Searching for Predictive Subgroups with Enhanced Treatment Effect in Clinical Trials using SHAPES

Lei Wang

A thesis

submitted in partial fulfillment of the

requirements for the degree of

Master of Science

University of Washington

2017

Committee:

Susanne May, Chair

Andrea Cook

Program Authorized to Offer Degree:

Biostatistics

University of Washington

Abstract

Searching for Predictive Subgroups with Enhanced Treatment Effect in Clinical Trials using SHAPES

Lei Wang

Chair of the Supervisory Committee:
Professor Susanne May
Biostatistics

The complexity of the human genome and variability of individual health histories, living environments, and lifestyles have motivated the development of precision medicine with the goal to enable health care providers to tailor treatments to each patients' unique characteristics. Targeted delivery of treatments/preventative strategies to subgroups who benefit sufficiently or solely can optimize treatments and improve the benefit/risk ratio for many patients. Therefore, effective methods of identifying subgroups with sufficiently large benefit are highly desired in clinical trials. However, the number of factors under consideration to define subgroups can be large and performing multiple tests to identify subgroups requires appropriate control of the overall Type I error. Here, we focus on investigating the Type I error rate for a new approach, SHAPES, which was recently developed by Prince (2015). SHAPES is an approach to search for the subgroups by directly restricting the type of subgroups that will be accepted. SHAPES subgroups must be convexly or co-convexly connected in the Boolean space. Prince explored the performance of SHAPES in terms of Type I error control and power under various scenarios considering up to four covariates for subgroup definition. We expand the evaluation of Type I error rate and the settings for evaluation as follows: 1) increasing the total number of covariates considered for

subgroup definition from 4 up to 50; 2) increasing the maximum number of covariates that can be used in the definition of a subgroup from 4 up to 6; 3) considering both independent and correlated covariates; 4) considering the prevalence of all covariates to be either 0.5 or 0.2.

To evaluate the Type I error rate of SHAPES, we simulate data under different scenarios. Briefly, two-arm randomized trials with either binary or continuous outcomes and various number of binary covariates are simulated under the null, i.e. in all simulations it is assumed that in truth there is no treatment benefit for the study population overall or for any subgroup. Qualified SHAPES subgroups are listed, and stratification models (for selected groups) as well as interaction models (for full population) are fitted. We obtain critical values from 5,000 (and up to 15,000 for selected scenarios) simulations under the null hypothesis where all the covariates are generated to be mutually independent. The critical values are then used in another 5,000 (and up to 15,000 for selected scenarios) simulations with independent or correlated covariates to calculate the Type I error rate. The number of covariates under consideration here is limited by the amount of time required for the simulations.

When the covariates are independent, the overall Type I error of SHAPES is maintained around the pre-specified α level and appears very robust for the scenarios we simulate. No monotonic trend is observed as the number of covariates considered for subgroup definition increases (up to 50) or the number of covariates for subgroup definition increases (up to 6). There is no obvious difference between the results of the stratification models and the interaction models. However, when some or all of the covariates are correlated, the overall Type I error is not as consistently close to the nominal α level as in the independent scenarios. For a binary endpoint with correlated covariates, the Type I error rate is slightly higher than the pre-specified α level, while for a continuous endpoint, the overall Type I error decreases as the correlation between covariates increases and in some scenarios is significantly lower than the nominal level.

In summary, for most of the simulated scenarios the Type I error rate of SHAPES was close or reasonably close to the nominal rate for both binary and continuous outcomes as well as independent and correlated covariates. Future research might focus on reducing the time required for the simulations to allow investigation of larger number of covariates.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I would first extend my sincere gratitude to my thesis advisor, Dr. Susanne May at the University of Washington. She has been my best support through research and writing, and steered me to the right direction whenever I need it. What I learned from her, especially the time management and multitasking skills in research, will benefit me in my future career path.

I would also like to thank my thesis committee member, Dr. Andrea Cook, for her suggestions to my thesis research and her encouragement. Without her passionate participation and input, this thesis could not have been successfully completed.

I must acknowledge Dr. Noah Simon, Dr. Lurdes Inoue, Dr. Sen Zhao, Dr. Yalan Xing, Kelsey Grinde, Phuong Vu, and other friends who generously helped me solve the programming problems as well as review my thesis.

Thanks to our program director, Gitana Garofalo, for her contribution on bonding all the Biostatistics students together. She is always available when I need support or suggestions. I deeply appreciate the books and resources that she shared with me, which helped me keep positive and confident.

Thanks to Dr. Patrick Heagerty and Dr. Susan Shortreed for leading me into the world of Electronic Medical Record (EMR) research. It was always so joyful to read papers and discuss

questions with them; and their valuable time is sincerely appreciated. Many thanks to Elizabeth Krantz for teaching me about data management and effective communications with others.

Last but not least, I want to give my thanks to Dr. Sarah Holte for her constant strong support throughout my years of study and research. Her excellent guidance in research, inspiration for me to overcome the fear to uncertainty, and her great patience and feedback while I was not my best self, all have impacted my career path profoundly.

DEDICATION

I dedicate my dissertation work to my parents, to my husband Dr. Yan Han, and to my brother, Nachuan Wang, for their unconditional love and continuous encouragement throughout my years of study, pursuing of research, and thesis writing process.

Chapter 1. INTRODUCTION

In this chapter, we will introduce the concepts of precision medicine, subgroup analysis, and the basic idea of SHAPES. We will discuss how to define and search for SHAPES subgroups and compare it with other approaches.

1.1 PRECISION MEDICINE

Precision medicine, which is known as personalized medicine or individualized treatments, is a medical method that classifies individuals into subpopulations[1] [2]. These subpopulations differ in either their susceptibility to a particular disease, in the biology or prognosis of those diseases they may develop, or their response to a specific treatment. Preventive or therapeutic interventions can be delivered to those who are most likely to benefit, which can reduce spare expenses and avoid side effects for others. A meta-analysis of Phase II studies reveals that a personalized strategy was associated with better outcomes and fewer toxic deaths[3]. Therefore, identifying subgroups is critical in applying precision medicine.

1.2 SUBGROUP ANALYSIS

Subgroup analysis is defined as evaluating the treatment effects for a specific endpoint in sub-groups of patients based on some baseline characteristics. We provide a simple conceptual example of subgroup analysis by using gender and age (old vs. young) in Figure 1.1. These subgroup analyses are commonly used in randomized controlled trials (RCTs). The subgroup analysis answers the question of whether the treatment effect varies across a subset of the patients

defined by their predictive baseline factors[4]. More importantly, it also addresses the question of whether any subgroup has better treatment response compared to the full population. The relationship between the subgroup effect and the full population effect can be complicated and we created three hypothetical examples in Figure 1.2 to illustrate it. Examples A and B have evidence of subgroup effect for human epidermal growth factor receptor 2 positive (HER2+) patients and age>50 patients, respectively. HER2- patients in example A are not responsive to the treatment on average, while the treatment in example B is harmful for patients that are younger than 50. Example C presents a situation where both subgroups have no statistical significant treatment effect even though there are tendencies for males and females. What we are interested in is to search for the subgroups that are similar to the HER2+ subgroup (patients with positive HER2) in example A and the age >50 subgroup in example B. There is a strong demand to identify the predictive subgroups with enhanced treatment effect if they exist[5].

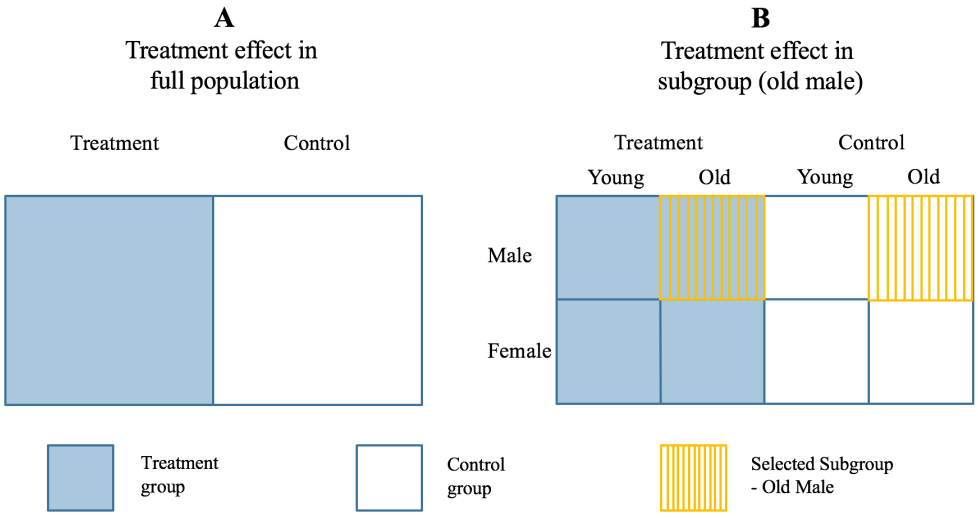


Figure 1.1. A. The full population is randomized into the treatment group and the control group. Testing the effect of treatment on the full population level is to test the difference of outcomes (continuous outcome) between these two groups. B. Full population is split into four subgroups: young males, young females, old males, and old females. Testing the treatment effect within the subgroup of old males is to test the difference of outcomes between old males in the treatment group vs old males in the control group.

Many drugs have been approved by FDA for subpopulations. For instance, Ibrance was initially approved in 2015 for postmenopausal women with estrogen receptor (ER)-positive, human epidermal growth factor receptor 2 (HER2) – negative, metastatic breast cancer who have not received an endocrine-based therapy[6]. Here the subgroup is defined by five factors: postmenopausal women, ER+, HER2-, metastatic type of breast cancer, and not yet received endocrine-based therapy. In 2017, Ibrance (palbociclib) was granted full FDA approval for use in combination with letrozole (Femara) in the frontline setting of postmenopausal women who have ER-positive, HER2-negative metastatic breast cancer[7], which is the same subgroup as before. In

addition, Alimta combined with cisplatin is used for initial treatment of advanced non-small cell lung cancer (NSCLC) for patients with non-squamous histology. Empirical subgroup identification efforts with subsequent confirmation in Phase III trials showed that it had no effect on patients with squamous NSCLC[8]. In this example, the subgroup is defined by two factors: advanced NSCLC and non-squamous histology.

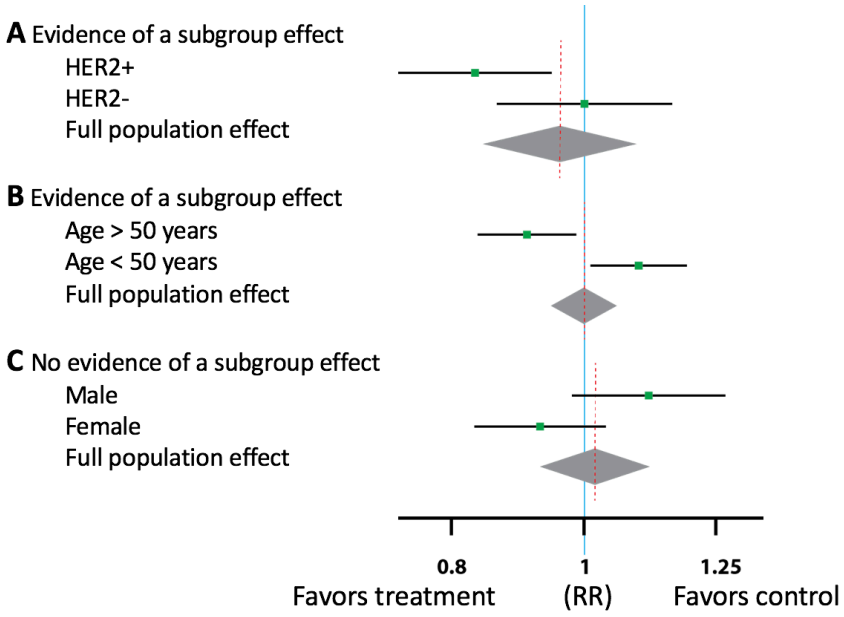


Figure 1.2. Three hypothetical example results of subgroup analyses. Gray diamonds represent the estimated relative risk of death and 95% confidence interval for full population level. Segments represent the estimated relative risk of death and 95% confidence interval for subgroup population. Here only one factor is used to define the subgroup in each example.

Subgroup analyses can lead to misleading results and Type I error control can be challenging when many covariates are under consideration. It can also cause problems, including over interpretation of subgroup analyses, lack of clear guidelines on covariate selection, lack of pre-specification for

a large number of subgroups without using appropriate adjustment for multiple comparisons[9,10]. Type I error rates will be inflated when multiple comparisons are used to test for potential subgroups without proper adjustment[11]. Performing a large number of tests on the same population may increase the Type I error rate. Consider the Framingham Heart Study, which enrolled 15,447 participants (6 cohorts) in total since 1948 and has published 3,740 papers over the decades. On average, one publication for every 4.13 participants. The ratio would be even lower if any subgroup analyses in these studies are counted in. Specifically, if we divide the population into many subgroups, then a large number of tests on the same population will be required. For example, we need to test 65,536 times if we want to find all possible subgroups on a population of e.g. 500 patients and four covariates, meaning that the number of tests is much larger than the population size. Simultaneously, when the full population is divided into subgroups and the treatment effect is homogeneous, statistical power of detecting the true difference of treatment among subgroups (if one exists) will be lower than in the full population due to the reduced sample size[12].

The general guidance of avoiding inflated Type I error when testing in subgroups is that 1) the subgroups should be specified before looking at the data, 2) subgroups are biologically plausible, 3) testing should be adjusted for multiple comparison, and 4) results should be interpreted with caution[13]. Therefore, it is critical to identify the subgroups in the study design. SHAPES is conceptualized as a method designed to identify subgroups in Phase II studies, which provides support for pre-specifying subgroups in confirmatory RCTs (Phase III study)[14]. Compared to Logic Regression which was also discussed by Prince for subgroup identification, SHAPES decreases the number of possible subgroups dramatically. In Phase II studies, the sample size is

usually much smaller than in Phase III studies and this may be an additional challenge in finding subgroups with enhanced treatment effect.

1.3 SHAPES

SHAPES is a method that restricts the form of subgroups to be convexly connected or co-convexly connected in the Boolean space, and thus substantially reduces the number of subgroups considered for testing compared to testing all (technically) possible subgroups.

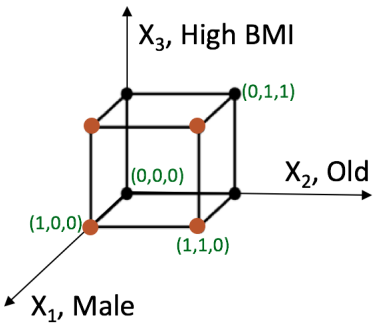
The Boolean algebra defines "$\wedge$"and "$\vee$" to represent "and" and "or", respectively.

- $X_1 \wedge X_2$ $X_1=1$ and $X_2=1$
- $X_1 \vee X_2$ $X_1=1$ or $X_2=1$
- $X_1^c \wedge X_2$ $X_1=0$ and $X_2=1$

A subgroup defined by using Boolean algebra can be visualized in a graph. Figure 1.3 and Figure 1.4 present examples of subgroups when we up to 3 covariates are considered for defining a subgroup. The selected points and the lines connecting them form a subgraph (subgroup) of the whole graph (full population). A subgraph is convexly connected if and only if all the shortest paths connecting any two points are included in the subgraph. Alternatively, a subgraph is co-convexly connected if and only if the set of points not in the subgraph is convexly connected.

The goal of SHAPES is to search for the subgroup (including the full population) with the most statistical significant enhanced treatment effect under certain constraints on the choices of subgroups. The main benefits of SHAPES include:

1) Reducing the multi-comparison problem. For example, if we consider a maximum of 4 covariates and allow up to four covariates to define the SHAPES subgroups, the number of tests would be 154, which is much smaller than all 65,536 possible subgroups.
2) The selected subgroups avoid counterintuitive combinations of subgroups. In Figure 1.4 (j), the subgroup defined by the two selected points represents "young male with low BMI or old female with high BMI", which is a union of two opposite groups of patients. That the treatment would be beneficial in only the combination of these two subgroups is counterintuitive, because a treatment is unlikely to benefit two subpopulations that are opposite on the graph and have opposing characteristics. In this subgroup example males benefit only if they are young and have low BMI, whereas females benefit from the same treatment only if they are old and have high BMI.



X1	X2	X3
1	1	1
1	1	0
1	0	1
1	0	0
0	1	1
0	1	0
0	0	1
0	0	0

Figure 1.3. Assume we have three covariates: sex (male and female), age (old and young), and BMI (high and low). The graph on the left side shows the combinations that correspond to the table on the right side. The four orange points represent a subgroup that is defined by only one

factor: male. The shortest paths between any pair of the orange points do not cover other points, meaning that the orange points are convexly connected.

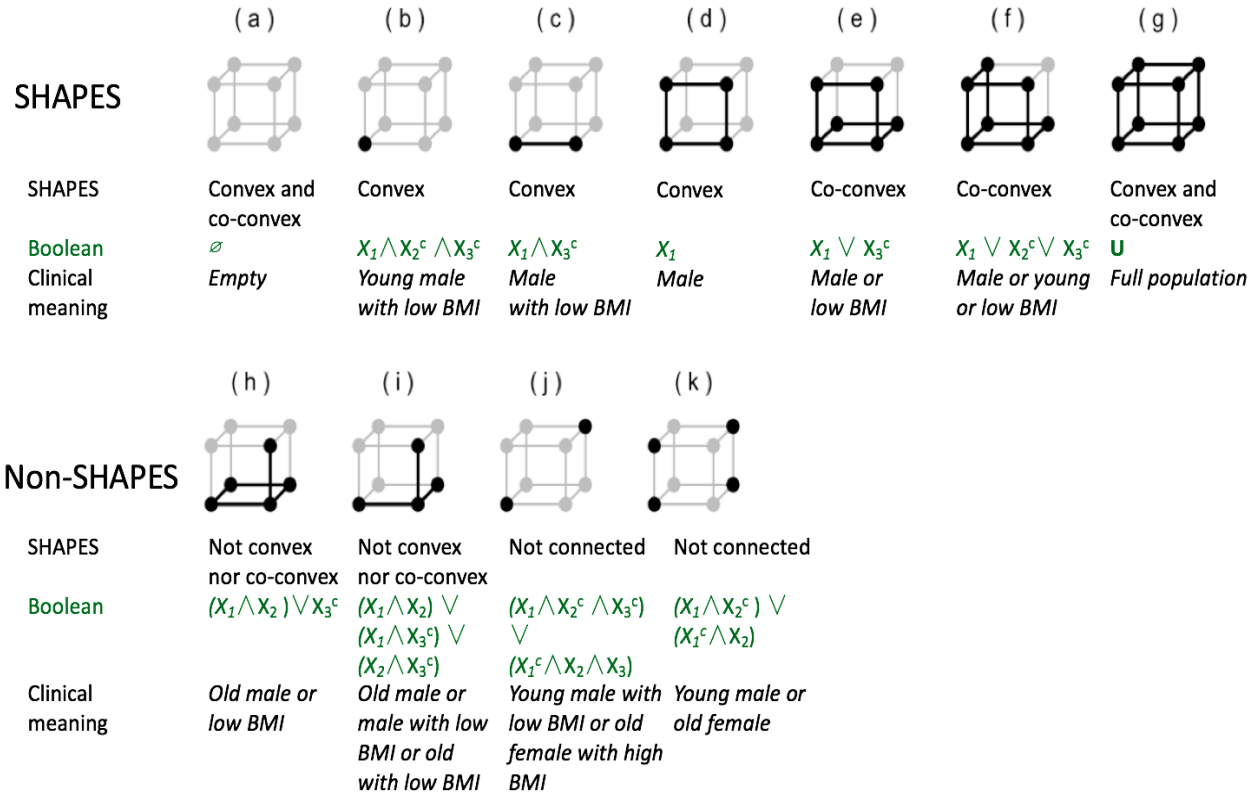


Figure 1.4. Assume that we have the same three covariates as in Figure 1.3. The first row shows subgroups that qualified as SHAPES subgroups and the second shows a few non-SHAPES subgroups. Under each graph, the SHAPES justification criteria, the combination in Boolean space, and the subgroup characteristics are shown. Scenario (a) will not be used in the subsequent simulations because it is empty. Scenario (g) will be tested as the full population.

Three parameters are defined in the SHAPES approach:

- **k** - the maximum number of covariates considered for subgroup definitions. For instance, the example in Figure 1.4 has 3 covariates (k=3).

- **L** - the maximum number of covariates considered in defining SHAPES subgroups. All SHAPES subgroups defined by 1, 2, … L factors (with $L \leq k$) are considered. Take Figure 1.4 for example (k=3, therefore $L \leq 3$), if L=1, then all six possible subgroups in the form of (d) will be selected for SHAPES subgroups; if L=2, then (c), (d), and (e) will be considered as qualified SHAPES subgroups; if L=3, the subgroups in the forms of (b), (c), (d), (e), and (f) will be considered.
- **d** – depth, the number of covariates in a subgroup that is chosen by the SHAPES approach to have a beneficial treatment effect, where $d \leq L \leq k$. If the subgroup $X_1 \wedge X_2$ is chosen as showing beneficial treatment effect (over other SHAPES subgroups that are considered), then d=2.

All subgroups that qualify as SHAPES subgroups are generated for each k by the following steps. These steps identify all subgroups that are convexly connected in the Boolean space. The complementary groups of convexly connected subgroup are co-convexly connected in the Boolean space. Take k=3 in Figure 1.3 for example, there are 8 points that are expressed by 3 digits (i.e. "1 1 1").

- Step 1 – List all the possible subgroups of points. Each subgroup is a set of points. The number of possible subgroups/sets is (with k=3) $2^{2^k} = 2^8 = 256$. The number of points in each set is from 1 to 8. The subgroup with only one point is qualified for the SHAPES subgroup restriction.

- Step 2 – Sequentially select one set if the set has two or more points. All the combinations of any two points in this set are listed and sequentially select one pair of points starting from the top of the list. Calculate the minimum steps between the selected two points and list all the shortest paths as well as the points on the paths as shown in Figure 1.5. The number of shortest steps equals the number of the different digits between two points (i.e. "1 0 0" and "1 1 1" have two different digits). If only one digit is different, then the two points are one step away and continue to select another pair of points in that set (a set with only two connected points is a qualified SHAPES subgroup and the check is done). If the two points are more than one step away, then list the points on their shortest paths. For example, "1 0 0" and "1 1 1" are two steps away and there are two shortest paths connecting them: "1 0 0"→ "1 0 1"→ "1 1 1" (green color path on Figure 1.5) and "1 0 0"→ "1 1 0"→ "1 1 1" (blue color path in Figure 1.5). Check whether the points "1 0 1" and "1 1 0" are included in the set/subgroup. If not, then this set is not a qualified SHAPES subgroup, otherwise, continue to choose the next pair of points from the list. And repeat checking whether the points on the shortest paths are included in the set. If it is true that the shortest paths between any pairs of points in the set are included in the set, the set is confirmed to be a qualified SHAPES subgroup.

