, in contrast to the model based analysis the number of clusters does not meaningfully affect the Type I error rate in the permutation analyses – as we change the number of clusters from 20 to 40, the Type I error rates are almost constant.

Under the alternative condition, $\theta = 0.21$ for S7 and $\theta = 0.80$ for S8 & S9. As expected, GEE models without a time effect still give biased treatment effect; while GEE models with a time effect are always unbiased. Also as expected the model with exchangeable correlation structure (G3) has more power than the model with independent correlation structure (G2), indicating higher efficiency. The power for G3 is good under all alternative scenarios. The power for G1 and G2 is not very high in S7-S9. For S8 and S9, the power is higher for independent GEE models with a time effect (G2) compared to those without a time effect (G1). For S7, the power for GEE independent models is the same regardless of inclusion of time effect. However, power is higher for models with an

the permutation distribution is 0.24 for null conditions, 0.28 for alternative condition S7, and about 0.74 for alternative conditions S8 and S9 respectively.

								G1	**G2**	**G3**
Scenario /models								GEE independent without time effect	GEE independent with time effect	GEE exchangeable with time effect
			F	R	T	C	Corr			
S1	Null	20	√		√	√	NA	0.05	0.05	0.05
S2	Null	20	√	√	√	√	0	0.03	0.07	0.08
S3	Null	20	√	√	√	√	0.3	0.03	0.06	0.08
S4	Null	40	√		√	√	NA	0.04	0.05	0.07
S5	Null	40	√	√	√	√	0	0.01	0.06	0.05
S6	Null	40	√	√	√	√	0.3	0.03	0.06	0.06
S7 (0.21)	Alt	40	√		√	√	NA	0.08	0.07	0.71
S8 (0.8)	Alt	40	√	√	√	√	0	0.11	0.21	0.69
S9 (0.8)	Alt	40	√	√	√	√	0.3	0.11	0.21	0.69

Table 6: GEE results based on 500 datasets, 1000 permutations / dataset, permutation results
Results are presented as: alpha-level
Note: Null: null condition (treatment effect = 0)
Alternative: alternative condition (treatment effect = 0.21 or 0.8)
20 vs. 40: 20 clusters vs. 40 clusters
F: fixed treatment effect
R: random treatment effect
T: time effect (0, 0.2, 0.3, 0.4, 0.5)
C: random cluster effect
Corr = NA: the correlation between intercept and slope is not applicable
Corr = 0.3: the correlation between intercept and slope is 0.3

3.4 LMM Permutation Test

We also conduct LMM permutation based analysis using the permutation distribution of the z-statistics shown in Table 7 (see Appendix A for results based on the permutation distribution of the estimated treatment effect parameter). Similar to the asymptotic results shown previously, LMMs without fixed time effects (L0 and L1) give biased estimates of the treatment effect (data not shown). Also similarly, LMMs with fixed time effects (L2, L3 & L4) give unbiased estimates of the treatment effect (data not shown). However, in contrast to

the model-based results, LMMs without fixed time effects give type I error rates and power that are identical to the corresponding model that includes a fixed time effect (e.g. compare L1 and L2). Overall, an LMM model that does not include random cluster and treatment effects (independence model) (L0), maintains the type I error rate at the nominal level for scenarios without a random treatment effect (S1 and S4) and is close to the nominal level for scenarios with a random treatment effect (S2, S3, S5, S6). In contrast, for LMM with a random cluster effect but without a random treatment effect (L1 and L2) the type I error rate is maintained at or near the nominal level for scenarios without a random treatment effect (S1 and S4) but the type I error rate is significantly inflated for scenarios with a random treatment effect (S2, S3, S5, S6). Interestingly, in contrast to the model-based LMM results, the Type I error rates for permutation-based tests using LMM models that include a random treatment effect (L3, L4) give inflate type I error rates under scenarios with a random treatment effect (S2, S3, S5, S6) even though the models perfectly match the corresponding scenarios, which is unexpected. There is some suggestion that increasing the number of clusters from 20 to 40 reduces the type I error inflation for L1 – L4 but more simulations are needed to confirm this.

Under the alternative condition, the effect size is 0.21 for S7 and 0.70 for S8 & S9. L1 – L4 only have nominal Type I error rate under scenarios without a random treatment effect; therefore, we only compare its power under scenario S7. The power for L1, L2, L3 and L4 are the same under S7. The power for L0 is much smaller than the other four.

								L0	**L1**	**L2**	**L3**	**L4**
Scenario /models								LMM without time effect, random treatment effect and cluster	LMM without time effect and random treatment effect	LMM with time effect but without random treatment effect	LMM with time effect and random treatment effect + slope and intercept are correlated	LMM with time effect and random treatment effect + slope and intercept are NOT

			F	R	T	C	Corr					
S1	Null	20	√		√	√	NA	0.05	0.04	0.04	0.04	0.04
S2	Null	20	√	√	√	√	0	0.07	0.18	0.18	0.10	0.10
S3	Null	20	√	√	√	√	0.3	0.08	0.18	0.18	0.10	0.10
S4	Null	40	√		√	√	NA	0.05	0.08	0.08	0.07	0.07
S5	Null	40	√	√	√	√	0	0.06	0.16	0.16	0.07	0.07
S6	Null	40	√	√	√	√	0.3	0.07	0.16	0.16	0.07	0.07
S7 (0.21)	Alt	40	√		√	√	NA	0.08	0.73	0.73	0.73	0.73
S8 (0.7)	Alt	40	√	√	√	√	0	0.21	0.77	0.77	0.66	0.66
S9 (0.7)	Alt	40	√	√	√	√	0.3	0.19	0.78	0.78	0.66	0.66

Table 7: LMM results based on 500 datasets, 1000 permutations / dataset, permutation results
Results are presented as: alpha-level
Null: null condition (treatment effect = 0)
Alternative: alternative condition (treatment effect = 0.21 or 0.7)
20 vs. 40: 20 clusters vs. 40 clusters
F: fixed treatment effect
R: random treatment effect
T: time effect (0, 0.2, 0.3, 0.4, 0.5)
C: random cluster effect
Corr = NA: the correlation between intercept and slope is not applicable
Corr = 0.3: the correlation between intercept and slope is 0.3
Bolded models: correct models under each scenario

3.5 GEE & LMM Permutation Test based on Coefficients

We also conduct permutation-based analyses based on coefficients instead of z-scores (Table 8, 9). The Type I error rates and power are approximately the same for G1 and G2 for permutation tests based on coefficients as was seen for z-score based tests. However, the Type I error rate for GEE models with exchangeable correlation structure (G3) is inflated under scenarios with a random treatment effect (S2, S3, S5 and S7). For LMM, the permutation results using coefficients give similar type I error rates and power to the results using z-scores for models L0 – L2. However, the permutation results using coefficients have a higher inflation in Type I error rates for models L3 and L4 under scenarios with a random treatment effect (S2, S3, S5, S6), compared to the Type I error rates in z-score based permutation analysis. Note that models G3 and L2 give identical results in tables 8 and 9 as the coefficients from these two models are identical.

								G1	G2	G3
Scenario /models								GEE independent without time effect	GEE independent with time effect	GEE exchangeable with time effect
			F	R	T	C	Corr			
S1	Null	20	√		√	√	NA	0.07	0.07	0.04
S2	Null	20	√	√	√	√	0	0.09	0.09	0.18
S3	Null	20	√	√	√	√	0.3	0.07	0.07	0.18
S4	Null	40	√		√	√	NA	0.04	0.04	0.07
S5	Null	40	√	√	√	√	0	0.06	0.06	0.17
S6	Null	40	√	√	√	√	0.3	0.06	0.06	0.16
S7 (0.21)	Alt	40	√		√	√	NA	0.07	0.07	0.73
S8 (0.8)	Alt	40	√	√	√	√	0	0.26	0.26	0.84
S9 (0.8)	Alt	40	√	√	√	√	0.3	0.26	0.26	0.84

Table 8: GEE results based on 500 datasets, 1000 permutations / dataset, permutation results, coefficient based.
Results are presented as: alpha-level
Note: Null: null condition (treatment effect = 0)
Alternative: alternative condition (treatment effect = 0.21 or 0.8)
20 vs. 40: 20 clusters vs. 40 clusters
F: fixed treatment effect
R: random treatment effect
T: time effect (0, 0.2, 0.3, 0.4, 0.5)
C: random cluster effect
Corr = NA: the correlation between intercept and slope is not applicable
Corr = 0.3: the correlation between intercept and slope is 0.

								L0	L1	L2	L3	L4
Scenario /models								LMM without time effect, random treatment effect and cluster effect	No time effect No random treatment	Time effect No random treatment	Time effect Random treatment Slope and intercept are correlated	Time effect Random treatment Slope and intercept are NOT correlated
			F	R	T	C	Corr					
S1	Null	20	√		√	√	NA	0.07	0.04	0.04	0.04	0.04
S2	Null	20	√	√	√	√	0	0.08	0.18	0.18	0.20	0.20
S3	Null	20	√	√	√	√	0.3	0.07	0.18	0.18	0.19	0.19
S4	Null	40	√		√	√	NA	0.04	0.07	0.07	0.08	0.09

S6	Null	40	√	√	√	√	0.3	0.06	0.17	0.17	0.18	0.18
S7 (0.21)	Alt	40	√		√	√	NA	0.07	0.73	0.73	0.73	0.72
S8 (0.7)	Alt	40	√	√	√	√	0	0.21	0.78	0.78	0.82	0.82
S9 (0.7)	Alt	40	√	√	√	√	0.3	0.19	0.78	0.78	0.81	0.81

Table 9: LMM results based on 500 datasets, 1000 permutations / dataset, permutation results, coefficient based
Results are presented as: alpha-level
Note: Null: null condition (treatment effect = 0)
Alternative: alternative condition (treatment effect = 0.21 or 0.7)
20 vs. 40: 20 clusters vs. 40 clusters
F: fixed treatment effect
R: random treatment effect (2)
T: time effect (0, 0.2, 0.3, 0.4, 0.5)
C: random cluster effect
Corr = NA: the correlation between intercept and slope is not applicable
Corr = 0.3: the correlation between intercept and slope is 0.3
Bolded models: correct models under each scenario

4. Discussion

We conducted both model-based and permutation-based analysis of data from stepped wedge design studies to compare the robustness, efficiency, Type I error rate under null condition and power under alternative condition among GEE and LMM models under each of 9 scenarios. In general, correct model specification under model-based analysis is more important to LMM than to GEE, and over-fitting of LMM models performs better than under-fitting of LMM models. Specifically, the model-based results show that LMM models with random cluster and treatment effect produce similar levels of bias, efficiency and Type I error rate as the correctly specified LMM model, even if there is no random treatment effect in the correctly specified model. In contrast, if a random treatment effect truly exists under the scenario, the model-based results for a LMM model without a random treatment effect perform extremely poorly and have an inflated Type I error rate.

In these simulations, GEE with exchangeable correlation structure led to smaller variance of the estimated treatment effect compared to an independence correlation structure

since the exchangeable structure corresponds more closely to the true correlation structure. In addition, in model-based analysis, the number of clusters has more effect on GEE than LMM. As we increase the number of clusters from 20 to 40, the model-based GEE simulations provide Type I error rates closer to the nominal level. The Type I error rates for model-based LMM simulations do not noticeably changes as number of clusters increases.

Using permutation tests, the primary quantities of interest are the Type I error rate under null conditions and the power under alternative conditions. GEE models generate similar Type I error rates or power under all scenarios when the permutation test is based on z-scores, regardless of the correlation structure and the model-specification. The number of clusters does not have an effect on Type I error rate when the permutation distribution is used for testing. The permutation results for LMM models are not what we expected. As we add a random treatment effect in the scenario, the LMM models provide inflated Type I error rates even if the underlying model is correctly specified. As noted by Hughes, et.al (2018), the variance of the permutation distribution assumed a constant variance-covariance matrix across clusters, an assumption that is violated when random treatment effects are included in the simulation.

Future research can correct for some of the limitations in our study, such as extremely high intra-cluster correlation in simulations (0.8) and the data we simulated are all normal. Further, future researchers may also wish to explore more approaches besides GEE and LMM, or conduct simulations using models with non-linear links such as binary data. From our study, we hope the results can contribute to the literature on permutation tests in stepped-wedge design, and can help researchers with appropriate model selection.

BIBLIOGRAPHY

1. Brown, C. A. and Lilford, R. J. (2006). **The stepped wedge trial design: A systematic review**. *BMC Med. Res. Methodol.* **6** 54.

2. Leyrat C, Morgan E. K., Leurent B, Kahan C. B. **Cluster randomized trials with a small number of clusters: which analyses should be used?** *International Journal of Epidemiology*, dyx169.

3. Diggle, P., Heagerty, P., Liang, K., Zeger, S. Analysis of longitudinal data. 2nd ed. Oxford University Press,; 2002.

4. Hussey, M. A. and Hughes, J. P. (2007). **Design and analysis of stepped wedge cluster randomized trials.** *Contemp. Clin. Trials* **28** 182–191.

5. Hughes, J.P., Patrick Heagerty, Yuqi Ren (2018). **Robust inference in the stepped wedge design.** In preparation

6. Ji, Xinyao; Fink, Gunther; Robyn, Paul Jacob; Small, Dylan S. (2017). **Randomization inference for stepped-wedge cluster-randomized trials: An application to community-based health insurance.** *Ann. Appl. Stat. no. 1*, **1--20**. doi:10.1214/16-AOAS969.

7. Mdege, N. D., Man, M. S., Taylor, C. A. and Torgerson, D. J. (2011). **Systematic review of stepped wedge cluster randomized trials shows that design is particularly used to evaluate interventions during routine implementation.** *J. Clin. Epidemiol.* **64** 936–948.

8. Rhoda, D. A., Murray, D. M., Andridge, R. R., Pennell, M. L. and Hade, E. M. (2011). **Studies with staggered starts: Multiple baseline designs and group-randomized trials.** *Am. J. Publ. Health* **101** 2164–2169.

9. Sharples, K., Breslow, N. Regression analysis of correlated binary data: some small sample results for the estimating equation approach. *J Stat Comput Simul*. 1992;42:1–20.

10. Woertman, W., de Hoop, E., Moerbeek, M., Zuidema, S. U., Gerritsen, D. L. and Teerenstra, S. (2013). **Stepped wedge designs could reduce the required sample size in cluster randomized trials**. *J. Clin. Epidemiol.* **66** 752–758.

APPENDIX A

ASYMPTOTIC ANALYSIS

 # GEE
 # Null condition, 20 clusters, with time effect, no random treatment effect, correlation = NA

```
geeIndResult = NULL
geeExResult = NULL
geeIndResultNotime = NULL
geeIndZscore = NULL
geeExZscore = NULL
geeIndZscoreNotime = NULL

for(i in 1:500) {
  design <- swDsn(clusters=c(5,5,5,5), extra.time=0, all.ctl.time0=TRUE)
  # mu1 is the difference between a0 and a1
  # tau: random intercept
  # eta: random treatment
  swGenData.nScalar <- swSim(design,family=gaussian(link="identity"), n=100, mu0=3,
                        mu1=3, time.effect=c(0,.2,.3,.4,.5), sigma=1, tau=2, eta=0,
                        rho=0, seed=i, retTimeOnTx=FALSE)

  geeInd <- gee(response.var~tx.var+as.factor(time.var), data = swGenData.nScalar, family =
"gaussian",id = cluster.var, corstr = "independence")
  geeIndResult[i] = geeInd$coefficients[[2]][1]
  geeIndZscore[i] = summary(geeInd)$coefficients[2,5]

  geeEx <- gee(response.var~tx.var+as.factor(time.var), data = swGenData.nScalar, family =
"gaussian",id = cluster.var, corstr = "exchangeable")
  geeExResult[i] = geeEx$coefficients[[2]][1]
  geeExZscore[i] = summary(geeEx)$coefficients[2,5]

  geeIndNotime <- gee(response.var~tx.var, data = swGenData.nScalar, family =
"gaussian",id = cluster.var, corstr = "exchangeable")
  geeIndResultNotime[i] = geeIndNotime$coefficients[[2]][1]
  geeIndZscoreNotime[i] = summary(geeIndNotime)$coefficients[2,5]
}
```

```
mean(geeIndResult)
mean(geeExResult)
mean(geeIndResultNotime)

sd(geeIndResult)
sd(geeExResult)
sd(geeIndResultNotime)


# Compute the p-values
geeIndCount = 0
geeExCount = 0
geeIndCountNotime = 0
for (i in 1:500) {
  if(abs(geeIndZscore[i]) >= 1.96) {
    geeIndCount = geeIndCount+1
  }
  if(abs(geeExZscore[i]) >= 1.96) {
    geeExCount = geeExCount+1
  }
  if(abs(geeIndZscoreNotime[i]) >= 1.96) {
    geeIndCountNotime = geeIndCountNotime + 1
  }
}
print(geeIndCount/500)
print(geeExCount/500)
print(geeIndCountNotime/500)
```

 # Null condition, 20 clusters, with random treatment effect, time effect, correlation = 0

```
geeIndResult1 = NULL
geeExResult1 = NULL
geeIndZscore1 = NULL
geeExZscore1 = NULL
geeIndResultNotime1 = NULL
geeIndZscoreNotime1 = NULL

for(i in 1:500) {
  design <- swDsn(clusters=c(5,5,5,5), extra.time=0, all.ctl.time0=TRUE)
  # mu1 is the difference between a0 and a1
  # tau: random intercept
  # eta: random treatment
  swGenData.nScalar <- swSim(design,family=gaussian(link="identity"), n=100, mu0=3,
                             mu1=3, time.effect=c(0,.2,.3,.4,.5), sigma=1, tau=2, eta=2,
                             rho=0, seed=i, retTimeOnTx=FALSE)

  geeInd1 <- gee(response.var~tx.var+as.factor(time.var), data = swGenData.nScalar, family
```

```
  geeIndResult1[i] = geeInd1$coefficients[[2]][1]
  geeIndZscore1[i] = summary(geeInd1)$coefficients[2,5]

  geeEx1 <- gee(response.var~tx.var+as.factor(time.var), data = swGenData.nScalar, family =
"gaussian",id = cluster.var, corstr = "exchangeable")
  geeExResult1[i] = geeEx1$coefficients[[2]][1]
  geeExZscore1[i] = summary(geeEx1)$coefficients[2,5]

  geeIndNotime1 <- gee(response.var~tx.var, data = swGenData.nScalar, family =
"gaussian",id = cluster.var, corstr = "independence")
  geeIndResultNotime1[i] = geeIndNotime1$coefficients[[2]][1]
  geeIndZscoreNotime1[i] = summary(geeIndNotime1)$coefficients[2,5]
}

mean(geeIndResult1)
mean(geeExResult1)
mean(geeIndResultNotime1)

sd(geeIndResult1)
sd(geeExResult1)
sd(geeIndResultNotime1)
boxplot(geeIndResult1, geeExResult1)

# Compute the p-values
geeIndCount1 = 0
geeExCount1 = 0
geeIndCountNotime1 = 0
for (i in 1:500) {
  if(abs(geeIndZscore1[i]) >= 1.96) {
    geeIndCount1 = geeIndCount1+1
  }
  if(abs(geeExZscore1[i]) >= 1.96) {
    geeExCount1 = geeExCount1+1
  }
  if(abs(geeIndZscoreNotime1[i]) >= 1.96) {
    geeIndCountNotime1 = geeIndCountNotime1 + 1
  }
}
print(geeIndCount1/500)
print(geeExCount1/500)
print(geeIndCountNotime1/500)
```

 # Null condition, 20 clusters, with random treatment effect, time effect, correlation = 0.3

```
geeIndResult2 = NULL
geeExResult2 = NULL
geeIndZscore2 = NULL
geeExZscore2 = NULL
```

```
geeIndZscoreNotime2 = NULL

for(i in 1:500) {
  design <- swDsn(clusters=c(5,5,5,5), extra.time=0, all.ctl.time0=TRUE)
  # mu1 is the difference between a0 and a1
  # tau: random intercept
  # eta: random treatment
  swGenData.nScalar <- swSim(design,family=gaussian(link="identity"), n=100, mu0=3,
                        mu1=3, time.effect=c(0,.2,.3,.4,.5), sigma=1, tau=2, eta=2,
                        rho=0.3, seed=i, retTimeOnTx=FALSE)

  geeInd2 <- gee(response.var~tx.var+as.factor(time.var), data = swGenData.nScalar, family
= "gaussian",id = cluster.var, corstr = "independence")
  geeIndResult2[i] = geeInd2$coefficients[[2]][1]
  geeIndZscore2[i] = summary(geeInd2)$coefficients[2,5]

  geeEx2 <- gee(response.var~tx.var+as.factor(time.var), data = swGenData.nScalar, family =
"gaussian",id = cluster.var, corstr = "exchangeable")
  geeExResult2[i] = geeEx2$coefficients[[2]][1]
  geeExZscore2[i] = summary(geeEx2)$coefficients[2,5]

  geeIndNotime2 <- gee(response.var~tx.var, data = swGenData.nScalar, family =
"gaussian",id = cluster.var, corstr = "independence")
  geeIndResultNotime2[i] = geeIndNotime2$coefficients[[2]][1]
  geeIndZscoreNotime2[i] = summary(geeIndNotime2)$coefficients[2,5]
}

mean(geeIndResult2)
mean(geeExResult2)
mean(geeIndResultNotime2)

sd(geeIndResult2)
sd(geeExResult2)
sd(geeIndResultNotime2)
boxplot(geeIndResult2, geeExResult2)

# Compute the p-values
geeIndCount2 = 0
geeExCount2 = 0
geeIndCountNotime2 = 0
for (i in 1:500) {
  if(abs(geeIndZscore2[i]) >= 1.96) {
    geeIndCount2 = geeIndCount2+1
  }
  if(abs(geeExZscore2[i]) >= 1.96) {
    geeExCount2 = geeExCount2+1
  }
  if(abs(geeIndZscoreNotime2[i]) >= 1.96) {
    geeIndCountNotime2 = geeIndCountNotime2+1
  }
```

```
}
print(geeIndCount2/500)
print(geeExCount2/500)
print(geeIndCountNotime2/500)
```

 # Null condition, 40 clusters, no random treatment effect, with time effect, correlation = NA

```
geeIndResult = NULL
geeExResult = NULL
geeIndResultNotime = NULL
geeIndZscore = NULL
geeExZscore = NULL
geeIndZscoreNotime = NULL

for(i in 1:500) {
  design <- swDsn(clusters=c(10,10,10,10), extra.time=0, all.ctl.time0=TRUE)
  # mu1 is the difference between a0 and a1
  # tau: random intercept
  # eta: random treatment
  swGenData.nScalar <- swSim(design,family=gaussian(link="identity"), n=100, mu0=3,
                        mu1=3, time.effect=c(0,.2,.3,.4,.5), sigma=1, tau=2, eta=0,
                        rho=0, seed=i, retTimeOnTx=FALSE)

  #geeInd <- gee(response.var~tx.var+as.factor(time.var), data = swGenData.nScalar, family
= "gaussian",id = cluster.var, corstr = "independence")
  #geeIndResult[i] = geeInd$coefficients[[2]][1]
  #geeIndZscore[i] = summary(geeInd)$coefficients[2,5]

  #geeEx <- gee(response.var~tx.var+as.factor(time.var), data = swGenData.nScalar, family =
"gaussian",id = cluster.var, corstr = "exchangeable")
  #geeExResult[i] = geeEx$coefficients[[2]][1]
  #geeExZscore[i] = summary(geeEx)$coefficients[2,5]

  geeIndNotime <- gee(response.var~tx.var, data = swGenData.nScalar, family =
"gaussian",id = cluster.var, corstr = "independence")
  geeIndResultNotime[i] = geeIndNotime$coefficients[[2]][1]
  geeIndZscoreNotime[i] = summary(geeIndNotime)$coefficients[2,5]
}

#mean(geeIndResult)
#mean(geeExResult)
mean(geeIndResultNotime)

#sd(geeIndResult)
#sd(geeExResult)
sd(geeIndResultNotime)
```

```
# Compute the p-values
#geeIndCount = 0
#geeExCount = 0
geeIndCountNotime = 0
for (i in 1:500) {
  #if(abs(geeIndZscore[i]) >= 1.96) {
  #  geeIndCount = geeIndCount+1
  #}
  #if(abs(geeExZscore[i]) >= 1.96) {
  #  geeExCount = geeExCount+1
  #}
  if(abs(geeIndZscoreNotime[i]) >= 1.96) {
    geeIndCountNotime = geeIndCountNotime + 1
  }
}
#print(geeIndCount/500)
#print(geeExCount/500)
print(geeIndCountNotime/500)
```

 # Null condition, 40 clusters, with random treatment effect, with time effect, correlation = 0

```
geeIndResult = NULL
geeExResult = NULL
geeIndResultNotime = NULL
geeIndZscore = NULL
geeExZscore = NULL
geeIndZscoreNotime = NULL

for(i in 1:500) {
  design <- swDsn(clusters=c(10,10,10,10), extra.time=0, all.ctl.time0=TRUE)
  # mu1 is the difference between a0 and a1
  # tau: random intercept
  # eta: random treatment
  swGenData.nScalar <- swSim(design,family=gaussian(link="identity"), n=100, mu0=3,
                        mu1=3, time.effect=c(0,.2,.3,.4,.5), sigma=1, tau=2, eta=2,
                        rho=0, seed=i, retTimeOnTx=FALSE)

  #geeInd <- gee(response.var~tx.var+as.factor(time.var), data = swGenData.nScalar, family
= "gaussian",id = cluster.var, corstr = "independence")
  #geeIndResult[i] = geeInd$coefficients[[2]][1]
  #geeIndZscore[i] = summary(geeInd)$coefficients[2,5]

  #geeEx <- gee(response.var~tx.var+as.factor(time.var), data = swGenData.nScalar, family =
"gaussian",id = cluster.var, corstr = "exchangeable")
  #geeExResult[i] = geeEx$coefficients[[2]][1]
  #geeExZscore[i] = summary(geeEx)$coefficients[2,5]
```

```
  geeIndNotime <- gee(response.var~tx.var, data = swGenData.nScalar, family =
"gaussian",id = cluster.var, corstr = "independence")
  geeIndResultNotime[i] = geeIndNotime$coefficients[[2]][1]
  geeIndZscoreNotime[i] = summary(geeIndNotime)$coefficients[2,5]
}

#mean(geeIndResult)
#mean(geeExResult)
mean(geeIndResultNotime)

#sd(geeIndResult)
#sd(geeExResult)
sd(geeIndResultNotime)

# Compute the p-values
#geeIndCount = 0
#geeExCount = 0
geeIndCountNotime = 0
for (i in 1:500) {
  #if(abs(geeIndZscore[i]) >= 1.96) {
  #  geeIndCount = geeIndCount+1
  #}
  #if(abs(geeExZscore[i]) >= 1.96) {
  #  geeExCount = geeExCount+1
  #}
  if(abs(geeIndZscoreNotime[i]) >= 1.96) {
    geeIndCountNotime = geeIndCountNotime + 1
  }
}
#print(geeIndCount/500)
#print(geeExCount/500)
print(geeIndCountNotime/500)
```

 # Null condition, 40 clusters, with random treatment effect, time effect, correlation = 0.3

```
geeIndResult = NULL
geeExResult = NULL
geeIndResultNotime = NULL
geeIndZscore = NULL
geeExZscore = NULL
geeIndZscoreNotime = NULL

for(i in 1:500) {
  design <- swDsn(clusters=c(10,10,10,10), extra.time=0, all.ctl.time0=TRUE)
  # mu1 is the difference between a0 and a1
  # tau: random intercept
  # eta: random treatment
```

```
                    mu1=3, time.effect=c(0,.2,.3,.4,.5), sigma=1, tau=2, eta=2,
                    rho=0.3, seed=i, retTimeOnTx=FALSE)

  #geeInd <- gee(response.var~tx.var+as.factor(time.var), data = swGenData.nScalar, family
= "gaussian",id = cluster.var, corstr = "independence")
  #geeIndResult[i] = geeInd$coefficients[[2]][1]
  #geeIndZscore[i] = summary(geeInd)$coefficients[2,5]

  #geeEx <- gee(response.var~tx.var+as.factor(time.var), data = swGenData.nScalar, family =
"gaussian",id = cluster.var, corstr = "exchangeable")
  #geeExResult[i] = geeEx$coefficients[[2]][1]
  #geeExZscore[i] = summary(geeEx)$coefficients[2,5]

  geeIndNotime <- gee(response.var~tx.var, data = swGenData.nScalar, family =
"gaussian",id = cluster.var, corstr = "independence")
  geeIndResultNotime[i] = geeIndNotime$coefficients[[2]][1]
  geeIndZscoreNotime[i] = summary(geeIndNotime)$coefficients[2,5]
}

#mean(geeIndResult)
#mean(geeExResult)
mean(geeIndResultNotime)

#sd(geeIndResult)
#sd(geeExResult)
sd(geeIndResultNotime)

# Compute the p-values
#geeIndCount = 0
#geeExCount = 0
geeIndCountNotime = 0
for (i in 1:500) {
  #if(abs(geeIndZscore[i]) >= 1.96) {
  #  geeIndCount = geeIndCount+1
  #}
  #if(abs(geeExZscore[i]) >= 1.96) {
  #  geeExCount = geeExCount+1
  #}
  if(abs(geeIndZscoreNotime[i]) >= 1.96) {
    geeIndCountNotime = geeIndCountNotime + 1
  }
}
#print(geeIndCount/500)
#print(geeExCount/500)
print(geeIndCountNotime/500)
```

 # Alternative condition (tx effect = 0.08), 40 clusters, no random treatment effect, with

```
geeIndResult = NULL
geeExResult = NULL
geeIndResultNotime = NULL
geeIndZscore = NULL
geeExZscore = NULL
geeIndZscoreNotime = NULL

for(i in 1:500) {
  design <- swDsn(clusters=c(10,10,10,10), extra.time=0, all.ctl.time0=TRUE)
  # mu1 is the difference between a0 and a1
  # tau: random intercept
  # eta: random treatment
  swGenData.nScalar <- swSim(design,family=gaussian(link="identity"), n=100, mu0=3,
                        mu1=3.08, time.effect=c(0,.2,.3,.4,.5), sigma=1, tau=2, eta=0,
                        rho=0, seed=i, retTimeOnTx=FALSE)

  #geeInd <- gee(response.var~tx.var+as.factor(time.var), data = swGenData.nScalar, family
= "gaussian",id = cluster.var, corstr = "independence")
  #geeIndResult[i] = geeInd$coefficients[[2]][1]
  #geeIndZscore[i] = summary(geeInd)$coefficients[2,5]

  #geeEx <- gee(response.var~tx.var+as.factor(time.var), data = swGenData.nScalar, family =
"gaussian",id = cluster.var, corstr = "exchangeable")
  #geeExResult[i] = geeEx$coefficients[[2]][1]
  #geeExZscore[i] = summary(geeEx)$coefficients[2,5]

  geeIndNotime <- gee(response.var~tx.var, data = swGenData.nScalar, family =
"gaussian",id = cluster.var, corstr = "independence")
  geeIndResultNotime[i] = geeIndNotime$coefficients[[2]][1]
  geeIndZscoreNotime[i] = summary(geeIndNotime)$coefficients[2,5]
}

#mean(geeIndResult)
#mean(geeExResult)
mean(geeIndResultNotime)

#sd(geeIndResult)
#sd(geeExResult)
sd(geeIndResultNotime)

# Compute the p-values
#geeIndCount = 0
#geeExCount = 0
geeIndCountNotime = 0
for (i in 1:500) {
  #if(abs(geeIndZscore[i]) >= 1.96) {
  #  geeIndCount = geeIndCount+1
  #}
```

```
  #  geeExCount = geeExCount+1
  #}
  if(abs(geeIndZscoreNotime[i]) >= 1.96) {
    geeIndCountNotime = geeIndCountNotime + 1
  }
}
#print(geeIndCount/500)
#print(geeExCount/500)
print(geeIndCountNotime/500)
```

 # Alternative condition (tx effect = 1), 40 clusters, with random treatment effect, time effect, correlation = 0

```
geeIndResult = NULL
geeExResult = NULL
geeIndResultNotime = NULL
geeIndZscore = NULL
geeExZscore = NULL
geeIndZscoreNotime = NULL

for(i in 1:500) {
  design <- swDsn(clusters=c(10,10,10,10), extra.time=0, all.ctl.time0=TRUE)
  # mu1 is the difference between a0 and a1
  # tau: random intercept
  # eta: random treatment
  swGenData.nScalar <- swSim(design,family=gaussian(link="identity"), n=100, mu0=3,
                        mu1=4, time.effect=c(0,.2,.3,.4,.5), sigma=1, tau=2, eta=2,
                        rho=0, seed=i, retTimeOnTx=FALSE)

  #geeInd <- gee(response.var~tx.var+as.factor(time.var), data = swGenData.nScalar, family
= "gaussian",id = cluster.var, corstr = "independence")
  #geeIndResult[i] = geeInd$coefficients[[2]][1]
  #geeIndZscore[i] = summary(geeInd)$coefficients[2,5]

  #geeEx <- gee(response.var~tx.var+as.factor(time.var), data = swGenData.nScalar, family =
"gaussian",id = cluster.var, corstr = "exchangeable")
  #geeExResult[i] = geeEx$coefficients[[2]][1]
  #geeExZscore[i] = summary(geeEx)$coefficients[2,5]

  geeIndNotime <- gee(response.var~tx.var, data = swGenData.nScalar, family =
"gaussian",id = cluster.var, corstr = "independence")
  geeIndResultNotime[i] = geeIndNotime$coefficients[[2]][1]
  geeIndZscoreNotime[i] = summary(geeIndNotime)$coefficients[2,5]
}

#mean(geeIndResult)
#mean(geeExResult)
mean(geeIndResultNotime)
```

```
#sd(geeIndResult)
#sd(geeExResult)
sd(geeIndResultNotime)

# Compute the p-values
#geeIndCount = 0
#geeExCount = 0
geeIndCountNotime = 0
for (i in 1:500) {
  #if(abs(geeIndZscore[i]) >= 1.96) {
  #  geeIndCount = geeIndCount+1
  #}
  #if(abs(geeExZscore[i]) >= 1.96) {
  #  geeExCount = geeExCount+1
  #}
  if(abs(geeIndZscoreNotime[i]) >= 1.96) {
    geeIndCountNotime = geeIndCountNotime + 1
  }
}
#print(geeIndCount/500)
#print(geeExCount/500)
print(geeIndCountNotime/500)
```

 # Alternative condition (tx effect = 1), 40 clusters, with random treatment effect, with time effect, correlation = 0.3

```
geeIndResult = NULL
geeExResult = NULL
geeIndResultNotime = NULL
geeIndZscore = NULL
geeExZscore = NULL
geeIndZscoreNotime = NULL

for(i in 1:500) {
  design <- swDsn(clusters=c(10,10,10,10), extra.time=0, all.ctl.time0=TRUE)
  # mu1 is the difference between a0 and a1
  # tau: random intercept
  # eta: random treatment
  swGenData.nScalar <- swSim(design,family=gaussian(link="identity"), n=100, mu0=3,
                        mu1=4, time.effect=c(0,.2,.3,.4,.5), sigma=1, tau=2, eta=2,
                        rho=0.3, seed=i, retTimeOnTx=FALSE)

  #geeInd <- gee(response.var~tx.var+as.factor(time.var), data = swGenData.nScalar, family
= "gaussian",id = cluster.var, corstr = "independence")
  #geeIndResult[i] = geeInd$coefficients[[2]][1]
  #geeIndZscore[i] = summary(geeInd)$coefficients[2,5]
```

```
  #geeEx <- gee(response.var~tx.var+as.factor(time.var), data = swGenData.nScalar, family =
"gaussian",id = cluster.var, corstr = "exchangeable")
  #geeExResult[i] = geeEx$coefficients[[2]][1]
  #geeExZscore[i] = summary(geeEx)$coefficients[2,5]

  geeIndNotime <- gee(response.var~tx.var, data = swGenData.nScalar, family =
"gaussian",id = cluster.var, corstr = "independence")
  geeIndResultNotime[i] = geeIndNotime$coefficients[[2]][1]
  geeIndZscoreNotime[i] = summary(geeIndNotime)$coefficients[2,5]
}

#mean(geeIndResult)
#mean(geeExResult)
mean(geeIndResultNotime)

#sd(geeIndResult)
#sd(geeExResult)
sd(geeIndResultNotime)

# Compute the p-values
#geeIndCount = 0
#geeExCount = 0
geeIndCountNotime = 0
for (i in 1:500) {
  #if(abs(geeIndZscore[i]) >= 1.96) {
  #  geeIndCount = geeIndCount+1
  #}
  #if(abs(geeExZscore[i]) >= 1.96) {
  #  geeExCount = geeExCount+1
  #}
  if(abs(geeIndZscoreNotime[i]) >= 1.96) {
    geeIndCountNotime = geeIndCountNotime + 1
  }
}
#print(geeIndCount/500)
#print(geeExCount/500)
print(geeIndCountNotime/500)
```

 # GLMM

 # Null condition, 20 clusters, no random effects, corr = NA

```
gm1a = NULL
gm2a = NULL
gmz1 = NULL
gmz2 = NULL
```

```
gm4a = NULL
gmz3 = NULL
gmz4 = NULL
for(i in 1:500) {
  design <- swDsn(clusters=c(5,5,5,5), extra.time=0, all.ctl.time0=TRUE)
  # mu1 is the difference between a0 and a1
  # tau: random intercept
  # eta: random treatment
  swGenData.nScalar <- swSim(design,family=gaussian(link="identity"), n=100, mu0=3,
                        mu1=3, time.effect=c(0,.2,.3,.4,.5), sigma=1, tau=2, eta=0,
                        rho=0, seed=i, retTimeOnTx=FALSE)
  # time effect, no random treatment effect
  glmod1 <- lmer(response.var ~ tx.var+as.factor(time.var)+(1|cluster.var), data =
swGenData.nScalar)
  gm1a[i] = fixef(glmod1)[[2]][1]
  gmz1[i] = summary(glmod1)$coefficients[2,3]
  # intercept and slope are correlated
  glmod2 <- lmer(response.var ~ tx.var+as.factor(time.var)+(1+tx.var|cluster.var), data =
swGenData.nScalar)
  gm2a[i] = fixef(glmod2)[[2]][1]
  gmz2[i] = summary(glmod2)$coefficients[2,3]
  # no time effect, no random treatment effect
  glmod3 <- lmer(response.var ~ tx.var+(1|cluster.var), data = swGenData.nScalar)
  gm3a[i] = fixef(glmod3)[[2]][1]
  gmz3[i] = summary(glmod3)$coefficients[2,3]
  # intercept and slope are not correlated
  glmod4 <- lmer(response.var ~ tx.var+ as.factor(time.var) + (1|cluster.var) + (tx.var-
1|cluster.var), data = swGenData.nScalar)
  gm4a[i] = fixef(glmod4)[[2]][1]
  gmz4[i] = summary(glmod4)$coefficients[2,3]
}

mean(gm1a)
mean(gm2a)
mean(gm3a)
mean(gm4a)

sd(gm1a)
sd(gm2a)
sd(gm3a)
sd(gm4a)
# Compute the p-values
glmcount1 = 0
glmcount2 = 0
glmcount3 = 0
for (i in 1:500) {
  if(abs(gmz1[i]) >= 1.96) {
    glmcount1 = glmcount1+1
  }
```

```
  glmcount2 = glmcount2+1
 }
 if(abs(gmz3[i]) >= 1.96) {
  glmcount3 = glmcount3+1
 }
}
print(glmcount1/500)
print(glmcount2/500)
print(glmcount3/500)
```

 # Null condition, 20 clusters, with random treatment effect, time effect, correlation = 0

```
gm1a = NULL
gm2a = NULL
gmz1 = NULL
gmz2 = NULL
gm3a = NULL
gm4a = NULL
gmz3 = NULL
gmz4 = NULL
for(i in 1:500) {
 design <- swDsn(clusters=c(5,5,5,5), extra.time=0, all.ctl.time0=TRUE)
 # mu1 is the difference between a0 and a1
 # tau: random intercept
 # eta: random treatment
 swGenData.nScalar <- swSim(design,family=gaussian(link="identity"), n=100, mu0=3,
                  mu1=3, time.effect=c(0,.2,.3,.4,.5), sigma=1, tau=2, eta=2,
                  rho=0, seed=i, retTimeOnTx=FALSE)
 # time effect, no random treatment effect
 glmod1 <- lmer(response.var ~ tx.var+as.factor(time.var)+(1|cluster.var), data =
swGenData.nScalar)
 gm1a[i] = fixef(glmod1)[[2]][1]
 gmz1[i] = summary(glmod1)$coefficients[2,3]
 # intercept and slope are correlated
 glmod2 <- lmer(response.var ~ tx.var+as.factor(time.var)+(1+tx.var|cluster.var), data =
swGenData.nScalar)
 gm2a[i] = fixef(glmod2)[[2]][1]
 gmz2[i] = summary(glmod2)$coefficients[2,3]
 # no time effect, no random treatment effect
 glmod3 <- lmer(response.var ~ tx.var+(1|cluster.var), data = swGenData.nScalar)
 gm3a[i] = fixef(glmod3)[[2]][1]
 gmz3[i] = summary(glmod3)$coefficients[2,3]
 # intercept and slope are not correlated
 glmod4 <- lmer(response.var ~ tx.var+ as.factor(time.var) + (1|cluster.var) + (tx.var-
1|cluster.var), data = swGenData.nScalar)
 gm4a[i] = fixef(glmod4)[[2]][1]
 gmz4[i] = summary(glmod4)$coefficients[2,3]
}
```

```
mean(gm1a)
mean(gm2a)
mean(gm3a)
mean(gm4a)
sd(gm1a)
sd(gm2a)
sd(gm3a)
sd(gm4a)

# Compute the p-values
sum(abs(gmz1)>=1.96)/500
sum(abs(gmz2)>=1.96)/500
sum(abs(gmz3)>=1.96)/500
sum(abs(gmz4)>=1.96)/500
```

 # Null condition, 20 clusters, with random treatment effect, time effect, correlation = 0.3

```
gm1a = NULL
gm2a = NULL
gmz1 = NULL
gmz2 = NULL
gm3a = NULL
gm4a = NULL
gmz3 = NULL
gmz4 = NULL
for(i in 1:500) {
  design <- swDsn(clusters=c(5,5,5,5), extra.time=0, all.ctl.time0=TRUE)
  # mu1 is the difference between a0 and a1
  # tau: random intercept
  # eta: random treatment
  swGenData.nScalar <- swSim(design,family=gaussian(link="identity"), n=100, mu0=3,
                        mu1=3, time.effect=c(0,.2,.3,.4,.5), sigma=1, tau=2, eta=2,
                        rho=0.3, seed=i, retTimeOnTx=FALSE)
  # time effect, no random treatment effect
  glmod1 <- lmer(response.var ~ tx.var+as.factor(time.var)+(1|cluster.var), data =
swGenData.nScalar)
  gm1a[i] = fixef(glmod1)[[2]][1]
  gmz1[i] = summary(glmod1)$coefficients[2,3]
  # intercept and slope are correlated
  glmod2 <- lmer(response.var ~ tx.var+as.factor(time.var)+(1+tx.var|cluster.var), data =
swGenData.nScalar)
  gm2a[i] = fixef(glmod2)[[2]][1]
  gmz2[i] = summary(glmod2)$coefficients[2,3]
  # no time effect, no random treatment effect
  glmod3 <- lmer(response.var ~ tx.var+(1|cluster.var), data = swGenData.nScalar)
  gm3a[i] = fixef(glmod3)[[2]][1]
```

```
  # intercept and slope are not correlated
  glmod4 <- lmer(response.var ~ tx.var+ as.factor(time.var) + (1|cluster.var) + (tx.var-
1|cluster.var), data = swGenData.nScalar)
  gm4a[i] = fixef(glmod4)[[2]][1]
  gmz4[i] = summary(glmod4)$coefficients[2,3]
}

mean(gm1a)
mean(gm2a)
mean(gm3a)
mean(gm4a)
sd(gm1a)
sd(gm2a)
sd(gm3a)
sd(gm4a)

# Compute the p-values
sum(abs(gmz1)>=1.96)/500
sum(abs(gmz2)>=1.96)/500
sum(abs(gmz3)>=1.96)/500
sum(abs(gmz4)>=1.96)/500
```

 # Null condition, 40 clusters, no random treatment effect, with time effect, correlation = NA

```
gm1a = NULL
gm2a = NULL
gmz1 = NULL
gmz2 = NULL
gm3a = NULL
gm4a = NULL
gmz3 = NULL
gmz4 = NULL
for(i in 1:500) {
  design <- swDsn(clusters=c(10,10,10,10), extra.time=0, all.ctl.time0=TRUE)
  # mu1 is the difference between a0 and a1
  # tau: random intercept
  # eta: random treatment
  swGenData.nScalar <- swSim(design,family=gaussian(link="identity"), n=100, mu0=3,
                        mu1=3, time.effect=c(0,.2,.3,.4,.5), sigma=1, tau=2, eta=0,
                        rho=0, seed=i, retTimeOnTx=FALSE)
  # time effect, no random treatment effect
  glmod1 <- glmer(response.var ~ tx.var+as.factor(time.var)+(1|cluster.var), data =
swGenData.nScalar, gaussian, nAGQ = 10)
  gm1a[i] = fixef(glmod1)[[2]][1]
  gm1se[i] = summary(glmod1)$coefficients[2,2]
  gmz1[i] = summary(glmod1)$coefficients[2,3]
```

```
  glmod2 <- glmer(response.var ~ tx.var+as.factor(time.var)+(1+tx.var|cluster.var), data =
swGenData.nScalar, gaussian, nAGQ = 10)
  gm2a[i] = fixef(glmod2)[[2]][1]
  gm2se[i] = summary(glmod2)$coefficients[2,2]
  gmz2[i] = summary(glmod2)$coefficients[2,3]
  # no time effect, no random treatment effect
  glmod3 <- glmer(response.var ~ tx.var+(1|cluster.var), data = swGenData.nScalar, gaussian,
nAGQ = 10)
  gm3a[i] = fixef(glmod3)[[2]][1]
  gm3se[i] = summary(glmod3)$coefficients[2,2]
  gmz3[i] = summary(glmod3)$coefficients[2,3]
  # intercept and slope are not correlated
  #glmod4 <- lmer(response.var ~ tx.var+ as.factor(time.var) + (1|cluster.var) +
(tx|cluster.var), data = swGenData.nScalar, gaussian, nAGQ = 10)
  #gm4a[i] = fixef(glmod4)[[2]][1]
  #gm4se[i] = summary(glmod4)$coefficients[2,2]
  #gmz4[i] = summary(glmod4)$coefficients[2,3]
}

mean(gm1a)
mean(gm2a)
mean(gm3a)

sd(gm1a)
sd(gm2a)
sd(gm3a)

# Compute the p-values
glmcount1 = 0
glmcount2 = 0
glmcount3 = 0
for (i in 1:500) {
  if(abs(gmz1[i]) >= 1.96) {
    glmcount1 = glmcount1+1
  }
  if(abs(gmz2[i]) >= 1.96) {
    glmcount2 = glmcount2+1
  }
  if(abs(gmz3[i]) >= 1.96) {
    glmcount3 = glmcount3+1
  }
}
print(glmcount1/500)
print(glmcount2/500)
print(glmcount3/500)
```

 # Null condition: 40 clusters with random treatment effect, time effect, correlation = 0

```
gm1a = NULL
gm2a = NULL
gmz1 = NULL
gmz2 = NULL
gm3a = NULL
gm4a = NULL
gmz3 = NULL
gmz4 = NULL
for(i in 1:500) {
  design <- swDsn(clusters=c(10,10,10,10), extra.time=0, all.ctl.time0=TRUE)
  # mu1 is the difference between a0 and a1
  # tau: random intercept
  # eta: random treatment
  swGenData.nScalar <- swSim(design,family=gaussian(link="identity"), n=100, mu0=3,
                        mu1=3, time.effect=c(0,.2,.3,.4,.5), sigma=1, tau=2, eta=0,
                        rho=0, seed=i, retTimeOnTx=FALSE)
  # time effect, no random treatment effect
  glmod1 <- glmer(response.var ~ tx.var+as.factor(time.var)+(1|cluster.var), data =
swGenData.nScalar, gaussian, nAGQ = 10)
  gm1a[i] = fixef(glmod1)[[2]][1]
  gm1se[i] = summary(glmod1)$coefficients[2,2]
  gmz1[i] = summary(glmod1)$coefficients[2,3]
  # intercept and slope are correlated
  glmod2 <- glmer(response.var ~ tx.var+as.factor(time.var)+(1+tx.var|cluster.var), data =
swGenData.nScalar, gaussian, nAGQ = 10)
  gm2a[i] = fixef(glmod2)[[2]][1]
  gm2se[i] = summary(glmod2)$coefficients[2,2]
  gmz2[i] = summary(glmod2)$coefficients[2,3]
  # no time effect, no random treatment effect
  glmod3 <- glmer(response.var ~ tx.var+(1|cluster.var), data = swGenData.nScalar, gaussian,
nAGQ = 10)
  gm3a[i] = fixef(glmod3)[[2]][1]
  gm3se[i] = summary(glmod3)$coefficients[2,2]
  gmz3[i] = summary(glmod3)$coefficients[2,3]
  # intercept and slope are not correlated
  #glmod4 <- lmer(response.var ~ tx.var+ as.factor(time.var) + (1|cluster.var) +
(tx|cluster.var), data = swGenData.nScalar, gaussian, nAGQ = 10)
  #gm4a[i] = fixef(glmod4)[[2]][1]
  #gm4se[i] = summary(glmod4)$coefficients[2,2]
  #gmz4[i] = summary(glmod4)$coefficients[2,3]
}

mean(gm1a)
mean(gm2a)
mean(gm3a)

sd(gm1a)
sd(gm2a)
```

```
# Compute the p-values
glmcount1 = 0
glmcount2 = 0
glmcount3 = 0
for (i in 1:500) {
  if(abs(gmz1[i]) >= 1.96) {
    glmcount1 = glmcount1+1
  }
  if(abs(gmz2[i]) >= 1.96) {
    glmcount2 = glmcount2+1
  }
  if(abs(gmz3[i]) >= 1.96) {
    glmcount3 = glmcount3+1
  }
}
print(glmcount1/500)
print(glmcount2/500)
print(glmcount3/500)
```

 # Null condition, 40 cluster, with random treatment effect, time effect, correlation = 0

```
gm1a = NULL
gm2a = NULL
gmz1 = NULL
gmz2 = NULL
gm3a = NULL
gm4a = NULL
gmz3 = NULL
gmz4 = NULL
for(i in 1:500) {
  design <- swDsn(clusters=c(10,10,10,10), extra.time=0, all.ctl.time0=TRUE)
  # mu1 is the difference between a0 and a1
  # tau: random intercept
  # eta: random treatment
  swGenData.nScalar <- swSim(design,family=gaussian(link="identity"), n=100, mu0=3,
                        mu1=3, time.effect=c(0,.2,.3,.4,.5), sigma=1, tau=2, eta=2,
                        rho=0, seed=i, retTimeOnTx=FALSE)
  # time effect, no random treatment effect
  glmod1 <- glmer(response.var ~ tx.var+as.factor(time.var)+(1|cluster.var), data =
swGenData.nScalar, gaussian, nAGQ = 10)
  gm1a[i] = fixef(glmod1)[[2]][1]
  gm1se[i] = summary(glmod1)$coefficients[2,2]
  gmz1[i] = summary(glmod1)$coefficients[2,3]
  # intercept and slope are correlated
  glmod2 <- glmer(response.var ~ tx.var+as.factor(time.var)+(1+tx.var|cluster.var), data =
```

```
  gm2a[i] = fixef(glmod2)[[2]][1]
  gm2se[i] = summary(glmod2)$coefficients[2,2]
  gmz2[i] = summary(glmod2)$coefficients[2,3]
  # no time effect, no random treatment effect
  glmod3 <- glmer(response.var ~ tx.var+(1|cluster.var), data = swGenData.nScalar, gaussian,
nAGQ = 10)
  gm3a[i] = fixef(glmod3)[[2]][1]
  gm3se[i] = summary(glmod3)$coefficients[2,2]
  gmz3[i] = summary(glmod3)$coefficients[2,3]
  # intercept and slope are not correlated
  #glmod4 <- lmer(response.var ~ tx.var+ as.factor(time.var) + (1|cluster.var) +
(tx|cluster.var), data = swGenData.nScalar, gaussian, nAGQ = 10)
  #gm4a[i] = fixef(glmod4)[[2]][1]
  #gm4se[i] = summary(glmod4)$coefficients[2,2]
  #gmz4[i] = summary(glmod4)$coefficients[2,3]
}

mean(gm1a)
mean(gm2a)
mean(gm3a)

sd(gm1a)
sd(gm2a)
sd(gm3a)

# Compute the p-values
glmcount1 = 0
glmcount2 = 0
glmcount3 = 0
for (i in 1:500) {
  if(abs(gmz1[i]) >= 1.96) {
    glmcount1 = glmcount1+1
  }
  if(abs(gmz2[i]) >= 1.96) {
    glmcount2 = glmcount2+1
  }
  if(abs(gmz3[i]) >= 1.96) {
    glmcount3 = glmcount3+1
  }
}
print(glmcount1/500)
print(glmcount2/500)
print(glmcount3/500)
```