- Step 3 – Translate all the qualified SHAPES subgroups and the complementary subgroups into the form of Boolean expression as in Figure 1.4.

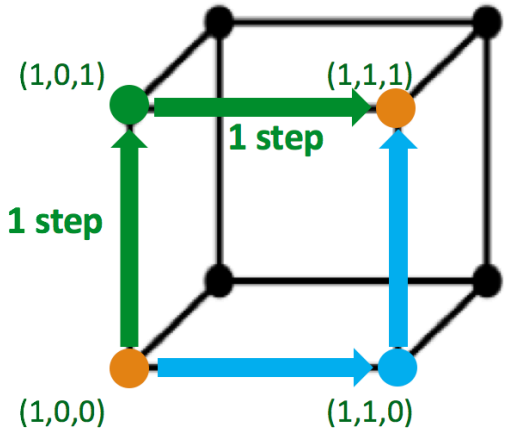


Figure 1.5. Take 3 covariates for example, each point is expressed by 3 digits. There are minimum of two steps between point (1,0,0) and point (1,1,1) by counting the number of green arrows or blue arrows between them. There are two shortest paths that cover one green point on each path. Our aim is to figure out: 1) the minimum steps between two points, 2) all the shortest paths to connect these two points, and 3) the points on the shortest paths.

We recognized a pattern in that only intersections like "$X_1 \wedge X_2 \wedge X_3$" and unions like "$X_1 \vee X_2 \vee X_3$" are qualified SHAPES subgroups formats, while any mixture of unions and intersections like "$(X_1 \wedge X_2) \vee X_3$" is not a qualified SHAPES subgroup.

The steps work well when the overall number of covariates under consideration is less than or equal to 4 ($k \leq 4$) but work slowly when $k>4$ because the number of possible subgroups that need to be checked is more than 4.3×10^9. We take advantage of the pattern recognized when $k \leq 4$ and use that pattern for settings with more than 4 covariates. It is obvious that all intersections are convexly connected SHAPES subgroups and unions are co-convexly connected subgroups. In the following we show that any mixture of unions and intersections is neither convex nor co-convex and thus violates the requirements for SHAPES subgroups. We show that for all mixtures we can find a point on the shortest pathway between two points (which are in the set) that is not included

in the set. Take $(X_1 \wedge X_2 \wedge X_3) \vee (X_4 \wedge X_5 \wedge X_6)$ when k=6 for example. Two points that are in this set are: "1 1 1 0 0 0" and "0 0 0 1 1 1". The minimum number of steps between these two points is 6, which means that all the other points are part of the shortest paths, but are not included in the set. E.g. point "1 1 1 0 1 0" is one step away from "1 1 1 0 0 0", is on one of the shortest paths between "1 1 1 0 0 0" and "0 0 0 1 1 1", but is not included in the set. Take $(X_1 \wedge X_2) \vee X^c_3$ (k=3) for an additional example, we can find out two points "1 1 1" and "0 0 0" from the set that have 3 steps between them. The 3-D figure is shown in Figure 1.4 (h), which indicates that all the points that are one/two steps away from either "1 1 1" or "0 0 0" would need to be in the set. However, points "0 0 1", "1 0 1", and "0 1 1" on the shortest pathways are not included in the set, proving that $(X_1 \wedge X_2) \vee X^c_3$ is not a convexly connected subgroup.

The qualified SHAPES subgroups formats are summarized in Figure 1.6: all the unions and intersections are qualified SHAPES subgroups. Each binary covariate defines two subgroups, like sex define male (sex=1) and female (sex=0). For example, we define X_1 to represent X_1=1 (male) and X_1^c to represent X_1=0 (female). Since L is the maximum number of covariates in the SHAPES subgroup, thus L=3 allows the full population (depth=0), subgroups defined by one covariate (depth=1), subgroups defined by two covariates (depth=2), and subgroups defined by three covariates (depth=3). The full population is qualified for the definition of SHAPES but it will not be considered as a subgroup. When depth=2, the unions of two covariates could be the unions of any 2 out of k covariates. The value of each covariate could be 0 or 1. Therefore, the number of SHAPES subgroups is $2 * k \; for \; d = 1$ and $\binom{k}{d} * 2^d * 2 \; for \; d \geq 2$. The number is doubled when $d \geq 2$ because the intersections are the complementary sets of unions.

SHAPES			
depth=0	Full		L=1
depth=1	X_i^a	X_i^a	L=2
depth=2	$X_i^a \wedge X_j^b$	$X_i^a \vee X_j^b$	L=3
depth=3	$X_i^a \wedge X_j^b \wedge X_h^f$	$X_i^a \vee X_j^b \vee X_h^f$	
…			

$Where\ \ i,j,h \in \{1,2,\dots k\} and\ a,b,f \in \{\emptyset, c\}$.

Figure 1.6. All the possible combinations of SHAPES subgroups. All the combinations of unions and intersections are qualified SHAPES subgroups. Here ∅ represents the characteristic (e.g. males) and c represents the complement of the characteristic (e.g. females).

Table 1.1 provides the number of qualified SHAPES subgroups under each combination of k and L. We can see that the number of qualified SHAPES subgroups increases when either L or k increases. Specifically, the number increases faster as L increases relatively to k. This number relates to the length of computing processing time. We are interested in not only just how far we can push k when L=1, but also in pushing both k and L to a higher level together. The number of the SHAPES subgroups is the key factor of running time. The right column shows the number of all possible combinations of points of each k (the number of subgroups tested in the logic regression subgroup method[14]). With the restriction of SHAPES, the number of subgroups that will be tested is significantly reduced compared to all possible combinations. The null and full are counted as qualified SHAPES subgroups but they are not considered as subgroups in the subsequent tests because the full set is for the full population level searching and the null set could not be tested.

Table 1.1. Number of subgroups in the SHAPES method compare to the number of all possible combinations.

	SHAPES							All Possible Combinations
	L=1	L=2	L=3	L=4	L=5	L=6	L=7	
k= 1	4							4
k= 2	6	14						16
k= 3	8	32	48					256
k= 4	10	58	122	154				65536
k= 5	12	92	252	412	476			4294967296
k= 6	14	134	454	934	1318	1446		1.84E+19
k= 7	16	184	744	1864	3208	4104	4360	3.40E+38
k= 8	18	242	1138	3378	6962	10546	12594	1.16E+77
k= 9	20	308	1652	5684	13748	24500	33716	1.34E+154
k= 10	22	382	2302	9022	25150	52030	82750	$\rightarrow\infty$
k= 11	24	464	3104	13664	43232	102368	186848	$\rightarrow\infty$
k= 12	26	554	4074	19914	70602	188874	391626	$\rightarrow\infty$
k= 13	28	652	5228	28108	110476	330124	769420	$\rightarrow\infty$
k= 14	30	758	6582	38614	166742	551126	1429718	$\rightarrow\infty$
k= 15	32	872	8152	51832	244024	884664	2532024	$\rightarrow\infty$
k= 16	34	994	9954	68194	347746	1372770	4301410	$\rightarrow\infty$
k= 17	36	1124	12004	88164	484196	2068324	7047012	$\rightarrow\infty$
k= 18	38	1262	14318	112238	660590	3036782	11183726	$\rightarrow\infty$
k= 19	40	1408	16912	140944	885136	4358032	17257360	$\rightarrow\infty$
k= 20	42	1562	19802	174842	1167098	6128378	25973498	$\rightarrow\infty$
k= 25	52	2452	39252	444052	3844372	26513172	149572372	$\rightarrow\infty$
k= 30	62	3542	68502	945462	10065846	86069046	607233846	$\rightarrow\infty$
k= 35	72	4832	109552	1785072	22561520	230326000	1951803120	$\rightarrow\infty$
k= 40	82	6322	164402	3088882	45201394	536514034	5309265394	$\rightarrow\infty$
k= 45	92	8012	235052	5002892	83195468	1125763148	12742945868	$\rightarrow\infty$
k= 50	102	9902	323502	7693102	143293742	2177303342	27747709742	$\rightarrow\infty$
k= 100	202	39802	2627002	128106202	4946507482	1.58E+11	4.26E+12	$\rightarrow\infty$
k= 200	402	159602	21174002	2091092402	1.64E+11	1.07E+13	5.95E+14	$\rightarrow\infty$
k= 300	602	359402	71641002	10656958602	1.26E+12	1.25E+14	1.05E+16	$\rightarrow\infty$
k= 400	802	639202	170028002	33793704802	5.36E+12	7.07E+14	7.96E+16	$\rightarrow\infty$
k= 500	1002	999002	332335002	82669331002	1.64E+13	2.71E+15	3.83E+17	$\rightarrow\infty$

1.4 OTHER APPROACHES

There are some other methods whose aims include identifying subgroups and controlling overall Type I error. One method is called subgroup enrichment designs which allows testing of a few pre-specified subgroups in randomized trial study designs[15,16]. The predictive enrichment strategy selects patients who are more likely to respond to a particular treatment based on some specific aspects of the disease history or other characteristics related to drug mechanism. SHAPES is more flexible and does not require pre-specifying the subgroups.

Another available subgroup method is the Cross Validated Adaptive Signature Design (CVASD) [17] [18]. The method uses a similar alpha allocation mechanism as SHAPES. The overall α-level is split into the full population search as well as the subgroup search. However, the analysis method is different from SHAPES. CVASD divides the participants into two cohorts, training and testing, and then incorporates cross validation to identify subgroups. SHAPES does not use this two cohorts method for verification because often the sample size is limited.

A third subgroup search method is called Subgroup Identification based on Differential Effect Search (SIDES)[19]. It selects a small number of biomarkers with the highest predictive ability first and then identifies subgroups with enhanced treatment effect by using the selected biomarkers. SIDES uses a recursive algorithm on a list of covariates from 5 to 100 with a full population size larger than 300. It can identify the subgroups that are only defined by one or two factors (e.g. current age>52 and first onset age<39). In comparison, SHAPES allows for more factors in defining subgroups and clarifies how to make the decision when more than one qualified SHAPES

subgroups are statistical significant (e.g. choose the subgroup with the lowest standardized p-value when three qualified SHAPES subgroups have standardized p-values below a significance level).

A fourth method is the Bayesian credible subgroup approach to identify a pair of bounding subgroups (one has treatment effect exceeding a threshold and the other does not) [20]. The credible subgroup is identified by estimating the posterior probability. Another subgroup method is the two-stage strategy which is designed for a time-to-event endpoint. This method estimates patient specific Multi-marker Molecular Signatures (MMMS) to identify biomarker-driven subgroups with enhanced treatment effect[21]. MMMS method applies elastic net on the entire population by building a large model including all the interaction terms between covariates and the treatment indicator. The subgroups are defined by the biomarker exceeding a threshold and therefore it can work for both binary and continuous covariates. The subgroups are ordered by descending treatment effect in each bootstrap first and then all the bootstrap's results are summarized and the subgroup with the highest frequency at the top of the ordered list is identified. The overall Type I error is approximately maintained well in the simulation scenarios.

Chapter 2. METHODS

In this chapter, we will first present the notation of SHAPES and how to identify SHAPES subgroups. Then we will show the analysis methods as well as the strategies in controlling Type I error. At the end of this chapter, we will discuss the correlation structure that we used in the simulations for correlated covariates.

2.1 SIMULATION

We assume a two-arm randomized clinical trial with outcome variable Y, treatment group indicator T (1=treated and 0=control), and covariates **X** (X_1, X_2, X_3, …). Y is either a binary outcome or a continuous outcome. The covariates **X** have a dimension from 1 to 50 and all of them are binary. If we are interested in continuous covariates, then we can dichotomize the continuous variables into binary. The covariates have two scenarios: 1) all covariates are independent to each other, and 2) partial or all of them are correlated. The goal is to detect a subgroup of patients, defined by S, that have enhanced treatment effect on Y. S is an indicator for SHAPES subgroup and it is created once we list all qualified SHAPES subgroups. For instance, the subgroup can be X_1, $X_1 \wedge X_2$, or $X_1 \vee X_2$, etc.

Compared to Prince's work, we extend the exploration range of investigating the properties of SHAPES. The simulation settings here are different from Prince's work in: 1) adding continuous outcomes in addition to binary outcomes, 2) outcomes are generated under the null hypothesis instead of under either null or alternative hypothesis, 3) all covariates have the identical prevalence, 4) allowing covariates to be correlated together, and 5) pushing both k and L to higher levels. We

mainly focus on testing how does SHAPES control for Type I error under the null hypothesis (no treatment effect difference in full population or subgroup). The simulation strategies and programming (conducted in R 3.3.2) are attached in the Appendix A.

2.2 IDENTIFY SHAPES SUBGROUPS

As presented in Figure 1.6, the SHAPES subgroups for each depth are the intersection and union sets of covariates. The indicator variable, S, was created to represent the potential SHAPES subgroup. First, we list out the subset of subgroups that satisfy the SHAPES restriction and then a list of all possible subgroups S will be created for each SHAPES subgroup. For example, if we have four covariates (k=4) and we plan to test whether there is a SHAPES subgroup defined by up to two covariates (L=2), then all the 56 qualified SHAPES subgroups will be considered and 56 S variables will be created. Second, S=1 for selected subgroup patients and S=0 for the unselected patients (complimentary group). Below are two examples of how we generate S when we have 9 covariates under consideration, where i and j are from 1 to the number of all qualified SHAPES subgroups.

When the subgroup is defined as "$X_2 \wedge X_9^c$", then

$$S_i = \begin{cases} 1 & if\ X_2 = 1 \text{ and } X_9 = 0, \\ 0 & else. \end{cases}$$

Similarly, when the subgroup is defined as "$X_1 \vee X_3^c \vee X_5 \vee X_7^c \vee X_9$", then

$$S_j = \begin{cases} 1 & if\ X_1 = 1 \text{ or } X_3 = 0 \text{ or } X_5 = 1 \text{ or } X_7 = 0 \text{ or } X_9 = 1, \\ 0 & else. \end{cases}$$

2.3 Analysis Methods

To test for a significant treatment effect, we build four different models, which are presented and defined in detail below.

Stratification model for binary outcome is:
$$logit\left(P(Y=1|T,X,S=1)\right) = \alpha + \beta T \quad \text{(M1)}$$

Interaction model for binary outcome is:
$$logit\left(P(Y=1|T,X,S)\right) = \alpha + \beta T + \gamma S + \varphi TS \quad \text{(M2)}$$

Stratification linear model for continuous outcome is:
$$E(Y|T,X,S=1) = \alpha + \beta T \quad \text{(M3)}$$

Linear model with interaction for continuous outcome is:
$$E(Y|T,X,S) = \alpha + \beta T + \gamma S + \varphi TS \quad \text{(M4)}$$

The SHAPES subgroup (indicated by S) is evaluated as a potential effect modifier. Two main models are used for the evaluation: a stratified model (M1 or M3) and an interaction model (M2 or M4). Prince used stratified models when presenting results. We will use both models (stratified and interaction model) to study whether there is a difference between them. The stratification model is fitted within the SHAPES subgroup population (S=1) for subgroup level search and it will also be fitted to the full population. In the stratified model M1 for a binary outcome, the coefficient β estimates the log odds ratio of recovery (suppose recovery is the outcome) between treatment group and control group within the subgroup. In stratification model M3 for continuous outcome, β estimates the mean difference of outcome between treatment group and control group within the subgroup. Next, we will introduce the coefficients in the interaction models. In the interaction model M2 for binary outcomes, the coefficient β estimates the log odds ratio comparing treatment group and control group among the complementary subjects of SHAPES subgroup (S=0), the coefficient φ for the interaction term estimates the difference between the log odds ratio

comparing treatment vs control among SHAPES subgroup and the log odds ratio comparing treatment vs control among the complementary subjects of SHAPES subgroup. We are interested in the linear combination of $\beta + \varphi$ because it estimates the log odds ratio comparing treatment group and control group within SHAPES subgroup. Similarly, in the interaction model M4 for continuous outcome, we are interested in the linear combination of $\beta + \varphi$ because it estimates the mean outcome difference comparing treatment group and control group within SHAPES subgroup. When we apply the interaction models for subgroup level searching, the full population level searching model is the same model as in the stratification model, that is, no interaction terms for full population testing. The hypothesis for the stratification models is: Ho: $\beta \leq 0$, Ha: $\beta > 0$; while the hypothesis for interaction models is: Ho: $\beta + \varphi \leq 0$, Ha: $\beta + \varphi > 0$. All the tests will be one-sided since we are interested in subgroups with enhanced treatment (one direction).

After defining the qualified SHAPES subgroups, stratification models and interaction models will be fitted one by one for the full population and every subgroup. For example, when k=4 and L=2 (4 independent covariates are considered and the maximum number of covariates that could be used to define a subgroup is 2), 57 tests (1 for full population, 8 for one covariate defined subgroups, and 48 for two covariates defined subgroups) will be performed.

2.4 TYPE I ERROR CONTROL

We adapted the alpha allocation mechanism from Prince[14] to distribute the overall significance level for each test. The overall Type I error α_T is 0.1 and it is split into the full population (α_F) as well as subgroup ($\alpha_s = \alpha_T - \alpha_F$). Then α_s will be equally split into each depth. If L=3, then the Type I error for each depth of subgroup testing is $(\alpha_T - \alpha_F)/3$. This alpha allocation scheme

allows flexibility in assessing the treatment effect among entire population or subgroups[13]. Full population and subgroups are tested separately with different significance levels. The advantage of this alpha allocation scheme is that it allows us to put more weight on the full population or subgroups based on previous knowledge or other prioritization. In total, we need to make sure that the $\Pr(p_F < c_F) \cup \Pr(p_S < c_S) < \alpha_T$, where p_F & p_S are the one-sided p-values and c_F & c_S are the critical values on the p-value scale for the full population and subgroups, respectively. Since SHAPES contains L depths for each analysis, the c_S contains L elements ($c_{d1}, c_{d2} \dots, c_{dL}$) for each depth.

There are three ways of obtaining critical values ($c_F, c_{d1}, c_{d2}, \dots c_{dL}$) on p-value scale. These critical values will be used to define the statistical significance.

- Alpha allocation values (unadjusted): $c_F = \alpha_F$, and for the i-th depth $c_{di} = (\alpha_T - \alpha_F)/L$. This unadjusted way is expected to lead to Type I inflation in the testing, because it insufficiently accounts for multiple comparisons.
- Critical values from simulations[14]: ($c_F, c_{d1}, c_{d2}, \dots c_{dL}$) will be selected from 5,000 simulation results that are performed under the null assuming no correlation between covariates. The results are combined as a matrix of 5,000 rows and L+1 columns that consist of one-sided p-values. Since the p-values are correlated across depths and our goal is to maintain the overall Type I error to be α_T, thus a cumulative way of calculating the percentiles is used (steps in Figure 2.1).
    - Step 1 - c_F equals to the $(100 * \alpha_F)th$ percentile of the 5,000 one-sided p-values from full population level tests. We use this percentile instead of α_F , which is different from Prince's work. Then the rows with $p_F \leq c_F$ will be deleted.

For example, if α_F=0.02, then c_F is the 2th percentile of 5,000 p values. One hundred rows in the result matrix will be deleted and 4,900 rows will be used in the next step.

- Step 2 - Within the remaining results, the critical value for 1st depth c_{d1} equals to the $(100 * \frac{\alpha_T - \alpha_F}{L})$th percentile of p-values in depth=1. The rows with p-values in depth=1 smaller or equal than c_{d1} will be deleted, the remaining result matrix will be used in the next step.
- Step 3 – Continue the process as in Step 2 for the next depth until we have c_{dL} for the L-th depth.
- Step 4 – Record $(c_F, c_{d1}, c_{d2}, \ldots c_{dL})$ as a vector.

- Bonferroni adjusted alpha vector: $c_F = \alpha_F$, and for the i-th depth $c_{di} = \frac{(\alpha_T - \alpha_F)/L}{\binom{k}{d} * 2^d * 2}$. For each depth, divide the allocated alpha value by the number of SHAPES subgroups under the depth. We expect that this is a very conservative way of controlling the overall Type I error.

P value results from 5000 repeated simulations with L=2.
Total alpha 0.1 is allocated to three levels: 0.02, 0.04, 0.04 respectively.

Full Population	Depth=1	Depth=2
p_{1_1}	p_{2_1}	p_{3_1}
p_{1_2}	p_{2_2}	p_{3_2}
p_{1_3}	p_{2_3}	p_{3_3}
p_{1_4}	p_{2_4}	p_{3_4}
p_{1_5}	p_{2_5}	p_{3_5}
…	…	…
p_{1_i}	p_{2_i}	p_{3_i}
…	…	…
p_{1_4997}	p_{2_4997}	p_{3_4997}
p_{1_4998}	p_{2_4998}	p_{3_4998}
p_{1_4999}	p_{2_4999}	p_{3_4999}
p_{1_5000}	p_{2_5000}	p_{3_5000}

5000

1st Step:
Sort the matrix by descending the column of "Full Population" and choose p_{1_m} as the 1st critical value.

Full Population	Depth=1	Depth=2
p_{1_719}	p_{2_719}	p_{3_719}
p_{1_811}	p_{2_811}	p_{3_811}
p_{1_7}	p_{2_7}	p_{3_7}
p_{1_4999}	p_{2_4999}	p_{3_4999}
p_{1_5}	p_{2_5}	p_{3_5}
…	…	…
p_{1_i}	p_{2_i}	p_{3_i}
…	…	…
p_{1_5}	p_{2_5}	p_{3_5}
p_{1_m}	**p_{2_m}**	**p_{3_m}**
…	…	…
p_{1_86}	**p_{2_86}**	**p_{3_86}**

5000 *0.02 =100

2st Step:
Remove the 100 rows at bottom
Sort the matrix by descending the column of "Depth=1"
Choose p_{2_n} as the 2st critical value.

Full Population	Depth=1	Depth=2
p_{1_3}	p_{2_3}	p_{3_3}
p_{1_4665}	p_{2_4665}	p_{3_4665}
…	…	…
p_{1_i}	p_{2_i}	p_{3_i}
…	…	…
p_{1_811}	p_{2_811}	p_{3_811}
p_{1_n}	**p_{2_n}**	**p_{3_n}**
…	…	…
p_{1_719}	**p_{2_719}**	**p_{3_719}**

5000 *0.04 =200

3rd Step:
Remove the 200 rows at bottom
Sort the matrix by descending the column of "Depth=2"
Choose p_{3_h} as the 3st critical value.