 # Null condition, 40 cluster, with random treatment effect, time effect, correlation = 0.3

```
gm1a = NULL
```

```
gmz1 = NULL
gmz2 = NULL
gm3a = NULL
gm4a = NULL
gmz3 = NULL
gmz4 = NULL
for(i in 1:500) {
  design <- swDsn(clusters=c(10,10,10,10), extra.time=0, all.ctl.time0=TRUE)
  # mu1 is the difference between a0 and a1
  # tau: random intercept
  # eta: random treatment
  swGenData.nScalar <- swSim(design,family=gaussian(link="identity"), n=100, mu0=3,
                        mu1=3, time.effect=c(0,.2,.3,.4,.5), sigma=1, tau=2, eta=0,
                        rho=0.3, seed=i, retTimeOnTx=FALSE)
  # time effect, no random treatment effect
  glmod1 <- glmer(response.var ~ tx.var+as.factor(time.var)+(1|cluster.var), data =
swGenData.nScalar, gaussian, nAGQ = 10)
  gm1a[i] = fixef(glmod1)[[2]][1]
  gm1se[i] = summary(glmod1)$coefficients[2,2]
  gmz1[i] = summary(glmod1)$coefficients[2,3]
  # intercept and slope are correlated
  glmod2 <- glmer(response.var ~ tx.var+as.factor(time.var)+(1+tx.var|cluster.var), data =
swGenData.nScalar, gaussian, nAGQ = 10)
  gm2a[i] = fixef(glmod2)[[2]][1]
  gm2se[i] = summary(glmod2)$coefficients[2,2]
  gmz2[i] = summary(glmod2)$coefficients[2,3]
  # no time effect, no random treatment effect
  glmod3 <- glmer(response.var ~ tx.var+(1|cluster.var), data = swGenData.nScalar, gaussian,
nAGQ = 10)
  gm3a[i] = fixef(glmod3)[[2]][1]
  gm3se[i] = summary(glmod3)$coefficients[2,2]
  gmz3[i] = summary(glmod3)$coefficients[2,3]
  # intercept and slope are not correlated
  #glmod4 <- lmer(response.var ~ tx.var+ as.factor(time.var) + (1|cluster.var) +
(tx|cluster.var), data = swGenData.nScalar, gaussian, nAGQ = 10)
  #gm4a[i] = fixef(glmod4)[[2]][1]
  #gm4se[i] = summary(glmod4)$coefficients[2,2]
  #gmz4[i] = summary(glmod4)$coefficients[2,3]
}

mean(gm1a)
mean(gm2a)
mean(gm3a)
mean(gm4a)

sd(gm1a)
sd(gm2a)
sd(gm3a)
sd(gm4a)
```

```
# Compute the p-values
glmcount1 = 0
glmcount2 = 0
glmcount3 = 0
glmcount4 = 0
for (i in 1:500) {
  if(abs(gmz1[i]) >= 1.96) {
    glmcount1 = glmcount1+1
  }
  if(abs(gmz2[i]) >= 1.96) {
    glmcount2 = glmcount2+1
  }
  if(abs(gmz3[i]) >= 1.96) {
    glmcount3 = glmcount3+1
  }
if(abs(gmz4[i]) >= 1.96) {
    glmcount4 = glmcount4+1
  }
}
print(glmcount1/500)
print(glmcount2/500)
print(glmcount3/500)
print(glmcount4/500)
```

 # Alternative condition, 40 cluster, tx effect = 1=0.08, no random treatment effect, correlation = NA

```
gm1a = NULL
gm2a = NULL
gmz1 = NULL
gmz2 = NULL
gm3a = NULL
gm4a = NULL
gmz3 = NULL
gmz4 = NULL
for(i in 1:500) {
  design <- swDsn(clusters=c(10,10,10,10), extra.time=0, all.ctl.time0=TRUE)
  # mu1 is the difference between a0 and a1
  # tau: random intercept
  # eta: random treatment
  swGenData.nScalar <- swSim(design,family=gaussian(link="identity"), n=100, mu0=3,
                        mu1=3.08, time.effect=c(0,.2,.3,.4,.5), sigma=1, tau=2, eta=0,
                        rho=0, seed=i, retTimeOnTx=FALSE)
  # time effect, no random treatment effect
  glmod1 <- lmer(response.var ~ tx.var+as.factor(time.var)+(1|cluster.var), data =
swGenData.nScalar)
  gm1a[i] = fixef(glmod1)[[2]][1]
  gm1se[i] = summary(glmod1)$coefficients[2,2]
  gmz1[i] = summary(glmod1)$coefficients[2,3]
```

```
  glmod2 <- lmer(response.var ~ tx.var+as.factor(time.var)+(tx.var|cluster.var), data =
swGenData.nScalar)
  gm2a[i] = fixef(glmod2)[[2]][1]
  gm2se[i] = summary(glmod2)$coefficients[2,2]
  gmz2[i] = summary(glmod2)$coefficients[2,3]
  # no time effect, no random treatment effect
  glmod3 <- lmer(response.var ~ tx.var+(1|cluster.var), data = swGenData.nScalar)
  gm3a[i] = fixef(glmod3)[[2]][1]
  gm3se[i] = summary(glmod3)$coefficients[2,2]
  gmz3[i] = summary(glmod3)$coefficients[2,3]
  # intercept and slope are not correlated
  glmod4 <- lmer(response.var ~ tx.var+ as.factor(time.var) + (1|cluster.var) + (tx.var-
1|cluster.var), data = swGenData.nScalar)
  gm4a[i] = fixef(glmod4)[[2]][1]
  gm4se[i] = summary(glmod4)$coefficients[2,2]
  gmz4[i] = summary(glmod4)$coefficients[2,3]
}

mean(gm1a)
mean(gm2a)
mean(gm3a)
mean(gm4a)

sd(gm1a)
sd(gm2a)
sd(gm3a)
sd(gm4a)

# Compute the p-values
glmcount1 = 0
glmcount2 = 0
glmcount3 = 0
glmcount4 = 0
for (i in 1:500) {
  if(abs(gmz1[i]) >= 1.96) {
    glmcount1 = glmcount1+1
  }
  if(abs(gmz2[i]) >= 1.96) {
    glmcount2 = glmcount2+1
  }
  if(abs(gmz3[i]) >= 1.96) {
    glmcount3 = glmcount3+1
  }
  if(abs(gmz4[i]) >= 1.96) {
    glmcount4 = glmcount4+1
  }
}
print(glmcount1/500)
print(glmcount2/500)
```

```
print(glmcount4/500)
```

 # Alternative condition, 40 cluster, tx effect = 0.8, with random treatment effect, correlation = 0

```
gm1a = NULL
gm2a = NULL
gmz1 = NULL
gmz2 = NULL
gm3a = NULL
gm4a = NULL
gmz3 = NULL
gmz4 = NULL
for(i in 1:500) {
  design <- swDsn(clusters=c(10,10,10,10), extra.time=0, all.ctl.time0=TRUE)
  # mu1 is the difference between a0 and a1
  # tau: random intercept
  # eta: random treatment
  swGenData.nScalar <- swSim(design,family=gaussian(link="identity"), n=100, mu0=3,
                        mu1=3.8, time.effect=c(0,.2,.3,.4,.5), sigma=1, tau=2, eta=2,
                        rho=0, seed=i, retTimeOnTx=FALSE)
  # time effect, no random treatment effect
  glmod1 <- lmer(response.var ~ tx.var+as.factor(time.var)+(1|cluster.var), data =
swGenData.nScalar)
  gm1a[i] = fixef(glmod1)[[2]][1]
  gm1se[i] = summary(glmod1)$coefficients[2,2]
  gmz1[i] = summary(glmod1)$coefficients[2,3]
  # intercept and slope are correlated
  glmod2 <- lmer(response.var ~ tx.var+as.factor(time.var)+(tx.var|cluster.var), data =
swGenData.nScalar)
  gm2a[i] = fixef(glmod2)[[2]][1]
  gm2se[i] = summary(glmod2)$coefficients[2,2]
  gmz2[i] = summary(glmod2)$coefficients[2,3]
  # no time effect, no random treatment effect
  glmod3 <- lmer(response.var ~ tx.var+(1|cluster.var), data = swGenData.nScalar)
  gm3a[i] = fixef(glmod3)[[2]][1]
  gm3se[i] = summary(glmod3)$coefficients[2,2]
  gmz3[i] = summary(glmod3)$coefficients[2,3]
  # intercept and slope are not correlated
  glmod4 <- lmer(response.var ~ tx.var+ as.factor(time.var) + (1|cluster.var) + (tx.var-
1|cluster.var), data = swGenData.nScalar)
  gm4a[i] = fixef(glmod4)[[2]][1]
  gm4se[i] = summary(glmod4)$coefficients[2,2]
  gmz4[i] = summary(glmod4)$coefficients[2,3]
}

mean(gm1a)
```

```
mean(gm3a)
mean(gm4a)

sd(gm1a)
sd(gm2a)
sd(gm3a)
sd(gm4a)

# Compute the p-values
glmcount1 = 0
glmcount2 = 0
glmcount3 = 0
glmcount4 = 0
for (i in 1:500) {
  if(abs(gmz1[i]) >= 1.96) {
    glmcount1 = glmcount1+1
  }
  if(abs(gmz2[i]) >= 1.96) {
    glmcount2 = glmcount2+1
  }
  if(abs(gmz3[i]) >= 1.96) {
    glmcount3 = glmcount3+1
  }
  if(abs(gmz4[i]) >= 1.96) {
    glmcount4 = glmcount4+1
  }
}
print(glmcount1/500)
print(glmcount2/500)
print(glmcount3/500)
print(glmcount4/500)
```

 # Alternative condition, 40 cluster, tx effect = 0.8, with random treatment effect, correlation = 0.3

```
gm1a = NULL
gm2a = NULL
gmz1 = NULL
gmz2 = NULL
gm3a = NULL
gm4a = NULL
gmz3 = NULL
gmz4 = NULL
for(i in 1:500) {
  design <- swDsn(clusters=c(10,10,10,10), extra.time=0, all.ctl.time0=TRUE)
  # mu1 is the difference between a0 and a1
  # tau: random intercept
  # eta: random treatment
```

```
                    mu1=3.8, time.effect=c(0,.2,.3,.4,.5), sigma=1, tau=2, eta=2,
                    rho=0.3, seed=i, retTimeOnTx=FALSE)
  # time effect, no random treatment effect
  glmod1 <- lmer(response.var ~ tx.var+as.factor(time.var)+(1|cluster.var), data =
swGenData.nScalar)
  gm1a[i] = fixef(glmod1)[[2]][1]
  gm1se[i] = summary(glmod1)$coefficients[2,2]
  gmz1[i] = summary(glmod1)$coefficients[2,3]
  # intercept and slope are correlated
  glmod2 <- lmer(response.var ~ tx.var+as.factor(time.var)+(tx.var|cluster.var), data =
swGenData.nScalar)
  gm2a[i] = fixef(glmod2)[[2]][1]
  gm2se[i] = summary(glmod2)$coefficients[2,2]
  gmz2[i] = summary(glmod2)$coefficients[2,3]
  # no time effect, no random treatment effect
  glmod3 <- lmer(response.var ~ tx.var+(1|cluster.var), data = swGenData.nScalar)
  gm3a[i] = fixef(glmod3)[[2]][1]
  gm3se[i] = summary(glmod3)$coefficients[2,2]
  gmz3[i] = summary(glmod3)$coefficients[2,3]
  # intercept and slope are not correlated
  glmod4 <- lmer(response.var ~ tx.var+ as.factor(time.var) + (1|cluster.var) + (tx.var-
1|cluster.var), data = swGenData.nScalar)
  gm4a[i] = fixef(glmod4)[[2]][1]
  gm4se[i] = summary(glmod4)$coefficients[2,2]
  gmz4[i] = summary(glmod4)$coefficients[2,3]
}

mean(gm1a)
mean(gm2a)
mean(gm3a)
mean(gm4a)

sd(gm1a)
sd(gm2a)
sd(gm3a)
sd(gm4a)

# Compute the p-values
glmcount1 = 0
glmcount2 = 0
glmcount3 = 0
glmcount4 = 0
for (i in 1:500) {
  if(abs(gmz1[i]) >= 1.96) {
    glmcount1 = glmcount1+1
  }
  if(abs(gmz2[i]) >= 1.96) {
    glmcount2 = glmcount2+1
  }
```

```
   glmcount3 = glmcount3+1
  }
  if(abs(gmz4[i]) >= 1.96) {
   glmcount4 = glmcount4+1
  }
}
print(glmcount1/500)
print(glmcount2/500)
print(glmcount3/500)
print(glmcount4/500)
```

PERMUTATION ANALYSIS

GEE

```
library(swCRTdesign)
library(gee)
```

S1

```
myfunc <- function(n, seed) {
  design <- swDsn(clusters=c(5,5,5,5), extra.time=0, all.ctl.time0=TRUE)
  # first generate the original dataset with a time effect and 0 treatment effect,
  # rho is the correlation between slope and intercept
  swGenData.nScalar <- swSim(design,family=gaussian(link="identity"), n=10, mu0=3,
                        mu1=3, time.effect=c(0,.2,.3,.4,.5), sigma=1, tau=2, eta=0,
                        rho=0, seed = seed, retTimeOnTx=FALSE)
  data = swSummary(response.var, tx.var, time.var, cluster.var, swGenData.nScalar,
type="mean", digits=3)$swDsn
  newdata = NULL
  listn=c(1:n)
  x = sample(listn)
  newdata=data[x,]
  return(newdata)
}


checkfunc = function(numberOfPermutation, nCluster, seed) {
  design <- swDsn(clusters=c(5,5,5,5), extra.time=0, all.ctl.time0=TRUE)
  # first generate the original dataset with a time effect and 0 treatment effect,
  # rho is the correlation between slope and intercept
  swdata <- swSim(design,family=gaussian(link="identity"), n=10, mu0=3,
              mu1=3, time.effect=c(0,.2,.3,.4,.5), sigma=1, tau=2, eta=0,
              rho=0, seed = seed, retTimeOnTx=FALSE)
```

```
  storep1_z = 0
  storep2 = 0
  storep2_z = 0
  storep3 = 0
  storep3_z = 0

  matrix = myfunc(nCluster, seed)

  mod1 <- gee(response.var~tx.var+as.factor(time.var), data = swdata, family = "gaussian",id
= cluster.var, corstr = "independence")
  result1 = mod1$coefficients[[2]][1]
  zscore1 = summary(mod1)$coefficients[2,5]

mod2 <- gee(response.var~tx.var+as.factor(time.var), data = swdata, family = "gaussian",id =
cluster.var, corstr = "exchangeable")
  result2 = mod2$coefficients[[2]][1]
  zscore2 = summary(mod2)$coefficients[2,5]

mod3 <- gee(response.var~tx.var, data = swdata, family = "gaussian",id = cluster.var, corstr
= "independence")
  result3 = mod3$coefficients[[2]][1]
  zscore3 = summary(mod3)$coefficients[2,5]

  dataPerm = swdata # copy the original dataset

  result1_1 = NULL
  zscore1_1 = NULL
  result2_1 = NULL
  zscore2_1 = NULL
  result3_1 = NULL
  zscore3_1 = NULL

  for(i in 1:numberOfPermutation) { # permutation starts
    newtx=NULL
    newdata = NULL
    listn=c(1:nCluster)
    x = sample(listn)
    newdata = matrix[x,]
    for(a in 1:nrow(newdata)) {
      for(b in 1:ncol(newdata)){
        if(newdata[a,b]==0) {
          newtx = append(newtx, rep(0,10))
        } else{
          newtx = append(newtx, rep(1,10))
        }
      }
    }

    dataPerm$tx.var = as.vector(t(newtx))
```

```
    mod1_1 <- gee(response.var~tx.var+as.factor(time.var), data = dataPerm, family =
"gaussian",id = cluster.var, corstr = "independence")
    result1_1[i] = mod1_1$coefficients[[2]][1]
    zscore1_1[i] = summary(mod1_1)$coefficients[2,5]

mod2_1 <- gee(response.var~tx.var+as.factor(time.var), data = dataPerm, family =
"gaussian",id = cluster.var, corstr = "exchangeable")
    result2_1[i] = mod2_1$coefficients[[2]][1]
    zscore2_1[i] = summary(mod2_1)$coefficients[2,5]

mod3_1 <- gee(response.var~tx.var, data = dataPerm, family = "gaussian",id = cluster.var,
corstr = "independence")
    result3_1[i] = mod3_1$coefficients[[2]][1]
    zscore3_1[i] = summary(mod3_1)$coefficients[2,5]

  }

  sort1 = sort(result1_1)
  sort2 = sort(result2_1)
  sort3 = sort(result3_1)

  sortz1 = sort(zscore1_1)
  sortz2 = sort(zscore2_1)
  sortz3 = sort(zscore3_1)

  # z score
  if((zscore1< sortz1[25]) | (zscore1> sortz1[975])) {
    storep1_z = 1}
  else {storep1_z = 0}
  if((zscore2< sortz2[25]) | (zscore2> sortz2[975])) {
    storep2_z = 1}
  else {storep2_z = 0}
  if((zscore3< sortz3[25]) | (zscore3> sortz3[975])) {
    storep3_z = 1}
  else {storep3_z = 0}

  # coefficients
  if((result1< sort1[25]) | (result1> sort1[975])) {
    storep1 = 1}
  else {storep1 = 0}
  if((result2< sort2[25]) | (result2> sort2[975])) {
    storep2 = 1}
  else {storep2 = 0}
  if((result3< sort3[25]) | (result3> sort3[975])) {
    storep3 = 1}
  else {storep3 = 0}

  return(c(storep1, storep1_z, storep2, storep2_z, storep3, storep3_z))
}
```

```
finalp1 = 0
finalp1_z = 0
finalp2 = 0
finalp2_z = 0
finalp3 = 0
finalp3_z = 0

for(i in 1:500) {
  resultList = NULL
  resultList = checkfunc(1000, 20, i)
  finalp1 = finalp1 + resultList[1]
  finalp1_z = finalp1_z + resultList[2]
  finalp2 = finalp2 + resultList[3]
  finalp2_z = finalp2_z + resultList[4]
  finalp3 = finalp3 + resultList[5]
  finalp3_z = finalp3_z + resultList[6]
}

print(finalp1/500)
print(finalp1_z/500)
print(finalp2/500)
print(finalp2_z/500)
print(finalp3/500)
print(finalp3_z/500)
```

 # S2

```
myfunc <- function(n, seed) {
  design <- swDsn(clusters=c(5,5,5,5), extra.time=0, all.ctl.time0=TRUE)
  # first generate the original dataset with a time effect and 0 treatment effect,
  # rho is the correlation between slope and intercept
  swGenData.nScalar <- swSim(design,family=gaussian(link="identity"), n=10, mu0=3,
                        mu1=3, time.effect=c(0,.2,.3,.4,.5), sigma=1, tau=2, eta=2,
                        rho=0, seed = seed, retTimeOnTx=FALSE)
  data = swSummary(response.var, tx.var, time.var, cluster.var, swGenData.nScalar,
type="mean", digits=3)$swDsn
  newdata = NULL
  listn=c(1:n)
  x = sample(listn)
  newdata=data[x,]
  return(newdata)
}


checkfunc = function(numberOfPermutation, nCluster, seed) {
```

```
  # first generate the original dataset with a time effect and 0 treatment effect,
  # rho is the correlation between slope and intercept
  swdata <- swSim(design,family=gaussian(link="identity"), n=10, mu0=3,
                  mu1=3, time.effect=c(0,.2,.3,.4,.5), sigma=1, tau=2, eta=2,
                  rho=0, seed = seed, retTimeOnTx=FALSE)

  storep1 = 0
  storep1_z = 0
  storep2 = 0
  storep2_z = 0
  storep3 = 0
  storep3_z = 0

  matrix = myfunc(nCluster, seed)

  mod1 <- gee(response.var~tx.var+as.factor(time.var), data = swdata, family = "gaussian",id
= cluster.var, corstr = "independence")
  result1 = mod1$coefficients[[2]][1]
  zscore1 = summary(mod1)$coefficients[2,5]

mod2 <- gee(response.var~tx.var+as.factor(time.var), data = swdata, family = "gaussian",id =
cluster.var, corstr = "exchangeable")
  result2 = mod2$coefficients[[2]][1]
  zscore2 = summary(mod2)$coefficients[2,5]

mod3 <- gee(response.var~tx.var, data = swdata, family = "gaussian",id = cluster.var, corstr
= "independence")
  result3 = mod3$coefficients[[2]][1]
  zscore3 = summary(mod3)$coefficients[2,5]

  dataPerm = swdata # copy the original dataset

  result1_1 = NULL
  zscore1_1 = NULL
  result2_1 = NULL
  zscore2_1 = NULL
  result3_1 = NULL
  zscore3_1 = NULL

  for(i in 1:numberOfPermutation) { # permutation starts
    newtx=NULL
    newdata = NULL
    listn=c(1:nCluster)
    x = sample(listn)
    newdata = matrix[x,]
    for(a in 1:nrow(newdata)) {
      for(b in 1:ncol(newdata)){
        if(newdata[a,b]==0) {
          newtx = append(newtx, rep(0,10))
```

```
        newtx = append(newtx, rep(1,10))
      }
    }
  }

  dataPerm$tx.var = as.vector(t(newtx))

  mod1_1 <- gee(response.var~tx.var+as.factor(time.var), data = dataPerm, family =
"gaussian",id = cluster.var, corstr = "independence")
  result1_1[i] = mod1_1$coefficients[[2]][1]
  zscore1_1[i] = summary(mod1_1)$coefficients[2,5]

mod2_1 <- gee(response.var~tx.var+as.factor(time.var), data = dataPerm, family =
"gaussian",id = cluster.var, corstr = "exchangeable")
  result2_1[i] = mod2_1$coefficients[[2]][1]
  zscore2_1[i] = summary(mod2_1)$coefficients[2,5]

mod3_1 <- gee(response.var~tx.var, data = dataPerm, family = "gaussian",id = cluster.var,
corstr = "independence")
  result3_1[i] = mod3_1$coefficients[[2]][1]
  zscore3_1[i] = summary(mod3_1)$coefficients[2,5]

 }

 sort1 = sort(result1_1)
 sort2 = sort(result2_1)
 sort3 = sort(result3_1)

 sortz1 = sort(zscore1_1)
 sortz2 = sort(zscore2_1)
 sortz3 = sort(zscore3_1)

 # z score
 if((zscore1< sortz1[25]) | (zscore1> sortz1[975])) {
  storep1_z = 1}
 else {storep1_z = 0}
 if((zscore2< sortz2[25]) | (zscore2> sortz2[975])) {
  storep2_z = 1}
 else {storep2_z = 0}
 if((zscore3< sortz3[25]) | (zscore3> sortz3[975])) {
  storep3_z = 1}
 else {storep3_z = 0}

 # coefficients
 if((result1< sort1[25]) | (result1> sort1[975])) {
  storep1 = 1}
 else {storep1 = 0}
 if((result2< sort2[25]) | (result2> sort2[975])) {
  storep2 = 1}
```

```
  if((result3< sort3[25]) | (result3> sort3[975])) {
    storep3 = 1}
  else {storep3 = 0}

  return(c(storep1, storep1_z, storep2, storep2_z, storep3, storep3_z))
}

finalp1 = 0
finalp1_z = 0
finalp2 = 0
finalp2_z = 0
finalp3 = 0
finalp3_z = 0

for(i in 1:500) {
  resultList = NULL
  resultList = checkfunc(1000, 20, i)
  finalp1 = finalp1 + resultList[1]
  finalp1_z = finalp1_z + resultList[2]
  finalp2 = finalp2 + resultList[3]
  finalp2_z = finalp2_z + resultList[4]
  finalp3 = finalp3 + resultList[5]
  finalp3_z = finalp3_z + resultList[6]
}

print(finalp1/500)
print(finalp1_z/500)
print(finalp2/500)
print(finalp2_z/500)
print(finalp3/500)
print(finalp3_z/500)
```

 # S3

```
myfunc <- function(n, seed) {
  design <- swDsn(clusters=c(5,5,5,5), extra.time=0, all.ctl.time0=TRUE)
  # first generate the original dataset with a time effect and 0 treatment effect,
  # rho is the correlation between slope and intercept
  swGenData.nScalar <- swSim(design,family=gaussian(link="identity"), n=10, mu0=3,
                        mu1=3, time.effect=c(0,.2,.3,.4,.5), sigma=1, tau=2, eta=2,
                        rho=0.3, seed = seed, retTimeOnTx=FALSE)
  data = swSummary(response.var, tx.var, time.var, cluster.var, swGenData.nScalar,
type="mean", digits=3)$swDsn
  newdata = NULL
  listn=c(1:n)
```

```
  newdata=data[x,]
  return(newdata)
}


checkfunc = function(numberOfPermutation, nCluster, seed) {
  design <- swDsn(clusters=c(5,5,5,5), extra.time=0, all.ctl.time0=TRUE)
  # first generate the original dataset with a time effect and 0 treatment effect,
  # rho is the correlation between slope and intercept
  swdata <- swSim(design,family=gaussian(link="identity"), n=10, mu0=3,
                  mu1=3, time.effect=c(0,.2,.3,.4,.5), sigma=1, tau=2, eta=2,
                  rho=0.3, seed = seed, retTimeOnTx=FALSE)

  storep1 = 0
  storep1_z = 0
  storep2 = 0
  storep2_z = 0
  storep3 = 0
  storep3_z = 0

  matrix = myfunc(nCluster, seed)

  mod1 <- gee(response.var~tx.var+as.factor(time.var), data = swdata, family = "gaussian",id
= cluster.var, corstr = "independence")
  result1 = mod1$coefficients[[2]][1]
  zscore1 = summary(mod1)$coefficients[2,5]

mod2 <- gee(response.var~tx.var+as.factor(time.var), data = swdata, family = "gaussian",id =
cluster.var, corstr = "exchangeable")
  result2 = mod2$coefficients[[2]][1]
  zscore2 = summary(mod2)$coefficients[2,5]

mod3 <- gee(response.var~tx.var, data = swdata, family = "gaussian",id = cluster.var, corstr
= "independence")
  result3 = mod3$coefficients[[2]][1]
  zscore3 = summary(mod3)$coefficients[2,5]

  dataPerm = swdata # copy the original dataset

  result1_1 = NULL
  zscore1_1 = NULL
  result2_1 = NULL
  zscore2_1 = NULL
  result3_1 = NULL
  zscore3_1 = NULL

  for(i in 1:numberOfPermutation) { # permutation starts
    newtx=NULL
    newdata = NULL
```

```
   x = sample(listn)
   newdata = matrix[x,]
   for(a in 1:nrow(newdata)) {
    for(b in 1:ncol(newdata)){
     if(newdata[a,b]==0) {
      newtx = append(newtx, rep(0,10))
     } else{
      newtx = append(newtx, rep(1,10))
     }
    }
   }

   dataPerm$tx.var = as.vector(t(newtx))

   mod1_1 <- gee(response.var~tx.var+as.factor(time.var), data = dataPerm, family =
"gaussian",id = cluster.var, corstr = "independence")
   result1_1[i] = mod1_1$coefficients[[2]][1]
   zscore1_1[i] = summary(mod1_1)$coefficients[2,5]

mod2_1 <- gee(response.var~tx.var+as.factor(time.var), data = dataPerm, family =
"gaussian",id = cluster.var, corstr = "exchangeable")
   result2_1[i] = mod2_1$coefficients[[2]][1]
   zscore2_1[i] = summary(mod2_1)$coefficients[2,5]

mod3_1 <- gee(response.var~tx.var, data = dataPerm, family = "gaussian",id = cluster.var,
corstr = "independence")
   result3_1[i] = mod3_1$coefficients[[2]][1]
   zscore3_1[i] = summary(mod3_1)$coefficients[2,5]
  }

  sort1 = sort(result1_1)
  sort2 = sort(result2_1)
  sort3 = sort(result3_1)

  sortz1 = sort(zscore1_1)
  sortz2 = sort(zscore2_1)
  sortz3 = sort(zscore3_1)

  # z score
  if((zscore1< sortz1[25]) | (zscore1> sortz1[975])) {
   storep1_z = 1}
  else {storep1_z = 0}
  if((zscore2< sortz2[25]) | (zscore2> sortz2[975])) {
   storep2_z = 1}
  else {storep2_z = 0}
  if((zscore3< sortz3[25]) | (zscore3> sortz3[975])) {
   storep3_z = 1}
  else {storep3_z = 0}

  # coefficient
```

```
  if((result1< sort1[25]) | (result1> sort1[975])) {
    storep1 = 1}
  else {storep1 = 0}
  if((result2< sort2[25]) | (result2> sort2[975])) {
    storep2 = 1}
  else {storep2 = 0}
  if((result3< sort3[25]) | (result3> sort3[975])) {
    storep3 = 1}
  else {storep3 = 0}

  return(c(storep1, storep1_z, storep2, storep2_z, storep3, storep3_z))
}

finalp1 = 0
finalp1_z = 0
finalp2 = 0
finalp2_z = 0
finalp3 = 0
finalp3_z = 0

for(i in 1:500) {
  resultList = NULL
  resultList = checkfunc(1000, 20, i)
  finalp1 = finalp1 + resultList[1]
  finalp1_z = finalp1_z + resultList[2]
  finalp2 = finalp2 + resultList[3]
  finalp2_z = finalp2_z + resultList[4]
  finalp3 = finalp3 + resultList[5]
  finalp3_z = finalp3_z + resultList[6]
}

print(finalp1/500)
print(finalp1_z/500)
print(finalp2/500)
print(finalp2_z/500)
print(finalp3/500)
print(finalp3_z/500)
```

 # S4

```
myfunc <- function(n, seed) {
  design <- swDsn(clusters=c(10,10,10,10), extra.time=0, all.ctl.time0=TRUE)
  # first generate the original dataset with a time effect and 0 treatment effect,
  # rho is the correlation between slope and intercept
  swGenData.nScalar <- swSim(design,family=gaussian(link="identity"), n=10, mu0=3,
```

```
                    rho=0, seed = seed, retTimeOnTx=FALSE)
  data = swSummary(response.var, tx.var, time.var, cluster.var, swGenData.nScalar,
type="mean", digits=3)$swDsn
  newdata = NULL
  listn=c(1:n)
  x = sample(listn)
  newdata=data[x,]
  return(newdata)
}


checkfunc = function(numberOfPermutation, nCluster, seed) {
  design <- swDsn(clusters=c(10,10,10,10), extra.time=0, all.ctl.time0=TRUE)
  # first generate the original dataset with a time effect and 0 treatment effect,
  # rho is the correlation between slope and intercept
  swdata <- swSim(design,family=gaussian(link="identity"), n=10, mu0=3,
            mu1=3, time.effect=c(0,.2,.3,.4,.5), sigma=1, tau=2, eta=0,
            rho=0, seed = seed, retTimeOnTx=FALSE)

  storep1 = 0
  storep1_z = 0
  storep2 = 0
  storep2_z = 0
  storep3 = 0
  storep3_z = 0

  matrix = myfunc(nCluster, seed)

  mod1 <- gee(response.var~tx.var+as.factor(time.var), data = swdata, family = "gaussian",id
= cluster.var, corstr = "independence")
  result1 = mod1$coefficients[[2]][1]
  zscore1 = summary(mod1)$coefficients[2,5]

mod2 <- gee(response.var~tx.var+as.factor(time.var), data = swdata, family = "gaussian",id =
cluster.var, corstr = "exchangeable")
  result2 = mod2$coefficients[[2]][1]
  zscore2 = summary(mod2)$coefficients[2,5]

mod3 <- gee(response.var~tx.var, data = swdata, family = "gaussian",id = cluster.var, corstr
= "independence")
  result3 = mod3$coefficients[[2]][1]
  zscore3 = summary(mod3)$coefficients[2,5]


  dataPerm = swdata # copy the original dataset

  result1_1 = NULL
  zscore1_1 = NULL
  result2_1 = NULL
```

```
  result3_1 = NULL
  zscore3_1 = NULL

  for(i in 1:numberOfPermutation) { # permutation starts
    newtx=NULL
    newdata = NULL
    listn=c(1:nCluster)
    x = sample(listn)
    newdata = matrix[x,]
    for(a in 1:nrow(newdata)) {
      for(b in 1:ncol(newdata)){
        if(newdata[a,b]==0) {
          newtx = append(newtx, rep(0,10))
        } else{
          newtx = append(newtx, rep(1,10))
        }
      }
    }

    dataPerm$tx.var = as.vector(t(newtx))

    mod1_1 <- gee(response.var~tx.var+as.factor(time.var), data = dataPerm, family =
"gaussian",id = cluster.var, corstr = "independence")
    result1_1[i] = mod1_1$coefficients[[2]][1]
    zscore1_1[i] = summary(mod1_1)$coefficients[2,5]

mod2_1 <- gee(response.var~tx.var+as.factor(time.var), data = dataPerm, family =
"gaussian",id = cluster.var, corstr = "exchangeable")
    result2_1[i] = mod2_1$coefficients[[2]][1]
    zscore2_1[i] = summary(mod2_1)$coefficients[2,5]

mod3_1 <- gee(response.var~tx.var, data = dataPerm, family = "gaussian",id = cluster.var,
corstr = "independence")
    result3_1[i] = mod3_1$coefficients[[2]][1]
    zscore3_1[i] = summary(mod3_1)$coefficients[2,5]

  }

  sort1 = sort(result1_1)
  sort2 = sort(result2_1)
  sort3 = sort(result3_1)

  sortz1 = sort(zscore1_1)
  sortz2 = sort(zscore2_1)
  sortz3 = sort(zscore3_1)

  # z score
  if((zscore1< sortz1[25]) | (zscore1> sortz1[975])) {
    storep1_z = 1}
```

```
  if((zscore2< sortz2[25]) | (zscore2> sortz2[975])) {
    storep2_z = 1}
  else {storep2_z = 0}
  if((zscore3< sortz3[25]) | (zscore3> sortz3[975])) {
    storep3_z = 1}
  else {storep3_z = 0}

  # coefficients
  if((result1< sort1[25]) | (result1> sort1[975])) {
    storep1 = 1}
  else {storep1 = 0}
  if((result2< sort2[25]) | (result2> sort2[975])) {
    storep2 = 1}
  else {storep2 = 0}
  if((result3< sort3[25]) | (result3> sort3[975])) {
    storep3 = 1}
  else {storep3 = 0}

  return(c(storep1, storep1_z, storep2, storep2_z, storep3, storep3_z))
}

finalp1 = 0
finalp1_z = 0
finalp2 = 0
finalp2_z = 0
finalp3 = 0
finalp3_z = 0

for(i in 1:500) {
  resultList = NULL
  resultList = checkfunc(1000, 40, i)
  finalp1 = finalp1 + resultList[1]
  finalp1_z = finalp1_z + resultList[2]
  finalp2 = finalp2 + resultList[3]
  finalp2_z = finalp2_z + resultList[4]
  finalp3 = finalp3 + resultList[5]
  finalp3_z = finalp3_z + resultList[6]
}

print(finalp1/500)
print(finalp1_z/500)
print(finalp2/500)
print(finalp2_z/500)
print(finalp3/500)
print(finalp3_z/500)
```

 # S5

```
myfunc <- function(n, seed) {
  design <- swDsn(clusters=c(10,10,10,10), extra.time=0, all.ctl.time0=TRUE)
  # first generate the original dataset with a time effect and 0 treatment effect,
  # rho is the correlation between slope and intercept
  swGenData.nScalar <- swSim(design,family=gaussian(link="identity"), n=10, mu0=3,
                        mu1=3, time.effect=c(0,.2,.3,.4,.5), sigma=1, tau=2, eta=2,
                        rho=0, seed = seed, retTimeOnTx=FALSE)
  data = swSummary(response.var, tx.var, time.var, cluster.var, swGenData.nScalar,
type="mean", digits=3)$swDsn
  newdata = NULL
  listn=c(1:n)
  x = sample(listn)
  newdata=data[x,]
  return(newdata)
}

checkfunc = function(numberOfPermutation, nCluster, seed) {
  design <- swDsn(clusters=c(10,10,10,10), extra.time=0, all.ctl.time0=TRUE)
  # first generate the original dataset with a time effect and 0 treatment effect,
  # rho is the correlation between slope and intercept
  swdata <- swSim(design,family=gaussian(link="identity"), n=10, mu0=3,
                mu1=3, time.effect=c(0,.2,.3,.4,.5), sigma=1, tau=2, eta=2,
                rho=0, seed = seed, retTimeOnTx=FALSE)

  storep1 = 0
  storep1_z = 0
  storep2 = 0
  storep2_z = 0
  storep3 = 0
  storep3_z = 0

  matrix = myfunc(nCluster, seed)

  mod1 <- gee(response.var~tx.var+as.factor(time.var), data = swdata, family = "gaussian",id
= cluster.var, corstr = "independence")
  result1 = mod1$coefficients[[2]][1]
  zscore1 = summary(mod1)$coefficients[2,5]

mod2 <- gee(response.var~tx.var+as.factor(time.var), data = swdata, family = "gaussian",id =
cluster.var, corstr = "exchangeable")
  result2 = mod2$coefficients[[2]][1]
  zscore2 = summary(mod2)$coefficients[2,5]

mod3 <- gee(response.var~tx.var, data = swdata, family = "gaussian",id = cluster.var, corstr
= "independence")
  result3 = mod3$coefficients[[2]][1]
```

```
  dataPerm = swdata # copy the original dataset

  result1_1 = NULL
  zscore1_1 = NULL
  result2_1 = NULL
  zscore2_1 = NULL
  result3_1 = NULL
  zscore3_1 = NULL

  for(i in 1:numberOfPermutation) { # permutation starts
    newtx=NULL
    newdata = NULL
    listn=c(1:nCluster)
    x = sample(listn)
    newdata = matrix[x,]
    for(a in 1:nrow(newdata)) {
      for(b in 1:ncol(newdata)){
        if(newdata[a,b]==0) {
          newtx = append(newtx, rep(0,10))
        } else{
          newtx = append(newtx, rep(1,10))
        }
      }
    }

    dataPerm$tx.var = as.vector(t(newtx))

    mod1_1 <- gee(response.var~tx.var+as.factor(time.var), data = dataPerm, family =
"gaussian",id = cluster.var, corstr = "independence")
    result1_1[i] = mod1_1$coefficients[[2]][1]
    zscore1_1[i] = summary(mod1_1)$coefficients[2,5]

mod2_1 <- gee(response.var~tx.var+as.factor(time.var), data = dataPerm, family =
"gaussian",id = cluster.var, corstr = "exchangeable")
    result2_1[i] = mod2_1$coefficients[[2]][1]
    zscore2_1[i] = summary(mod2_1)$coefficients[2,5]

mod3_1 <- gee(response.var~tx.var, data = dataPerm, family = "gaussian",id = cluster.var,
corstr = "independence")
    result3_1[i] = mod3_1$coefficients[[2]][1]
    zscore3_1[i] = summary(mod3_1)$coefficients[2,5]

  }

  sort1 = sort(result1_1)
  sort2 = sort(result2_1)
  sort3 = sort(result3_1)
```

```
  sortz2 = sort(zscore2_1)
  sortz3 = sort(zscore3_1)

  # z score
  if((zscore1< sortz1[25]) | (zscore1> sortz1[975])) {
    storep1_z = 1}
  else {storep1_z = 0}
  if((zscore2< sortz2[25]) | (zscore2> sortz2[975])) {
    storep2_z = 1}
  else {storep2_z = 0}
  if((zscore3< sortz3[25]) | (zscore3> sortz3[975])) {
    storep3_z = 1}
  else {storep3_z = 0}

  # coefficients
  if((result1< sort1[25]) | (result1> sort1[975])) {
    storep1 = 1}
  else {storep1 = 0}
  if((result2< sort2[25]) | (result2> sort2[975])) {
    storep2 = 1}
  else {storep2 = 0}
  if((result3< sort3[25]) | (result3> sort3[975])) {
    storep3 = 1}
  else {storep3 = 0}

  return(c(storep1, storep1_z, storep2, storep2_z, storep3, storep3_z))
}

finalp1 = 0
finalp1_z = 0
finalp2 = 0
finalp2_z = 0
finalp3 = 0
finalp3_z = 0

for(i in 1:500) {
  resultList = NULL
  resultList = checkfunc(1000, 40, i)
  finalp1 = finalp1 + resultList[1]
  finalp1_z = finalp1_z + resultList[2]
  finalp2 = finalp2 + resultList[3]
  finalp2_z = finalp2_z + resultList[4]
  finalp3 = finalp3 + resultList[5]
  finalp3_z = finalp3_z + resultList[6]
}

print(finalp1/500)
print(finalp1_z/500)
print(finalp2/500)
```

```
print(finalp3/500)
print(finalp3_z/500)
```

 # S6

```
myfunc <- function(n, seed) {
  design <- swDsn(clusters=c(10,10,10,10), extra.time=0, all.ctl.time0=TRUE)
  # first generate the original dataset with a time effect and 0 treatment effect,
  # rho is the correlation between slope and intercept
  swGenData.nScalar <- swSim(design,family=gaussian(link="identity"), n=10, mu0=3,
                        mu1=3, time.effect=c(0,.2,.3,.4,.5), sigma=1, tau=2, eta=2,
                        rho=0.3, seed = seed, retTimeOnTx=FALSE)
  data = swSummary(response.var, tx.var, time.var, cluster.var, swGenData.nScalar,
type="mean", digits=3)$swDsn
  newdata = NULL
  listn=c(1:n)
  x = sample(listn)
  newdata=data[x,]
  return(newdata)
}


checkfunc = function(numberOfPermutation, nCluster, seed) {
  design <- swDsn(clusters=c(10,10,10,10), extra.time=0, all.ctl.time0=TRUE)
  # first generate the original dataset with a time effect and 0 treatment effect,
  # rho is the correlation between slope and intercept
  swdata <- swSim(design,family=gaussian(link="identity"), n=10, mu0=3,
                mu1=3, time.effect=c(0,.2,.3,.4,.5), sigma=1, tau=2, eta=2,
                rho=0.3, seed = seed, retTimeOnTx=FALSE)

  storep1 = 0
  storep1_z = 0
  storep2 = 0
  storep2_z = 0
  storep3 = 0
  storep3_z = 0

  matrix = myfunc(nCluster, seed)

  mod1 <- gee(response.var~tx.var+as.factor(time.var), data = swdata, family = "gaussian",id
= cluster.var, corstr = "independence")
  result1 = mod1$coefficients[[2]][1]
  zscore1 = summary(mod1)$coefficients[2,5]
```

```
mod2 <- gee(response.var~tx.var+as.factor(time.var), data = swdata, family = "gaussian",id =
cluster.var, corstr = "exchangeable")
  result2 = mod2$coefficients[[2]][1]
  zscore2 = summary(mod2)$coefficients[2,5]

mod3 <- gee(response.var~tx.var, data = swdata, family = "gaussian",id = cluster.var, corstr
= "independence")
  result3 = mod3$coefficients[[2]][1]
  zscore3 = summary(mod3)$coefficients[2,5]


  dataPerm = swdata # copy the original dataset

  result1_1 = NULL
  zscore1_1 = NULL
  result2_1 = NULL
  zscore2_1 = NULL
  result3_1 = NULL
  zscore3_1 = NULL

  for(i in 1:numberOfPermutation) { # permutation starts
    newtx=NULL
    newdata = NULL
    listn=c(1:nCluster)
    x = sample(listn)
    newdata = matrix[x,]
    for(a in 1:nrow(newdata)) {
      for(b in 1:ncol(newdata)){
        if(newdata[a,b]==0) {
          newtx = append(newtx, rep(0,10))
        } else{
          newtx = append(newtx, rep(1,10))
        }
      }
    }

    dataPerm$tx.var = as.vector(t(newtx))

    mod1_1 <- gee(response.var~tx.var+as.factor(time.var), data = dataPerm, family =
"gaussian",id = cluster.var, corstr = "independence")
    result1_1[i] = mod1_1$coefficients[[2]][1]
    zscore1_1[i] = summary(mod1_1)$coefficients[2,5]

mod2_1 <- gee(response.var~tx.var+as.factor(time.var), data = dataPerm, family =
"gaussian",id = cluster.var, corstr = "exchangeable")
    result2_1[i] = mod2_1$coefficients[[2]][1]
    zscore2_1[i] = summary(mod2_1)$coefficients[2,5]

mod3_1 <- gee(response.var~tx.var, data = dataPerm, family = "gaussian",id = cluster.var,
```

```
  result3_1[i] = mod3_1$coefficients[[2]][1]
  zscore3_1[i] = summary(mod3_1)$coefficients[2,5]

 }

 sort1 = sort(result1_1)
 sort2 = sort(result2_1)
 sort3 = sort(result3_1)

 sortz1 = sort(zscore1_1)
 sortz2 = sort(zscore2_1)
 sortz3 = sort(zscore3_1)

 # z score
 if((zscore1< sortz1[25]) | (zscore1> sortz1[975])) {
  storep1_z = 1}
 else {storep1_z = 0}
 if((zscore2< sortz2[25]) | (zscore2> sortz2[975])) {
  storep2_z = 1}
 else {storep2_z = 0}
 if((zscore3< sortz3[25]) | (zscore3> sortz3[975])) {
  storep3_z = 1}
 else {storep3_z = 0}

 # coefficients
 if((result1< sort1[25]) | (result1> sort1[975])) {
  storep1 = 1}
 else {storep1 = 0}
 if((result2< sort2[25]) | (result2> sort2[975])) {
  storep2 = 1}
 else {storep2 = 0}
 if((result3< sort3[25]) | (result3> sort3[975])) {
  storep3 = 1}
 else {storep3 = 0}

 return(c(storep1, storep1_z, storep2, storep2_z, storep3, storep3_z))
}

finalp1 = 0
finalp1_z = 0
finalp2 = 0
finalp2_z = 0
finalp3 = 0
finalp3_z = 0

for(i in 1:500) {
 resultList = NULL
 resultList = checkfunc(1000, 40, i)
 finalp1 = finalp1 + resultList[1]
```

```
  finalp2 = finalp2 + resultList[3]
  finalp2_z = finalp2_z + resultList[4]
  finalp3 = finalp3 + resultList[5]
  finalp3_z = finalp3_z + resultList[6]
}

print(finalp1/500)
print(finalp1_z/500)
print(finalp2/500)
print(finalp2_z/500)
print(finalp3/500)
print(finalp3_z/500)
```

 # S7

```
myfunc <- function(n, seed) {
  design <- swDsn(clusters=c(10,10,10,10), extra.time=0, all.ctl.time0=TRUE)
  # first generate the original dataset with a time effect and 0 treatment effect,
  # rho is the correlation between slope and intercept
  swGenData.nScalar <- swSim(design,family=gaussian(link="identity"), n=10, mu0=3,
                        mu1=3.21, time.effect=c(0,.2,.3,.4,.5), sigma=1, tau=2, eta=0,
                        rho=0, seed = seed, retTimeOnTx=FALSE)
  data = swSummary(response.var, tx.var, time.var, cluster.var, swGenData.nScalar,
type="mean", digits=3)$swDsn
  newdata = NULL
  listn=c(1:n)
  x = sample(listn)
  newdata=data[x,]
  return(newdata)
}

checkfunc = function(numberOfPermutation, nCluster, seed) {
  design <- swDsn(clusters=c(10,10,10,10), extra.time=0, all.ctl.time0=TRUE)
  # first generate the original dataset with a time effect and 0 treatment effect,
  # rho is the correlation between slope and intercept
  swdata <- swSim(design,family=gaussian(link="identity"), n=10, mu0=3,
                mu1=3.21, time.effect=c(0,.2,.3,.4,.5), sigma=1, tau=2, eta=0,
                rho=0, seed = seed, retTimeOnTx=FALSE)

  storep1 = 0
  storep1_z = 0
  storep2 = 0
  storep2_z = 0
  storep3 = 0
  storep3_z = 0
```

```
matrix = myfunc(nCluster, seed)

  mod1 <- gee(response.var~tx.var+as.factor(time.var), data = swdata, family = "gaussian",id
= cluster.var, corstr = "independence")
  result1 = mod1$coefficients[[2]][1]
  zscore1 = summary(mod1)$coefficients[2,5]

mod2 <- gee(response.var~tx.var+as.factor(time.var), data = swdata, family = "gaussian",id =
cluster.var, corstr = "exchangeable")
  result2 = mod2$coefficients[[2]][1]
  zscore2 = summary(mod2)$coefficients[2,5]

mod3 <- gee(response.var~tx.var, data = swdata, family = "gaussian",id = cluster.var, corstr
= "independence")
  result3 = mod3$coefficients[[2]][1]
  zscore3 = summary(mod3)$coefficients[2,5]

  dataPerm = swdata # copy the original dataset

  result1_1 = NULL
  zscore1_1 = NULL
  result2_1 = NULL
  zscore2_1 = NULL
  result3_1 = NULL
  zscore3_1 = NULL

  for(i in 1:numberOfPermutation) { # permutation starts
    newtx=NULL
    newdata = NULL
    listn=c(1:nCluster)
    x = sample(listn)
    newdata = matrix[x,]
    for(a in 1:nrow(newdata)) {
      for(b in 1:ncol(newdata)){
        if(newdata[a,b]==0) {
          newtx = append(newtx, rep(0,10))
        } else{
          newtx = append(newtx, rep(1,10))
        }
      }
    }

    dataPerm$tx.var = as.vector(t(newtx))

    mod1_1 <- gee(response.var~tx.var+as.factor(time.var), data = dataPerm, family =
"gaussian",id = cluster.var, corstr = "independence")
    result1_1[i] = mod1_1$coefficients[[2]][1]
    zscore1_1[i] = summary(mod1_1)$coefficients[2,5]
```

```
mod2_1 <- gee(response.var~tx.var+as.factor(time.var), data = dataPerm, family =
"gaussian",id = cluster.var, corstr = "exchangeable")
   result2_1[i] = mod2_1$coefficients[[2]][1]
   zscore2_1[i] = summary(mod2_1)$coefficients[2,5]

mod3_1 <- gee(response.var~tx.var, data = dataPerm, family = "gaussian",id = cluster.var,
corstr = "independence")
   result3_1[i] = mod3_1$coefficients[[2]][1]
   zscore3_1[i] = summary(mod3_1)$coefficients[2,5]

  }

  sort1 = sort(result1_1)
  sort2 = sort(result2_1)
  sort3 = sort(result3_1)

  sortz1 = sort(zscore1_1)
  sortz2 = sort(zscore2_1)
  sortz3 = sort(zscore3_1)

  # z score
  if((zscore1< sortz1[25]) | (zscore1> sortz1[975])) {
    storep1_z = 1}
  else {storep1_z = 0}
  if((zscore2< sortz2[25]) | (zscore2> sortz2[975])) {
    storep2_z = 1}
  else {storep2_z = 0}
  if((zscore3< sortz3[25]) | (zscore3> sortz3[975])) {
    storep3_z = 1}
  else {storep3_z = 0}

  # coefficients
  if((result1< sort1[25]) | (result1> sort1[975])) {
    storep1 = 1}
  else {storep1 = 0}
  if((result2< sort2[25]) | (result2> sort2[975])) {
    storep2 = 1}
  else {storep2 = 0}
  if((result3< sort3[25]) | (result3> sort3[975])) {
    storep3 = 1}
  else {storep3 = 0}

  return(c(storep1, storep1_z, storep2, storep2_z, storep3, storep3_z))
}

finalp1 = 0
finalp1_z = 0
finalp2 = 0
finalp2_z = 0
```

```
finalp3_z = 0

for(i in 1:500) {
  resultList = NULL
  resultList = checkfunc(1000, 40, i)
  finalp1 = finalp1 + resultList[1]
  finalp1_z = finalp1_z + resultList[2]
  finalp2 = finalp2 + resultList[3]
  finalp2_z = finalp2_z + resultList[4]
  finalp3 = finalp3 + resultList[5]
  finalp3_z = finalp3_z + resultList[6]
}

print(finalp1/500)
print(finalp1_z/500)
print(finalp2/500)
print(finalp2_z/500)
print(finalp3/500)
print(finalp3_z/500)
```