Full Population	Depth=1	Depth=2
p_{1_619}	p_{2_619}	p_{3_619}
…	…	…
p_{1_914}	p_{2_914}	p_{3_914}
p_{1_h}	**p_{2_h}**	**p_{3_h}**
…	…	…
p_{1_511}	**p_{2_511}**	**p_{3_511}**

5000 *0.04 =200

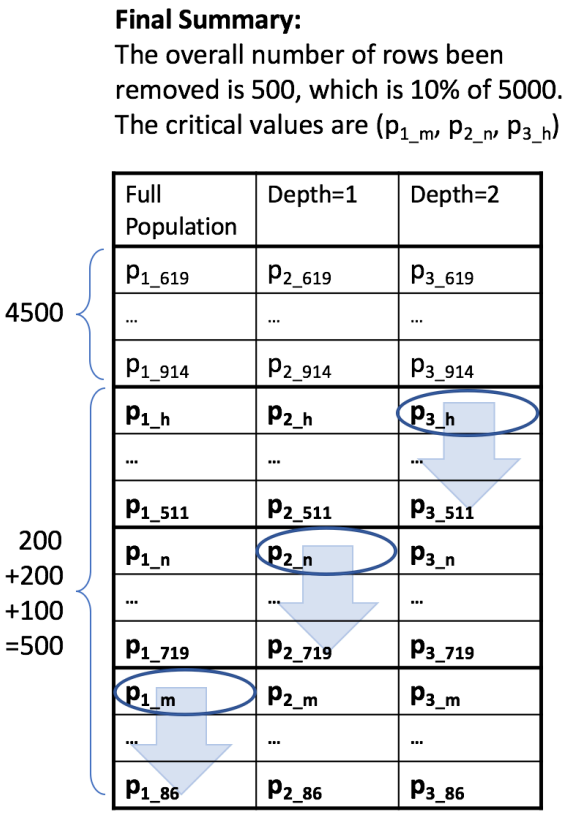


Figure 2.1. Steps of obtaining critical values from simulations. Here we use L=2 as an example, which means that the maximum number of covariates that can be used to define a SHAPES subgroup is 2. It will not be affected by the number of covariates (k) that are considered for subgroup analysis.

It's complicated to compare the results of multiple tests and make decisions regarding which group (full or subgroup) to choose when multiple of them are significant. The allocated α values for the full population and each depth are different, indicating that we cannot directly compare the p value from the full population test with the p-values from subgroup tests or compare the p-values across depths . To make sure that the results between depths, or between subgroups, vs full population are comparable, we created standardized *p* values.

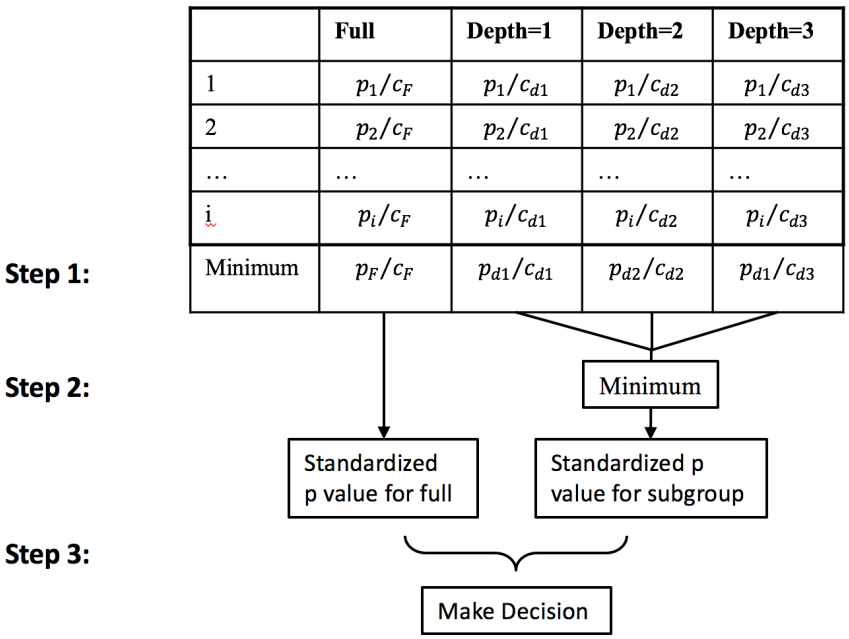


Figure 2.2. Calculating standardized p-values and comparing them between levels when L=3.

The standardized p values are calculated by dividing the p values by the corresponding critical values $(e.g.\ p_F/c_F, p_{di}/c_{di})$. As we discussed before, the critical values could be the alpha allocation values (identical for all subgroup depths) or could be those obtained from simulations (unique for each depth). The standardized p value below 1 means that the subgroup (or full population) has a significant enhanced treatment effect. Take the stratification model for example, for each simulation, we obtain a vector of p values $(p_F, p_{d1}, p_{d2}, \ldots p_{dL})$ for full population test, depth=1 subgroups test, depth=2 subgroups test, …, and depth=L subgroups test, respectively. The following steps are described in Figure 2.2. Within each depth, the subgroup with the smallest standardized p value is selected (Step 1). Comparing all the depths of subgroup searching, the subgroup with the smallest standardized p value across subgroups (depths) is selected (Step 2). When both the standardized p values of full and at least one subgroup are below 1, three utilities

will then be used to make the decision [14] (Step 3). Under the null hypothesis, the Type I errors by using these three utilities are theoretically identical, however, the proportion of the results that is contributed by full or subgroup varies among utilities.

- Prefer full - Select full population when both the results of full and subgroup are significant. If only one of them is significant and the other one is not, then select the significant one. This approach might be preferable when a treatment has low rates of side effects or is likely to benefit to all patients.
- Prefer subgroup - Select subgroup when both the results of full and subgroup are significant. If only one of them is significant and the other one is not, then select the significant one. This approach might be preferable when a treatment has high rate of adverse events, especially severe adverse events, but it has a clinically important benefit for selected patients.
- Minimum standardized p value: select the group with smaller standardized p value (min $(p_F/c_F, p_{di}/c_{di})$) when either the results of the full population or the subgroups are significant.

2.5 CORRELATION OF COVARIATES

Covariates are assumed to be independent in the simulations for determining critical values, however, in real practice, some covariates are typically correlated with each other. Therefore, the Type I error rates under the correlated scenarios will be explored by using the critical values obtained under independent situations. We do not consider simulating correlated covariates when determining critical values, because we don't know the true correlation structure in the real world. We will first let two of the covariates be correlated and then let more covariates be correlated at a

specific level of r. For example, if X_3 and X_4 are correlated, then $corr(X_3, X_4) = r$ and $corr(X_i, X_j) = 0$ where (i, j) is not a combination of (3, 4) or (4, 3) and $1 \leq i, j \leq k$. We will explore the properties of SHAPES when 1) the number of correlated covariates is 2 to the number of covariates considered in subgroup definition, and 2) the correlation magnitude among covariates from 0.1 to 0.6. The correlation structure for a dataset with four covariates is shown in Figure 2.3.

Correlation	X_1	X_2	X_3	X_4
X_1	1	0	0	0
X_2	0	1	0	0
X_3	0	0	1	0
X_4	0	0	0	1

All covariates are independent

Correlation	X_1	X_2	X_3	X_4
X_1	1	**r**	0	0
X_2	**r**	1	0	0
X_3	0	0	1	0
X_4	0	0	0	1

Two covariates are correlated

Correlation	X_1	X_2	X_3	X_4
X_1	1	**r**	**r**	0
X_2	**r**	1	**r**	0
X_3	**r**	**r**	1	0
X_4	0	0	0	1

Three covariates are correlated

Correlation	X_1	X_2	X_3	X_4
X_1	1	**r**	**r**	**r**
X_2	**r**	1	**r**	**r**
X_3	**r**	**r**	1	**r**
X_4	**r**	**r**	**r**	1

Four covariates are correlated

Figure 2.3. Correlation structure for 4 covariates in the dataset (k=4). The correlation between each pair of covariates is equal to r, and r is selected from 0.1, 0.2, 0.3, 0.4, 0.5, and 0.6.

Chapter 3. RESULTS

To gain insight into Type I error rate of SHAPES under different scenarios, we present the simulation results in two sections, for binary outcomes in Section 3.1 and continuous outcomes in Section 3.2. Within each outcome section, we summarize the results following the subsequent steps. First, we illustrate the simulation set-up, the selection of sample size and alpha allocation mechanism that will be used in the following simulations. Under each setting, we obtain the critical values from simulations under the null hypothesis with independent covariates, and then use the critical values as cut-off points of overall Type I error in other simulations. Second, we explore the overall Type I error rate of SHAPES when the number of covariates increase and the dimension of SHAPES subgroups changes. Third, we consider correlated covariates and explore the properties of SHAPES under this situation. The covariates could all be correlated, or a mixture of correlated and independent covariates.

3.1 BINARY OUTCOME SIMULATION

We start from the binary endpoint because it is representative in many clinical trials. All the simulations in this section are under the null hypothesis, which means that there is no overall treatment effect for the full population and no treatment effect for any subgroup. Both the probability of a positive outcome in the treatment group and the control group are equal to 0.3 as in Prince's research.

3.1.1 *Sample size*

To investigate the overall Type I error rate on different sample settings, we simulated the data with sample size from 100 to 1000. All of them are simulated based on k=4, L=2, α_T=0.1, and α_F=0.02. The results are presented in Figure 3.1. We used three methods of determining c_F and c_S as we described before: alpha allocation (unadjusted), critical values from simulations, and Bonferroni adjusted alpha vector. The overall Type I errors calculated by critical values from the simulations are very close to α_T and are consistent across different sample sizes. No difference between stratification and interaction models is observed by using this method. Consistent with our expectation, the overall Type I error rate of using alpha allocation method (unadjusted) is around 0.45, which is much higher than 0.1. Stratification models perform slightly better than interaction models, but are still very elevated using the allocation method. The overall Type I error rate of using Bonferroni adjusted alpha vector is substantially below 0.1, indicating that the Bonferroni method is too conservative in subgroup searching. The interaction model has a better estimate of Type I error compared to the stratification model, but is still conservative. The overall Type I error rates of using either the allocation or Bonferroni are slightly lower when the sample size is 100 compared to sample sizes larger than 200, while the method of critical values from simulations does not have this issue.

For the subsequent simulations, a sample size (n) of 500 is chosen based on two considerations. First, the sample size in Phase II studies is usually from dozens to hundreds, suggesting that a sample size of 500 is representative for the situation in the real world. Second, since we will push the number of covariates in SHAPES subgroups (L) from 4 to 6, a sample size of 500 will make

sure that most of the subgroups are larger than a minimum sample size. If the subgroup is smaller than 6 then it will not be considered in the subgroup search.

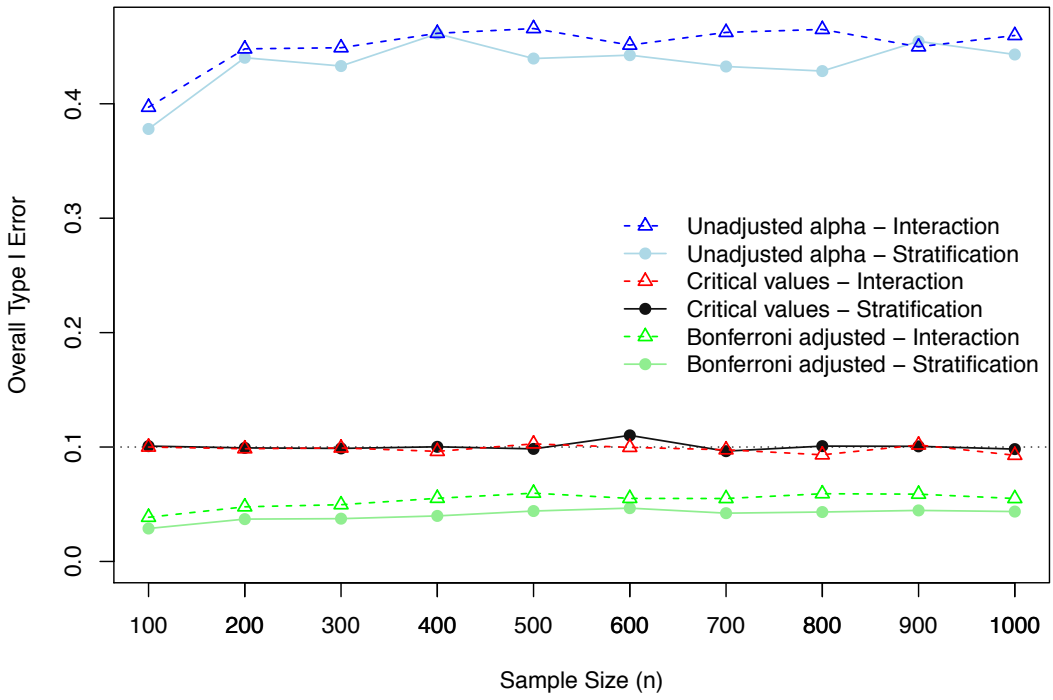


Figure 3.1. Comparing overall Type I error rate of SHAPES when sample size changes from 100 to 1000 with binary endpoint. Simulated for 5,000 replicates with fixed k=4, L=2, α_T=0.1, and α_F=0.02. Three pairs of lines represent using alpha allocation vector (unadjusted), critical values from simulations, and Bonferroni adjusted alpha vector as cut-off points. Each pair contains test results of stratification model (solid line) and interaction model (dashed line). The horizontal line of α_T=0.1can be used to compare the overall Type I error rates. Values at α_T = 0.1 are optimal.

In addition, we would like to discuss the sample size of SHAPES subgroups when L is larger than 4. For each depth, the subgroups can be classified as two groups by their Boolean expressions: the intersection group and the union group. The intersection group of subgroups have sample sizes

around $n * 0.5^d$ by assuming that all the covariates have the same prevalence of 0.5. Since the sample size of full population is 500, thus half of the subgroups will have a sample size around $500 * 0.5^5 \approx 7.8$ for the sixth depth when L=6. As we noted before, a subgroup of less than 6 subjects will not be tested. To observe whether the sample size will cause problems in the modeling, we generated a histogram (Figure 3.2) to summarize the minimum sample size on each depth when k=7 and L=6. When depth=1, the minimum sample size is close to 500*0.5=250. When depth is between 1 and 4, we don't need to be concerned about very small sample sizes of subgroups because even the minimum sample size is above 6. When depth=5, the minimum sample size is around 6 but we would not worry about this because the mean sample size of the intersection subgroups is $500 * 0.5^5 \approx 15.6$ and therefore sufficient for testing subgroups. When depth=6, every simulation has at least a subgroup that is smaller than 5. Half of the subgroups have a sample size around 7.8 but a small proportion of subgroups are smaller than 6 (10.75%, detail in Appendix B.1). If we increase L to 7, almost half of the 7^{th} depth subgroups will be smaller than 6 (40.78%, detail in Appendix B.2), while the other half of the subgroups are almost the same as the full population. Therefore, testing the 7^{th} depth subgroups is similar as testing the full population. If interest is in a SHAPES subgroup that is defined by more than 6 covariates, then the sample size of the full population should be larger than 1000.

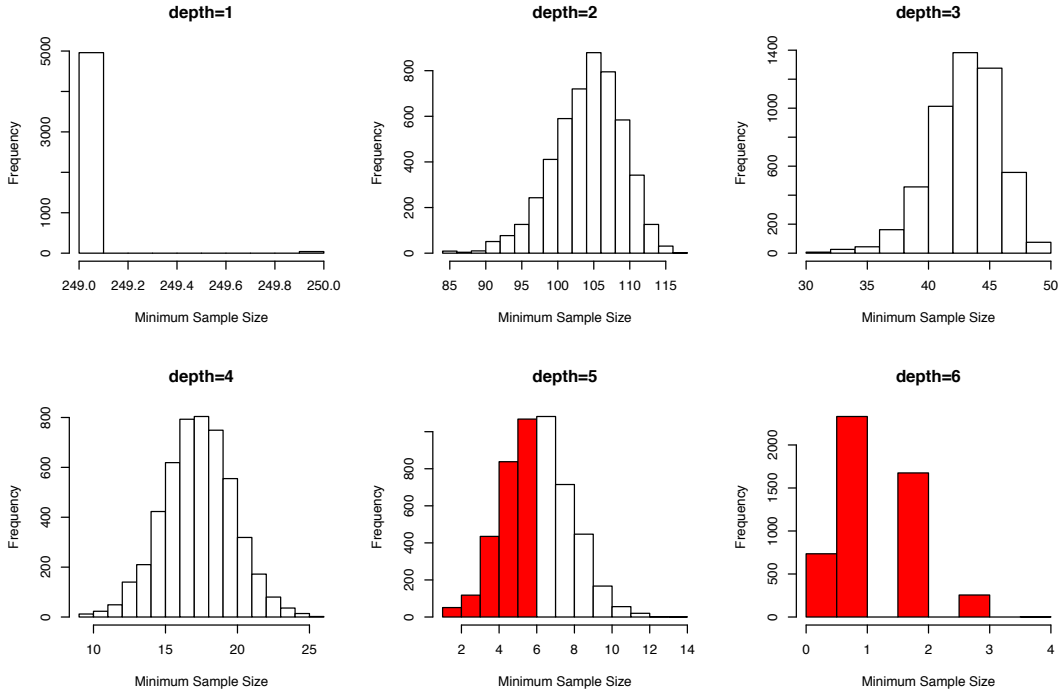


Figure 3.2. Distribution of minimum sample size of each depth when k=7 and L=6. The sample size is 500 and the prevalence of 7 covariates is 0.5. All covariates are independent to each other. Red color means that the sample size is below 6.

3.1.2 *Alpha allocation*

Splitting the overall Type I error ($\alpha_T = 0.1$) into full population search (α_F) and subgroups search is flexible. We can distribute any portion of α_F into full population level search based on our previous knowledge. We simulated data with α_F between 0 and 0.095 to investigate whether the overall Type I error will be changed. The simulations have sample size of 500, k=4 and L=2 because this setting is representative. The results are summarized in Figure 3.3. The overall Type I error rate by SHAPES using critical values from simulation is maintained around 0.1. No monotonic trend is observed across the proportions of α_F.

The overall Type I error rates of using unadjusted critical values as critical values start from 0.5 and decreases to 0.1. The overall Type I error rates of using Bonferroni adjusted critical values start from 0.05 and increases to 0.1. The overall Type I error rate of interaction models in these two methods is still higher than the stratification models, suggesting that the stratification model for unadjusted method and the interaction model for Bonferroni adjusted method perform better in controlling the Type I errors. When $\alpha_F = 1$, both the alpha allocation method and the Bonferroni method will control the Type I error around 0.1.

We select the α_F= 0.02 in the simulations of pushing k and L to higher levels because of a high weight of subgroup testing (α_s= 0.08) will help us observe the properties of SHAPES better. We plan to put more weight on the SHAPES subgroups analysis to investigate whether a subgroup or the full population will be identified as a group with positive treatment effect.

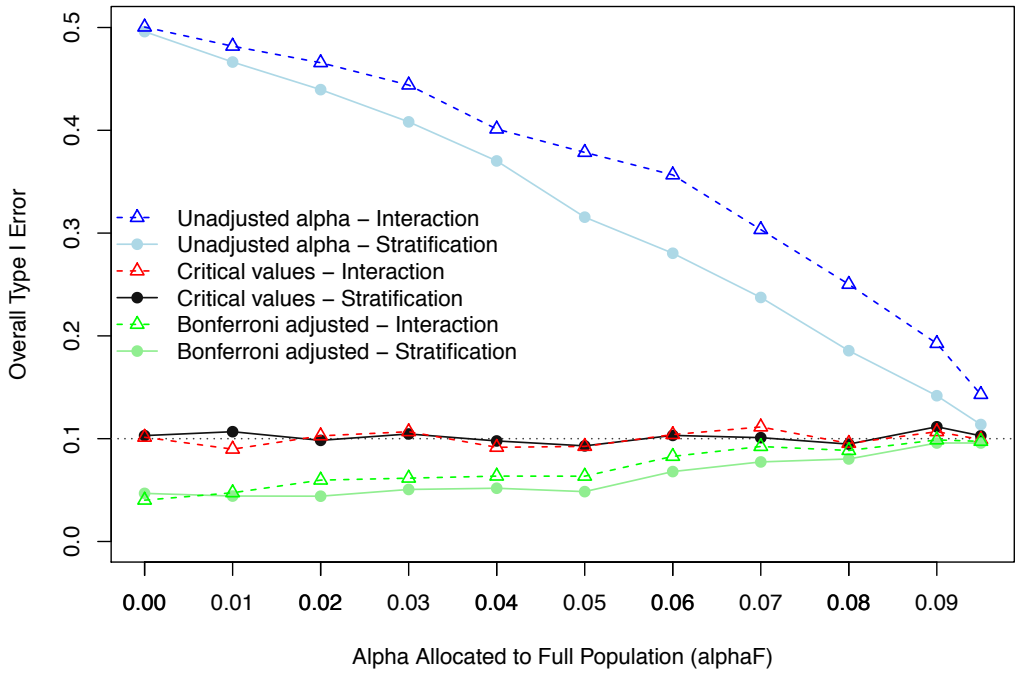


Figure 3.3. Comparing Type I error rate of SHAPES when the Type I error allocated to full population changes from 0 to 0.095. Simulated for 5,000 replicates with fixed k=4, L=2, α_T=0.1, and n=500. Three pairs of lines represent using alpha allocation vector, critical values from simulations, or Bonferroni adjusted alpha vector as cut-off points. Each pair contains test results of stratified model (solid line) and interaction model (dashed line). Values near α_T = 0.1 are optimal.

3.1.3 *Prevalence of covariates*

In Prince's research[14], the prevalence of covariates varies but he fixed the prevalence of all subgroups to be 0.5. For example, if there are four covariates X_1, X_2, X_3, and X_4 in the dataset and he selected the subgroup that is defined by $X_1 \wedge X_2$, then the following prevalence assignment

was designed: $P(X_1 = 1) = P(X_2 = 1) = 0.71$, $P(X_3 = 1) = P(X_4 = 1) = 0.5$, and $P(X_1 \wedge X_2 = 1) = 0.5$.

In this study, all covariates have the identical prevalence and the prevalence of subgroups is varying. To explore the effect of covariate prevalence on the overall Type I error rate, we simulated for two levels of prevalence: 0.5 and 0.2. In Figure 3.4, we increase the number of covariates from 2 to 10 with fixed L=2, n=500, α_T=0.1, and α_F=0.02. The overall Type I error rates of using critical values from simulation are consistently close to 0.1 in both scenarios. Both interaction models and stratification models perform very well. Since our next set of simulations will push the number of covariates in SHAPES subgroups (L) to a higher number we have chosen to set the covariates prevalence to be 0.5 to ensure the minimum sample size of subgroups is large enough.

The overall Type I error rates by using the unadjusted alpha allocation as critical values have a big issue in controlling for Type I errors. As k approaches 10, the overall Type I error increases to 0.8. As k increases then it is likely to increase to close to 1. The overall Type I error from Bonferroni adjusted methods remained around 0.05 and increases a little bit as k increases. The result from interaction models is slightly higher than the result from stratification models but is still conservative.

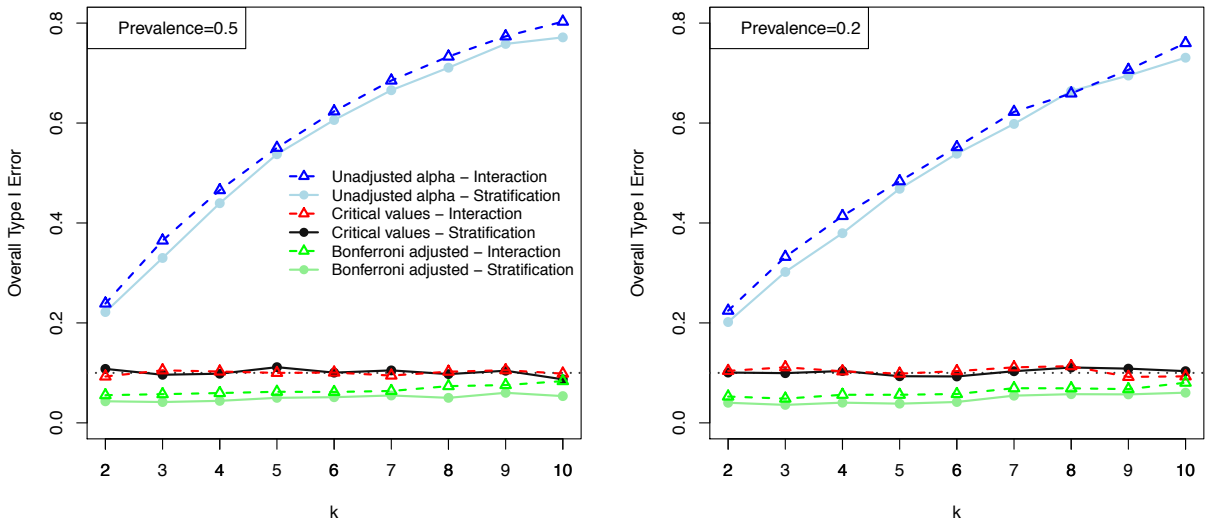


Figure 3.4. Overall Type I error of SHAPES when the prevalence of covariates is 0.5 (left side) and 0.2 (right side). The number of covariates (k) increases from 2 to 10. Simulated for 5,000 replications with binary outcome, n=500, L=2, α_T=0.1, α_F=0.02, and independent covariates. Three pairs of lines represent using alpha allocation vector, critical values from simulations, or Bonferroni adjusted alpha vector as cut-off points. Each pair contains test results of stratified model (solid line) and interaction model (dashed line). Values near α_T = 0.1 are optimal.

3.1.4 *Expanding k and L*

To explore the overall Type I error rate for each combination of k and L, we increase both k and L to higher levels. First, we fix L=1 (the maximum number of covariates being used to define a SHAPES subgroup is 1) and expand k (the number of covariates under consideration for subgroup definition) from 2 to 50. Second, we extend L from 1 to 6. The 5,000 simulations have identical setting of α_T=0.1, α_F=0.02, covariates prevalence=0.5, and n=500. The test results of 5,000 simulations can be considered as a binomial distribution with p=0.1, suggesting that the standard

error for the Type I error should be $\sqrt{\frac{0.1*0.9}{5000}} = 0.00424$ and the 95% confidence interval is $0.1 \pm (1.96 * 0.00424) \approx [0.09, 0.11]$. Therefore, we used the range of 0.1 ± 0.01 in the figures for comparing the variation of the overall Type I error rates to this range.

The combinations that have been investigated are shown in Figure 3.5. Here we only discuss the critical values from simulations. Each pair of lines represent stratification model and interaction model. The results are organized by the level of L. For each L, the 0.1 α level line and 0.1±0.01 lines are plotted for comparison. When the maximum number of covariates in SHAPES subgroup is 1 or 2 (L=1 or L=2), up to 50 covariates were simulated. It shows that overall Type I error is still robust when k is large. No monotonic trend is observed across the k ranges. Most of the overall Type I error rates are within 0.1±0.01 and only a few of points exceed this range. When k is between 2 and 10, we can observe that as L increases up to 6, the overall Type I error is still maintained very well around 0.1. When L=3 or L=4, the lines shows no trend along with k. The performance of stratification model and interaction model are similar without systematic differences.

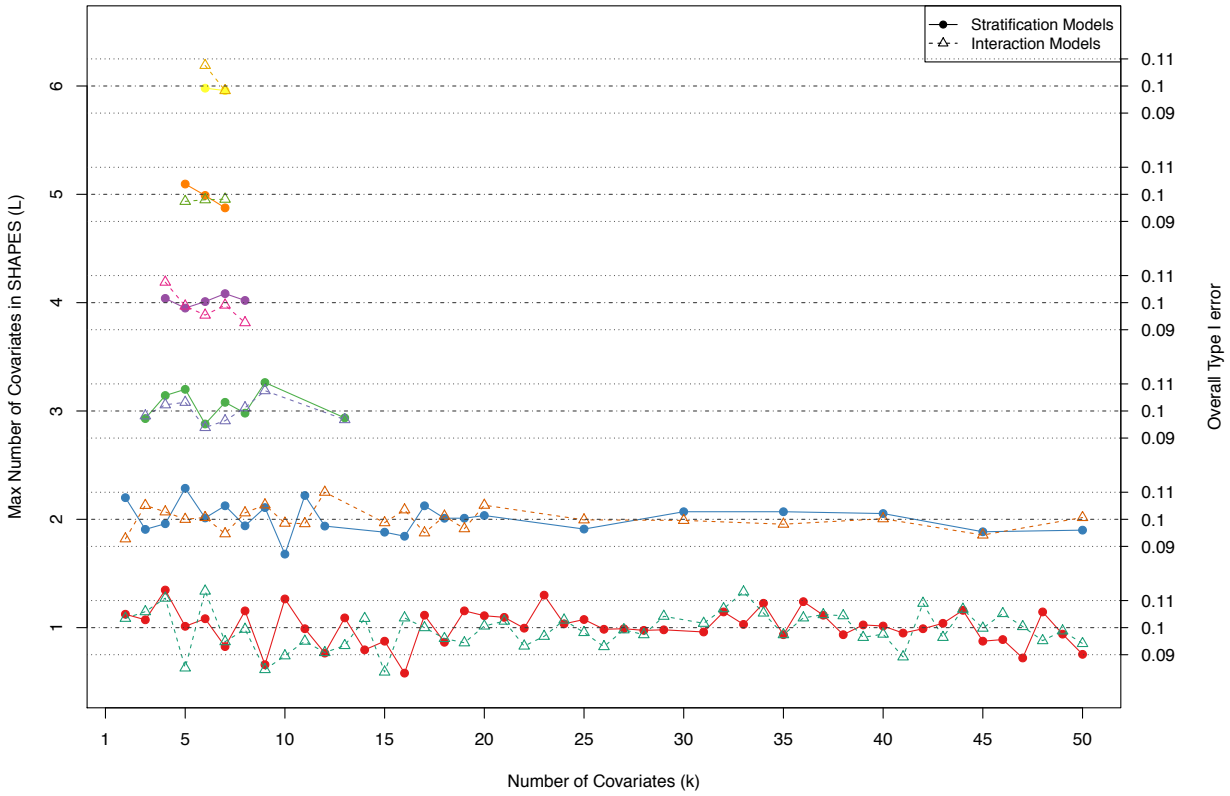


Figure 3.5. Comparing Type I error rates of SHAPES when k changes from 2 to 50 and L varies from 1 to 6. Simulated for 5,000 replicates with binary outcome, α_T =0.1, α_F =0.02, prevalence=0.5, and n=500. Each pair of lines contains test results of stratified model (circle, solid line) and interaction model (triangle, dashed line). Horizontal dashed lines represent the overall Type I error range of 0.1±0.01. Values at α_T = 0.1 are optimal.

3.1.5 *Correlated covariates*

In the simulations above, we assume that all the covariates are independent from each other. However, in practice, a group of covariates are typically correlated. Therefore, it is valuable to explore the performance of SHAPES by applying critical values generated from independent simulations on the correlated simulations. We simulated data with 4 covariates and allow up to 4 covariates to define a subgroup. First, we let two covariates be correlated and change the strength

of the correlation from 0.1 to 0.6. Second, then we let 3 or all of the 4 covariates be correlated. The correlation structure is described in Figure 2.3 and the results are summarized in Figure 3.6. For each number of correlated covariates, the overall Type I error rate is estimated via stratification models (solid line and points) and interaction models (dashed line and triangles). In general, the overall Type I error rates are higher than 0.1. The stratification model lines are slightly above the interaction model lines when two covariates are correlated together, which means that the interaction model controls the overall Type I error better in this situation. The overall Type I error lines are higher than 0.1 except the interaction models for 2 correlated covariates. The stratification model line for data with 4 correlated covariates has a slightly monotonic decrease trend as the correlation magnitude increases, indicating that the overall Type I error estimated via stratification models decreases when all the covariates are highly correlated. This trend is consistent with the trend of Type I error inflation of multiple comparisons: the Type I error inflation rate of multiple comparison on higher number of correlated factors is lower than those on the same number of independent factors [22].

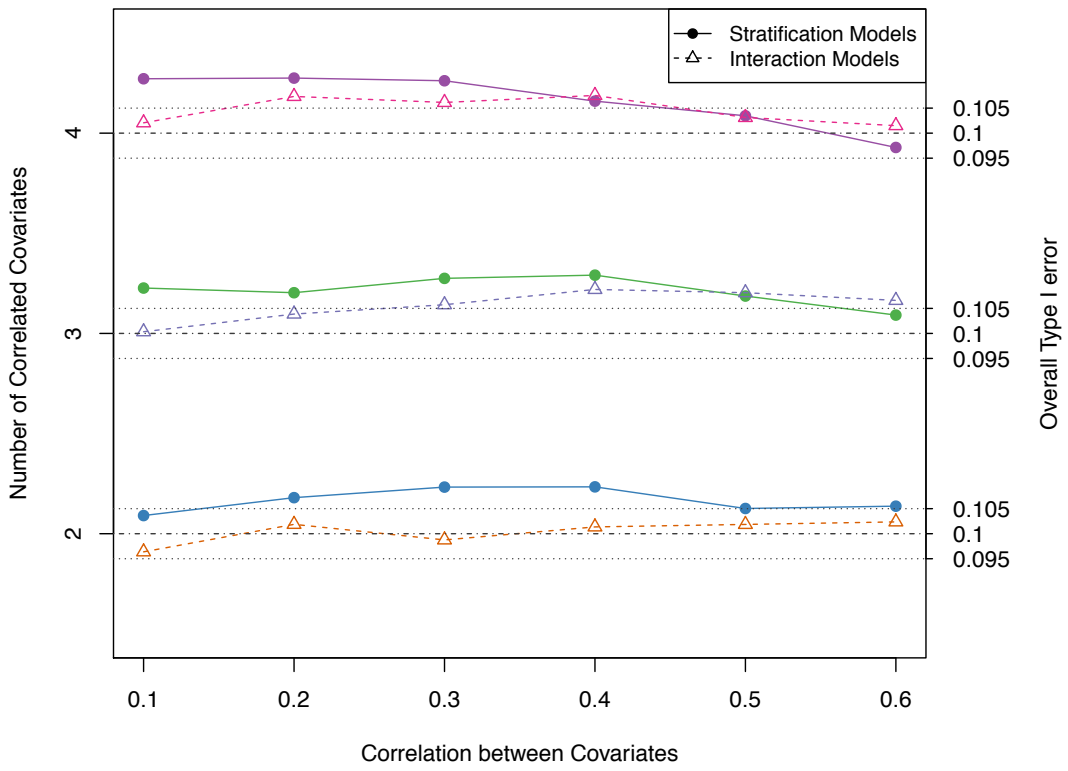


Figure 3.6. Overall Type I error when some or all the covariates are correlated together with various correlation levels. Simulated 15,000 replications for binary outcome to reduce variability in the simulation results. The number of covariates is 4, the maximum number of covariates in SHAPES subgroups is 4, n=500, α_T=0.1, and α_F=0.02. Stratification models (point, solid line) and interaction model (triangle, dashed line) are performed separately to obtain the overall Type I error. Values at α_T = 0.1 are optimal. Some of the estimated Type I error rates are outside of 95% confidence intervals (0.1±0.005).

3.2 CONTINUOUS OUTCOME SIMULATIONS

Continuous outcomes are frequently used in clinical trials and it is of interest to investigate the properties of SHAPES under the null hypothesis. This section uses the same organization as in 3.1.

3.2.1 *Sample size*

To investigate the overall Type I error rate on different sample size scenarios for continuous outcomes, we simulated the data with sample size from 100 to 1000 (Figure 3.7). The setting is the same as in 3.1.1: k=4, L=2, α_T=0.1, and α_F=0.02. When applying the critical values from simulations method, the overall Type I error rates are controlled very well. Both the stratification model line and the interaction model line are consistently close to 0.1. No trend is observed when the sample size increases. Therefore, sample size of 500 is appropriate in the following simulations. In addition, the performance of unadjusted alpha allocation method and the Bonferroni adjusted method are similar as in 3.1.1. The unadjusted lines are much higher than 0.1 with the stratification model performing better than interaction model. The Bonferroni adjusted lines are substantially below 0.1 with the interaction model performing better.

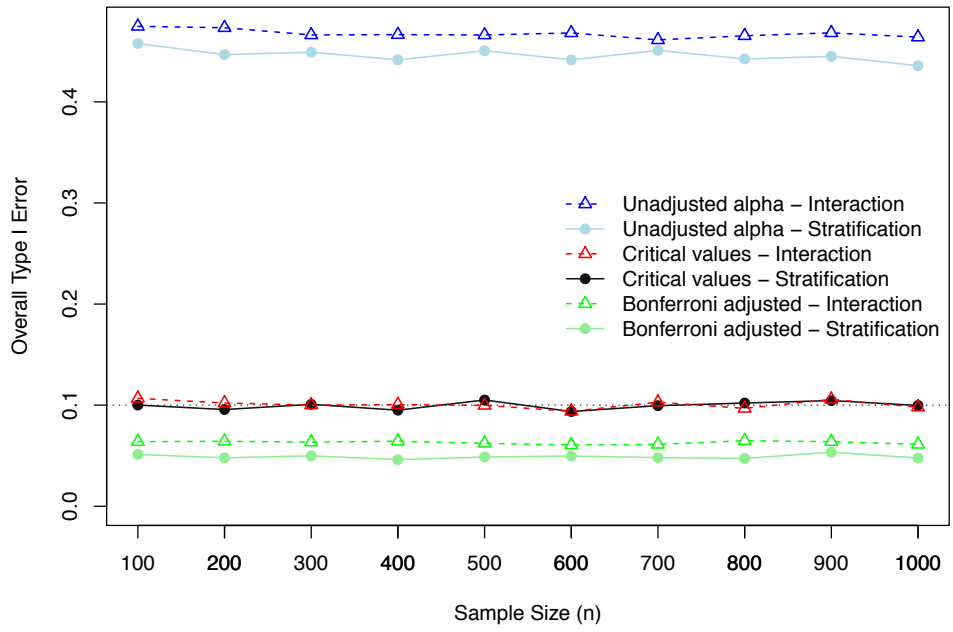


Figure 3.7. Comparing overall Type I error rate of SHAPES when sample size changes from 100 to 1000 with continuous endpoint. Simulated for 5,000 replicates with fixed k=4, L=2, α_T=0.1, and α_F=0.02. Three pairs of lines represent using alpha allocation vector (unadjusted), critical values from simulations, and Bonferroni adjusted alpha vector as cut-off points. Each pair contains test results of stratification model (solid line) and interaction model (dashed line). Values at α_T = 0.1 are optimal.

3.2.2 *Alpha allocation*

The simulations in Figure 3.8 use a serial proportion of α level that is distributed into population level. The α_T is 0.1 and α_F is from 0 to 0.095. The other settings in the simulations are the same: 500 subjects, 4 covariates, and allow up to 2 covariates to define a SHAPES subgroup. First, by using the critical values from simulations, the overall Type I error rates are maintained very well. No special trend or fluctuation is observed. Therefore, an α_F of 0.02 could be used in the following

simulations because we want to put more weight on the subgroup level searching. Second, the overall Type I errors of unadjusted method start from 0.5 and decrease to 0.1 with stratification model controls better. Third, the Bonferroni adjusted method is conservative when α_F is small and then goes to 0.1 as more proportion of α level distributed to full population. This is consistent with our expectation because of that the tests are similar as full population test as we put more weight on full level searching.

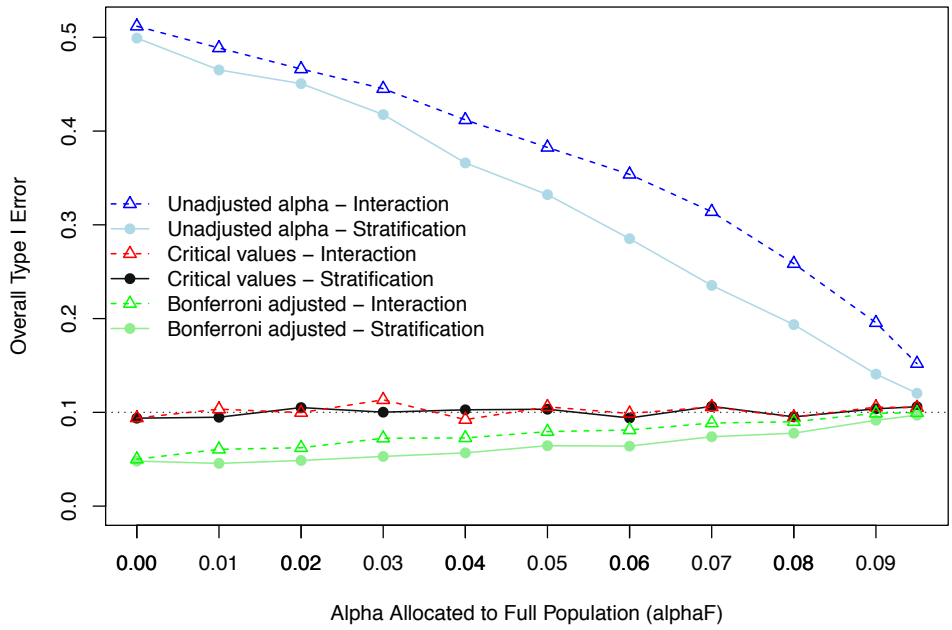


Figure 3.8. Comparing Type I error rate of SHAPES when the Type I error allocated to full population changes from 0 to 0.095 for continuous endpoint Simulated for 5,000 replicates with fixed k=4, L=2, α_T=0.1, and n=500. Three pairs of lines represent three methods of obtaining critical values from the alpha allocation vector, from simulations, and from Bonferroni adjusted alpha vector. Each pair contains test results of stratified model (solid line) and interaction model (dashed line). Values at α_T = 0.1 are optimal.

3.2.3 *Prevalence of covariates*

We simulated the data with all the prevalence of covariates equal to 0.5 or 0.2 to compare if there is a difference between them. The number of covariates considered for subgroup definition changes from 2 to 10 but the maximum number of SHAPES subgroups is restricted at 2. The sample size is fixed at 500 and the α_T=0.1 as well as α_F=0.02. As shown in Figure 3.9, the overall Type I error lines for both prevalence of 0.5 and 0.2 are close to 0.1. There are no large differences between the results of these two settings. Same as in the binary outcome simulations, each covariate with prevalence of 0.5 will be used in the following simulations. If the prevalence is below 0.5, the sample size of SHAPES subgroup for large L will be very small. The unadjusted method has overall Type I errors inflated to 0.8 when there are 10 covariates, while the Bonferroni adjusted method over controls the Type I error below 0.1.

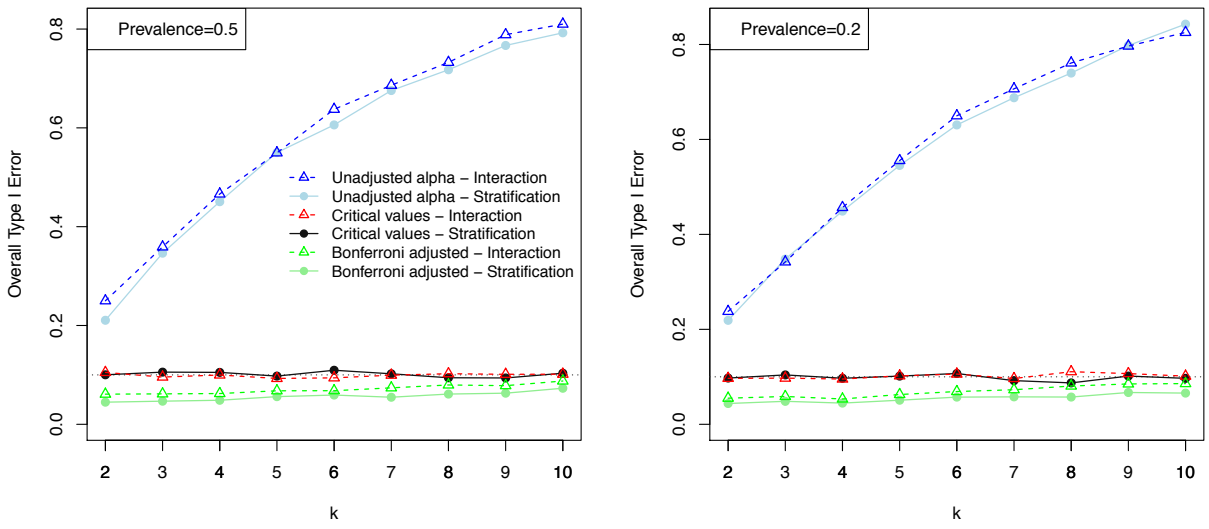


Figure 3.9. Overall Type I error of SHAPES when the prevalence of covariates is 0.5 (left) or 0.2 (right). Simulated for continuous outcome, n=500, L=2, α_T=0.1, α_F=0.02, independent covariates, and repeated for 5000 times. Three pairs of lines represent using alpha allocation vector, critical

values from simulations, or Bonferroni adjusted alpha vector as cut-off points. Each pair contains test results of stratified model (solid line) and interaction model (dashed line). Values at α_T = 0.1 are optimal.

3.2.4 *Expanding k and L*

Same as in the binary endpoint settings, the number of covariates (k) ranges from 2 to 50 and the maximum number of covariates in SHAPES subgroups (L) is from 1 to 6. The results are presented in Figure 3.10. For each L, there is a pair of lines that represent the overall Type I error from stratification models and interaction models. When L=1 or L=2, we simulated over a broad range of k. The overall Type I errors are maintained within the range of 0.1±0.01 and they do not follow a monotonic trend when k increases. When L increases from 1 to 6, the overall Type I errors are still controlled between 0.09 and 0.11. They are centered at 0.1 and have no trend across L levels. The stratification model results are similar to the interaction model results.