 # S8

```
myfunc <- function(n, seed) {
  design <- swDsn(clusters=c(10,10,10,10), extra.time=0, all.ctl.time0=TRUE)
  # first generate the original dataset with a time effect and 0 treatment effect,
  # rho is the correlation between slope and intercept
  swGenData.nScalar <- swSim(design,family=gaussian(link="identity"), n=10, mu0=3,
                        mu1=3.9, time.effect=c(0,.2,.3,.4,.5), sigma=1, tau=2, eta=2,
                        rho=0.3, seed = seed, retTimeOnTx=FALSE)
  data = swSummary(response.var, tx.var, time.var, cluster.var, swGenData.nScalar,
type="mean", digits=3)$swDsn
  newdata = NULL
  listn=c(1:n)
  x = sample(listn)
  newdata=data[x,]
  return(newdata)
}

checkfunc = function(numberOfPermutation, nCluster, seed) {
  design <- swDsn(clusters=c(10,10,10,10), extra.time=0, all.ctl.time0=TRUE)
  # first generate the original dataset with a time effect and 0 treatment effect,
  # rho is the correlation between slope and intercept
  swdata <- swSim(design,family=gaussian(link="identity"), n=10, mu0=3,
                mu1=3.9, time.effect=c(0,.2,.3,.4,.5), sigma=1, tau=2, eta=2,
```

```
  storep1 = 0
  storep1_z = 0
  storep2 = 0
  storep2_z = 0
  storep3 = 0
  storep3_z = 0

  matrix = myfunc(nCluster, seed)

  mod1 <- gee(response.var~tx.var+as.factor(time.var), data = swdata, family = "gaussian",id
= cluster.var, corstr = "independence")
  result1 = mod1$coefficients[[2]][1]
  zscore1 = summary(mod1)$coefficients[2,5]

mod2 <- gee(response.var~tx.var+as.factor(time.var), data = swdata, family = "gaussian",id =
cluster.var, corstr = "exchangeable")
  result2 = mod2$coefficients[[2]][1]
  zscore2 = summary(mod2)$coefficients[2,5]

mod3 <- gee(response.var~tx.var, data = swdata, family = "gaussian",id = cluster.var, corstr
= "independence")
  result3 = mod3$coefficients[[2]][1]
  zscore3 = summary(mod3)$coefficients[2,5]

  dataPerm = swdata # copy the original dataset

  result1_1 = NULL
  zscore1_1 = NULL
  result2_1 = NULL
  zscore2_1 = NULL
  result3_1 = NULL
  zscore3_1 = NULL

  for(i in 1:numberOfPermutation) { # permutation starts
    newtx=NULL
    newdata = NULL
    listn=c(1:nCluster)
    x = sample(listn)
    newdata = matrix[x,]
    for(a in 1:nrow(newdata)) {
      for(b in 1:ncol(newdata)){
        if(newdata[a,b]==0) {
          newtx = append(newtx, rep(0,10))
        } else{
          newtx = append(newtx, rep(1,10))
        }
      }
    }
```

```
   dataPerm$tx.var = as.vector(t(newtx))

   mod1_1 <- gee(response.var~tx.var+as.factor(time.var), data = dataPerm, family =
"gaussian",id = cluster.var, corstr = "independence")
   result1_1[i] = mod1_1$coefficients[[2]][1]
   zscore1_1[i] = summary(mod1_1)$coefficients[2,5]

mod2_1 <- gee(response.var~tx.var+as.factor(time.var), data = dataPerm, family =
"gaussian",id = cluster.var, corstr = "exchangeable")
   result2_1[i] = mod2_1$coefficients[[2]][1]
   zscore2_1[i] = summary(mod2_1)$coefficients[2,5]

mod3_1 <- gee(response.var~tx.var, data = dataPerm, family = "gaussian",id = cluster.var,
corstr = "independence")
   result3_1[i] = mod3_1$coefficients[[2]][1]
   zscore3_1[i] = summary(mod3_1)$coefficients[2,5]

  }

  sort1 = sort(result1_1)
  sort2 = sort(result2_1)
  sort3 = sort(result3_1)

  sortz1 = sort(zscore1_1)
  sortz2 = sort(zscore2_1)
  sortz3 = sort(zscore3_1)

  # z score
  if((zscore1< sortz1[25]) | (zscore1> sortz1[975])) {
   storep1_z = 1}
  else {storep1_z = 0}
  if((zscore2< sortz2[25]) | (zscore2> sortz2[975])) {
   storep2_z = 1}
  else {storep2_z = 0}
  if((zscore3< sortz3[25]) | (zscore3> sortz3[975])) {
   storep3_z = 1}
  else {storep3_z = 0}

  # coefficients
  if((result1< sort1[25]) | (result1> sort1[975])) {
   storep1 = 1}
  else {storep1 = 0}
  if((result2< sort2[25]) | (result2> sort2[975])) {
   storep2 = 1}
  else {storep2 = 0}
  if((result3< sort3[25]) | (result3> sort3[975])) {
   storep3 = 1}
  else {storep3 = 0}
```

```
}

finalp1 = 0
finalp1_z = 0
finalp2 = 0
finalp2_z = 0
finalp3 = 0
finalp3_z = 0

for(i in 1:500) {
  resultList = NULL
  resultList = checkfunc(1000, 40, i)
  finalp1 = finalp1 + resultList[1]
  finalp1_z = finalp1_z + resultList[2]
  finalp2 = finalp2 + resultList[3]
  finalp2_z = finalp2_z + resultList[4]
  finalp3 = finalp3 + resultList[5]
  finalp3_z = finalp3_z + resultList[6]
}

print(finalp1/500)
print(finalp1_z/500)
print(finalp2/500)
print(finalp2_z/500)
print(finalp3/500)
print(finalp3_z/500)
```

 # S9

```
myfunc <- function(n, seed) {
  design <- swDsn(clusters=c(10,10,10,10), extra.time=0, all.ctl.time0=TRUE)
  # first generate the original dataset with a time effect and 0 treatment effect,
  # rho is the correlation between slope and intercept
  swGenData.nScalar <- swSim(design,family=gaussian(link="identity"), n=10, mu0=3,
                             mu1=3.8, time.effect=c(0,.2,.3,.4,.5), sigma=1, tau=2, eta=2,
                             rho=0, seed = seed, retTimeOnTx=FALSE)
  data = swSummary(response.var, tx.var, time.var, cluster.var, swGenData.nScalar,
type="mean", digits=3)$swDsn
  newdata = NULL
  listn=c(1:n)
  x = sample(listn)
  newdata=data[x,]
  return(newdata)
}
```

```
checkfunc = function(numberOfPermutation, nCluster, seed) {
  design <- swDsn(clusters=c(10,10,10,10), extra.time=0, all.ctl.time0=TRUE)
  # first generate the original dataset with a time effect and 0 treatment effect,
  # rho is the correlation between slope and intercept
  swdata <- swSim(design,family=gaussian(link="identity"), n=10, mu0=3,
                  mu1=3.8, time.effect=c(0,.2,.3,.4,.5), sigma=1, tau=2, eta=2,
                  rho=0, seed = seed, retTimeOnTx=FALSE)

  storep1 = 0
  storep1_z = 0
  storep2 = 0
  storep2_z = 0
  storep3 = 0
  storep3_z = 0

  matrix = myfunc(nCluster, seed)

  mod1 <- gee(response.var~tx.var+as.factor(time.var), data = swdata, family = "gaussian",id
= cluster.var, corstr = "independence")
  result1 = mod1$coefficients[[2]][1]
  zscore1 = summary(mod1)$coefficients[2,5]

mod2 <- gee(response.var~tx.var+as.factor(time.var), data = swdata, family = "gaussian",id =
cluster.var, corstr = "exchangeable")
  result2 = mod2$coefficients[[2]][1]
  zscore2 = summary(mod2)$coefficients[2,5]

mod3 <- gee(response.var~tx.var, data = swdata, family = "gaussian",id = cluster.var, corstr
= "independence")
  result3 = mod3$coefficients[[2]][1]
  zscore3 = summary(mod3)$coefficients[2,5]

  dataPerm = swdata # copy the original dataset

  result1_1 = NULL
  zscore1_1 = NULL
  result2_1 = NULL
  zscore2_1 = NULL
  result3_1 = NULL
  zscore3_1 = NULL

  for(i in 1:numberOfPermutation) { # permutation starts
    newtx=NULL
    newdata = NULL
    listn=c(1:nCluster)
    x = sample(listn)
    newdata = matrix[x,]
    for(a in 1:nrow(newdata)) {
      for(b in 1:ncol(newdata)){
```

```
        newtx = append(newtx, rep(0,10))
      } else{
        newtx = append(newtx, rep(1,10))
      }
    }
  }

  dataPerm$tx.var = as.vector(t(newtx))

  mod1_1 <- gee(response.var~tx.var+as.factor(time.var), data = dataPerm, family =
"gaussian",id = cluster.var, corstr = "independence")
  result1_1[i] = mod1_1$coefficients[[2]][1]
  zscore1_1[i] = summary(mod1_1)$coefficients[2,5]

mod2_1 <- gee(response.var~tx.var+as.factor(time.var), data = dataPerm, family =
"gaussian",id = cluster.var, corstr = "exchangeable")
  result2_1[i] = mod2_1$coefficients[[2]][1]
  zscore2_1[i] = summary(mod2_1)$coefficients[2,5]

mod3_1 <- gee(response.var~tx.var, data = dataPerm, family = "gaussian",id = cluster.var,
corstr = "independence")
  result3_1[i] = mod3_1$coefficients[[2]][1]
  zscore3_1[i] = summary(mod3_1)$coefficients[2,5]

 }

 sort1 = sort(result1_1)
 sort2 = sort(result2_1)
 sort3 = sort(result3_1)

 sortz1 = sort(zscore1_1)
 sortz2 = sort(zscore2_1)
 sortz3 = sort(zscore3_1)

 # z score
 if((zscore1< sortz1[25]) | (zscore1> sortz1[975])) {
  storep1_z = 1}
 else {storep1_z = 0}
 if((zscore2< sortz2[25]) | (zscore2> sortz2[975])) {
  storep2_z = 1}
 else {storep2_z = 0}
 if((zscore3< sortz3[25]) | (zscore3> sortz3[975])) {
  storep3_z = 1}
 else {storep3_z = 0}

 # coefficients
 if((result1< sort1[25]) | (result1> sort1[975])) {
  storep1 = 1}
 else {storep1 = 0}
```

```
   storep2 = 1}
  else {storep2 = 0}
  if((result3< sort3[25]) | (result3> sort3[975])) {
   storep3 = 1}
  else {storep3 = 0}

  return(c(storep1, storep1_z, storep2, storep2_z, storep3, storep3_z))
}

finalp1 = 0
finalp1_z = 0
finalp2 = 0
finalp2_z = 0
finalp3 = 0
finalp3_z = 0

for(i in 1:500) {
  resultList = NULL
  resultList = checkfunc(1000, 40, i)
  finalp1 = finalp1 + resultList[1]
  finalp1_z = finalp1_z + resultList[2]
  finalp2 = finalp2 + resultList[3]
  finalp2_z = finalp2_z + resultList[4]
  finalp3 = finalp3 + resultList[5]
  finalp3_z = finalp3_z + resultList[6]
}

print(finalp1/500)
print(finalp1_z/500)
print(finalp2/500)
print(finalp2_z/500)
print(finalp3/500)
print(finalp3_z/500)
```

 # GLMM

```
library(lme4)
```

 # S2

```
myfunc <- function(n, seed) {
  design <- swDsn(clusters=c(5,5,5,5), extra.time=0, all.ctl.time0=TRUE)
  # first generate the original dataset with a time effect and 0 treatment effect,
```

```
  swGenData.nScalar <- swSim(design,family=gaussian(link="identity"), n=10, mu0=3,
                        mu1=3, time.effect=c(0,.2,.3,.4,.5), sigma=1, tau=2, eta=2,
                        rho=0, seed = seed, retTimeOnTx=FALSE)
  data = swSummary(response.var, tx.var, time.var, cluster.var, swGenData.nScalar,
type="mean", digits=3)$swDsn
  newdata = NULL
  listn=c(1:n)
  x = sample(listn)
  newdata=data[x,]
  return(newdata)
}

checkfunc = function(numberOfPermutation, nCluster, seed) {
  design <- swDsn(clusters=c(5,5,5,5), extra.time=0, all.ctl.time0=TRUE)
  # first generate the original dataset with a time effect and 0 treatment effect,
  # rho is the correlation between slope and intercept
  data <- swSim(design,family=gaussian(link="identity"), n=10, mu0=3,
              mu1=3, time.effect=c(0,.2,.3,.4,.5), sigma=1, tau=2, eta=2,
              rho=0, seed = seed, retTimeOnTx=FALSE)

  storep1 = 0
  storep1_z = 0
  storep2 = 0
  storep2_z = 0
  storep3 = 0
  storep3_z = 0
  storep4 = 0
  storep4_z = 0

  matrix = myfunc(nCluster, seed)

  # no time effect, no random treatment
  glmod1 <- lmer(response.var ~ tx.var+(1|cluster.var), data = data)
  gm1a = fixef(glmod1)[[2]][1]
  gmz1 = summary(glmod1)$coefficients[2,3]

  # with time effect, no random treatment effect
  glmod2 <-  lmer(response.var ~ tx.var+as.factor(time.var)+(1|cluster.var), data = data)
  gm2a = fixef(glmod2)[[2]][1]
  gmz2 = summary(glmod2)$coefficients[2,3]

  # with random treatment and time effect, intercept and slope are not correlated
  glmod3 <- lmer(response.var ~ tx.var+as.factor(time.var) + (1|cluster.var) + (tx.var-
1|cluster.var), data = data)
  gm3a = fixef(glmod3)[[2]][1]
  gmz3 = summary(glmod3)$coefficients[2,3]

  # with random treatment and time effect, intercept and slope are correlated
  glmod4 <- lmer(response.var ~ tx.var+as.factor(time.var) + (tx.var+1|cluster.var), data =
```

```
  gm4a = fixef(glmod4)[[2]][1]
  gmz4 = summary(glmod4)$coefficients[2,3]

  dataPerm = data # copy the original dataset

  gm1a_1 = NULL
  gmz1_1 = NULL
  gm2a_1 = NULL
  gmz2_1 = NULL
  gm3a_1 = NULL
  gmz3_1 = NULL
  gm4a_1 = NULL
  gmz4_1 = NULL

  for(i in 1:numberOfPermutation) { # permutation starts
    newtx=NULL
    newdata = NULL
    listn=c(1:nCluster)
    x = sample(listn)
    newdata = matrix[x,]
    for(a in 1:nrow(newdata)) {
      for(b in 1:ncol(newdata)){
        if(newdata[a,b]==0) {
          newtx = append(newtx, rep(0,10))
        } else{
          newtx = append(newtx, rep(1,10))
        }
      }
    }


    dataPerm$tx.var = as.vector(t(newtx))

    # no time effect, no random treatment
    glmod1_1 <- lmer(response.var ~ tx.var+(1|cluster.var), data = dataPerm)
    gm1a_1[i] = fixef(glmod1_1)[[2]][1]
    gmz1_1[i] = summary(glmod1_1)$coefficients[2,3]

    # with time effect, no random treatment
    glmod2_1 <-  lmer(response.var ~ tx.var+as.factor(time.var)+(1|cluster.var), data =
dataPerm)
    gm2a_1[i] = fixef(glmod2_1)[[2]][1]
    gmz2_1[i] = summary(glmod2_1)$coefficients[2,3]

    # with random treatment and time effect, intercept and slope are not correlated
    glmod3_1 <- lmer(response.var ~ tx.var+as.factor(time.var) + (1|cluster.var) + (tx.var-
1|cluster.var), data = dataPerm)
    gm3a_1[i] = fixef(glmod3_1)[[2]][1]
    gmz3_1[i] = summary(glmod3_1)$coefficients[2,3]
```

```
    # with random treatment and time effect, intercept and slope are correlated
    glmod4_1 <- lmer(response.var ~ tx.var+as.factor(time.var) + (tx.var+1|cluster.var), data =
dataPerm)
    gm4a_1[i] = fixef(glmod4_1)[[2]][1]
    gmz4_1[i] = summary(glmod4_1)$coefficients[2,3]

  }

  sort1 = sort(gm1a_1)
  sort2 = sort(gm2a_1)
  sort3 = sort(gm3a_1)
  sort4 = sort(gm4a_1)

  sortz1 = sort(gmz1_1)
  sortz2 = sort(gmz2_1)
  sortz3 = sort(gmz3_1)
  sortz4 = sort(gmz4_1)

  # z score
  if((gmz1< sortz1[25]) | (gmz1> sortz1[975])) {
    storep1_z = 1}
  else {storep1_z = 0}
  if((gmz2< sortz2[25]) | (gmz2> sortz2[975])) {
    storep2_z = 1}
  else {storep2_z = 0}
  if((gmz3< sortz3[25]) | (gmz3> sortz3[975])) {
    storep3_z = 1}
  else {storep3_z = 0}
  if((gmz4< sortz4[25]) | (gmz4> sortz4[975])) {
    storep4_z = 1}
  else {storep4_z = 0}

  # coefficients
  if((gm1a< sort1[25]) | (gm1a> sort1[975])) {
    storep1 = 1}
  else {storep1 = 0}
  if((gm2a< sort2[25]) | (gm2a> sort2[975])) {
    storep2 = 1}
  else {storep2 = 0}
  if((gm3a< sort3[25]) | (gm3a> sort3[975])) {
    storep3 = 1}
  else {storep3 = 0}
  if((gm4a< sort4[25]) | (gm4a> sort4[975])) {
    storep4 = 1}
  else {storep4 = 0}

  return(c(storep1, storep1_z, storep2, storep2_z, storep3, storep3_z, storep4, storep4_z))
}
```

```
finalp1_z = 0
finalp2 = 0
finalp2_z = 0
finalp3 = 0
finalp3_z = 0
finalp4 = 0
finalp4_z = 0

for(i in 1:500) {
  resultList = NULL
  resultList = checkfunc(1000, 20, i)
  finalp1 = finalp1 + resultList[1]
  finalp1_z = finalp1_z + resultList[2]
  finalp2 = finalp2 + resultList[3]
  finalp2_z = finalp2_z + resultList[4]
  finalp3 = finalp3 + resultList[5]
  finalp3_z = finalp3_z + resultList[6]
  finalp4 = finalp4 + resultList[7]
  finalp4_z = finalp4_z + resultList[8]
}

print(finalp1/500)
print(finalp1_z/500)
print(finalp2/500)
print(finalp2_z/500)
print(finalp3/500)
print(finalp3_z/500)
print(finalp4/500)
print(finalp4_z/500)
```

 # S1

```
myfunc <- function(n, seed) {
  design <- swDsn(clusters=c(5,5,5,5), extra.time=0, all.ctl.time0=TRUE)
  # first generate the original dataset with a time effect and 0 treatment effect,
  # rho is the correlation between slope and intercept
  swGenData.nScalar <- swSim(design,family=gaussian(link="identity"), n=10, mu0=3,
                        mu1=3, time.effect=c(0,.2,.3,.4,.5), sigma=1, tau=2, eta=0,
                        rho=0, seed = seed, retTimeOnTx=FALSE)
  data = swSummary(response.var, tx.var, time.var, cluster.var, swGenData.nScalar,
type="mean", digits=3)$swDsn
  newdata = NULL
  listn=c(1:n)
  x = sample(listn)
  newdata=data[x,]
```

```
}


checkfunc = function(numberOfPermutation, nCluster, seed) {
  design <- swDsn(clusters=c(5,5,5,5), extra.time=0, all.ctl.time0=TRUE)
  # first generate the original dataset with a time effect and 0 treatment effect,
  # rho is the correlation between slope and intercept
  data <- swSim(design,family=gaussian(link="identity"), n=10, mu0=3,
              mu1=3, time.effect=c(0,.2,.3,.4,.5), sigma=1, tau=2, eta=0,
              rho=0, seed = seed, retTimeOnTx=FALSE)

  storep1 = 0
  storep1_z = 0
  storep2 = 0
  storep2_z = 0
  storep3 = 0
  storep3_z = 0
  storep4 = 0
  storep4_z = 0

  matrix = myfunc(nCluster, seed)

  # no time effect, no random treatment
  glmod1 <- lmer(response.var ~ tx.var+(1|cluster.var), data = data)
  gm1a = fixef(glmod1)[[2]][1]
  gmz1 = summary(glmod1)$coefficients[2,3]

  # with time effect, no random treatment effect
  glmod2 <-  lmer(response.var ~ tx.var+as.factor(time.var)+(1|cluster.var), data = data)
  gm2a = fixef(glmod2)[[2]][1]
  gmz2 = summary(glmod2)$coefficients[2,3]

  # with random treatment and time effect, intercept and slope are not correlated
  glmod3 <- lmer(response.var ~ tx.var+as.factor(time.var) + (1|cluster.var) + (tx.var-
1|cluster.var), data = data)
  gm3a = fixef(glmod3)[[2]][1]
  gmz3 = summary(glmod3)$coefficients[2,3]

  # with random treatment and time effect, intercept and slope are correlated
  glmod4 <- lmer(response.var ~ tx.var+as.factor(time.var) + (tx.var+1|cluster.var), data =
data)
  gm4a = fixef(glmod4)[[2]][1]
  gmz4 = summary(glmod4)$coefficients[2,3]

  dataPerm = data # copy the original dataset

  gm1a_1 = NULL
  gmz1_1 = NULL
  gm2a_1 = NULL
```

```
  gm3a_1 = NULL
  gmz3_1 = NULL
  gm4a_1 = NULL
  gmz4_1 = NULL

  for(i in 1:numberOfPermutation) { # permutation starts
    newtx=NULL
    newdata = NULL
    listn=c(1:nCluster)
    x = sample(listn)
    newdata = matrix[x,]
    for(a in 1:nrow(newdata)) {
      for(b in 1:ncol(newdata)){
        if(newdata[a,b]==0) {
          newtx = append(newtx, rep(0,10))
        } else{
          newtx = append(newtx, rep(1,10))
        }
      }
    }

    dataPerm$tx.var = as.vector(t(newtx))

    # no time effect, no random treatment
    glmod1_1 <- lmer(response.var ~ tx.var+(1|cluster.var), data = dataPerm)
    gm1a_1[i] = fixef(glmod1_1)[[2]][1]
    gmz1_1[i] = summary(glmod1_1)$coefficients[2,3]

    # with time effect, no random treatment
    glmod2_1 <-  lmer(response.var ~ tx.var+as.factor(time.var)+(1|cluster.var), data =
dataPerm)
    gm2a_1[i] = fixef(glmod2_1)[[2]][1]
    gmz2_1[i] = summary(glmod2_1)$coefficients[2,3]

    # with random treatment and time effect, intercept and slope are not correlated
    glmod3_1 <- lmer(response.var ~ tx.var+as.factor(time.var) + (1|cluster.var) + (tx.var-
1|cluster.var), data = dataPerm)
    gm3a_1[i] = fixef(glmod3_1)[[2]][1]
    gmz3_1[i] = summary(glmod3_1)$coefficients[2,3]

    # with random treatment and time effect, intercept and slope are correlated
    glmod4_1 <- lmer(response.var ~ tx.var+as.factor(time.var) + (tx.var+1|cluster.var), data =
dataPerm)
    gm4a_1[i] = fixef(glmod4_1)[[2]][1]
    gmz4_1[i] = summary(glmod4_1)$coefficients[2,3]

  }

  sort1 = sort(gm1a_1)
```

```
 sort3 = sort(gm3a_1)
 sort4 = sort(gm4a_1)

 sortz1 = sort(gmz1_1)
 sortz2 = sort(gmz2_1)
 sortz3 = sort(gmz3_1)
 sortz4 = sort(gmz4_1)

 # z score
 if((gmz1< sortz1[25]) | (gmz1> sortz1[975])) {
   storep1_z = 1}
 else {storep1_z = 0}
 if((gmz2< sortz2[25]) | (gmz2> sortz2[975])) {
   storep2_z = 1}
 else {storep2_z = 0}
 if((gmz3< sortz3[25]) | (gmz3> sortz3[975])) {
   storep3_z = 1}
 else {storep3_z = 0}
 if((gmz4< sortz4[25]) | (gmz4> sortz4[975])) {
   storep4_z = 1}
 else {storep4_z = 0}

 # coefficients
 if((gm1a< sort1[25]) | (gm1a> sort1[975])) {
   storep1 = 1}
 else {storep1 = 0}
 if((gm2a< sort2[25]) | (gm2a> sort2[975])) {
   storep2 = 1}
 else {storep2 = 0}
 if((gm3a< sort3[25]) | (gm3a> sort3[975])) {
   storep3 = 1}
 else {storep3 = 0}
 if((gm4a< sort4[25]) | (gm4a> sort4[975])) {
   storep4 = 1}
 else {storep4 = 0}

 return(c(storep1, storep1_z, storep2, storep2_z, storep3, storep3_z, storep4, storep4_z))
}

finalp1 = 0
finalp1_z = 0
finalp2 = 0
finalp2_z = 0
finalp3 = 0
finalp3_z = 0
finalp4 = 0
finalp4_z = 0

for(i in 1:500) {
```

```
  resultList = checkfunc(1000, 20, i)
  finalp1 = finalp1 + resultList[1]
  finalp1_z = finalp1_z + resultList[2]
  finalp2 = finalp2 + resultList[3]
  finalp2_z = finalp2_z + resultList[4]
  finalp3 = finalp3 + resultList[5]
  finalp3_z = finalp3_z + resultList[6]
  finalp4 = finalp4 + resultList[7]
  finalp4_z = finalp4_z + resultList[8]
}

print(finalp1/500)
print(finalp1_z/500)
print(finalp2/500)
print(finalp2_z/500)
print(finalp3/500)
print(finalp3_z/500)
print(finalp4/500)
print(finalp4_z/500)
```