Compared to the result in 3.1.4, there's no significant difference between binary endpoint simulation results and continuous endpoint simulation results. The overall Type I error is controlled around 0.1 for the combinations of k and L that we simulated. Visually, the variation of the overall Type I error for continuous endpoint is slightly lower than the variation for binary endpoint.

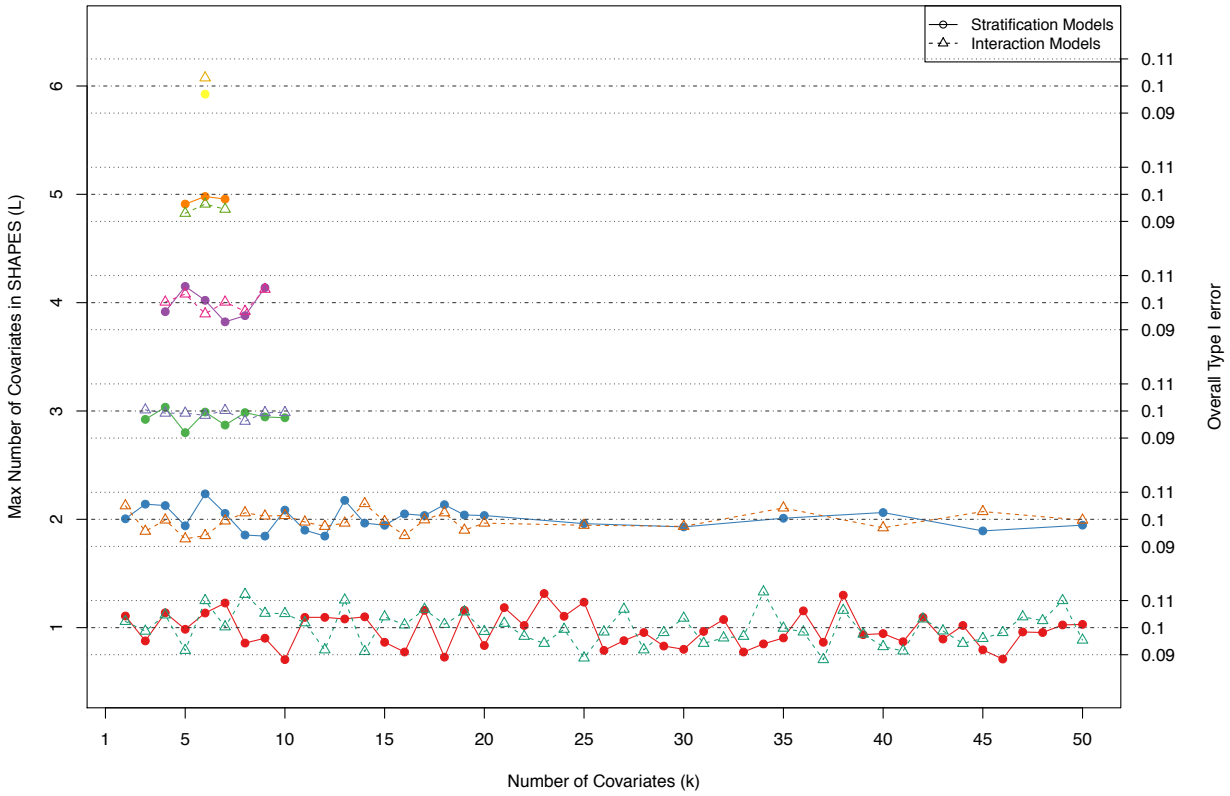


Figure 3.10. Comparing Type I error rate of SHAPES when k changes from 2 to 50 and L changes from 1 to 6 with continuous endpoint. Simulated for 5,000 replications with fixed α_T=0.1, α_F=0.02, and n=500. Each pair of the lines contain the test results of stratified model (circle, solid line) and interaction model (triangle, dashed line). Horizontal dashed lines represent the overall Type I error range of 0.1±0.01. Values at α_T = 0.1 are optimal.

3.2.5 *Correlated covariates*

The correlation structure for the continuous outcome is the same as for the binary outcome. Some or all of the covariates are correlated at a specific level. The critical values are obtained from independent covariate simulations and then are applied to the correlated data simulation results. We simulated data with 4 covariates and allow up to 4 covariates in the SHAPES subgroup. The sample size is 500, the total α level α_T is 0.1, and the α level distributed into full population α_F is

0.02. As shown in Figure 3.11, the number of correlated covariates is from 2 to 4 and the correlation magnitude between them is from 0.1 to 0.6. When 2 covariates are correlated, both the overall Type I errors of stratification model and interaction model are maintained around 0.1 across correlation magnitudes. When 3 out of 4 covariates are correlated, the overall Types I errors are close to 0.1 but they decrease from weak correlations to strong correlations. When all of the 4 covariates are correlated, a decreasing trend in overall Type I error as the correlation increases is observed. The overall Type I errors are around 0.1 at the beginning and decrease when the correlation gets stronger. The stratification model line decreases slightly faster than the interaction model line. The trend here is more obvious compared to the results in binary endpoints. It shows that the Type I error inflation rate of multiple comparisons is reduced when the data are highly correlated. When the correlation is equal to 0.6, the overall Type I error from stratification models and interaction models are 0.08 and 0.085, respectively, which is outside of the 95% confidence interval. We will reject less conservative when more strongly correlated covariates exist.

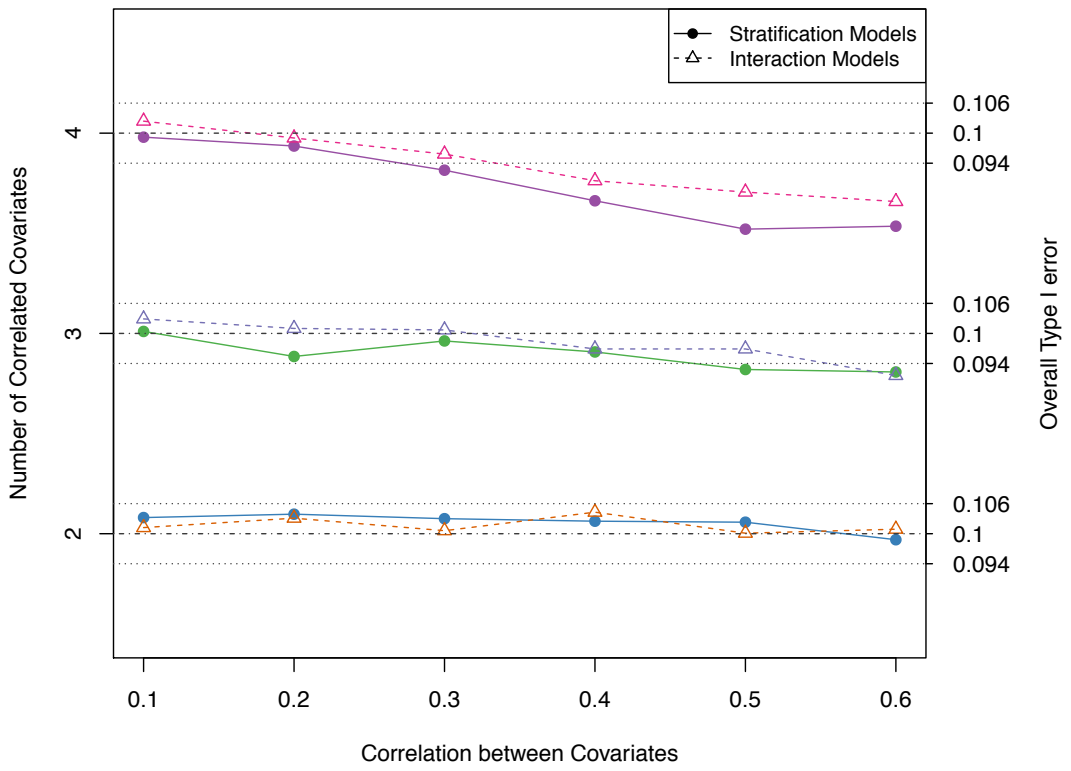


Figure 3.11. Overall Type I error when some or all the covariates are correlated together with various correlation levels. Simulated 10,000 replications for continuous outcome. The number of the covariates is 4, the maximum number of covariates in SHAPES subgroups is 4, n=500, α_T=0.1, and α_F=0.02. Stratification models (point, solid line) and interaction model (triangle, dashed line) are performed separately to obtain the overall Type I error. Values at α_T = 0.1 are optimal. Some of the estimated Type I error rates are outside of the 95% confidence intervals (0.1±0.006).

Chapter 4. LIMITATIONS

The types of endpoints that we simulated are binary and continuous. It would be of interest to simulate other types of endpoints such as time to event endpoints, count data or categorical endpoints with more than two categories. Some statistical models can be fitted for new endpoints, for instance, Cox proportional hazard model for time to event endpoint, Poisson regression or over dispersion Poisson regression for counts endpoint, and multinomial regression for categorical endpoint. Based on how similar the results for the binary and continuous endpoints are, we expect that the overall Type I error of SHAPES method will also be controlled in most scenarios if simulated similar to the scenarios evaluated here.

All the covariates that we simulated are binary, but other types of covariates can be transformed to fit in SHAPES. Continuous covariates can be dichotomized into binary variables by selecting a threshold. The threshold can be decided based on biological rational, for instance, systolic blood pressure and blood glucose have well-designed standards of defining categories in clinical practice. In addition, the prevalence of covariates that we considered are 0.5 and 0.2. Among them, 0.5 is used for most of the simulations. However, in real data, not all the covariates have such a high prevalence. If we are going to investigate the properties of SHAPES when the prevalence of covariates is limited at a low level, we would suggest to use a larger sample size and a small L. The main reason for this choice is to maintain the sample size of SHAPES subgroups. Otherwise, the subgroup with only a few subjects will cause problem in modeling particularly for binary outcomes. Moreover, it is highly possible that the covariates prevalence is mixed by high prevalence (e.g. male, low BMI or wild genotype) and low prevalence (e.g. mutated genotype, biomarker positive). The correlation structure of the covariates could also be more complex.

Simulations accounting for flexible correlation structures and prevalence levels could be done in the next step.

We used 0.1 as the overall Type I error (α_T) in the simulations, however, the overall Type I error could be different from 0.1. Modifying the overall Type I error from 0.1 to 0.2 or 0.05 will help us understand the properties of SHAPES deeply. Additional simulations could be performed to test whether the overall Type I error will be affected by α_T. We discussed the relationship between the overall Type I error and the α level distributed into full population (α_F) in 3.1.2 and 3.2.2. Since there is no difference in that setting, we choose $\alpha_F = 0.02$ in the other settings. However, it may be possible that the conclusions in simulations for independent covariates (i.e. 3.1.4 and 3.2.4) or correlated covariates (i.e. 3.1.5 and 3.2.5) will be affected by the α_F level. Therefore α_F =0.05 or 0.1 should be covered in more simulations. In addition, the selection of critical values from simulations as a cut off point for full population level Type I error is valuable to be investigated further. We use the percentile from the simulation results instead of using α_F as in Prince's work because of that these two values are not always consistent in simulations.

When we generate the outcomes, a baseline effect of 0.3 is used for binary outcomes and a standard deviation of 1 is used for continuous outcomes. More work could be done to explore the fitting issues with rare outcomes and high variability of the outcomes. Methods can have different performance if the outcome is more variable or rarer. When fitting the logistic regression on dataset with rare outcome, the odds ratio becomes more variable as the outcome becomes rarer. This could affect the Type I error rate and make it lower or higher than desired in simulation studies. Having

a more variable continuous outcome could also cause a problem with Type I error estimation with the same sample size based on small sample performance issues.

Pushing k to a higher level will expand the usage of SHAPES because the number of covariates in statistical genetics or biomarker studies is much higher than 50. The maximum number of covariates (k) in the simulations is 50, which is limited by the length of processing time of running programs on clusters. Suggestions for improving the computer programming efficiency are listed in the Appendix. According to Table 1.1, the processing time depends mostly on the number of SHAPES subgroups, which is more sensitive to L compared to k. For instance, running k=10 and L=4 is similar as running k=50 and L=2. Improving the R programming and the simulations algorithm or running programs on a faster cluster (i.e. GPU) will be helpful for exploring the properties of SHAPES on a broader range of data settings.

In a summary, to explore the properties of SHAPES deeper, existing scenarios can be covered more comprehensively. The important aspects in the simulations including the number of the covariates, the maximum number of covariates in SHAPES subgroups, the sample size, the overall α level, the α level distributed into full and subgroups, the prevalence of covariates, the correlation structure, and the type of endpoints can be modified to expand the evaluation of SHAPES.

Chapter 5. DISCUSSION

The major goal of this study is to investigate the properties of SHAPES, especially in controlling the overall Type I error. SHAPES is developed to identify subgroups with sufficiently large treatment effect compared to the overall population. In contrast, some studies use the interaction model and only consider the coefficient of the covariate interaction term [11,17,18,19,23]. They compare the treatment effect of subgroup to the complementary group (i.e. ASD, SIDES), to identify subgroups with enhanced treatment effect over complementary subgroup. This methodology may identify a subgroup as benefit subgroup when its complementary group has harmful treatment effect. Since subgroups with harmful treatment effect are not within the scope of this study, we purposefully used one-sided test and focused only on identifying subgroups with enhanced treatment effect than the population-average effect, which are clinically meaningful and beneficial for patients.

The sample size of subgroups is critical for identifying subgroups with truly enhanced treatment effect. The size of subgroup is determined by the population size and prevalence of the covariates. A typical Phase II study commonly enrolls dozens or a few hundreds of patients, with exceptions such as large-scale cardiovascular studies. On the other hand, the prevalence of some covariates in the real data is much lower than 50% that we used in the simulations. For instance, in a full-population with 500 subjects, if the subgroup is defined by two independent covariates $X_1 \wedge X_2$ (X_1=1 and X_2=1) with a low prevalence of $P(X_1=1) = P(X_2=1) = 0.1$, then $P(X_1 \wedge X_2 =1)= P(X_1=1) * P(X_2=1)=0.01$ and the sample size of this subgroup is around 5. As we defined in the testing, a

subgroup with less or equal than 5 subjects is too small to fit a statistical model. With such a low prevalence, the maximum of number covariates in the SHAPES subgroups should be 1.

Comparing the performance of SHAPES with other subgroup identification methods is challenging yet valuable. One of the challenge is that the subgroups identified in one study may not have the same enhanced treatment effect in another study. SHAPES is a promising method in controlling the Type I error and identifying true subgroups under alternative hypothesis simulations as stated in Prince's work[14]. Therefore, simulating data under the alternative hypothesis and comparing the result with other approaches like CVASD, SIDES or the Bayesian credible subgroup approach will be very meaningful. One of the challenges is that there is no well-designed R package(s) for these methods that can be used directly for comparison. Some software is designed based on other languages, for example, SIDESxl is designed to identify subgroups by applying SIDES method on Excel dataset[24]. Therefore, ready-to-use programs for these methods should be developed in the future. Another major challenge was evoked by long computer processing time, which largely limited the number of covariates used in SHAPES, making it difficult to apply SHAPES on a dataset with hundreds of covariates.

The correlation structure of the covariates in real data set is usually unknown. Since some of the covariates are correlated based on biological rational while others are independent, we explored the overall Type I error in two scenarios: independent and correlated. When two or more of the covariates are correlated, collinearity and multicollinearity can give rise to Type I errors if not accounted for. Some of the overall Type I error rates of SHAPES are outside of the 95% confidence intervals, if covariates are correlated. Moreover, the Type I error rates decrease along with stronger

correlation among covariates. Even though, the rates are still above 0.08 when the correlation is 0.6.

There are some additional comments for the subgroup analysis regarding pre-specifying subgroup analysis in the study design and only test for the primary. For those who are interested in SHAPES' application in his/her own study, the following guidelines / considerations may be helpful:

1) If the number of covariates considered for subgroup definition and the maximum number of covariates in the SHAPES subgroups are similar to our simulation study settings, applying SHAPES should maintain the overall type I error fairly well unless the outcome is continuous and more than two covariates are highly correlated. Selecting a small subset of the covariates based on the scientific background when large number of covariates are under consideration is efficient in the subgroup search.
2) Specify the maximum number of covariates in SHAPES subgroup (L). Would a two-factor definition be sufficient (i.e. old male), or it requires more factors to define the scenario? For instance, scenario with postmenopausal women who are estrogen receptor (ER)-positive and human epidermal growth factor receptor 2 (HER2)-negative, as we described in 1.1.2, requires five factors. In addition, the choice of L should also depend on the overall sample size and the prevalence of covariates. A higher L means smaller subgroups will be tested in the analyses.
3) Consider the correlation structure among covariates. SHAPES is robust when the correlations between covariates are small, however, control of overall Type I error is not as robust when the correlations are strong.

In conclusion, SHAPES mostly performs well in controlling the overall Type I errors under the simulated scenarios when it is employed to identify a subgroup in clinical trials. The overall Type I error rates of SHAPES in simulations with binary and continuous outcomes are comparable. When the covariates are independent, the overall Type I error rates are around the assigned overall alpha level of 0.1, while when many covariates are strongly correlated, the overall Type I error rates may be outside of the 95% confidence interval. For future research, it is important to extend the evaluation of performance of SHAPES from a maximum of 50 covariates to thousands. Results of such evaluation are critical for scenarios where thousands of genes are to be tested in subgroup analyses.

BIBLIOGRAPHY

1. Bahcall O. Precision medicine. Nature. Nature Publishing Group, a division of Macmillan Publishers Limited. 2015 Oct; 526:335–335.

2. Lu YF, Goldstein DB, Angrist M, Cavalleri G. Personalized Medicine and Human Genetic Diversity. Cold Spring Harbor Perspectives in Medicine. 2014 Sep;4:a008581–a008581.

3. Schwaederle M, Zhao M, Lee JJ, Eggermont AM, Schilsky RL, Mendelsohn J, Lazar V, Kurzrock R. Impact of Precision Medicine in Diverse Cancers: A Meta-Analysis of Phase II Clinical Trials. JCO. 2015 Nov;33:3817–3825.

4. Brookes ST, Whitley E, Peters TJ, Mulheran PA, Egger M, Davey Smith G. Subgroup analyses in randomised controlled trials: quantifying the risks of false-positives and false-negatives. Health Technology Assessment. 2001;5.

5. Ruberg SJ, Lei Chen, Yanping Wang. The mean does not mean as much anymore: finding sub-groups for tailored therapeutics. Clinical Trials. 2010 Oct;7:574–583.

6. Pfizer Inc. IBRANCE® (palbociclib) Receives FDA Regular Approval and Expanded Indication for First-Line HR+, HER2- Metastatic Breast Cancer. httppress.pfizer.compress-releaseibrance-palbociclib-receives-fda-regular-approval-and-expanded-indication-first-line-h. 2017 Aug;:1–3.

7. Broderick JM. Ibrance Gets Full FDA Approval for Breast Cancer. httpwww.curetoday.comarticlesibrance-gets-full-fda-approval-for-breast-cancer. 2017. pp. 1–3.

8. NIH. FDA Approval for Pemetrexed Disodium - National Cancer Institute. httpswww.cancer.govabout-cancertreatmentdrugsfda-pemetrexed-disodium. 2017. pp. 1–4.

9. Hernández AV, Boersma E, Murray GD, Habbema JDF, Steyerberg EW. Subgroup analyses in therapeutic cardiovascular clinical trials: Are most of them misleading? American Heart Journal. 2006 Feb;151:257–264.

10. Rothwell PM. Subgroup analysis in randomised controlled trials: importance, indications, and interpretation. The Lancet. 2005 Jan;365:176–186.

11. Brookes ST, Whitley E, Peters TJ, Mulheran PA, Egger M, Smith GD. Subgroup analyses in randomised controlled trials: quantifying the risks of false-positives and false-negatives. Health Technology Assessment. 2001 Sep;:1–64.

12. Assmann SF, Pocock SJ, Enos LE, Kasten LE. Subgroup analysis and other (mis)uses of baseline data in clinical trials. The Lancet. 2000 Mar;355:1064–1069.

13. FDA. Enrichment Strategies for Clinical Trials to Support Approval of Human Drugs and Biological Products. httpwww.fda.govDrugsGuidanceComplianceRegulatoryInformationGuidancesdefault.htm . 2012 Nov;:1–42.

14. Prince D. Searching for Predictive Subgroups. PhD Dissertation, Department of Biostatistics, University of Washington. 2015 Apr.

15. Jenkins M, Stone A, Jennison C. An adaptive seamless phase II/III design for oncology trials with subpopulation selection using correlated survival endpoints†. http://www.crcpress.com. 2010 Dec;10:347–356.

16. KE P, D-G C. Clinical trial methodology. http://www.crcpress.com. 2010.

17. Freidlin B, Jiang W, Simon R. The Cross-Validated Adaptive Signature Design. Clinical Cancer Research. 2010 Jan;16:691–698.

18. Freidlin B. Adaptive Signature Design: An Adaptive Clinical Trial Design for Generating and Prospectively Testing A Gene Expression Signature for Sensitive Patients. Clinical Cancer Research. 2005 Nov;11:7872–7878.

19. Lipkovich I, Dmitrienko A. Strategies for identifying predictive biomarkers and subgroups with enhanced treatment effect in clinical trials using SIDES. Journal of Biopharmaceutical Statistics. 2014 Jan;24:1–33.

20. Schnell PM, Tang Q, Offen WW, Carlin BP. A Bayesian credible subgroups approach to identifying patient subgroups with positive treatment effects. Biom. 2016 May;72:1026–1036.

21. Li L, Guennel T, Marshall S, Cheung LW-K. A multi-marker molecular signature approach for treatment-specific subgroup identification with survival outcomes. The Pharmacogenomics Journal. Nature Publishing Group; 2014 Mar;14:439–445.

22. Blakesley R, Mazumdar S, Dew MA, Houck P, Tang G, Reynolds C, Butters M. Comparisons of Methods for Multiple Hypothesis Testing in Neuropsychological Research. Neuropsychology. 2009;:1–18.

23. Pocock SJ, Stuar, Assmann SE, Enos LE, Kasten LE. Subgroup analysis, covariate adjustment and baseline comparisons in clinical trial reporting: current practice and problems. Pocock SJ, Assmann SE, Enos LE, Kasten LE, editors. Statist. Med. 2002 Sep;:1–14.

24. SIDESxl Installation and User's Guide. httpwww.biopharmnet.comdocinstallation.pdf. 2014 Aug;:1–16.

APPENDIX

The simulations of SHAPES are performed in R 3.3.2. In this chapter, we will introduce the simulation strategies (A.1) and the R programming (A.2-A.6). The running time for each simulation is very long when either L or k is large and running the program on a single processor, for instance, running k=50 and L=2 would require around 459 days. It encouraged me to run parallel jobs on clusters. The 5000 replicated simulations for each combination (i.e. k=50, L=2, α_F=0.02, n=500) was split into 50 small jobs with 100 replicates in each job. Then all the simulated results will be merged and analyzed. This way significantly reduced the time of running programs to 7 days and this way is easy for debugging.