 # S3

```
myfunc <- function(n, seed) {
  design <- swDsn(clusters=c(5,5,5,5), extra.time=0, all.ctl.time0=TRUE)
  # first generate the original dataset with a time effect and 0 treatment effect,
  # rho is the correlation between slope and intercept
  swGenData.nScalar <- swSim(design,family=gaussian(link="identity"), n=10, mu0=3,
                        mu1=3, time.effect=c(0,.2,.3,.4,.5), sigma=1, tau=2, eta=2,
                        rho=0.3, seed = seed, retTimeOnTx=FALSE)
  newdata = NULL
  listn=c(1:n)
  x = sample(listn)
  newdata=data[x,]
  return(newdata)
}

checkfunc = function(numberOfPermutation, nCluster, seed) {
  design <- swDsn(clusters=c(5,5,5,5), extra.time=0, all.ctl.time0=TRUE)
  # first generate the original dataset with a time effect and 0 treatment effect,
  # rho is the correlation between slope and intercept
  data <- swSim(design,family=gaussian(link="identity"), n=10, mu0=3,
              mu1=3, time.effect=c(0,.2,.3,.4,.5), sigma=1, tau=2, eta=2,
              rho=0.3, seed = seed, retTimeOnTx=FALSE)
```

```
  storep1_z = 0
  storep2 = 0
  storep2_z = 0
  storep3 = 0
  storep3_z = 0
  storep4 = 0
  storep4_z = 0

  matrix = myfunc(nCluster, seed)

  # no time effect, no random treatment
  glmod1 <- lmer(response.var ~ tx.var+(1|cluster.var), data = data)
  gm1a = fixef(glmod1)[[2]][1]
  gmz1 = summary(glmod1)$coefficients[2,3]

  # with time effect, no random treatment effect
  glmod2 <-  lmer(response.var ~ tx.var+as.factor(time.var)+(1|cluster.var), data = data)
  gm2a = fixef(glmod2)[[2]][1]
  gmz2 = summary(glmod2)$coefficients[2,3]

  # with random treatment and time effect, intercept and slope are not correlated
  glmod3 <- lmer(response.var ~ tx.var+as.factor(time.var) + (1|cluster.var) + (tx.var-
1|cluster.var), data = data)
  gm3a = fixef(glmod3)[[2]][1]
  gmz3 = summary(glmod3)$coefficients[2,3]

  # with random treatment and time effect, intercept and slope are correlated
  glmod4 <- lmer(response.var ~ tx.var+as.factor(time.var) + (tx.var+1|cluster.var), data =
data)
  gm4a = fixef(glmod4)[[2]][1]
  gmz4 = summary(glmod4)$coefficients[2,3]


  dataPerm = data # copy the original dataset

  gm1a_1 = NULL
  gmz1_1 = NULL
  gm2a_1 = NULL
  gmz2_1 = NULL
  gm3a_1 = NULL
  gmz3_1 = NULL
  gm4a_1 = NULL
  gmz4_1 = NULL

  for(i in 1:numberOfPermutation) { # permutation starts
    newtx=NULL
    newdata = NULL
    listn=c(1:nCluster)
    x = sample(listn)
```

```
   for(a in 1:nrow(newdata)) {
     for(b in 1:ncol(newdata)){
       if(newdata[a,b]==0) {
         newtx = append(newtx, rep(0,10))
       } else{
         newtx = append(newtx, rep(1,10))
       }
     }
   }

   dataPerm$tx.var = as.vector(t(newtx))

   # no time effect, no random treatment
   glmod1_1 <- lmer(response.var ~ tx.var+(1|cluster.var), data = dataPerm)
   gm1a_1[i] = fixef(glmod1_1)[[2]][1]
   gmz1_1[i] = summary(glmod1_1)$coefficients[2,3]

   # with time effect, no random treatment
   glmod2_1 <-  lmer(response.var ~ tx.var+as.factor(time.var)+(1|cluster.var), data =
dataPerm)
   gm2a_1[i] = fixef(glmod2_1)[[2]][1]
   gmz2_1[i] = summary(glmod2_1)$coefficients[2,3]

   # with random treatment and time effect, intercept and slope are not correlated
   glmod3_1 <- lmer(response.var ~ tx.var+as.factor(time.var) + (1|cluster.var) + (tx.var-
1|cluster.var), data = dataPerm)
   gm3a_1[i] = fixef(glmod3_1)[[2]][1]
   gmz3_1[i] = summary(glmod3_1)$coefficients[2,3]

   # with random treatment and time effect, intercept and slope are correlated
   glmod4_1 <- lmer(response.var ~ tx.var+as.factor(time.var) + (tx.var+1|cluster.var), data =
dataPerm)
   gm4a_1[i] = fixef(glmod4_1)[[2]][1]
   gmz4_1[i] = summary(glmod4_1)$coefficients[2,3]

  }

  sort1 = sort(gm1a_1)
  sort2 = sort(gm2a_1)
  sort3 = sort(gm3a_1)
  sort4 = sort(gm4a_1)

  sortz1 = sort(gmz1_1)
  sortz2 = sort(gmz2_1)
  sortz3 = sort(gmz3_1)
  sortz4 = sort(gmz4_1)

  # z score
  if((gmz1< sortz1[25]) | (gmz1> sortz1[975])) {
```

```
  else {storep1_z = 0}
  if((gmz2< sortz2[25]) | (gmz2> sortz2[975])) {
    storep2_z = 1}
  else {storep2_z = 0}
  if((gmz3< sortz3[25]) | (gmz3> sortz3[975])) {
    storep3_z = 1}
  else {storep3_z = 0}
  if((gmz4< sortz4[25]) | (gmz4> sortz4[975])) {
    storep4_z = 1}
  else {storep4_z = 0}

  # coefficients
  if((gm1a< sort1[25]) | (gm1a> sort1[975])) {
    storep1 = 1}
  else {storep1 = 0}
  if((gm2a< sort2[25]) | (gm2a> sort2[975])) {
    storep2 = 1}
  else {storep2 = 0}
  if((gm3a< sort3[25]) | (gm3a> sort3[975])) {
    storep3 = 1}
  else {storep3 = 0}
  if((gm4a< sort4[25]) | (gm4a> sort4[975])) {
    storep4 = 1}
  else {storep4 = 0}

  return(c(storep1, storep1_z, storep2, storep2_z, storep3, storep3_z, storep4, storep4_z))
}


finalp1 = 0
finalp1_z = 0
finalp2 = 0
finalp2_z = 0
finalp3 = 0
finalp3_z = 0
finalp4 = 0
finalp4_z = 0

for(i in 1:500) {
  resultList = NULL
  resultList = checkfunc(1000, 20, i)
  finalp1 = finalp1 + resultList[1]
  finalp1_z = finalp1_z + resultList[2]
  finalp2 = finalp2 + resultList[3]
  finalp2_z = finalp2_z + resultList[4]
  finalp3 = finalp3 + resultList[5]
  finalp3_z = finalp3_z + resultList[6]
  finalp4 = finalp4 + resultList[7]
  finalp4_z = finalp4_z + resultList[8]
```

```
print(finalp1/500)
print(finalp1_z/500)
print(finalp2/500)
print(finalp2_z/500)
print(finalp3/500)
print(finalp3_z/500)
print(finalp4/500)
print(finalp4_z/500)
```

 # S4

```
myfunc <- function(n, seed) {
  design <- swDsn(clusters=c(10,10,10,10), extra.time=0, all.ctl.time0=TRUE)
  # first generate the original dataset with a time effect and 0 treatment effect,
  # rho is the correlation between slope and intercept
  swGenData.nScalar <- swSim(design,family=gaussian(link="identity"), n=10, mu0=3,
                        mu1=3, time.effect=c(0,.2,.3,.4,.5), sigma=1, tau=2, eta=0,
                        rho=0, seed = seed, retTimeOnTx=FALSE)
  data = swSummary(response.var, tx.var, time.var, cluster.var, swGenData.nScalar,
type="mean", digits=3)$swDsn
  newdata = NULL
  listn=c(1:n)
  x = sample(listn)
  newdata=data[x,]
  return(newdata)
}


checkfunc = function(numberOfPermutation, nCluster, seed) {
  design <- swDsn(clusters=c(10,10,10,10), extra.time=0, all.ctl.time0=TRUE)
  # first generate the original dataset with a time effect and 0 treatment effect,
  # rho is the correlation between slope and intercept
  data <- swSim(design,family=gaussian(link="identity"), n=10, mu0=3,
              mu1=3, time.effect=c(0,.2,.3,.4,.5), sigma=1, tau=2, eta=0,
              rho=0, seed = seed, retTimeOnTx=FALSE)

  storep1 = 0
  storep1_z = 0
  storep2 = 0
  storep2_z = 0
  storep3 = 0
```

```
  storep4 = 0
  storep4_z = 0

  matrix = myfunc(nCluster, seed)

  # no time effect, no random treatment
  glmod1 <- lmer(response.var ~ tx.var+(1|cluster.var), data = data)
  gm1a = fixef(glmod1)[[2]][1]
  gmz1 = summary(glmod1)$coefficients[2,3]

  # with time effect, no random treatment effect
  glmod2 <-  lmer(response.var ~ tx.var+as.factor(time.var)+(1|cluster.var), data = data)
  gm2a = fixef(glmod2)[[2]][1]
  gmz2 = summary(glmod2)$coefficients[2,3]

  # with random treatment and time effect, intercept and slope are not correlated
  glmod3 <- lmer(response.var ~ tx.var+as.factor(time.var) + (1|cluster.var) + (tx.var-
1|cluster.var), data = data)
  gm3a = fixef(glmod3)[[2]][1]
  gmz3 = summary(glmod3)$coefficients[2,3]

  # with random treatment and time effect, intercept and slope are correlated
  glmod4 <- lmer(response.var ~ tx.var+as.factor(time.var) + (tx.var+1|cluster.var), data =
data)
  gm4a = fixef(glmod4)[[2]][1]
  gmz4 = summary(glmod4)$coefficients[2,3]

  dataPerm = data # copy the original dataset

  gm1a_1 = NULL
  gmz1_1 = NULL
  gm2a_1 = NULL
  gmz2_1 = NULL
  gm3a_1 = NULL
  gmz3_1 = NULL
  gm4a_1 = NULL
  gmz4_1 = NULL

  for(i in 1:numberOfPermutation) { # permutation starts
    newtx=NULL
    newdata = NULL
    listn=c(1:nCluster)
    x = sample(listn)
    newdata = matrix[x,]
    for(a in 1:nrow(newdata)) {
      for(b in 1:ncol(newdata)){
        if(newdata[a,b]==0) {
          newtx = append(newtx, rep(0,10))
        } else{
```

```
    }
   }
  }


  dataPerm$tx.var = as.vector(t(newtx))

  # no time effect, no random treatment
  glmod1_1 <- lmer(response.var ~ tx.var+(1|cluster.var), data = dataPerm)
  gm1a_1[i] = fixef(glmod1_1)[[2]][1]
  gmz1_1[i] = summary(glmod1_1)$coefficients[2,3]

  # with time effect, no random treatment
  glmod2_1 <-  lmer(response.var ~ tx.var+as.factor(time.var)+(1|cluster.var), data =
dataPerm)
  gm2a_1[i] = fixef(glmod2_1)[[2]][1]
  gmz2_1[i] = summary(glmod2_1)$coefficients[2,3]

  # with random treatment and time effect, intercept and slope are not correlated
  glmod3_1 <- lmer(response.var ~ tx.var+as.factor(time.var) + (1|cluster.var) + (tx.var-
1|cluster.var), data = dataPerm)
  gm3a_1[i] = fixef(glmod3_1)[[2]][1]
  gmz3_1[i] = summary(glmod3_1)$coefficients[2,3]

  # with random treatment and time effect, intercept and slope are correlated
  glmod4_1 <- lmer(response.var ~ tx.var+as.factor(time.var) + (tx.var+1|cluster.var), data =
dataPerm)
  gm4a_1[i] = fixef(glmod4_1)[[2]][1]
  gmz4_1[i] = summary(glmod4_1)$coefficients[2,3]

 }

 sort1 = sort(gm1a_1)
 sort2 = sort(gm2a_1)
 sort3 = sort(gm3a_1)
 sort4 = sort(gm4a_1)

 sortz1 = sort(gmz1_1)
 sortz2 = sort(gmz2_1)
 sortz3 = sort(gmz3_1)
 sortz4 = sort(gmz4_1)

 # z score
 if((gmz1< sortz1[25]) | (gmz1> sortz1[975])) {
  storep1_z = 1}
 else {storep1_z = 0}
 if((gmz2< sortz2[25]) | (gmz2> sortz2[975])) {
  storep2_z = 1}
 else {storep2_z = 0}
```

```
   storep3_z = 1}
  else {storep3_z = 0}
  if((gmz4< sortz4[25]) | (gmz4> sortz4[975])) {
   storep4_z = 1}
  else {storep4_z = 0}

  # coefficients
  if((gm1a< sort1[25]) | (gm1a> sort1[975])) {
   storep1 = 1}
  else {storep1 = 0}
  if((gm2a< sort2[25]) | (gm2a> sort2[975])) {
   storep2 = 1}
  else {storep2 = 0}
  if((gm3a< sort3[25]) | (gm3a> sort3[975])) {
   storep3 = 1}
  else {storep3 = 0}
  if((gm4a< sort4[25]) | (gm4a> sort4[975])) {
   storep4 = 1}
  else {storep4 = 0}

  return(c(storep1, storep1_z, storep2, storep2_z, storep3, storep3_z, storep4, storep4_z))
}

finalp1 = 0
finalp1_z = 0
finalp2 = 0
finalp2_z = 0
finalp3 = 0
finalp3_z = 0
finalp4 = 0
finalp4_z = 0

for(i in 1:500) {
  resultList = NULL
  resultList = checkfunc(1000, 40, i)
  finalp1 = finalp1 + resultList[1]
  finalp1_z = finalp1_z + resultList[2]
  finalp2 = finalp2 + resultList[3]
  finalp2_z = finalp2_z + resultList[4]
  finalp3 = finalp3 + resultList[5]
  finalp3_z = finalp3_z + resultList[6]
  finalp4 = finalp4 + resultList[7]
  finalp4_z = finalp4_z + resultList[8]
}

print(finalp1/500)
print(finalp1_z/500)
print(finalp2/500)
print(finalp2_z/500)
```

```
print(finalp3_z/500)
print(finalp4/500)
print(finalp4_z/500)
```

 # S5

```
myfunc <- function(n, seed) {
  design <- swDsn(clusters=c(10,10,10,10), extra.time=0, all.ctl.time0=TRUE)
  # first generate the original dataset with a time effect and 0 treatment effect,
  # rho is the correlation between slope and intercept
  swGenData.nScalar <- swSim(design,family=gaussian(link="identity"), n=10, mu0=3,
                        mu1=3, time.effect=c(0,.2,.3,.4,.5), sigma=1, tau=2, eta=2,
                        rho=0, seed = seed, retTimeOnTx=FALSE)
  data = swSummary(response.var, tx.var, time.var, cluster.var, swGenData.nScalar,
type="mean", digits=3)$swDsn
  newdata = NULL
  listn=c(1:n)
  x = sample(listn)
  newdata=data[x,]
  return(newdata)
}

checkfunc = function(numberOfPermutation, nCluster, seed) {
  design <- swDsn(clusters=c(10,10,10,10), extra.time=0, all.ctl.time0=TRUE)
  # first generate the original dataset with a time effect and 0 treatment effect,
  # rho is the correlation between slope and intercept
  data <- swSim(design,family=gaussian(link="identity"), n=10, mu0=3,
              mu1=3, time.effect=c(0,.2,.3,.4,.5), sigma=1, tau=2, eta=2,
              rho=0, seed = seed, retTimeOnTx=FALSE)

  storep1 = 0
  storep1_z = 0
  storep2 = 0
  storep2_z = 0
  storep3 = 0
  storep3_z = 0
  storep4 = 0
  storep4_z = 0

  matrix = myfunc(nCluster, seed)

  # no time effect, no random treatment
  glmod1 <- lmer(response.var ~ tx.var+(1|cluster.var), data = data)
  gm1a = fixef(glmod1)[[2]][1]
```

```
  # with time effect, no random treatment effect
  glmod2 <-  lmer(response.var ~ tx.var+as.factor(time.var)+(1|cluster.var), data = data)
  gm2a = fixef(glmod2)[[2]][1]
  gmz2 = summary(glmod2)$coefficients[2,3]

  # with random treatment and time effect, intercept and slope are not correlated
  glmod3 <- lmer(response.var ~ tx.var+as.factor(time.var) + (1|cluster.var) + (tx.var-
1|cluster.var), data = data)
  gm3a = fixef(glmod3)[[2]][1]
  gmz3 = summary(glmod3)$coefficients[2,3]

  # with random treatment and time effect, intercept and slope are correlated
  glmod4 <- lmer(response.var ~ tx.var+as.factor(time.var) + (tx.var+1|cluster.var), data =
data)
  gm4a = fixef(glmod4)[[2]][1]
  gmz4 = summary(glmod4)$coefficients[2,3]

  dataPerm = data # copy the original dataset

  gm1a_1 = NULL
  gmz1_1 = NULL
  gm2a_1 = NULL
  gmz2_1 = NULL
  gm3a_1 = NULL
  gmz3_1 = NULL
  gm4a_1 = NULL
  gmz4_1 = NULL

  for(i in 1:numberOfPermutation) { # permutation starts
    newtx=NULL
    newdata = NULL
    listn=c(1:nCluster)
    x = sample(listn)
    newdata = matrix[x,]
    for(a in 1:nrow(newdata)) {
      for(b in 1:ncol(newdata)){
        if(newdata[a,b]==0) {
          newtx = append(newtx, rep(0,10))
        } else{
          newtx = append(newtx, rep(1,10))
        }
      }
    }

    dataPerm$tx.var = as.vector(t(newtx))

    # no time effect, no random treatment
    glmod1_1 <- lmer(response.var ~ tx.var+(1|cluster.var), data = dataPerm)
```

```
   gmz1_1[i] = summary(glmod1_1)$coefficients[2,3]

   # with time effect, no random treatment
   glmod2_1 <-  lmer(response.var ~ tx.var+as.factor(time.var)+(1|cluster.var), data =
dataPerm)
   gm2a_1[i] = fixef(glmod2_1)[[2]][1]
   gmz2_1[i] = summary(glmod2_1)$coefficients[2,3]

   # with random treatment and time effect, intercept and slope are not correlated
   glmod3_1 <- lmer(response.var ~ tx.var+as.factor(time.var) + (1|cluster.var) + (tx.var-
1|cluster.var), data = dataPerm)
   gm3a_1[i] = fixef(glmod3_1)[[2]][1]
   gmz3_1[i] = summary(glmod3_1)$coefficients[2,3]

   # with random treatment and time effect, intercept and slope are correlated
   glmod4_1 <- lmer(response.var ~ tx.var+as.factor(time.var) + (tx.var+1|cluster.var), data =
dataPerm)
   gm4a_1[i] = fixef(glmod4_1)[[2]][1]
   gmz4_1[i] = summary(glmod4_1)$coefficients[2,3]

  }

  sort1 = sort(gm1a_1)
  sort2 = sort(gm2a_1)
  sort3 = sort(gm3a_1)
  sort4 = sort(gm4a_1)

  sortz1 = sort(gmz1_1)
  sortz2 = sort(gmz2_1)
  sortz3 = sort(gmz3_1)
  sortz4 = sort(gmz4_1)

  # z score
  if((gmz1< sortz1[25]) | (gmz1> sortz1[975])) {
    storep1_z = 1}
  else {storep1_z = 0}
  if((gmz2< sortz2[25]) | (gmz2> sortz2[975])) {
    storep2_z = 1}
  else {storep2_z = 0}
  if((gmz3< sortz3[25]) | (gmz3> sortz3[975])) {
    storep3_z = 1}
  else {storep3_z = 0}
  if((gmz4< sortz4[25]) | (gmz4> sortz4[975])) {
    storep4_z = 1}
  else {storep4_z = 0}

  # coefficients
  if((gm1a< sort1[25]) | (gm1a> sort1[975])) {
    storep1 = 1}
```

```
  if((gm2a< sort2[25]) | (gm2a> sort2[975])) {
    storep2 = 1}
  else {storep2 = 0}
  if((gm3a< sort3[25]) | (gm3a> sort3[975])) {
    storep3 = 1}
  else {storep3 = 0}
  if((gm4a< sort4[25]) | (gm4a> sort4[975])) {
    storep4 = 1}
  else {storep4 = 0}

  return(c(storep1, storep1_z, storep2, storep2_z, storep3, storep3_z, storep4, storep4_z))
}

finalp1 = 0
finalp1_z = 0
finalp2 = 0
finalp2_z = 0
finalp3 = 0
finalp3_z = 0
finalp4 = 0
finalp4_z = 0

for(i in 1:500) {
  resultList = NULL
  resultList = checkfunc(1000, 40, i)
  finalp1 = finalp1 + resultList[1]
  finalp1_z = finalp1_z + resultList[2]
  finalp2 = finalp2 + resultList[3]
  finalp2_z = finalp2_z + resultList[4]
  finalp3 = finalp3 + resultList[5]
  finalp3_z = finalp3_z + resultList[6]
  finalp4 = finalp4 + resultList[7]
  finalp4_z = finalp4_z + resultList[8]
}

print(finalp1/500)
print(finalp1_z/500)
print(finalp2/500)
print(finalp2_z/500)
print(finalp3/500)
print(finalp3_z/500)
print(finalp4/500)
print(finalp4_z/500)
```

 # S6

```
myfunc <- function(n, seed) {
  design <- swDsn(clusters=c(10,10,10,10), extra.time=0, all.ctl.time0=TRUE)
  # first generate the original dataset with a time effect and 0 treatment effect,
  # rho is the correlation between slope and intercept
  swGenData.nScalar <- swSim(design,family=gaussian(link="identity"), n=10, mu0=3,
                        mu1=3, time.effect=c(0,.2,.3,.4,.5), sigma=1, tau=2, eta=2,
                        rho=0.3, seed = seed, retTimeOnTx=FALSE)
  data = swSummary(response.var, tx.var, time.var, cluster.var, swGenData.nScalar,
type="mean", digits=3)$swDsn
  newdata = NULL
  listn=c(1:n)
  x = sample(listn)
  newdata=data[x,]
  return(newdata)
}

checkfunc = function(numberOfPermutation, nCluster, seed) {
  design <- swDsn(clusters=c(10,10,10,10), extra.time=0, all.ctl.time0=TRUE)
  # first generate the original dataset with a time effect and 0 treatment effect,
  # rho is the correlation between slope and intercept
  data <- swSim(design,family=gaussian(link="identity"), n=10, mu0=3,
              mu1=3, time.effect=c(0,.2,.3,.4,.5), sigma=1, tau=2, eta=2,
              rho=0.3, seed = seed, retTimeOnTx=FALSE)

  storep1 = 0
  storep1_z = 0
  storep2 = 0
  storep2_z = 0
  storep3 = 0
  storep3_z = 0
  storep4 = 0
  storep4_z = 0

  matrix = myfunc(nCluster, seed)

  # no time effect, no random treatment
  glmod1 <- lmer(response.var ~ tx.var+(1|cluster.var), data = data)
  gm1a = fixef(glmod1)[[2]][1]
  gmz1 = summary(glmod1)$coefficients[2,3]

  # with time effect, no random treatment effect
  glmod2 <-  lmer(response.var ~ tx.var+as.factor(time.var)+(1|cluster.var), data = data)
  gm2a = fixef(glmod2)[[2]][1]
  gmz2 = summary(glmod2)$coefficients[2,3]

  # with random treatment and time effect, intercept and slope are not correlated
  glmod3 <- lmer(response.var ~ tx.var+as.factor(time.var) + (1|cluster.var) + (tx.var-
1|cluster.var), data = data)
  gm3a = fixef(glmod3)[[2]][1]
```

```
  # with random treatment and time effect, intercept and slope are correlated
  glmod4 <- lmer(response.var ~ tx.var+as.factor(time.var) + (tx.var+1|cluster.var), data =
data)
  gm4a = fixef(glmod4)[[2]][1]
  gmz4 = summary(glmod4)$coefficients[2,3]

  dataPerm = data # copy the original dataset

  gm1a_1 = NULL
  gmz1_1 = NULL
  gm2a_1 = NULL
  gmz2_1 = NULL
  gm3a_1 = NULL
  gmz3_1 = NULL
  gm4a_1 = NULL
  gmz4_1 = NULL

  for(i in 1:numberOfPermutation) { # permutation starts
    newtx=NULL
    newdata = NULL
    listn=c(1:nCluster)
    x = sample(listn)
    newdata = matrix[x,]
    for(a in 1:nrow(newdata)) {
      for(b in 1:ncol(newdata)){
        if(newdata[a,b]==0) {
          newtx = append(newtx, rep(0,10))
        } else{
          newtx = append(newtx, rep(1,10))
        }
      }
    }

    dataPerm$tx.var = as.vector(t(newtx))

    # no time effect, no random treatment
    glmod1_1 <- lmer(response.var ~ tx.var+(1|cluster.var), data = dataPerm)
    gm1a_1[i] = fixef(glmod1_1)[[2]][1]
    gmz1_1[i] = summary(glmod1_1)$coefficients[2,3]

    # with time effect, no random treatment
    glmod2_1 <-  lmer(response.var ~ tx.var+as.factor(time.var)+(1|cluster.var), data =
dataPerm)
    gm2a_1[i] = fixef(glmod2_1)[[2]][1]
    gmz2_1[i] = summary(glmod2_1)$coefficients[2,3]

    # with random treatment and time effect, intercept and slope are not correlated
    glmod3_1 <- lmer(response.var ~ tx.var+as.factor(time.var) + (1|cluster.var) + (tx.var-
```

```
    gm3a_1[i] = fixef(glmod3_1)[[2]][1]
    gmz3_1[i] = summary(glmod3_1)$coefficients[2,3]

    # with random treatment and time effect, intercept and slope are correlated
    glmod4_1 <- lmer(response.var ~ tx.var+as.factor(time.var) + (tx.var+1|cluster.var), data =
dataPerm)
    gm4a_1[i] = fixef(glmod4_1)[[2]][1]
    gmz4_1[i] = summary(glmod4_1)$coefficients[2,3]

  }

  sort1 = sort(gm1a_1)
  sort2 = sort(gm2a_1)
  sort3 = sort(gm3a_1)
  sort4 = sort(gm4a_1)

  sortz1 = sort(gmz1_1)
  sortz2 = sort(gmz2_1)
  sortz3 = sort(gmz3_1)
  sortz4 = sort(gmz4_1)

  # z score
  if((gmz1< sortz1[25]) | (gmz1> sortz1[975])) {
    storep1_z = 1}
  else {storep1_z = 0}
  if((gmz2< sortz2[25]) | (gmz2> sortz2[975])) {
    storep2_z = 1}
  else {storep2_z = 0}
  if((gmz3< sortz3[25]) | (gmz3> sortz3[975])) {
    storep3_z = 1}
  else {storep3_z = 0}
  if((gmz4< sortz4[25]) | (gmz4> sortz4[975])) {
    storep4_z = 1}
  else {storep4_z = 0}

  # coefficients
  if((gm1a< sort1[25]) | (gm1a> sort1[975])) {
    storep1 = 1}
  else {storep1 = 0}
  if((gm2a< sort2[25]) | (gm2a> sort2[975])) {
    storep2 = 1}
  else {storep2 = 0}
  if((gm3a< sort3[25]) | (gm3a> sort3[975])) {
    storep3 = 1}
  else {storep3 = 0}
  if((gm4a< sort4[25]) | (gm4a> sort4[975])) {
    storep4 = 1}
  else {storep4 = 0}
```

```
  return(c(storep1, storep1_z, storep2, storep2_z, storep3, storep3_z, storep4, storep4_z))
}

finalp1 = 0
finalp1_z = 0
finalp2 = 0
finalp2_z = 0
finalp3 = 0
finalp3_z = 0
finalp4 = 0
finalp4_z = 0

for(i in 1:500) {
  resultList = NULL
  resultList = checkfunc(1000, 40, i)
  finalp1 = finalp1 + resultList[1]
  finalp1_z = finalp1_z + resultList[2]
  finalp2 = finalp2 + resultList[3]
  finalp2_z = finalp2_z + resultList[4]
  finalp3 = finalp3 + resultList[5]
  finalp3_z = finalp3_z + resultList[6]
  finalp4 = finalp4 + resultList[7]
  finalp4_z = finalp4_z + resultList[8]
}

print(finalp1/500)
print(finalp1_z/500)
print(finalp2/500)
print(finalp2_z/500)
print(finalp3/500)
print(finalp3_z/500)
print(finalp4/500)
print(finalp4_z/500)
```

 # S7

```
myfunc <- function(n, seed) {
  design <- swDsn(clusters=c(10,10,10,10), extra.time=0, all.ctl.time0=TRUE)
  # first generate the original dataset with a time effect and 0 treatment effect,
  # rho is the correlation between slope and intercept
  swGenData.nScalar <- swSim(design,family=gaussian(link="identity"), n=10, mu0=3,
                             mu1=3.21, time.effect=c(0,.2,.3,.4,.5), sigma=1, tau=2, eta=0,
                             rho=0, seed = seed, retTimeOnTx=FALSE)
  data = swSummary(response.var, tx.var, time.var, cluster.var, swGenData.nScalar,
type="mean", digits=3)$swDsn
```

```
  listn=c(1:n)
  x = sample(listn)
  newdata=data[x,]
  return(newdata)
}


checkfunc = function(numberOfPermutation, nCluster, seed) {
  design <- swDsn(clusters=c(10,10,10,10), extra.time=0, all.ctl.time0=TRUE)
  # first generate the original dataset with a time effect and 0 treatment effect,
  # rho is the correlation between slope and intercept
  data <- swSim(design,family=gaussian(link="identity"), n=10, mu0=3,
              mu1=3.21, time.effect=c(0,.2,.3,.4,.5), sigma=1, tau=2, eta=0,
              rho=0, seed = seed, retTimeOnTx=FALSE)

  storep1 = 0
  storep1_z = 0
  storep2 = 0
  storep2_z = 0
  storep3 = 0
  storep3_z = 0
  storep4 = 0
  storep4_z = 0

  matrix = myfunc(nCluster, seed)

  # no time effect, no random treatment
  glmod1 <- lmer(response.var ~ tx.var+(1|cluster.var), data = data)
  gm1a = fixef(glmod1)[[2]][1]
  gmz1 = summary(glmod1)$coefficients[2,3]

  # with time effect, no random treatment effect
  glmod2 <-  lmer(response.var ~ tx.var+as.factor(time.var)+(1|cluster.var), data = data)
  gm2a = fixef(glmod2)[[2]][1]
  gmz2 = summary(glmod2)$coefficients[2,3]

  # with random treatment and time effect, intercept and slope are not correlated
  glmod3 <- lmer(response.var ~ tx.var+as.factor(time.var) + (1|cluster.var) + (tx.var-
1|cluster.var), data = data)
  gm3a = fixef(glmod3)[[2]][1]
  gmz3 = summary(glmod3)$coefficients[2,3]

  # with random treatment and time effect, intercept and slope are correlated
  glmod4 <- lmer(response.var ~ tx.var+as.factor(time.var) + (tx.var+1|cluster.var), data =
data)
  gm4a = fixef(glmod4)[[2]][1]
  gmz4 = summary(glmod4)$coefficients[2,3]

  dataPerm = data # copy the original dataset
```

```
  gm1a_1 = NULL
  gmz1_1 = NULL
  gm2a_1 = NULL
  gmz2_1 = NULL
  gm3a_1 = NULL
  gmz3_1 = NULL
  gm4a_1 = NULL
  gmz4_1 = NULL

  for(i in 1:numberOfPermutation) { # permutation starts
    newtx=NULL
    newdata = NULL
    listn=c(1:nCluster)
    x = sample(listn)
    newdata = matrix[x,]
    for(a in 1:nrow(newdata)) {
      for(b in 1:ncol(newdata)){
        if(newdata[a,b]==0) {
          newtx = append(newtx, rep(0,10))
        } else{
          newtx = append(newtx, rep(1,10))
        }
      }
    }


    dataPerm$tx.var = as.vector(t(newtx))

    # no time effect, no random treatment
    glmod1_1 <- lmer(response.var ~ tx.var+(1|cluster.var), data = dataPerm)
    gm1a_1[i] = fixef(glmod1_1)[[2]][1]
    gmz1_1[i] = summary(glmod1_1)$coefficients[2,3]

    # with time effect, no random treatment
    glmod2_1 <-  lmer(response.var ~ tx.var+as.factor(time.var)+(1|cluster.var), data =
dataPerm)
    gm2a_1[i] = fixef(glmod2_1)[[2]][1]
    gmz2_1[i] = summary(glmod2_1)$coefficients[2,3]

    # with random treatment and time effect, intercept and slope are not correlated
    glmod3_1 <- lmer(response.var ~ tx.var+as.factor(time.var) + (1|cluster.var) + (tx.var-
1|cluster.var), data = dataPerm)
    gm3a_1[i] = fixef(glmod3_1)[[2]][1]
    gmz3_1[i] = summary(glmod3_1)$coefficients[2,3]

    # with random treatment and time effect, intercept and slope are correlated
    glmod4_1 <- lmer(response.var ~ tx.var+as.factor(time.var) + (tx.var+1|cluster.var), data =
dataPerm)
    gm4a_1[i] = fixef(glmod4_1)[[2]][1]
```

```
 }

 sort1 = sort(gm1a_1)
 sort2 = sort(gm2a_1)
 sort3 = sort(gm3a_1)
 sort4 = sort(gm4a_1)

 sortz1 = sort(gmz1_1)
 sortz2 = sort(gmz2_1)
 sortz3 = sort(gmz3_1)
 sortz4 = sort(gmz4_1)

 # z score
 if((gmz1< sortz1[25]) | (gmz1> sortz1[975])) {
   storep1_z = 1}
 else {storep1_z = 0}
 if((gmz2< sortz2[25]) | (gmz2> sortz2[975])) {
   storep2_z = 1}
 else {storep2_z = 0}
 if((gmz3< sortz3[25]) | (gmz3> sortz3[975])) {
   storep3_z = 1}
 else {storep3_z = 0}
 if((gmz4< sortz4[25]) | (gmz4> sortz4[975])) {
   storep4_z = 1}
 else {storep4_z = 0}

 # coefficients
 if((gm1a< sort1[25]) | (gm1a> sort1[975])) {
   storep1 = 1}
 else {storep1 = 0}
 if((gm2a< sort2[25]) | (gm2a> sort2[975])) {
   storep2 = 1}
 else {storep2 = 0}
 if((gm3a< sort3[25]) | (gm3a> sort3[975])) {
   storep3 = 1}
 else {storep3 = 0}
 if((gm4a< sort4[25]) | (gm4a> sort4[975])) {
   storep4 = 1}
 else {storep4 = 0}

 return(c(storep1, storep1_z, storep2, storep2_z, storep3, storep3_z, storep4, storep4_z))
}

finalp1 = 0
finalp1_z = 0
finalp2 = 0
finalp2_z = 0
finalp3 = 0
```

```
finalp4 = 0
finalp4_z = 0

for(i in 1:500) {
  resultList = NULL
  resultList = checkfunc(1000, 40, i)
  finalp1 = finalp1 + resultList[1]
  finalp1_z = finalp1_z + resultList[2]
  finalp2 = finalp2 + resultList[3]
  finalp2_z = finalp2_z + resultList[4]
  finalp3 = finalp3 + resultList[5]
  finalp3_z = finalp3_z + resultList[6]
  finalp4 = finalp4 + resultList[7]
  finalp4_z = finalp4_z + resultList[8]
}

print(finalp1/500)
print(finalp1_z/500)
print(finalp2/500)
print(finalp2_z/500)
print(finalp3/500)
print(finalp3_z/500)
print(finalp4/500)
print(finalp4_z/500)
```

 # S8

```
myfunc <- function(n, seed) {
  design <- swDsn(clusters=c(10,10,10,10), extra.time=0, all.ctl.time0=TRUE)
  # first generate the original dataset with a time effect and 0 treatment effect,
  # rho is the correlation between slope and intercept
  swGenData.nScalar <- swSim(design,family=gaussian(link="identity"), n=10, mu0=3,
                        mu1=3.68, time.effect=c(0,.2,.3,.4,.5), sigma=1, tau=2, eta=2,
                        rho=0, seed = seed, retTimeOnTx=FALSE)
  data = swSummary(response.var, tx.var, time.var, cluster.var, swGenData.nScalar,
type="mean", digits=3)$swDsn
  newdata = NULL
  listn=c(1:n)
  x = sample(listn)
  newdata=data[x,]
  return(newdata)
}

checkfunc = function(numberOfPermutation, nCluster, seed) {
  design <- swDsn(clusters=c(10,10,10,10), extra.time=0, all.ctl.time0=TRUE)
  # first generate the original dataset with a time effect and 0 treatment effect,
```

```
  data <- swSim(design,family=gaussian(link="identity"), n=10, mu0=3,
              mu1=3.68, time.effect=c(0,.2,.3,.4,.5), sigma=1, tau=2, eta=2,
              rho=0, seed = seed, retTimeOnTx=FALSE)

  storep1 = 0
  storep1_z = 0
  storep2 = 0
  storep2_z = 0
  storep3 = 0
  storep3_z = 0
  storep4 = 0
  storep4_z = 0

  matrix = myfunc(nCluster, seed)

  # no time effect, no random treatment
  glmod1 <- lmer(response.var ~ tx.var+(1|cluster.var), data = data)
  gm1a = fixef(glmod1)[[2]][1]
  gmz1 = summary(glmod1)$coefficients[2,3]

  # with time effect, no random treatment effect
  glmod2 <-  lmer(response.var ~ tx.var+as.factor(time.var)+(1|cluster.var), data = data)
  gm2a = fixef(glmod2)[[2]][1]
  gmz2 = summary(glmod2)$coefficients[2,3]

  # with random treatment and time effect, intercept and slope are not correlated
  glmod3 <- lmer(response.var ~ tx.var+as.factor(time.var) + (1|cluster.var) + (tx.var-
1|cluster.var), data = data)
  gm3a = fixef(glmod3)[[2]][1]
  gmz3 = summary(glmod3)$coefficients[2,3]

  # with random treatment and time effect, intercept and slope are correlated
  glmod4 <- lmer(response.var ~ tx.var+as.factor(time.var) + (tx.var+1|cluster.var), data =
data)
  gm4a = fixef(glmod4)[[2]][1]
  gmz4 = summary(glmod4)$coefficients[2,3]

  dataPerm = data # copy the original dataset

  gm1a_1 = NULL
  gmz1_1 = NULL
  gm2a_1 = NULL
  gmz2_1 = NULL
  gm3a_1 = NULL
  gmz3_1 = NULL
  gm4a_1 = NULL
  gmz4_1 = NULL

  for(i in 1:numberOfPermutation) { # permutation starts
```

```
   newdata = NULL
   listn=c(1:nCluster)
   x = sample(listn)
   newdata = matrix[x,]
   for(a in 1:nrow(newdata)) {
     for(b in 1:ncol(newdata)){
       if(newdata[a,b]==0) {
         newtx = append(newtx, rep(0,10))
       } else{
         newtx = append(newtx, rep(1,10))
       }
     }
   }

   dataPerm$tx.var = as.vector(t(newtx))

   # no time effect, no random treatment
   glmod1_1 <- lmer(response.var ~ tx.var+(1|cluster.var), data = dataPerm)
   gm1a_1[i] = fixef(glmod1_1)[[2]][1]
   gmz1_1[i] = summary(glmod1_1)$coefficients[2,3]

   # with time effect, no random treatment
   glmod2_1 <-  lmer(response.var ~ tx.var+as.factor(time.var)+(1|cluster.var), data =
dataPerm)
   gm2a_1[i] = fixef(glmod2_1)[[2]][1]
   gmz2_1[i] = summary(glmod2_1)$coefficients[2,3]

   # with random treatment and time effect, intercept and slope are not correlated
   glmod3_1 <- lmer(response.var ~ tx.var+as.factor(time.var) + (1|cluster.var) + (tx.var-
1|cluster.var), data = dataPerm)
   gm3a_1[i] = fixef(glmod3_1)[[2]][1]
   gmz3_1[i] = summary(glmod3_1)$coefficients[2,3]

   # with random treatment and time effect, intercept and slope are correlated
   glmod4_1 <- lmer(response.var ~ tx.var+as.factor(time.var) + (tx.var+1|cluster.var), data =
dataPerm)
   gm4a_1[i] = fixef(glmod4_1)[[2]][1]
   gmz4_1[i] = summary(glmod4_1)$coefficients[2,3]

  }

  sort1 = sort(gm1a_1)
  sort2 = sort(gm2a_1)
  sort3 = sort(gm3a_1)
  sort4 = sort(gm4a_1)

  sortz1 = sort(gmz1_1)
  sortz2 = sort(gmz2_1)
  sortz3 = sort(gmz3_1)
```

```
  # z score
  if((gmz1< sortz1[25]) | (gmz1> sortz1[975])) {
    storep1_z = 1}
  else {storep1_z = 0}
  if((gmz2< sortz2[25]) | (gmz2> sortz2[975])) {
    storep2_z = 1}
  else {storep2_z = 0}
  if((gmz3< sortz3[25]) | (gmz3> sortz3[975])) {
    storep3_z = 1}
  else {storep3_z = 0}
  if((gmz4< sortz4[25]) | (gmz4> sortz4[975])) {
    storep4_z = 1}
  else {storep4_z = 0}

  # coefficients
  if((gm1a< sort1[25]) | (gm1a> sort1[975])) {
    storep1 = 1}
  else {storep1 = 0}
  if((gm2a< sort2[25]) | (gm2a> sort2[975])) {
    storep2 = 1}
  else {storep2 = 0}
  if((gm3a< sort3[25]) | (gm3a> sort3[975])) {
    storep3 = 1}
  else {storep3 = 0}
  if((gm4a< sort4[25]) | (gm4a> sort4[975])) {
    storep4 = 1}
  else {storep4 = 0}

  return(c(storep1, storep1_z, storep2, storep2_z, storep3, storep3_z, storep4, storep4_z))
}

finalp1 = 0
finalp1_z = 0
finalp2 = 0
finalp2_z = 0
finalp3 = 0
finalp3_z = 0
finalp4 = 0
finalp4_z = 0

for(i in 1:500) {
  resultList = NULL
  resultList = checkfunc(1000, 40, i)
  finalp1 = finalp1 + resultList[1]
  finalp1_z = finalp1_z + resultList[2]
  finalp2 = finalp2 + resultList[3]
  finalp2_z = finalp2_z + resultList[4]
  finalp3 = finalp3 + resultList[5]
```

```
  finalp4 = finalp4 + resultList[7]
  finalp4_z = finalp4_z + resultList[8]
}

print(finalp1/500)
print(finalp1_z/500)
print(finalp2/500)
print(finalp2_z/500)
print(finalp3/500)
print(finalp3_z/500)
print(finalp4/500)
print(finalp4_z/500)
```

 # S9

```
myfunc <- function(n, seed) {
  design <- swDsn(clusters=c(10,10,10,10), extra.time=0, all.ctl.time0=TRUE)
  # first generate the original dataset with a time effect and 0 treatment effect,
  # rho is the correlation between slope and intercept
  swGenData.nScalar <- swSim(design,family=gaussian(link="identity"), n=10, mu0=3,
                        mu1=3.7, time.effect=c(0,.2,.3,.4,.5), sigma=1, tau=2, eta=2,
                        rho=0.3, seed = seed, retTimeOnTx=FALSE)
  data = swSummary(response.var, tx.var, time.var, cluster.var, swGenData.nScalar,
type="mean", digits=3)$swDsn
  newdata = NULL
  listn=c(1:n)
  x = sample(listn)
  newdata=data[x,]
  return(newdata)
}

checkfunc = function(numberOfPermutation, nCluster, seed) {
  design <- swDsn(clusters=c(10,10,10,10), extra.time=0, all.ctl.time0=TRUE)
  # first generate the original dataset with a time effect and 0 treatment effect,
  # rho is the correlation between slope and intercept
  data <- swSim(design,family=gaussian(link="identity"), n=10, mu0=3,
              mu1=3.7, time.effect=c(0,.2,.3,.4,.5), sigma=1, tau=2, eta=2,
              rho=0.3, seed = seed, retTimeOnTx=FALSE)

  storep1 = 0
  storep1_z = 0
  storep2 = 0
  storep2_z = 0
  storep3 = 0
  storep3_z = 0
```

```
  storep4_z = 0

  matrix = myfunc(nCluster, seed)

  # no time effect, no random treatment
  glmod1 <- lmer(response.var ~ tx.var+(1|cluster.var), data = data)
  gm1a = fixef(glmod1)[[2]][1]
  gmz1 = summary(glmod1)$coefficients[2,3]

  # with time effect, no random treatment effect
  glmod2 <-  lmer(response.var ~ tx.var+as.factor(time.var)+(1|cluster.var), data = data)
  gm2a = fixef(glmod2)[[2]][1]
  gmz2 = summary(glmod2)$coefficients[2,3]

  # with random treatment and time effect, intercept and slope are not correlated
  glmod3 <- lmer(response.var ~ tx.var+as.factor(time.var) + (1|cluster.var) + (tx.var-
1|cluster.var), data = data)
  gm3a = fixef(glmod3)[[2]][1]
  gmz3 = summary(glmod3)$coefficients[2,3]

  # with random treatment and time effect, intercept and slope are correlated
  glmod4 <- lmer(response.var ~ tx.var+as.factor(time.var) + (tx.var+1|cluster.var), data =
data)
  gm4a = fixef(glmod4)[[2]][1]
  gmz4 = summary(glmod4)$coefficients[2,3]

  dataPerm = data # copy the original dataset

  gm1a_1 = NULL
  gmz1_1 = NULL
  gm2a_1 = NULL
  gmz2_1 = NULL
  gm3a_1 = NULL
  gmz3_1 = NULL
  gm4a_1 = NULL
  gmz4_1 = NULL

  for(i in 1:numberOfPermutation) { # permutation starts
    newtx=NULL
    newdata = NULL
    listn=c(1:nCluster)
    x = sample(listn)
    newdata = matrix[x,]
    for(a in 1:nrow(newdata)) {
      for(b in 1:ncol(newdata)){
        if(newdata[a,b]==0) {
          newtx = append(newtx, rep(0,10))
        } else{
          newtx = append(newtx, rep(1,10))
        }
```

```
    }
   }

   dataPerm$tx.var = as.vector(t(newtx))

   # no time effect, no random treatment
   glmod1_1 <- lmer(response.var ~ tx.var+(1|cluster.var), data = dataPerm)
   gm1a_1[i] = fixef(glmod1_1)[[2]][1]
   gmz1_1[i] = summary(glmod1_1)$coefficients[2,3]

   # with time effect, no random treatment
   glmod2_1 <-  lmer(response.var ~ tx.var+as.factor(time.var)+(1|cluster.var), data =
dataPerm)
   gm2a_1[i] = fixef(glmod2_1)[[2]][1]
   gmz2_1[i] = summary(glmod2_1)$coefficients[2,3]

   # with random treatment and time effect, intercept and slope are not correlated
   glmod3_1 <- lmer(response.var ~ tx.var+as.factor(time.var) + (1|cluster.var) + (tx.var-
1|cluster.var), data = dataPerm)
   gm3a_1[i] = fixef(glmod3_1)[[2]][1]
   gmz3_1[i] = summary(glmod3_1)$coefficients[2,3]

   # with random treatment and time effect, intercept and slope are correlated
   glmod4_1 <- lmer(response.var ~ tx.var+as.factor(time.var) + (tx.var+1|cluster.var), data =
dataPerm)
   gm4a_1[i] = fixef(glmod4_1)[[2]][1]
   gmz4_1[i] = summary(glmod4_1)$coefficients[2,3]

  }

  sort1 = sort(gm1a_1)
  sort2 = sort(gm2a_1)
  sort3 = sort(gm3a_1)
  sort4 = sort(gm4a_1)

  sortz1 = sort(gmz1_1)
  sortz2 = sort(gmz2_1)
  sortz3 = sort(gmz3_1)
  sortz4 = sort(gmz4_1)

  # z score
  if((gmz1< sortz1[25]) | (gmz1> sortz1[975])) {
   storep1_z = 1}
  else {storep1_z = 0}
  if((gmz2< sortz2[25]) | (gmz2> sortz2[975])) {
   storep2_z = 1}
  else {storep2_z = 0}
  if((gmz3< sortz3[25]) | (gmz3> sortz3[975])) {
   storep3_z = 1}
```

```
  if((gmz4< sortz4[25]) | (gmz4> sortz4[975])) {
    storep4_z = 1}
  else {storep4_z = 0}

  # coefficients
  if((gm1a< sort1[25]) | (gm1a> sort1[975])) {
    storep1 = 1}
  else {storep1 = 0}
  if((gm2a< sort2[25]) | (gm2a> sort2[975])) {
    storep2 = 1}
  else {storep2 = 0}
  if((gm3a< sort3[25]) | (gm3a> sort3[975])) {
    storep3 = 1}
  else {storep3 = 0}
  if((gm4a< sort4[25]) | (gm4a> sort4[975])) {
    storep4 = 1}
  else {storep4 = 0}

  return(c(storep1, storep1_z, storep2, storep2_z, storep3, storep3_z, storep4, storep4_z))
}

finalp1 = 0
finalp1_z = 0
finalp2 = 0
finalp2_z = 0
finalp3 = 0
finalp3_z = 0
finalp4 = 0
finalp4_z = 0

for(i in 1:500) {
  resultList = NULL
  resultList = checkfunc(1000, 40, i)
  finalp1 = finalp1 + resultList[1]
  finalp1_z = finalp1_z + resultList[2]
  finalp2 = finalp2 + resultList[3]
  finalp2_z = finalp2_z + resultList[4]
  finalp3 = finalp3 + resultList[5]
  finalp3_z = finalp3_z + resultList[6]
  finalp4 = finalp4 + resultList[7]
  finalp4_z = finalp4_z + resultList[8]
}

print(finalp1/500)
print(finalp1_z/500)
print(finalp2/500)
print(finalp2_z/500)
print(finalp3/500)
print(finalp3_z/500)
```

```
print(finalp4_z/500)
```

Test L0

S1

```
myfunc <- function(n, seed) {
  design <- swDsn(clusters=c(5,5,5,5), extra.time=0, all.ctl.time0=TRUE)
  # first generate the original dataset with a time effect and 0 treatment effect,
  # rho is the correlation between slope and intercept
  swGenData.nScalar <- swSim(design,family=gaussian(link="identity"), n=10, mu0=3,
                        mu1=3, time.effect=c(0,.2,.3,.4,.5), sigma=1, tau=2, eta=0,
                        rho=0, seed = seed, retTimeOnTx=FALSE)
  data = swSummary(response.var, tx.var, time.var, cluster.var, swGenData.nScalar,
type="mean", digits=3)$swDsn
  newdata = NULL
  listn=c(1:n)
  x = sample(listn)
  newdata=data[x,]
  return(newdata)
}

checkfunc = function(numberOfPermutation, nCluster, seed) {
  design <- swDsn(clusters=c(5,5,5,5), extra.time=0, all.ctl.time0=TRUE)
  # first generate the original dataset with a time effect and 0 treatment effect,
  # rho is the correlation between slope and intercept
  data <- swSim(design,family=gaussian(link="identity"), n=10, mu0=3,
              mu1=3, time.effect=c(0,.2,.3,.4,.5), sigma=1, tau=2, eta=0,
              rho=0, seed = seed, retTimeOnTx=FALSE)

  storep0 = 0
  storep0_z = 0

  matrix = myfunc(nCluster, seed)

  # no time effect, no random treatment
  mod0 <- lm(response.var~tx.var, data = data)
  gm0a = mod0$coefficients[2]
  gmz0 = summary(mod0)$coefficients[2,4]

  dataPerm = data # copy the original dataset

  gm0a_1 = NULL
  gmz0_1 = NULL
```

```
  for(i in 1:numberOfPermutation) { # permutation starts
    newtx=NULL
    newdata = NULL
    listn=c(1:nCluster)
    x = sample(listn)
    newdata = matrix[x,]
    for(a in 1:nrow(newdata)) {
      for(b in 1:ncol(newdata)){
        if(newdata[a,b]==0) {
          newtx = append(newtx, rep(0,10))
        } else{
          newtx = append(newtx, rep(1,10))
        }
      }
    }

    dataPerm$tx.var = as.vector(t(newtx))

    # no time effect, no random treatment
    mod0_1 <- lm(response.var~tx.var, data = dataPerm)
    gm0a_1[i] = mod0_1$coefficients[2]
    gmz0_1[i] = summary(mod0_1)$coefficients[2,4]

  }

  sort0 = sort(gm0a_1)
  sortz0 = sort(gmz0_1)

  # z score
  if((gmz0< sortz0[25]) | (gmz0> sortz0[975])) {
    storep0_z = 1}
  else {storep0_z = 0}

  # coefficients
  if((gm0a< sort0[25]) | (gm0a> sort0[975])) {
    storep0 = 1}
  else {storep0 = 0}

  return(c(storep0, storep0_z))
}


finalp0 = 0
finalp0_z = 0

for(i in 1:500) {
  resultList = NULL
  resultList = checkfunc(1000, 20, i)
  finalp0 = finalp0 + resultList[1]
```

```
}

print(finalp0/500)
print(finalp0_z/500)
```