Some occasional R function errors may happen when L is large, for instance, errors in the linear combination process following the interaction models. The probabilities of errors in the interactions models is higher than those in the stratification models. One of the reason is that when the treatment variable and the SHAPES indicator variable are highly correlated with each other, the interaction model cannot estimate the interaction coefficient, which causes errors when running *glht* function in calculating the linear combination p values. After checking the returned errors and figuring out it caused problems the *glht* step, we limited the p-values estimating situations and skip the linear combination step. The second type of modeling errors is caused by the small sample size of SHAPES subgroup. Therefore, when the sample size is below 6, we do not test the treatment effect for that subgroup.

SHAPES simulation steps:

1. Create a folder "*Thesis*" and ensure that the below files are in the directory.
    - *R_function.R* (A.2) – Contains all the R functions that we used.
    - *R_run.R* (A.3) – Run the simulations and save intermediate results.
    - *Run_Sheet.xlsx* (A.5) – I organized the simulation combinations here. The first column "row" is an index for each combination, which will be used as an argument in the shell scripts.
    - *bash.sh* (A.6) – Batch file for submitting the jobs on the clusters. There are two arguments: "row" is the row number in the Run_Sheet.xlsx, and "cluster" is the number of jobs.
    - *R_result_binary.R* and *R_result_continuous.R* (A.4) – When the simulations are done, run this program to merge all the outputs together. It will create the critical values use the null simulation datasets, and then calculate the overall Type I error by using the rep simulation datasets. It will generate the figures in the thesis.
    - Create a folder "*R_plots*" to save the figures.
2. Once all the above files are in place, connect to cluster (i.e. UW bayes), go to the directory of "*Thesis*". Then submit command as following:

```
qsub -v row=66 -v cluster=1 -q  normal.q -cwd bash.sh

qsub -v row=66 -v cluster=2 -q  normal.q -cwd bash.sh

qsub -v row=66 -v cluster=3 -q  normal.q -cwd bash.sh
```

The "qsub" process is used for job submission on "bayes". Check the cluster you will use for the specific job submission code. The arguments "row" and "cluster" will be passed into R code through the shell scripts "bash.sh", where "row" is the row number in the

Run_Sheet.xlsx and "cluster" is just an assigned number for the output results name. It would be efficient to write loops for running a list of jobs in a new shell scripts and then submit only one job on cluster. Specify "-v" before every argument/variable.

3. Check the R output for status. They are named as "*R_row_cluster.Rout*" (i.e. *R_66_1.Rout*) in the same folder.
4. Check the results. The result will be saved in a new folder under "*Thesis*" and the name is shown in the *Run_Sheet.xlsx*.
5. When the simulations are done or some of them are done, run the *R_result_binary.R* or *R_result_continuous.R* to check the final result.
6. Check errors, save results and update the coding.

A.1 Simulation strategies

Step 1 Generate covariates (binary, independent or correlated, prevalence=0.5)

Step 2 Generate treatment variable (binary, 1:1)

Step 3 Generate outcome under the null hypothesis

-Binary (baseline effect=0.3, same proportions)

-Continuous (SD=1, same means)

Step 4 Specify sample size, number of covariates (k), the maximum number of covariates of defining a subgroup (L), treatment effect for full population (=0), treatment effect for subgroup (=0)

Step 5 Generate simulated data

Step 6 Find out all qualified SHAPES subgroups combinations

Step 7 Fit both stratified models and interaction models for testing treatment effect in full population and subgroups

-Logistic regression for binary outcome

-Linear regression for continuous outcome

Step 8 Keep one-sided p value based on both Wald statistic

Step 9 Figure out the subgroup or the full population with the minimum standardized p value among all the tests

Step 10 Figure out the critical values as cut-off points by repeating the Steps 1-9 under null hypothesis for 5,000-15,000 times.

Step 11 Repeat the Step 1-9 for another 5,000-10,000 times under the null hypothesis and then use the critical value to cut off the minimum standardized p values. If the p value is below the critical value, then it is false positive. Summarize the Type I error and compare among different situations.

A.2 R_function.R

It contains all the functions about generating data, running SHAPES, obtaining critical values, and calculating Type I error.

```
######################################################################
# File:       R_function.R
# Purpose:    All functions for SHAPES simulations
# Written by: Lei Wang
# Last Date:  8/30/2017
######################################################################

library(foreign)

# Install multcomp for linear combination of interaction models
if (!require("multcomp")) {
  install.packages("multcomp", repos="http://cran.us.r-project.org")
  library("multcomp")
}

# Install bindata for the function rmvbin - sim correlated binary data
if (!require("bindata")) {
  install.packages("bindata", repos="http://cran.rstudio.com/")
  library("bindata")
}

##
### Generate datasets, outcome could be either binary or continuous
##
gen_data <- function(n, k, prev, scene, base, allEffect, subEffect, binary, sd, kc,
corr){
  #kc - the number of covariates that are correlated (correlation=corr) 0 means all
covariates are independent
  #kc <= k  !
  #corr - the correlation between covariates

  # Create matrix of covariates (n*k) and all values equal 0
  X <- matrix(rep(0, n*k), ncol=k)

  # If not all the covariates are independent
  if (kc!=0) {

    # binary correlation matrix
    corrM <- matrix(rep(corr, kc*kc), ncol=kc)
    diag(corrM) <- rep(1, kc)
    # create binary correlated variables
    X[, 1:kc] <- rmvbin(n,margprob=rep(prev, kc), bincorr=corrM)

    # Indenpendent covariates
    # randomly select n*prev observations to 1
    # thus the covariates are distributed exactly at 100*prev% and they are independent
    if (k>kc) {
      for (xi in (kc+1):k){X[,xi][sample(1:n, round(n*prev,0) )] <- 1}
    }
```

```
  }else {
    # If all covariates are Independent
    if (k>kc) {
      for (xi in (kc+1):k){X[,xi][sample(1:n, round(n*prev,0) )] <- 1}
    }
  }

  # Name the columns of X by using X1 X2 …
  colnames ( X ) <- paste ('X', 1:k, sep='')

  X <- data.frame(X)

  # Create Treatment (trt) based on even n or odd n
  if( (n %% 2)==0){ Trt <- c(rep(1, n/2), rep(0, n/2))
  }else         { Trt <- c(1, rep(1, floor(n/2)), rep(0, floor(n/2)))  }

  ### Create subgroup truth (pred) according to scenerio
  # This part has no impact under the NULL hypothesis
  # scene=1  , 2  , 3  , 4   ... corresponding to X1, X1^X2, X1^X2^X3, X1^X2^X3^X4....
(SHAPES)
  # scene=     2.5, 3.5, 4.5 ... corresponding to     X1|X2, X1|X2|X3, X1|X2|X3|X4....
(SHAPES) no 1.5
  # scene=0.3, truth=X1^X2orX3        (not SHAPES)  only when k>=3
  # scene=0.4, truth=(X1^X2)or(X3^X4) (not SHAPES)  only when k>=4

  pred <- NULL
  truth_label=NULL  # add this for checking the truth subgroup's name

  if       (scene==1) {
      pred <- X[,1]     # define truth subgroup
      truth_label="X1"  # label of truth subgroup

  }else if (k>=floor(scene) & (scene-floor(scene))==0  ){  #and (truth subgroup is SHAPES)
      pred <- do.call(pmin, X[,1:floor(scene)])
      truth_label <- paste0(paste("X",c(1:floor(scene)), sep=''), collapse="_")

  }else if (k>=floor(scene) & (scene-floor(scene))==0.5){  #or (truth subgroup is SHAPES)
      pred <- do.call(pmax, X[,1:floor(scene)])
      truth_label <- paste0(paste("X",c(1:floor(scene)), sep=''), collapse="or")

  }else if (k>=3 & scene==0.3) {
      pred <- pmax(pmin(X[,1], X[,2]), X[,3]) # truth is X1^X2orX3 (not SHAPES)  only
when k>=3
      truth_label <- "X1_X2orX3"

  }else if (k>=4 & scene==0.4) {
      pred <-   pred <- pmax(pmin(X[,1], X[,2]), pmin(X[,3], X[,4]))   # truth is
(X1^X2)or(X3^X4) (not SHAPES)  only when k>=4
      truth_label <- "X1_X2orX3_X4"

  }else      { return(print('Issue! Check parameter settings')) }


  ### Create outcome Y (check the total effect is below 1)
```

```
  if (base+allEffect+subEffect<1) {
       # Generate binary outcome
        if (binary=="binary") {
          yProb <- base + I(allEffect * Trt)  + I(subEffect*(Trt*pred))
          y <- rbinom(n, 1, yProb)
       # Generate continuous outcome
        } else if (binary=="continuous") {
          y <- base + I(allEffect * Trt)  + I(subEffect*(Trt*pred)) + rnorm(n,0, sd)
        } else {return(print('Check outcome settings'))}

  } else {return(print('Check values in generating outcomes'))}

  # Mydata contains outcome, treatment group, all covariates and one predictor
  mydata <- data.frame(cbind(y,Trt, X, pred)) #ncol=1+1+k+1 , pred is the truth

  # save the data for check
  # write.csv(mydata, file = "1_gen_data.csv")

  return(list(X, Trt, y, pred, truth_label))
}

##
### this function is to calculate all the combinations of k covariates (a table)
##
gen_comb_k <- function(k){
  aa <- list ()
  for (ai in 1:k){ aa [[ai]] <- c(1 ,0)}
  bb <- expand.grid ( aa )
  com_k_table <- bb [k :1]
  colnames ( com_k_table ) <- paste ('X', 1:k, sep ='')
  com_k_table <- as.matrix(com_k_table)
  return(com_k_table)
}

##
### calculate the number of shapes subgroup for each k and L (including total and null)
##
n_shapes <- function(k, ell){
  tot <- 0
  for (i in 0:ell) {
    if (i==0) {subtot <- 2                                    #full and null
    } else if (i==1) {subtot <- choose(k, i) * (2^i)          #X1, X1^c
    } else if (i>1)  {subtot <- choose(k, i) * (2^i) * 2      # i=2: X1&X2     , X1|X2
    }                                                        # i=3: X1 &X2 &X3, X1 |X2 |X3
    tot <- tot+subtot
  }
  return(tot)
}


##
### Create the table of k and L (Table 1.1)
##
shapes_matrix_func <- function(k, ell){
  shapes_matrix <- matrix(rep(1, k*ell), nrow = k)
```

```
  colnames(shapes_matrix) <- paste("ell=",c(1:ell))
  rownames(shapes_matrix) <- paste("k=", c(1:k))
  for (i in 1:k){
    for (j in 1:ell){
      shapes_matrix[i,j] <- n_shapes(i,j)
    }
  }
  return (shapes_matrix)
}

# write.csv(shapes_matrix_func(500, 8), "0_shapes_matrix.csv")

############################################################
### This is the main function, it contains:
###   1. alphaVec
###   2. test_n <- n_shapes(k, ell)
###   3. create a RAW dataset (check detail of list 1 2 3 4
###       - rows - sample size
###       _ columns contain the y, Xs, indicator
###   4. create an OUT matrix for test results
###       - nrow=(n_shapes(k,ell)-1) - w/o None
###        - col="Subgroup_Name", "Depth", "alpha_level", "sample_size", "stratify_p",
"LRT_p", "b", "c" - b,c just for adding
###   5. result_vec <- c(min_p_stratify_vec,min_lrt_p_vec, sample_size_vec)
############################################################

shapes_func <- function (triallist,n, k, ell, alphaT, alphaF, binary, shapesP=NA) {
  # dataset (triallist is generated in do_one function)
  X <- triallist[[1]]
  trt <- triallist[[2]]
  y <- triallist[[3]]

  full <- cbind(X, trt, y)

  ##### alpha vector length=(ell+1), alphaT is split equally into each depth and then
split equally into each SHAPES subgroup
  # full population is separate
  # alphaVec  <I defined this in main function, still define it here in the same way>
  alphaVec <- rep(NA, ell+1)
  alphaVec[1] <- alphaF
  alphaVec[2:(ell+1)] <- rep((alphaT-alphaF)/ell, ell) #normal alpha vector

  # create a table with all possible combinations/ same as in {scenerio} (not relate to
data, only relates to k)

  # total number of shapes (should use test_n - 2 because full and null is not in)
  # or use test_n-1 because only remove null, but keep full in the final output
  test_n <- n_shapes(k, ell)


  ###################
  ## Create Raw    ##
  ###################
  #### these two lists is preparing for choose subgroup
```

```
  list1 <- list() # choose(k,di) then repeat -- the combination of covariates (1 2
represent X1 X2)
  list2 <- list() # matrix of combinations (2^di) then repeat -- the values of covariates
(1:X=1, 0:X=0)
  # no full(depth=0) in these lists. Only half of the subgroups are here, we'll calculate
the complementary later
  # when depth>1, X1&X2 is the complementary of X1c|X2c
  for (di in 1:ell) {
    if (di==1) { #depth=1 is very special, only keep k subgroups here, the other k will
be later
      list1[[1]] <- t(combn(k, di))
      list2[[1]] <- matrix( rep(1,k), nrow= k )
    }else {
      list1[[di]] <- matrix( rep(t(combn(k, di)),each=2^di), nrow= (choose(k, di)*2^di))
      # choose(k, di) = nrow(t(combn(k, 1)))
list2[[di]] <- apply(gen_comb_k(di), 2, rep, choose(k, di))
      # these two lists have the same dimension. If cbind them together, all the rows
are unique
    }
  }

  # here we're going to generate the subgroups
  # two halfs: X1&X2 + X1|X2 (complementaries)
  raw <- matrix (rep(0, n*(n_shapes(k,ell)-2)), nrow=n)


  ##### Create the subgroup indicators/variables
  for (si in 1:ell){ # count from 1 to max(depth)

    # calculate number of shapes in each depth (only half)
    if (si==1){ n_depth<-k
    }else{  n_depth <- choose(k,si)*2^si }

    for (sdi in 1:n_depth){
      raw[,sdi+n_shapes(k,                                                              si-1)-
2][which(X[,list1[[si]][sdi,]]==matrix(rep(list2[[si]][sdi,], n), ncol=si),arr.ind=T)]
=1
      #complementary group
      raw[,sdi+n_shapes(k, si-1)-2+n_depth] <- (1-raw[,sdi+n_shapes(k, si-1)-2])^2
    }

  }
  #

  #### Add full as the first column
  raw <- data.frame(cbind(rep(1, n), raw, trt, y))

  # save raw for check
  # write.csv (raw, '3_list1.csv')

  #### Column names of raw  (add full as the 1st column)
  # X1.1_X2.0_X3.1    - means (X1=1 & X2=0 & X3=1)
  # X1.1_X2.0_X3.1-C  - the complementary, means (X1=0 or X2=1 or X3=0)
  list3 <- list2   #used below, could replace list3 by list2
  list4 <- list2   #just use the frame, the values will be replaced
```

```
  ##### get the colmames for raw

  # depth=1, prespecify the column names
  colname_raw <- c("full", paste("X",list1[[1]],".1", sep=""), paste("X",list1[[1]],
".0", sep="") ) # depth=1, ell=1

  # depth>1, ell>1
  if (ell>1){
    for (ci in 2:ell){
      for (cci in 1:ci){
        #  replace(list3[[ci]][,cci], which(list2[[ci]][,cci]==0), "c")
        list4[[ci]][,cci]<-paste("X",paste(list1[[ci]][,cci],list3[[ci]][,cci],
sep='.' ),sep="")
      }
      colname_raw <- c(colname_raw,
                       apply( list4[[ci]][,1:ci], 1 , paste , collapse = "_" ),
                       paste(apply(  list4[[ci]][,1:ci],  1  ,  paste  ,  collapse  =
"_" ),"C",sep="-"))
    }
  }
  colnames(raw) <- c(colname_raw, "trt", "y")

  #### save list for check
  # lapply(list1, function(x) write.table( data.frame(x), '3_list1.csv'  , append= T,
sep=','  ))
  # lapply(list2, function(x) write.table( data.frame(x), '3_list2.csv'  , append= T,
sep=','  ))
  # lapply(list4, function(x) write.table( data.frame(x), '3_list4.csv'  , append= T,
sep=','  ))



  #### depth vector
  depth_vec <- c(0,rep(1,2*k)) #full and depth=1
  if (ell>1){
    for (dvi in 2:ell){
      depth_vec <- c(depth_vec, rep(dvi, 2*(choose(k, dvi)*2^dvi)))
    }
  }

  ###### result matrix
  out <- matrix(NA, nrow=(n_shapes(k,ell)-1), ncol=8)
  out <- as.data.frame(out)
  colnames(out)   <-   c("Subgroup_Name",   "Depth",   "alpha_level",   "sample_size",
"stratify_p", "LRT_p", "interaction_p", "c" )
  # the alpha_level is useless in this step, may delete


  ### note: add columns: waldp, likeliratiop, scorep, samplesize, ...
  out[,'Subgroup_Name'] <- colname_raw
  out[,'Depth'] <- depth_vec

  depth_n <- as.data.frame(table(depth_vec))[,2]
```

```
  out[,'alpha_level'] <- rep(alphaVec, depth_n)


  for (ti in 1:(n_shapes(k,ell)-1)) { #n_shapes(k,ell)-1 = full + all shapes subgroups

    out[ti,'sample_size'] <- sum(raw[,ti])

    # if sample size is less or equal than 5, then don't test it
    if (sum(raw[,ti])>5) {

      m  <- NULL
      m1 <- NULL
      m2 <- NULL
      m3 <- NULL
      ############
      ## tests  ##
      ############
      if (binary=="binary") {

        ####################################
        ## stratification models for binary
        ####################################
        m  <- glm(y~ trt,data=raw[which(raw[,ti]==1),], family ='binomial') # logistic
regression for binary
        # test coefficient
        m1 <- summary (m)$ coef

        # check;
        # write.csv(m1, file = "5_model_stratify_bin.csv")


        # one-sided p value based on Wald (only those have positive treatment effect!!
great!!)
        if (dim(m1)[1]==2){ #when the sample size is very small, all y=trt=0
          out[ti,'stratify_p'] <- 1-pnorm(m1[2,1]/m1[2,2]) # one-sided p value
        }

        ####################################
        ## interaction models   for  binary
        ####################################
        Xshapes <- raw[,ti] # this is the new covariate that define the SHAPES subgroup
        m2  <- glm(y~ trt*Xshapes, family ='binomial') # logistic regression for binary
with interaction terms
        m3 <- summary(m2)$coef

        # check;
        # write.csv(summary(m3), file = "5_model_interaction_bin.csv")


        if (ti==1){  #because full population has no interaction model
                out[ti,'interaction_p'] <- 1-pnorm(m3[2,1]/m3[2,2]) # one-sided p value
        }else{  # 6/4/2017 linear combination of two terms, one-sided
          if (dim(m3)[1]==4){ #sometimes the coef of interaction term is NA 8/25
            out[ti,'interaction_p'] <-summary(glht(m2, linfct = c("trt + trt:Xshapes >=
0")))[[9]]$pvalues[1]
```

```
          }
        }

      } else if (binary=="continuous") {

        ########################################
        ## stratification models for continuous
        ########################################
        #m <- t.test(y~ trt,data=raw[which(raw[,ti]==1),], alternative="greater")
        m    <-  lm(y~  trt,data=raw[which(raw[,ti]==1),])    #  linear  regression  for
continuous
        m1 <- summary(m)$coef

        # check;
        # write.csv(summary(m1), file = "5_model_stratify_cts.csv")
        if (dim(m1)[1]==2){ #when the sample size is very small, all y=trt=0
          out[ti,'stratify_p'] <- 1-pt(m1[2,3], m$df) # one-sided p value t-distribution
        }

        ########################################
        ## interaction models for    continuous
        ########################################

        Xshapes <- raw[,ti] # this is the new covariate that define the SHAPES subgroup
        m2 <- lm(y~ trt*Xshapes)
        m3 <- summary(m2)$coef

        # check;
        # write.csv(summary(m2), file = "5_model_interaction_cts.csv")
        if (ti==1) { # full population is different
                out[ti,'interaction_p'] <- 1-pt(m3[2,3], m2$df) # one-sided p value
         }else{
           #6/4/2017 linear combination of two terms  one-sided
           #out[ti,'interaction_p'] <- 1-pt(m3[4,3], m2$df) # one-sided p value
           if (dim(m3)[1]==4){
            out[ti,'interaction_p'] <-summary(glht(m2, linfct = c("trt + trt:Xshapes  >=
0")))[[9]]$pvalues[1]
           }
        }
      }


    } #else {out[ti,'stratify_p'] <- NA
      #      out[ti,'interaction_p'] <- NA}

  }

  # write.csv(out, file = "5_out.csv")

  # a vector of min p values for each depth
  min_p_stratify_vec <- NULL
  min_p_interact_vec <- NULL
  # an minimum sample size for each depth
  sample_size_vec <- NULL
```

```
  for (mpi in 0:ell) {
    min_p_stratify_vec<-
c(min_p_stratify_vec,min(out[which(out[,'Depth']==mpi),'stratify_p']   , na.rm=T) )
    min_p_interact_vec<-
c(min_p_interact_vec,min(out[which(out[,'Depth']==mpi),'interaction_p'], na.rm=T))
    #min_lrt_p_vec    <-    c(min_lrt_p_vec,min(out[which(out[,'Depth']==mpi),'LRT_p'],
na.rm=T) )
    sample_size_vec                        <-                        c(sample_size_vec,
min(out[which(out[,'Depth']==mpi),'sample_size'], na.rm=T))
  }

  #result_vec <- c(min_p_stratify_vec,min_lrt_p_vec, sample_size_vec)
  result_vec <- c(min_p_stratify_vec,  min_p_interact_vec,             sample_size_vec)

  names(result_vec) <- c(paste("stratify_p_depth=", c(0:ell), sep=''),
                         paste("interact_p_depth=", c(0:ell), sep=''),
                         #paste("LRT_p_depth=", c(0:ell), sep=''),
                         paste("Min_Size_depth=", c(0:ell), sep='') )
  #return( list(min_p_stratify_vec,min_lrt_p_vec, sample_size_vec) )
  return(result_vec)
}


#################################################################
# Function for obtaining critical values (cut-off points vector)
# Different with David's. The first cut-off point
#################################################################

shapes_cut_off <- function ( data , alphaVec_cum, ell ){
  reps <- nrow ( data )    #  =nullReps   shapesP=t(nullD)
  cutoffSH <- rep(NA, (ell+1))

  shapesP <- data[,1:(ell+1)]  # full population, based on Wald test

  leftOver <- reps - floor ( quantile (1: reps , alphaVec_cum )) # 980      953      926
900  (alphaT=0.1, alphaF=0.2, ell=3)
  #remaining <- subset ( shapesP , shapesP [ ,1] >= alphaVec_cum [1])  #dim(remaining)
[1] 983   4
  remaining <- shapesP #7/21/2017 use the quantile for the first cutoffSH instead of
alphaF
  for (i in 1: (ell+1) ){
    ranker <- rank (1- remaining [,i])
    subTab <- remaining [,i][ ranker > leftOver [i]]
    cutoffSH [i] <- max ( subTab )
    remaining <- subset ( remaining , remaining [,i]> cutoffSH [i])
  }
  return ( cutoffSH )  #0.020000000 0.009478289 0.004750190 0.007591756
}



##
### Run one simulation
##
```

```
do_one <- function (n, k, prev, scene, base, allEffect, subEffect, ell, alphaT,
alphaF,binary, sd, kc, corr ){

  triallist <- gen_data ( n, k, prev, scene, base, allEffect, subEffect, binary, sd, kc,
corr)

  pResult <- shapes_func(triallist, n, k, ell, alphaT,alphaF, binary, shapesP=NA)

  return ( pResult)
}


#################################################################
# Function to calculate "prefer full", "min p", "prefer subgroup"
#
#################################################################
threeP <- function (pdata, alphaCut, ell,nNull, name1, name2) {

  #pdata - p value vector for full population +
  #      - p value matrix for subgroup tests (for each depth)
  #alphaCut - alpha level for cut off the full population tests +
  #         - alpha cut off vector for each depth of subgroup
  #name1 - "s" for stratification
  #        "i" for interaction
  #name2 - alphaVec, critical, naive


  aa <- pdata[,1]/alphaCut[1] #p/critical for full, use the critical value instead of
alphaF
  if (ell==1){bb <- pdata[,2]/alphaCut[2]
  }else{bb  <-  apply(sweep(pdata[,  2:(ell+1)],  2,  alphaCut[2:(ell+1)],  '/'  )  ,1,
min,na.rm=TRUE) # subgroup,
  }
  cc <- pmin(aa, bb,na.rm=TRUE) # minimum between full and subgroup (p/critical <1 means
significance )

      # prefer full (choose aa when aa and bb both <1)
      ff <- (aa<1)*aa + (aa>=1)*bb
      # if (aa<1) {ff<-aa} else {ff<-bb}

      # prefer subgroup
      ss <- (bb<1)*bb + (bb>=1)*aa


  # 7/21/2017 only keep one, enough, because they are all equal
  type1error_wald <- c( # mean( ff<1 , na.rm = T), # prefer full (always select full
population)
                        mean( cc<1 , na.rm = T)    # minimum p value
                        # ,mean( ss<1 , na.rm = T) # prefer subgroup (always select
subgroup population)
                        # ,
                        # # the percentage of combinations (f s both significant ,then
prefer full or prefer sub)
                        # mean((aa<1)*(bb<1)*((aa-bb)<0), na.rm=T),  #f and s both <1,
and f<s
```

```
                        # mean((aa<1)*(bb<1)*((aa-bb)>0), na.rm=T),  #f and s both <1,
and s<f
                        # mean((aa<1)*(bb>=1), na.rm=T),             #full <1 only
                        # mean((bb<1)*(aa>=1), na.rm=T)            # sub <1 only
                        # #mean((aa>=1)*(bb>=1), na.rm=T)          #neither <1
  )


  names(type1error_wald) <- paste(name1, "_Prefer_Full_", name2, sep="")

  return(type1error_wald)
}




##
### Gift fuction. If you don't want to run parallel on clusters. Then just run this
function is ok.
##
main_func_critical <- function(n, prev, k, ell, scene, base, allEffect, subEffect, alphaT,
alphaF, nSim, nNull, binary, sd, kc, corr){
  #                                  scene is used for justify the truth

  # alphaVec
  alphaVec <- rep(NA, ell+1)
  alphaVec[1] <- alphaF
  alphaVec[2:(ell+1)] <- rep((alphaT-alphaF)/ell, ell)
  alphaVec_cum <- cumsum(alphaVec) #cumulative

  # For depth=2, ell=3, alpha_vec[3] = ((alphaT-alphaF)/3)/48
  alpha_vec_naive <- alphaF
  for (afi in 1:ell){
    if (afi==1){alpha_vec_naive <- c(alpha_vec_naive, (alphaT-alphaF)/(ell*8))
    }else{alpha_vec_naive   <-c(   alpha_vec_naive,   (alphaT-alphaF)/(ell*2*choose(k,
afi)*2^afi) )}
  }

  # nullD returns p value/critical value matrix : (ell+1)*nullReps
  # nullD <- replicate ( nNull ,
  #                   shapes_func(gen_data ( n, k, pVec, scene, base, allEffect,binary,
subEffect), ell, alphaT, shapesP=NA)
  #                     , simplify = " array ")
  # note: kc is always 0 in the Null simulation
  nullD <- replicate ( nNull , do_one(n, k, prev, scene, base, allEffect, subEffect,
ell, alphaT , alphaF, binary, sd, 0, corr), simplify = "array")
  data <- t(nullD)

  # save the data for check
  # write.csv(data, file = "7_t_nullD_main_func_critical1.csv")

  # get the critical value
  pSH_stratify <- NULL
  pSH_interact <- NULL
```

```
  pSH_stratify <- shapes_cut_off(data[, 1:(ell+1)],        alphaVec_cum,ell )
  pSH_interact <- shapes_cut_off(data[, (ell+2):(2*ell+2)],alphaVec_cum,ell )

  # repeat again for 5000 times ---- nSim
  nullD <- replicate ( nSim , do_one(n, k, prev, scene, base, allEffect, subEffect, ell,
alphaT , alphaF, binary, sd, kc, corr), simplify = "array")
  data <- t(nullD)

  # save the data for check
  # write.csv(data, file = "7_t_nullD_main_func_critical2.csv")

  ###
  ### Decision results for 3 utilities based on Wald


#############################################################################################
############################
  # Use alphaVec - stratify
  type1error_stratify_alphaVec <- threeP (data[, 1:(ell+1)], alphaVec, ell,nNull,"s",
"alphaVec")

  # Use critical value - stratify
  type1error_stratify_critical <- threeP (data[, 1:(ell+1)], pSH_stratify,ell,nNull, "s",
"critical")

  # Use Bonferroni - stratify
  type1error_stratify_naive <- threeP (data[, 1:(ell+1)], alpha_vec_naive,ell,nNull, "s",
"naive")



#############################################################################################
############################
  # Use alphaVec (too loose) - interaction
  type1error_interact_alphaVec   <-   threeP   (data[,   (ell+2):(2*ell+2)],   alphaVec,
ell,nNull,"i", "alphaVec")

  # Use critical value - interaction
  type1error_interact_critical      <-      threeP      (data[,      (ell+2):(2*ell+2)],
pSH_interact,ell,nNull, "i", "critical") # use another pSH

  # Use Naive - interaction
  type1error_interact_naive <- threeP (data[, (ell+2):(2*ell+2)], alpha_vec_naive,ell,
nNull,"i", "naive")


  ##### ADD: specify the significant subgroups' name under ALternative
  label <- c(n, prev, k, ell, kc, corr, scene, base, allEffect, subEffect, alphaT,
alphaF, nSim, nNull, binary)
  names(label) <- c("Sample Size", "Prevalence", "k", "ell","Correlated covariates",
"Correlation","scene", "Base effect", "allEffect", "subEffect", "alphaT", "alphaF",
"nSim", "nNull", "Outcome")

  names(alphaVec)        <- paste("alphaVec_d=",        c(0:ell), sep='')
  names(pSH_stratify)    <- paste("pSH_stratify_d=",    c(0:ell), sep='')
```

```
  names(pSH_interact)    <- paste("pSH_interact_d=",    c(0:ell), sep='')
  names(alpha_vec_naive) <- paste("alpha_vec_naive_d=", c(0:ell), sep='')

  return (c (label,

             type1error_stratify_alphaVec,
             type1error_stratify_critical,
             type1error_stratify_naive,

             type1error_interact_alphaVec,
             type1error_interact_critical,
             type1error_interact_naive
             ,

             alphaVec,
             pSH_stratify,
             pSH_interact,
             alpha_vec_naive
  ))
}



#########################################################
# Use this function to combine the output datasets
# and then created the Type I error
# It will be used in the R_result_binary.R
# or R_result_continuous.R
#########################################################

get_result <- function ( folderdir){

  #source("R_function.R")
  file_names <- dir(folderdir) #where you have your files
  #setwd(folderdir)

  #your_data_frame <- do.call(rbind,lapply(file_names,read.csv))

  # combine all NULL results (will be used to generating critical values)
  nulldata <- do.call(rbind, lapply(paste0(folderdir,"/", file_names[substr(file_names,
1, 3)=="Nul"]), read.csv))[,-1]
  # combine all rep results (check the type I error)
  repdata  <- do.call(rbind, lapply(paste0(folderdir,"/", file_names[substr(file_names,
1, 3)=="Rep"]), read.csv))[,-1]

  #get the information of the simulation
  info <- unlist(strsplit(folderdir, "[_]"))
  info
  # these will be used in generating the alpha vector
  n <- as.numeric(info[2])
  k <- as.numeric(info[4])
  ell <- as.numeric(info[5])
  alphaT <- as.numeric(info[6])
  alphaF <- as.numeric(info[7])
```

```
  # these will be used in the final results (label name)
  binary <- info[1]
  prev <- as.numeric(info[3])
  kc <- as.numeric(info[8])
  corr <- as.numeric(info[9])
  #scene <-  no need to specify it in this analysis, it will be used in the alternative

  base    <- 0.3   # only works in binary
  allEffect <- 0   # under null hypothesis always
  subEffect <- 0   # under null hypothesis always
  sd      <- 1     # only works in continuous

  nNull <- dim(nulldata)[1]
  nSim <- dim(repdata)[1]


  ####### Alpha Vectors ########
  # alphaVec
  alphaVec <- rep(NA, ell+1)
  alphaVec[1] <- alphaF
  alphaVec[2:(ell+1)] <- rep((alphaT-alphaF)/ell, ell)
  alphaVec_cum <- cumsum(alphaVec) #cumulative

  # For depth=2, ell=3, alpha_vec[3] = ((alphaT-alphaF)/3)/48
  alpha_vec_naive <- alphaF
  for (afi in 1:ell){
    if (afi==1){alpha_vec_naive <- c(alpha_vec_naive, (alphaT-alphaF)/(ell*8))
    }else{alpha_vec_naive   <-c(   alpha_vec_naive,   (alphaT-alphaF)/(ell*2*choose(k,
afi)*2^afi) )}
  }


  ##### get the critical value by using NULL dataset
  pSH_stratify <- NULL
  pSH_interact <- NULL
  pSH_stratify <- shapes_cut_off(nulldata[, 1:(ell+1)],        alphaVec_cum,ell )
  pSH_interact <- shapes_cut_off(nulldata[, (ell+2):(2*ell+2)],alphaVec_cum,ell )

  alphaVec
  pSH_stratify
  pSH_interact


################################################################################
############################
  # Use alphaVec (too loose) - stratify
  type1error_stratify_alphaVec <- threeP (repdata[, 1:(ell+1)], alphaVec, ell,nNull,"s",
"alphaVec")

  # Use critical value - stratify
  type1error_stratify_critical <- threeP (repdata[, 1:(ell+1)], pSH_stratify,ell,nNull,
"s", "critical")

  # Use Naive - stratify
```

```
  type1error_stratify_naive <- threeP (repdata[, 1:(ell+1)], alpha_vec_naive,ell,nNull,
"s", "naive")

  type1error_stratify_alphaVec
  type1error_stratify_critical
  type1error_stratify_naive


################################################################################
############################
  # Use alphaVec (too loose) - interaction
  type1error_interact_alphaVec  <-  threeP  (repdata[,  (ell+2):(2*ell+2)],  alphaVec,
ell,nNull,"i", "alphaVec")

  # Use critical value - interaction
  type1error_interact_critical     <-     threeP     (repdata[,     (ell+2):(2*ell+2)],
pSH_interact,ell,nNull, "i", "critical") # use another pSH

  # Use Naive - interaction
  type1error_interact_naive      <-      threeP      (repdata[,      (ell+2):(2*ell+2)],
alpha_vec_naive,ell, nNull,"i", "naive")

  type1error_interact_alphaVec
  type1error_interact_critical
  type1error_interact_naive

  ##### ADD: specify the significant subgroups' name under ALternative
  label <- c(n, prev, k, ell, kc, corr, base, allEffect, subEffect, alphaT, alphaF,nNull,
nSim,  binary)
  names(label) <- c("Sample Size", "Prevalence", "k", "ell","Correlated covariates",
"Correlation",   "Base   effect",   "allEffect",   "subEffect",   "alphaT",   "alphaF",
"nNull","nRep",  "Outcome")

  names(alphaVec)        <- paste("alphaVec_d=",        c(0:ell), sep='')
  names(pSH_stratify)    <- paste("pSH_stratify_d=",    c(0:ell), sep='')
  names(pSH_interact)    <- paste("pSH_interact_d=",    c(0:ell), sep='')
  names(alpha_vec_naive) <- paste("alpha_vec_naive_d=", c(0:ell), sep='')

  return (c (label,

             type1error_stratify_alphaVec,
             type1error_stratify_critical,
             type1error_stratify_naive,

             type1error_interact_alphaVec,
             type1error_interact_critical,
             type1error_interact_naive
             # ,
             #
             # alphaVec,
             # pSH_stratify,
             # pSH_interact,
             # alpha_vec_naive
  ))
}
```

A.3 R_run.R

```
#############################################################################
# File:       R_run.R
# Purpose:    Run R
#   I found this way is the best way of running parallel on R cluster.
#   (foreach, doSNOW, doRNG don't work well)
# Summary: 1. run the simulations seperately.
#             Ex. if you need to run 50,000, then run 10,000 for 5 times,
#             then combine the output datasets
#          2. run nNull(independent) and nRep(=nSim, correlated maybe)
#             seperately, then created the critical alpha values in nNull
#             and genereate resutls in nRep
# Written by: Lei Wang
# Date Log:   Last modified 8/26/2017
#############################################################################

# The time helps to track the running status
Sys.time()

source("R_function.R")
time <- proc.time()

if (!require("xlsx")) {
  install.packages("xlsx", repos="http://cran.us.r-project.org")
  library("xlsx")
}

# Read in the EXCEL file with all the combinations
runsheet <- read.xlsx("Run_Sheet.xlsx", sheetName="Sheet1")

# args[1] is the row number in the Excel sheet, [2] is the cluster number
args = commandArgs(TRUE)
args

run.cluster <- as.numeric(args[[2]])

run.value <- runsheet[which(runsheet$runthisrow==as.numeric(args[[1]])), 2:12]
run.value

binary    <-  as.character(run.value[1]$binary)    #  binary  or  continuous  (transform
data.frame)

n       <- as.numeric(run.value[2])  # sample size for each simulation
prev    <- as.numeric(run.value[3])  # prevalence of covariates, all equal
k       <- as.numeric(run.value[4])  # number of covariates
ell       <- as.numeric(run.value[5])   # number of covariates been used to define a
subgroup

alphaT  <- as.numeric(run.value[6])  # total alpha level
alphaF  <- as.numeric(run.value[7])  # alpha level split to full population
```

```
kc      <- as.numeric(run.value[8])  # kc = 0 in the null simulations
corr    <- as.numeric(run.value[9])  # correlation between kc covariates


# run Null (nNull*run.cluster) and Rep (nSim*run.cluster) at the same time.  1000 or 500
nNull   <- as.numeric(run.value[10])
nSim    <- as.numeric(run.value[11])


# Usually, no need to change the parameters below
scene     <- 1     # useless
base      <- 0.3   # only works in binary
allEffect <- 0   # under null hypothesis always
subEffect <- 0   # under null hypothesis always
sd        <- 1       # only works in continuous


# Define alphaVec the same way as in R_function and R_run
alphaVec <- rep(NA, ell+1)
alphaVec[1] <- alphaF
alphaVec[2:(ell+1)] <- rep((alphaT-alphaF)/ell, ell)
alphaVec_cum <- cumsum(alphaVec) #cumulative

# For depth=2, ell=3, alpha_vec[3] = ((alphaT-alphaF)/3)/48
alpha_vec_naive <- alphaF
for (afi in 1:ell){
  if (afi==1){alpha_vec_naive <- c(alpha_vec_naive, (alphaT-alphaF)/(ell*8))
  }else{alpha_vec_naive    <-c(    alpha_vec_naive,    (alphaT-alphaF)/(ell*2*choose(k,
afi)*2^afi) )}
}


# OUTPUT dataset (it should be a group of datasets, then combine together)
# create FOLDER (check if exist, if not, then create a new one)
folder.name <- paste(sep="_", binary, n, prev, k, ell, alphaT, alphaF, kc, corr )
binary
folder.name
dir.create(file.path(folder.name), showWarnings = T)
setwd(file.path(folder.name))


# RUN the simulations and save the dataset as .csv files
# returns p value/critical value matrix
  # null simulations - 1st step
        set.seed(nNull*k*ell+alphaF*1000+10+run.cluster)
        # note: kc is always 0 in the Null simulation
        nullD <- replicate ( nNull , do_one(n, k, prev, scene, base, allEffect, subEffect,
ell, alphaT , alphaF, binary, sd, 0, corr), simplify = "array")
        dataNull <- t(nullD)
        out.name.null   <- paste(sep="_","Null", binary, n, prev, k, ell, alphaT, alphaF,
kc, corr,  nNull, run.cluster,".csv")

        #write.csv(dataNull, file = paste0("/",folder.name,"/", out.name.null))
        write.csv(dataNull, file =  out.name.null)
```

```
Sys.time()

  # rep  simulations - 2nd step
        set.seed(nSim*k*ell+alphaF*1000+20+run.cluster)
        repD <- replicate ( nSim , do_one(n, k, prev, scene, base, allEffect, subEffect,
ell, alphaT , alphaF, binary, sd, kc, corr), simplify = "array")
        dataRep <- t(repD)
        out.name.rep    <- paste(sep="_", "Rep",binary, n, prev, k, ell, alphaT, alphaF,
kc, corr,  nSim, run.cluster, ".csv")

        write.csv(dataRep, file =  out.name.rep)

Sys.time()

proc.time()-time
```