 # S2

```
myfunc <- function(n, seed) {
  design <- swDsn(clusters=c(5,5,5,5), extra.time=0, all.ctl.time0=TRUE)
  # first generate the original dataset with a time effect and 0 treatment effect,
  # rho is the correlation between slope and intercept
  swGenData.nScalar <- swSim(design,family=gaussian(link="identity"), n=10, mu0=3,
                         mu1=3, time.effect=c(0,.2,.3,.4,.5), sigma=1, tau=2, eta=2,
                         rho=0, seed = seed, retTimeOnTx=FALSE)
  data = swSummary(response.var, tx.var, time.var, cluster.var, swGenData.nScalar,
type="mean", digits=3)$swDsn
  newdata = NULL
  listn=c(1:n)
  x = sample(listn)
  newdata=data[x,]
  return(newdata)
}

checkfunc = function(numberOfPermutation, nCluster, seed) {
  design <- swDsn(clusters=c(5,5,5,5), extra.time=0, all.ctl.time0=TRUE)
  # first generate the original dataset with a time effect and 0 treatment effect,
  # rho is the correlation between slope and intercept
  data <- swSim(design,family=gaussian(link="identity"), n=10, mu0=3,
              mu1=3, time.effect=c(0,.2,.3,.4,.5), sigma=1, tau=2, eta=2,
              rho=0, seed = seed, retTimeOnTx=FALSE)

  storep0 = 0
  storep0_z = 0

  matrix = myfunc(nCluster, seed)

  # no time effect, no random treatment
  mod0 <- lm(response.var~tx.var, data = data)
  gm0a = mod0$coefficients[2]
  gmz0 = summary(mod0)$coefficients[2,4]

  dataPerm = data # copy the original dataset

  gm0a_1 = NULL
  gmz0_1 = NULL
```

```
  for(i in 1:numberOfPermutation) { # permutation starts
    newtx=NULL
    newdata = NULL
    listn=c(1:nCluster)
    x = sample(listn)
    newdata = matrix[x,]
    for(a in 1:nrow(newdata)) {
      for(b in 1:ncol(newdata)){
        if(newdata[a,b]==0) {
          newtx = append(newtx, rep(0,10))
        } else{
          newtx = append(newtx, rep(1,10))
        }
      }
    }

    dataPerm$tx.var = as.vector(t(newtx))

    # no time effect, no random treatment
    mod0_1 <- lm(response.var~tx.var, data = dataPerm)
    gm0a_1[i] = mod0_1$coefficients[2]
    gmz0_1[i] = summary(mod0_1)$coefficients[2,4]

  }

  sort0 = sort(gm0a_1)
  sortz0 = sort(gmz0_1)

  # z score
  if((gmz0< sortz0[25]) | (gmz0> sortz0[975])) {
    storep0_z = 1}
  else {storep0_z = 0}

  # coefficients
  if((gm0a< sort0[25]) | (gm0a> sort0[975])) {
    storep0 = 1}
  else {storep0 = 0}

  return(c(storep0, storep0_z))
}

finalp0 = 0
finalp0_z = 0

for(i in 1:500) {
  resultList = NULL
  resultList = checkfunc(1000, 20, i)
  finalp0 = finalp0 + resultList[1]
  finalp0_z = finalp0_z + resultList[2]
```

```
print(finalp0/500)
print(finalp0_z/500)
```

 # S3

```
myfunc <- function(n, seed) {
  design <- swDsn(clusters=c(5,5,5,5), extra.time=0, all.ctl.time0=TRUE)
  # first generate the original dataset with a time effect and 0 treatment effect,
  # rho is the correlation between slope and intercept
  swGenData.nScalar <- swSim(design,family=gaussian(link="identity"), n=10, mu0=3,
                        mu1=3, time.effect=c(0,.2,.3,.4,.5), sigma=1, tau=2, eta=2,
                        rho=0.3, seed = seed, retTimeOnTx=FALSE)
  data = swSummary(response.var, tx.var, time.var, cluster.var, swGenData.nScalar,
type="mean", digits=3)$swDsn
  newdata = NULL
  listn=c(1:n)
  x = sample(listn)
  newdata=data[x,]
  return(newdata)
}

checkfunc = function(numberOfPermutation, nCluster, seed) {
  design <- swDsn(clusters=c(5,5,5,5), extra.time=0, all.ctl.time0=TRUE)
  # first generate the original dataset with a time effect and 0 treatment effect,
  # rho is the correlation between slope and intercept
  data <- swSim(design,family=gaussian(link="identity"), n=10, mu0=3,
              mu1=3, time.effect=c(0,.2,.3,.4,.5), sigma=1, tau=2, eta=2,
              rho=0.3, seed = seed, retTimeOnTx=FALSE)

  storep0 = 0
  storep0_z = 0

  matrix = myfunc(nCluster, seed)

  # no time effect, no random treatment
  mod0 <- lm(response.var~tx.var, data = data)
  gm0a = mod0$coefficients[2]
  gmz0 = summary(mod0)$coefficients[2,4]

  dataPerm = data # copy the original dataset

  gm0a_1 = NULL
  gmz0_1 = NULL

  for(i in 1:numberOfPermutation) { # permutation starts
```

```
    newdata = NULL
    listn=c(1:nCluster)
    x = sample(listn)
    newdata = matrix[x,]
    for(a in 1:nrow(newdata)) {
      for(b in 1:ncol(newdata)){
        if(newdata[a,b]==0) {
          newtx = append(newtx, rep(0,10))
        } else{
          newtx = append(newtx, rep(1,10))
        }
      }
    }

    dataPerm$tx.var = as.vector(t(newtx))

    # no time effect, no random treatment
    mod0_1 <- lm(response.var~tx.var, data = dataPerm)
    gm0a_1[i] = mod0_1$coefficients[2]
    gmz0_1[i] = summary(mod0_1)$coefficients[2,4]

  }

  sort0 = sort(gm0a_1)
  sortz0 = sort(gmz0_1)

  # z score
  if((gmz0< sortz0[25]) | (gmz0> sortz0[975])) {
    storep0_z = 1}
  else {storep0_z = 0}

  # coefficients
  if((gm0a< sort0[25]) | (gm0a> sort0[975])) {
    storep0 = 1}
  else {storep0 = 0}

  return(c(storep0, storep0_z))
}

finalp0 = 0
finalp0_z = 0

for(i in 1:500) {
  resultList = NULL
  resultList = checkfunc(1000, 20, i)
  finalp0 = finalp0 + resultList[1]
  finalp0_z = finalp0_z + resultList[2]
}

  print(finalp0/500)
```

```
print(finalp0_z/500)
```

 # S4

```
myfunc <- function(n, seed) {
  design <- swDsn(clusters=c(10,10,10,10), extra.time=0, all.ctl.time0=TRUE)
  # first generate the original dataset with a time effect and 0 treatment effect,
  # rho is the correlation between slope and intercept
  swGenData.nScalar <- swSim(design,family=gaussian(link="identity"), n=10, mu0=3,
                        mu1=3, time.effect=c(0,.2,.3,.4,.5), sigma=1, tau=2, eta=0,
                        rho=0, seed = seed, retTimeOnTx=FALSE)
  data = swSummary(response.var, tx.var, time.var, cluster.var, swGenData.nScalar,
type="mean", digits=3)$swDsn
  newdata = NULL
  listn=c(1:n)
  x = sample(listn)
  newdata=data[x,]
  return(newdata)
}

checkfunc = function(numberOfPermutation, nCluster, seed) {
  design <- swDsn(clusters=c(10,10,10,10), extra.time=0, all.ctl.time0=TRUE)
  # first generate the original dataset with a time effect and 0 treatment effect,
  # rho is the correlation between slope and intercept
  data <- swSim(design,family=gaussian(link="identity"), n=10, mu0=3,
              mu1=3, time.effect=c(0,.2,.3,.4,.5), sigma=1, tau=2, eta=0,
              rho=0, seed = seed, retTimeOnTx=FALSE)

  storep0 = 0
  storep0_z = 0

  matrix = myfunc(nCluster, seed)

  # no time effect, no random treatment
  mod0 <- lm(response.var~tx.var, data = data)
  gm0a = mod0$coefficients[2]
  gmz0 = summary(mod0)$coefficients[2,4]

  dataPerm = data # copy the original dataset

  gm0a_1 = NULL
  gmz0_1 = NULL

  for(i in 1:numberOfPermutation) { # permutation starts
    newtx=NULL
    newdata = NULL
```

```
    x = sample(listn)
    newdata = matrix[x,]
    for(a in 1:nrow(newdata)) {
      for(b in 1:ncol(newdata)){
        if(newdata[a,b]==0) {
          newtx = append(newtx, rep(0,10))
        } else{
          newtx = append(newtx, rep(1,10))
        }
      }
    }

    dataPerm$tx.var = as.vector(t(newtx))

    # no time effect, no random treatment
    mod0_1 <- lm(response.var~tx.var, data = dataPerm)
    gm0a_1[i] = mod0_1$coefficients[2]
    gmz0_1[i] = summary(mod0_1)$coefficients[2,4]

  }

  sort0 = sort(gm0a_1)
  sortz0 = sort(gmz0_1)

  # z score
  if((gmz0< sortz0[25]) | (gmz0> sortz0[975])) {
    storep0_z = 1}
  else {storep0_z = 0}

  # coefficients
  if((gm0a< sort0[25]) | (gm0a> sort0[975])) {
    storep0 = 1}
  else {storep0 = 0}

  return(c(storep0, storep0_z))
}

finalp0 = 0
finalp0_z = 0

for(i in 1:500) {
  resultList = NULL
  resultList = checkfunc(1000, 40, i)
  finalp0 = finalp0 + resultList[1]
  finalp0_z = finalp0_z + resultList[2]
}

print(finalp0/500)
print(finalp0_z/500)
```

 # S5

```
myfunc <- function(n, seed) {
  design <- swDsn(clusters=c(10,10,10,10), extra.time=0, all.ctl.time0=TRUE)
  # first generate the original dataset with a time effect and 0 treatment effect,
  # rho is the correlation between slope and intercept
  swGenData.nScalar <- swSim(design,family=gaussian(link="identity"), n=10, mu0=3,
                        mu1=3, time.effect=c(0,.2,.3,.4,.5), sigma=1, tau=2, eta=2,
                        rho=0, seed = seed, retTimeOnTx=FALSE)
  data = swSummary(response.var, tx.var, time.var, cluster.var, swGenData.nScalar,
type="mean", digits=3)$swDsn
  newdata = NULL
  listn=c(1:n)
  x = sample(listn)
  newdata=data[x,]
  return(newdata)
}

checkfunc = function(numberOfPermutation, nCluster, seed) {
  design <- swDsn(clusters=c(10,10,10,10), extra.time=0, all.ctl.time0=TRUE)
  # first generate the original dataset with a time effect and 0 treatment effect,
  # rho is the correlation between slope and intercept
  data <- swSim(design,family=gaussian(link="identity"), n=10, mu0=3,
              mu1=3, time.effect=c(0,.2,.3,.4,.5), sigma=1, tau=2, eta=2,
              rho=0, seed = seed, retTimeOnTx=FALSE)

  storep0 = 0
  storep0_z = 0

  matrix = myfunc(nCluster, seed)

  # no time effect, no random treatment
  mod0 <- lm(response.var~tx.var, data = data)
  gm0a = mod0$coefficients[2]
  gmz0 = summary(mod0)$coefficients[2,4]

  dataPerm = data # copy the original dataset

  gm0a_1 = NULL
  gmz0_1 = NULL

  for(i in 1:numberOfPermutation) { # permutation starts
    newtx=NULL
    newdata = NULL
    listn=c(1:nCluster)
    x = sample(listn)
    newdata = matrix[x,]
    for(a in 1:nrow(newdata)) {
```

```
      if(newdata[a,b]==0) {
        newtx = append(newtx, rep(0,10))
      } else{
        newtx = append(newtx, rep(1,10))
      }
    }
  }

  dataPerm$tx.var = as.vector(t(newtx))

  # no time effect, no random treatment
  mod0_1 <- lm(response.var~tx.var, data = dataPerm)
  gm0a_1[i] = mod0_1$coefficients[2]
  gmz0_1[i] = summary(mod0_1)$coefficients[2,4]

 }

 sort0 = sort(gm0a_1)
 sortz0 = sort(gmz0_1)

 # z score
 if((gmz0< sortz0[25]) | (gmz0> sortz0[975])) {
  storep0_z = 1}
 else {storep0_z = 0}

 # coefficients
 if((gm0a< sort0[25]) | (gm0a> sort0[975])) {
  storep0 = 1}
 else {storep0 = 0}

 return(c(storep0, storep0_z))
}

finalp0 = 0
finalp0_z = 0

for(i in 1:500) {
 resultList = NULL
 resultList = checkfunc(1000, 40, i)
 finalp0 = finalp0 + resultList[1]
 finalp0_z = finalp0_z + resultList[2]
}

print(finalp0/500)
print(finalp0_z/500)
```

 # S6

```
myfunc <- function(n, seed) {
  design <- swDsn(clusters=c(10,10,10,10), extra.time=0, all.ctl.time0=TRUE)
  # first generate the original dataset with a time effect and 0 treatment effect,
  # rho is the correlation between slope and intercept
  swGenData.nScalar <- swSim(design,family=gaussian(link="identity"), n=10, mu0=3,
                        mu1=3, time.effect=c(0,.2,.3,.4,.5), sigma=1, tau=2, eta=2,
                        rho=0.3, seed = seed, retTimeOnTx=FALSE)
  data = swSummary(response.var, tx.var, time.var, cluster.var, swGenData.nScalar,
type="mean", digits=3)$swDsn
  newdata = NULL
  listn=c(1:n)
  x = sample(listn)
  newdata=data[x,]
  return(newdata)
}


checkfunc = function(numberOfPermutation, nCluster, seed) {
  design <- swDsn(clusters=c(10,10,10,10), extra.time=0, all.ctl.time0=TRUE)
  # first generate the original dataset with a time effect and 0 treatment effect,
  # rho is the correlation between slope and intercept
  data <- swSim(design,family=gaussian(link="identity"), n=10, mu0=3,
              mu1=3, time.effect=c(0,.2,.3,.4,.5), sigma=1, tau=2, eta=2,
              rho=0.3, seed = seed, retTimeOnTx=FALSE)

  storep0 = 0
  storep0_z = 0

  matrix = myfunc(nCluster, seed)

  # no time effect, no random treatment
  mod0 <- lm(response.var~tx.var, data = data)
  gm0a = mod0$coefficients[2]
  gmz0 = summary(mod0)$coefficients[2,4]


  dataPerm = data # copy the original dataset

  gm0a_1 = NULL
  gmz0_1 = NULL

  for(i in 1:numberOfPermutation) { # permutation starts
    newtx=NULL
    newdata = NULL
    listn=c(1:nCluster)
    x = sample(listn)
    newdata = matrix[x,]
    for(a in 1:nrow(newdata)) {
      for(b in 1:ncol(newdata)){
```

```
        newtx = append(newtx, rep(0,10))
      } else{
        newtx = append(newtx, rep(1,10))
      }
    }
  }

  dataPerm$tx.var = as.vector(t(newtx))

  # no time effect, no random treatment
  mod0_1 <- lm(response.var~tx.var, data = dataPerm)
  gm0a_1[i] = mod0_1$coefficients[2]
  gmz0_1[i] = summary(mod0_1)$coefficients[2,4]

 }

 sort0 = sort(gm0a_1)
 sortz0 = sort(gmz0_1)

 # z score
 if((gmz0< sortz0[25]) | (gmz0> sortz0[975])) {
  storep0_z = 1}
 else {storep0_z = 0}

 # coefficients
 if((gm0a< sort0[25]) | (gm0a> sort0[975])) {
  storep0 = 1}
 else {storep0 = 0}


 return(c(storep0, storep0_z))
}

finalp0 = 0
finalp0_z = 0

for(i in 1:500) {
 resultList = NULL
 resultList = checkfunc(1000, 40, i)
 finalp0 = finalp0 + resultList[1]
 finalp0_z = finalp0_z + resultList[2]
}

print(finalp0/500)
print(finalp0_z/500)
```

```
myfunc <- function(n, seed) {
  design <- swDsn(clusters=c(10,10,10,10), extra.time=0, all.ctl.time0=TRUE)
  # first generate the original dataset with a time effect and 0 treatment effect,
  # rho is the correlation between slope and intercept
  swGenData.nScalar <- swSim(design,family=gaussian(link="identity"), n=10, mu0=3,
                        mu1=3.21, time.effect=c(0,.2,.3,.4,.5), sigma=1, tau=2, eta=0,
                        rho=0, seed = seed, retTimeOnTx=FALSE)
  data = swSummary(response.var, tx.var, time.var, cluster.var, swGenData.nScalar,
type="mean", digits=3)$swDsn
  newdata = NULL
  listn=c(1:n)
  x = sample(listn)
  newdata=data[x,]
  return(newdata)
}

checkfunc = function(numberOfPermutation, nCluster, seed) {
  design <- swDsn(clusters=c(10,10,10,10), extra.time=0, all.ctl.time0=TRUE)
  # first generate the original dataset with a time effect and 0 treatment effect,
  # rho is the correlation between slope and intercept
  data <- swSim(design,family=gaussian(link="identity"), n=10, mu0=3,
              mu1=3.21, time.effect=c(0,.2,.3,.4,.5), sigma=1, tau=2, eta=0,
              rho=0, seed = seed, retTimeOnTx=FALSE)

  storep0 = 0
  storep0_z = 0

  matrix = myfunc(nCluster, seed)

  # no time effect, no random treatment
  mod0 <- lm(response.var~tx.var, data = data)
  gm0a = mod0$coefficients[2]
  gmz0 = summary(mod0)$coefficients[2,4]


  dataPerm = data # copy the original dataset

  gm0a_1 = NULL
  gmz0_1 = NULL


  for(i in 1:numberOfPermutation) { # permutation starts
    newtx=NULL
    newdata = NULL
    listn=c(1:nCluster)
    x = sample(listn)
    newdata = matrix[x,]
    for(a in 1:nrow(newdata)) {
```

```
      if(newdata[a,b]==0) {
        newtx = append(newtx, rep(0,10))
      } else{
        newtx = append(newtx, rep(1,10))
      }
    }
  }


  dataPerm$tx.var = as.vector(t(newtx))

  # no time effect, no random treatment
  mod0_1 <- lm(response.var~tx.var, data = dataPerm)
  gm0a_1[i] = mod0_1$coefficients[2]
  gmz0_1[i] = summary(mod0_1)$coefficients[2,4]

 }

 sort0 = sort(gm0a_1)
 sortz0 = sort(gmz0_1)

 # z score
 if((gmz0< sortz0[25]) | (gmz0> sortz0[975])) {
  storep0_z = 1}
 else {storep0_z = 0}

 # coefficients
 if((gm0a< sort0[25]) | (gm0a> sort0[975])) {
  storep0 = 1}
 else {storep0 = 0}


 return(c(storep0, storep0_z))
}

finalp0 = 0
finalp0_z = 0

for(i in 1:500) {
 resultList = NULL
 resultList = checkfunc(1000, 40, i)
 finalp0 = finalp0 + resultList[1]
 finalp0_z = finalp0_z + resultList[2]
}

print(finalp0/500)
print(finalp0_z/500)
```

 # S8

```
myfunc <- function(n, seed) {
  design <- swDsn(clusters=c(10,10,10,10), extra.time=0, all.ctl.time0=TRUE)
  # first generate the original dataset with a time effect and 0 treatment effect,
  # rho is the correlation between slope and intercept
  swGenData.nScalar <- swSim(design,family=gaussian(link="identity"), n=10, mu0=3,
                        mu1=3.7, time.effect=c(0,.2,.3,.4,.5), sigma=1, tau=2, eta=2,
                        rho=0, seed = seed, retTimeOnTx=FALSE)
  data = swSummary(response.var, tx.var, time.var, cluster.var, swGenData.nScalar,
type="mean", digits=3)$swDsn
  newdata = NULL
  listn=c(1:n)
  x = sample(listn)
  newdata=data[x,]
  return(newdata)
}



checkfunc = function(numberOfPermutation, nCluster, seed) {
  design <- swDsn(clusters=c(10,10,10,10), extra.time=0, all.ctl.time0=TRUE)
  # first generate the original dataset with a time effect and 0 treatment effect,
  # rho is the correlation between slope and intercept
  data <- swSim(design,family=gaussian(link="identity"), n=10, mu0=3,
              mu1=3.7, time.effect=c(0,.2,.3,.4,.5), sigma=1, tau=2, eta=2,
              rho=0, seed = seed, retTimeOnTx=FALSE)

  storep0 = 0
  storep0_z = 0

  matrix = myfunc(nCluster, seed)

  # no time effect, no random treatment
  mod0 <- lm(response.var~tx.var, data = data)
  gm0a = mod0$coefficients[2]
  gmz0 = summary(mod0)$coefficients[2,4]


  dataPerm = data # copy the original dataset

  gm0a_1 = NULL
  gmz0_1 = NULL


  for(i in 1:numberOfPermutation) { # permutation starts
    newtx=NULL
    newdata = NULL
```

```
    x = sample(listn)
    newdata = matrix[x,]
    for(a in 1:nrow(newdata)) {
      for(b in 1:ncol(newdata)){
        if(newdata[a,b]==0) {
          newtx = append(newtx, rep(0,10))
        } else{
          newtx = append(newtx, rep(1,10))
        }
      }
    }


    dataPerm$tx.var = as.vector(t(newtx))

    # no time effect, no random treatment
    mod0_1 <- lm(response.var~tx.var, data = dataPerm)
    gm0a_1[i] = mod0_1$coefficients[2]
    gmz0_1[i] = summary(mod0_1)$coefficients[2,4]

  }

  sort0 = sort(gm0a_1)

  sortz0 = sort(gmz0_1)

  # z score
  if((gmz0< sortz0[25]) | (gmz0> sortz0[975])) {
    storep0_z = 1}
  else {storep0_z = 0}

  # coefficients
  if((gm0a< sort0[25]) | (gm0a> sort0[975])) {
    storep0 = 1}
  else {storep0 = 0}


  return(c(storep0, storep0_z))
}


finalp0 = 0
finalp0_z = 0



for(i in 1:500) {
  resultList = NULL
  resultList = checkfunc(1000, 40, i)
```

```
  finalp0_z = finalp0_z + resultList[2]
}

print(finalp0/500)
print(finalp0_z/500)
```

 # S9

```
myfunc <- function(n, seed) {
  design <- swDsn(clusters=c(10,10,10,10), extra.time=0, all.ctl.time0=TRUE)
  # first generate the original dataset with a time effect and 0 treatment effect,
  # rho is the correlation between slope and intercept
  swGenData.nScalar <- swSim(design,family=gaussian(link="identity"), n=10, mu0=3,
                        mu1=3.7, time.effect=c(0,.2,.3,.4,.5), sigma=1, tau=2, eta=2,
                        rho=0.3, seed = seed, retTimeOnTx=FALSE)
  data = swSummary(response.var, tx.var, time.var, cluster.var, swGenData.nScalar,
type="mean", digits=3)$swDsn
  newdata = NULL
  listn=c(1:n)
  x = sample(listn)
  newdata=data[x,]
  return(newdata)
}


checkfunc = function(numberOfPermutation, nCluster, seed) {
  design <- swDsn(clusters=c(10,10,10,10), extra.time=0, all.ctl.time0=TRUE)
  # first generate the original dataset with a time effect and 0 treatment effect,
  # rho is the correlation between slope and intercept
  data <- swSim(design,family=gaussian(link="identity"), n=10, mu0=3,
              mu1=3.7, time.effect=c(0,.2,.3,.4,.5), sigma=1, tau=2, eta=2,
              rho=0.3, seed = seed, retTimeOnTx=FALSE)

  storep0 = 0
  storep0_z = 0

  matrix = myfunc(nCluster, seed)

  # no time effect, no random treatment
  mod0 <- lm(response.var~tx.var, data = data)
  gm0a = mod0$coefficients[2]
  gmz0 = summary(mod0)$coefficients[2,4]

  dataPerm = data # copy the original dataset

  gm0a_1 = NULL
```

```
  for(i in 1:numberOfPermutation) { # permutation starts
    newtx=NULL
    newdata = NULL
    listn=c(1:nCluster)
    x = sample(listn)
    newdata = matrix[x,]
    for(a in 1:nrow(newdata)) {
      for(b in 1:ncol(newdata)){
        if(newdata[a,b]==0) {
          newtx = append(newtx, rep(0,10))
        } else{
          newtx = append(newtx, rep(1,10))
        }
      }
    }


    dataPerm$tx.var = as.vector(t(newtx))

    # no time effect, no random treatment
    mod0_1 <- lm(response.var~tx.var, data = dataPerm)
    gm0a_1[i] = mod0_1$coefficients[2]
    gmz0_1[i] = summary(mod0_1)$coefficients[2,4]

  }

  sort0 = sort(gm0a_1)
  sortz0 = sort(gmz0_1)

  # z score
  if((gmz0< sortz0[25]) | (gmz0> sortz0[975])) {
    storep0_z = 1}
  else {storep0_z = 0}

  # coefficients
  if((gm0a< sort0[25]) | (gm0a> sort0[975])) {
    storep0 = 1}
  else {storep0 = 0}


  return(c(storep0, storep0_z))
}

finalp0 = 0
finalp0_z = 0

for(i in 1:500) {
  resultList = NULL
```

```
  finalp0 = finalp0 + resultList[1]
  finalp0_z = finalp0_z + resultList[2]
}

print(finalp0/500)
print(finalp0_z/500)
```