A.4 Running results and create tables as well as figures

```
############################################################
# File name:  R_result_binary.R"
# Purpose:    Plot results for binary endpoint
# Written by: Lei Wang
# Last date:  8/25/2017
############################################################
if (!require("RColorBrewer")) {
  install.packages("RColorBrewer")
  library(RColorBrewer)
}

# Plots will be saved under "P:/Thesis/R_plots"

# Location of simulation results
# The simulation results were saved under each folder
setwd("P:/Thesis")

# Extract all the files' name under the folder
file_names <- dir("P:/Thesis")

# List of folder name that contain the simulation results
dataAll <- file_names[substr(file_names, 1, 6) %in% c("binary", "contin")]

# R brewer color
colA <- brewer.pal(9, "Set1")
colB <- brewer.pal(8, "Dark2")
colP <- brewer.pal(12, "Paired")

#########################################################################
##### Figure - Type I error varies w/ n increases (ell fixed ) ######
#########################################################################

rows <- dataAll[c(which(substr(dataAll, 12, 31)=="0.5_4_2_0.1_0.02_0_0"),
                  which(substr(dataAll, 13, 32)=="0.5_4_2_0.1_0.02_0_0"))]

# run results here
# function get_result is designed in R_function.R
dat <- NULL
for (i in 1: length(rows)){
  dat <- rbind(dat,
               c(rows[i], get_result(rows[i])))
}
# save output for check
write.csv(dat, file = "Result_n.csv")

# import output
dat <- read.csv("Result_n.csv")
dat <- dat[order(dat$Sample.Size),] #sort dat by the variables we need

# these variables was in another function, but were moved out
# variable for x-axis
var=dat$Sample.Size
xname="Sample Size (n)"
yname="Overall Type I Error"
```

```
outfile="3.1.1-n"  # by sample size

    # save plot as pdf file
    pdf(paste0("P:\\Thesis\\R_plots\\", outfile,".pdf"),
        width=8, height=5.5)
    plot(c(min(var), max(var)), c(0, max(dat[, 17:22])),type="n" ,
         #    main="Figure 3.1  Type I error performance across different sample sizes",
         xlab=xname, ylab=yname)

    # Define X axis label
    axis(1, at = seq(min(var), max(var), by=100), labels = T)

    # horizontal line for alpha=0.1
    abline (h=0.1, col="Gray23", lwd=1.2, lty=3)

    ### plot the overall Type I error lines

    #critical - s
    points(x=var, y=dat$s_Prefer_Full_critical, type="o", col=colP[1],pch=19,lwd=1.2)
    #critical - i
    points(x=var,  y=dat$i_Prefer_Full_critical,  type="o",col=colP[2],  pch=2,  lty=2,
lwd=1.2)

    #alphaVec -s
    points(x=var, y=dat$s_Prefer_Full_alphaVec, type="o", col=colP[3], pch=19,lwd=1.2)
    #alphaVec -i
    points(x=var,  y=dat$i_Prefer_Full_alphaVec,  type="o",  col=colP[4],  pch=2,  lty=2,
lwd=1.2)

    #Bonfferoni -s
    points(x=var, y=dat$s_Prefer_Full_naive, type="o", col=colP[5],pch=19, lwd=1.2)
    points(x=var,   y=dat$i_Prefer_Full_naive,   type="o",   col=colP[6],pch=2,   lty=2,
lwd=1.2)

    # add legend
    legend("right", c(
      "Unadjusted alpha - Interaction","Unadjusted alpha - Stratification",
      "Critical values - Interaction","Critical values - Stratification",
      "Bonferroni adjusted - Interaction","Bonferroni adjusted - Stratification"),
      xpd=T, horiz=F,
      bty="n",  pch=c(2,19,2,19,2,19), lty=c(2,1,2,1,2,1), lwd=1.2,
      col=colP[c(4,3,2,1,6,5)],
      cex=1)

    # pdf file done
    dev.off()


######################################################################
##### Figure - Type I error varies w/ alphaF allocation        ######
######################################################################

rows <- dataAll[which(substr(dataAll, 1, 22)=="binary_500_0.5_4_2_0.1")]
rows <- rows[ -8] # remove alphaF=0.1
```

```
# run results here
dat <- NULL
for (i in 1: length(rows)){
  dat <- rbind(dat,
               c(rows[i], get_result(rows[i])))}
write.csv(dat, file = "Result_a.csv")


dat <- read.csv("Result_a.csv")
dat <- dat[order(dat$alphaF),] #sort dat by the variables we need

var=dat$alphaF
xname="Alpha Allocated to Full Population (alphaF)"
yname="Overall Type I Error"
outfile="3.1.2-alphaF" #Figure_alpha_lines"


  pdf(paste0("P:\\Thesis\\R_plots\\", outfile,".pdf"),
      width=8, height=5.5)
  plot(c(min(var), max(var)), c(0, max(dat[, 17:22])),type="n" ,
       #     main="Figure 3.1  Type I error performance across different sample sizes",
       xlab=xname, ylab=yname)

  axis(1, at = seq(min(var), max(var), by=0.01), labels = T)
  #axis(2, at = c( 0, 0.07, 0.08, 0.09, 0.1,0.11, 0.12, 0.13), labels=T)

  abline (h=0.1, col="Gray23", lwd=1.2, lty=3)
  #critical - s
  points(x=var, y=dat$s_Prefer_Full_critical, type="o", col=colP[1],pch=19,lwd=1.2)
  #critical - i
  points(x=var,   y=dat$i_Prefer_Full_critical,   type="o",col=colP[2],   pch=2,   lty=2,
lwd=1.2)

  #alphaVec -s
  points(x=var, y=dat$s_Prefer_Full_alphaVec, type="o", col=colP[3], pch=19,lwd=1.2)
  #alphaVec -i
  points(x=var,  y=dat$i_Prefer_Full_alphaVec,  type="o",  col=colP[4],  pch=2,  lty=2,
lwd=1.2)

  #Bonfferoni -s
  points(x=var, y=dat$s_Prefer_Full_naive, type="o", col=colP[5],pch=19, lwd=1.2)
  points(x=var, y=dat$i_Prefer_Full_naive, type="o", col=colP[6],pch=2, lty=2, lwd=1.2)

  legend("left", c(
    "Unadjusted alpha - Interaction","Unadjusted alpha - Stratification",
    "Critical values - Interaction","Critical values - Stratification",
    "Bonferroni adjusted - Interaction","Bonferroni adjusted - Stratification"),
    xpd=T, horiz=F,
    #inset =location, #c(max(var),max(dat[, 17:22])),
    bty="n",  pch=c(2,19,2,19,2,19), lty=c(2,1,2,1,2,1), lwd=1.2,
    col=colP[c(4,3,2,1,6,5)],
    cex=1)
  dev.off()
```

```
#######################################################################
#####       Figure - Prevalence 0.2 vs. 0.5                       ######
#######################################################################
rows <- c(
  'binary_500_0.5_2_2_0.1_0.02_0_0',
  'binary_500_0.5_3_2_0.1_0.02_0_0',
  'binary_500_0.5_4_2_0.1_0.02_0_0',
  'binary_500_0.5_5_2_0.1_0.02_0_0',
  'binary_500_0.5_6_2_0.1_0.02_0_0',
  'binary_500_0.5_7_2_0.1_0.02_0_0',
  'binary_500_0.5_8_2_0.1_0.02_0_0',
  'binary_500_0.5_9_2_0.1_0.02_0_0',
  'binary_500_0.5_10_2_0.1_0.02_0_0',

  'binary_500_0.2_2_2_0.1_0.02_0_0',
  'binary_500_0.2_3_2_0.1_0.02_0_0',
  'binary_500_0.2_4_2_0.1_0.02_0_0',
  'binary_500_0.2_5_2_0.1_0.02_0_0',
  'binary_500_0.2_6_2_0.1_0.02_0_0',
  'binary_500_0.2_7_2_0.1_0.02_0_0',
  'binary_500_0.2_8_2_0.1_0.02_0_0',
  'binary_500_0.2_9_2_0.1_0.02_0_0'
  #,
  #'binary_500_0.2_10_2_0.1_0.02_0_0'

)

dat <- NULL
for (i in 1: length(rows)){
  dat <- rbind(dat,
               c(rows[i], get_result(rows[i])))
}
write.csv(dat, file = "Result_prevalence.csv")


dat1 <- data.frame(read.csv("Result_prevalence.csv"))
dat1 <- dat1[order(dat1$Prevalence, dat1$k),] #sort dat by the variables we need

#prev=0.5
dat <- dat1[which(dat1$Prevalence==0.5),]

var=dat$k
xname="k"
yname="Overall Type I Error"
outfile="3.1.3-prev-0.5" # Figure_prev_0.5.pdf"

    pdf(paste0("P:\\Thesis\\R_plots\\", outfile,".pdf"))
    plot(c(min(var), max(var)), c(0, max(dat[, 17:22])),type="n" ,
         #    main="Figure 3.1  Type I error performance across different sample sizes",
         xlab=xname, ylab=yname)

    axis(1, at = seq(min(var), max(var), by=1), labels = T)
    #axis(2, at = c( 0, 0.07, 0.08, 0.09, 0.1,0.11, 0.12, 0.13), labels=T)

    abline (h=0.1, col="Gray23", lwd=1.2, lty=3)
```

```
    #critical - s
    points(x=var, y=dat$s_Prefer_Full_critical, type="o", col=colP[1],pch=19,lwd=1.2)
    #critical - i
    points(x=var,  y=dat$i_Prefer_Full_critical,  type="o",col=colP[2],  pch=2,  lty=2,
lwd=1.2)

    #alphaVec -s
    points(x=var, y=dat$s_Prefer_Full_alphaVec, type="o", col=colP[3], pch=19,lwd=1.2)
    #alphaVec -i
    points(x=var,  y=dat$i_Prefer_Full_alphaVec,  type="o",  col=colP[4],  pch=2,  lty=2,
lwd=1.2)

    #Bonfferoni -s
    points(x=var, y=dat$s_Prefer_Full_naive, type="o", col=colP[5],pch=19, lwd=1.2)
    points(x=var,   y=dat$i_Prefer_Full_naive,   type="o",   col=colP[6],pch=2,   lty=2,
lwd=1.2)

    legend("right", c(
      "Unadjusted alpha - Interaction","Unadjusted alpha - Stratification",
      "Critical values - Interaction","Critical values - Stratification",
      "Bonferroni adjusted - Interaction","Bonferroni adjusted - Stratification"),
      xpd=T, horiz=F,
      #inset =location, #c(max(var),max(dat[, 17:22])),
      bty="n",  pch=c(2,19,2,19,2,19), lty=c(2,1,2,1,2,1), lwd=1.2,
      col=colP[c(4,3,2,1,6,5)],
      cex=1)
    dev.off()

#prev=0.2
dat <- dat1[which(dat1$Prevalence==0.2),]

var=dat$k
xname="k"
yname="Overall Type I Error"
outfile="3.1.3-prev-0.2.pdf" #"Figure_prev_0.2.pdf"

    pdf(paste0("P:\\Thesis\\R_plots\\", outfile))
    plot(c(min(var), max(var)), c(0, max(dat[, 17:22])),type="n" ,
         #    main="Figure 3.1  Type I error performance across different sample sizes",
         xlab=xname, ylab=yname)

    axis(1, at = seq(min(var), max(var), by=1), labels = T)
    #axis(2, at = c( 0, 0.07, 0.08, 0.09, 0.1,0.11, 0.12, 0.13), labels=T)

    abline (h=0.1, col="Gray23", lwd=1.2, lty=3)
    #critical - s
    points(x=var, y=dat$s_Prefer_Full_critical, type="o", col=colP[1],pch=19,lwd=1.2)
    #critical - i
    points(x=var,  y=dat$i_Prefer_Full_critical,  type="o",col=colP[2],  pch=2,  lty=2,
lwd=1.2)

    #alphaVec -s
    points(x=var, y=dat$s_Prefer_Full_alphaVec, type="o", col=colP[3], pch=19,lwd=1.2)
    #alphaVec -i
```

```
    points(x=var, y=dat$i_Prefer_Full_alphaVec, type="o", col=colP[4], pch=2, lty=2,
lwd=1.2)

    #Bonfferoni -s
    points(x=var, y=dat$s_Prefer_Full_naive, type="o", col=colP[5],pch=19, lwd=1.2)
    points(x=var,   y=dat$i_Prefer_Full_naive,   type="o",   col=colP[6],pch=2,   lty=2,
lwd=1.2)

    legend("right", c(
      "Unadjusted alpha - Interaction","Unadjusted alpha - Stratification",
      "Critical values - Interaction","Critical values - Stratification",
      "Bonferroni adjusted - Interaction","Bonferroni adjusted - Stratification"),
      xpd=T, horiz=F,
      #inset =location, #c(max(var),max(dat[, 17:22])),
      bty="n",  pch=c(2,19,2,19,2,19), lty=c(2,1,2,1,2,1), lwd=1.2,
      col=colP[c(4,3,2,1,6,5)],
      cex=1)
    dev.off()




######################################################################
#####  Figure - Type I error various combinations of K and L   ######
######################################################################

rows<-c(
"binary_500_0.5_2_2_0.1_0.02_0_0"

,"binary_500_0.5_3_2_0.1_0.02_0_0"
,"binary_500_0.5_3_3_0.1_0.02_0_0"

,"binary_500_0.5_4_2_0.1_0.02_0_0"
,"binary_500_0.5_4_3_0.1_0.02_0_0"
,"binary_500_0.5_4_4_0.1_0.02_0_0"

,"binary_500_0.5_5_2_0.1_0.02_0_0"
,"binary_500_0.5_5_3_0.1_0.02_0_0"
,"binary_500_0.5_5_4_0.1_0.02_0_0"
,"binary_500_0.5_5_5_0.1_0.02_0_0"

,"binary_500_0.5_6_2_0.1_0.02_0_0"
,"binary_500_0.5_6_3_0.1_0.02_0_0"
,"binary_500_0.5_6_4_0.1_0.02_0_0"
,"binary_500_0.5_6_5_0.1_0.02_0_0"
#,"binary_500_0.5_6_6_0.1_0.02_0_0"

,"binary_500_0.5_7_2_0.1_0.02_0_0"
,"binary_500_0.5_7_3_0.1_0.02_0_0"
,"binary_500_0.5_7_4_0.1_0.02_0_0"
#,"binary_500_0.5_7_5_0.1_0.02_0_0"
#,"binary_500_0.5_7_6_0.1_0.02_0_0"
#,"binary_500_0.5_7_7_0.1_0.02_0_0"
```

```
,"binary_500_0.5_8_2_0.1_0.02_0_0"
,"binary_500_0.5_8_3_0.1_0.02_0_0"
,"binary_500_0.5_8_4_0.1_0.02_0_0"  #k=8, max L=4

,"binary_500_0.5_9_2_0.1_0.02_0_0"
,"binary_500_0.5_9_3_0.1_0.02_0_0"  #
#,"binary_500_0.5_9_4_0.1_0.02_0_0"

,"binary_500_0.5_10_2_0.1_0.02_0_0"
,"binary_500_0.5_11_2_0.1_0.02_0_0"
,"binary_500_0.5_12_2_0.1_0.02_0_0"
#,"binary_500_0.5_13_2_0.1_0.02_0_0" #check
#,"binary_500_0.5_14_2_0.1_0.02_0_0"
,"binary_500_0.5_15_2_0.1_0.02_0_0"
,"binary_500_0.5_16_2_0.1_0.02_0_0"
,"binary_500_0.5_17_2_0.1_0.02_0_0"
,"binary_500_0.5_18_2_0.1_0.02_0_0"
,"binary_500_0.5_19_2_0.1_0.02_0_0"
,"binary_500_0.5_20_2_0.1_0.02_0_0"
,"binary_500_0.5_25_2_0.1_0.02_0_0"
,"binary_500_0.5_30_2_0.1_0.02_0_0"
,"binary_500_0.5_35_2_0.1_0.02_0_0"
,"binary_500_0.5_40_2_0.1_0.02_0_0"
,"binary_500_0.5_45_2_0.1_0.02_0_0"
,"binary_500_0.5_50_2_0.1_0.02_0_0"
,

'binary_500_0.5_2_1_0.1_0.02_0_0', #L=1
'binary_500_0.5_3_1_0.1_0.02_0_0',
'binary_500_0.5_4_1_0.1_0.02_0_0',
'binary_500_0.5_5_1_0.1_0.02_0_0',
'binary_500_0.5_6_1_0.1_0.02_0_0',
'binary_500_0.5_7_1_0.1_0.02_0_0',
'binary_500_0.5_8_1_0.1_0.02_0_0',
'binary_500_0.5_9_1_0.1_0.02_0_0',
'binary_500_0.5_10_1_0.1_0.02_0_0',
'binary_500_0.5_11_1_0.1_0.02_0_0',
'binary_500_0.5_12_1_0.1_0.02_0_0',
'binary_500_0.5_13_1_0.1_0.02_0_0',
'binary_500_0.5_14_1_0.1_0.02_0_0',
'binary_500_0.5_15_1_0.1_0.02_0_0',
'binary_500_0.5_16_1_0.1_0.02_0_0',
'binary_500_0.5_17_1_0.1_0.02_0_0',
'binary_500_0.5_18_1_0.1_0.02_0_0',
'binary_500_0.5_19_1_0.1_0.02_0_0',
'binary_500_0.5_20_1_0.1_0.02_0_0',
'binary_500_0.5_21_1_0.1_0.02_0_0',
'binary_500_0.5_22_1_0.1_0.02_0_0',
'binary_500_0.5_23_1_0.1_0.02_0_0',
'binary_500_0.5_24_1_0.1_0.02_0_0',
'binary_500_0.5_25_1_0.1_0.02_0_0',
'binary_500_0.5_26_1_0.1_0.02_0_0',
'binary_500_0.5_27_1_0.1_0.02_0_0',
'binary_500_0.5_28_1_0.1_0.02_0_0',
'binary_500_0.5_29_1_0.1_0.02_0_0',
```

```
# 'binary_500_0.5_30_1_0.1_0.02_0_0',
'binary_500_0.5_31_1_0.1_0.02_0_0',
'binary_500_0.5_32_1_0.1_0.02_0_0',
'binary_500_0.5_33_1_0.1_0.02_0_0',
'binary_500_0.5_34_1_0.1_0.02_0_0',
'binary_500_0.5_35_1_0.1_0.02_0_0',
# 'binary_500_0.5_36_1_0.1_0.02_0_0',
# 'binary_500_0.5_37_1_0.1_0.02_0_0',
# 'binary_500_0.5_38_1_0.1_0.02_0_0',
# 'binary_500_0.5_39_1_0.1_0.02_0_0',
'binary_500_0.5_40_1_0.1_0.02_0_0',
# 'binary_500_0.5_41_1_0.1_0.02_0_0',
# 'binary_500_0.5_42_1_0.1_0.02_0_0',
# 'binary_500_0.5_43_1_0.1_0.02_0_0',
# 'binary_500_0.5_44_1_0.1_0.02_0_0',
'binary_500_0.5_45_1_0.1_0.02_0_0',
# 'binary_500_0.5_46_1_0.1_0.02_0_0',
# 'binary_500_0.5_47_1_0.1_0.02_0_0',
# 'binary_500_0.5_48_1_0.1_0.02_0_0',
# 'binary_500_0.5_49_1_0.1_0.02_0_0',
'binary_500_0.5_50_1_0.1_0.02_0_0'
)

# run results here
dat <- NULL
for (i in 1: length(rows)){
dat <- rbind(dat,
     c(rows[i], get_result(rows[i])) )
}
write.csv(dat, file = "Result_k_L.csv")

dat <- read.csv("Result_k_L.csv")
dat <- dat[order(dat$k, dat$ell),] #sort dat by the variables we need

# new dataset for plot
dat2<-dat
dat2$s_Prefer_Full_critical2 <- 25*(dat2$s_Prefer_Full_critical-0.1)+dat2$ell
dat2$i_Prefer_Full_critical2 <- 25*(dat2$i_Prefer_Full_critical-0.1)+dat2$ell
ml <- max(dat2$ell)

# output pdf files
    pdf("P:\\Thesis\\R_plots\\3.1.4-k_L.pdf",
      width=8, height=5.5)
    par(mar=c(5, 4, 1, 6) + 0.1) #extra room for axis
    plot(c(min(dat2$k), max(dat2$k)), c(0.5, 6.5),type="n" ,
         xlab="Number of Covariates (k)",
         ylab="Max Number of Covariates in SHAPES (L)",
         #main="Stratification Models",
         xaxt='n',yaxt='n')
    axis(1, at = c(1, seq(5,max(dat2$k), 5)), labels = T)
    axis(2, at = c( 1:ml), labels=T)

    # lables on right side
    mtext("Overall Type I error",side=4,line=4)
    axis(4,
```

```
         at=rep(c(1:ml),each=3)+rep(c(-0.25, 0, 0.25), ml),
         labels=rep (c(0.09, 0.1, 0.11),ml),
         las=1, tck=-0.01)
    # points for each L
    for (i in 1:ml){
      points(x=dat2[which(dat2$ell==i),]$k,
y=dat2[which(dat2$ell==i),]$s_Prefer_Full_critical2,
             type="o", lwd=1, col=colA[i], pch=19)
      points(x=dat2[which(dat2$ell==i),]$k,
y=dat2[which(dat2$ell==i),]$i_Prefer_Full_critical2,
             type="o", lwd=1, col=colB[i], pch=2, lty=2)
      # overall Type I error range lines
      abline (h=i, col="Gray23", lwd=1, lty=4)
      abline (h=i+0.25, col="Gray23", lwd=1, lty=3)
      abline (h=i-0.25, col="Gray23", lwd=1, lty=3)
    }
    legend("topright",pch=c(19,2), lty=c(1,2),lwd=1,cex=0.9,
           c("Stratification Models", "Interaction Models"))
    dev.off()
dat[,c(5:8, 14:15)]



######################################################################
#####           Figure - Correlated Covariates                  ######
######################################################################
rows <- c('binary_500_0.5_4_4_0.1_0.02_0_0',

'binary_500_0.5_4_4_0.1_0.02_2_0.1',
'binary_500_0.5_4_4_0.1_0.02_2_0.2',
'binary_500_0.5_4_4_0.1_0.02_2_0.3',
'binary_500_0.5_4_4_0.1_0.02_2_0.4',
'binary_500_0.5_4_4_0.1_0.02_2_0.5',
'binary_500_0.5_4_4_0.1_0.02_2_0.6',
'binary_500_0.5_4_4_0.1_0.02_2_0.7',
'binary_500_0.5_4_4_0.1_0.02_2_0.8',
'binary_500_0.5_4_4_0.1_0.02_2_0.9',

'binary_500_0.5_4_4_0.1_0.02_3_0.1',
'binary_500_0.5_4_4_0.1_0.02_3_0.2',
'binary_500_0.5_4_4_0.1_0.02_3_0.3',
'binary_500_0.5_4_4_0.1_0.02_3_0.4',
'binary_500_0.5_4_4_0.1_0.02_3_0.5',
'binary_500_0.5_4_4_0.1_0.02_3_0.6',
'binary_500_0.5_4_4_0.1_0.02_3_0.7',

'binary_500_0.5_4_4_0.1_0.02_4_0.1',
'binary_500_0.5_4_4_0.1_0.02_4_0.2',
'binary_500_0.5_4_4_0.1_0.02_4_0.3',
'binary_500_0.5_4_4_0.1_0.02_4_0.4',
'binary_500_0.5_4_4_0.1_0.02_4_0.5',
'binary_500_0.5_4_4_0.1_0.02_4_0.6'

)
dat <- NULL
```

```
for (i in 1: length(rows)){
dat <- rbind(dat,
     c(rows[i], get_result(rows[i])))
}
write.csv(dat, file = "Result_correlated_4_4.csv")
dat <- read.csv("Result_correlated_4_4.csv")

#dat <- result[which(result$X.1 %in% rows),]
dat  <-  dat[order(dat$Correlated.covariates,  dat$Correlation),]  #sort  dat  by  the
variables we need

dat2<-dat
dat2$s_Prefer_Full_critical2            <-            25*(dat2$s_Prefer_Full_critical-
0.1)+dat2$Correlated.covariates
dat2$i_Prefer_Full_critical2            <-            25*(dat2$i_Prefer_Full_critical-
0.1)+dat2$Correlated.covariates
ml <- max(dat2$Correlated.covariates)

pdf(paste0("P:\\Thesis\\R_plots\\", "3.1.5-corr.pdf"),
  width=8, height=5.5)
par(mar=c(5, 4, 1, 6) + 0.1) #extra room for axis
plot(c(min(dat2$Correlation), max(dat2$Correlation)), c(1.5, 4.5),type="n" ,
   xlab="Correlation between Covariates",
   ylab="Number of Correlated Covariates",
   #main="Stratification Models",
   xaxt='n',yaxt='n')
axis(1, at = seq(min(dat2$Correlation), max(dat2$Correlation), by=0.1), labels = T)
axis(2, at = c( 2:ml), labels=T)
# lables on right side
mtext("Overall Type I error",side=4,line=4)
# 95% CI is +-0.005 instead of +-0.01
axis(4,
   at=rep(c(2,3,4),each=3)+rep(c(-0.25/2, 0, 0.25/2), ml-1),
   labels=rep (c(0.095, 0.1, 0.105),ml-1),
   las=1, tck=-0.01)
for (i in 2:ml){
points(x=dat2[which(dat2$Correlated.covariates==i),]$Correlation,
       y=dat2[which(dat2$Correlated.covariates==i),]$s_Prefer_Full_critical2,
       type="o", lwd=1, col=colA[i], pch=19)
points(x=dat2[which(dat2$Correlated.covariates==i),]$Correlation,
       y=dat2[which(dat2$Correlated.covariates==i),]$i_Prefer_Full_critical2,
       type="o", lwd=1, col=colB[i], pch=2, lty=2)
abline (h=i, col="Gray23", lwd=1, lty=4)
abline (h=i+0.25/2, col="Gray23", lwd=1, lty=3)
abline (h=i-0.25/2, col="Gray23", lwd=1, lty=3)
}
legend("topright",pch=c(19,2), lty=c(1,2),lwd=1,cex=0.9,
     c("Stratification Models", "Interaction Models"))
dev.off()
```

```
##########################################################
```

```
# File name:  R_result_continuous.R"
# Purpose:    Plot results for continuous endpoint
# Written by: Lei Wang
# Last date:  8/25/2017
##########################################################

setwd("P:/Thesis")

# all the files' name
file_names <- dir("P:/Thesis")

# the folder that contain the simulation results
dataAll <- file_names[substr(file_names, 1, 6) %in% c("binary", "contin")]

# R brewer color
colA <- brewer.pal(9, "Set1")
colB <- brewer.pal(8, "Dark2")
colP <- brewer.pal(12, "Paired")

#########################################################################
##### Figure - Type I error varies w/ n increases (ell fixed ) ######
#########################################################################

rows <- c(
#'continuous_50_0.5_4_2_0.1_0.02_0_0',
'continuous_100_0.5_4_2_0.1_0.02_0_0',
'continuous_200_0.5_4_2_0.1_0.02_0_0',
'continuous_300_0.5_4_2_0.1_0.02_0_0',
'continuous_400_0.5_4_2_0.1_0.02_0_0',
'continuous_500_0.5_4_2_0.1_0.02_0_0',
'continuous_600_0.5_4_2_0.1_0.02_0_0',
'continuous_700_0.5_4_2_0.1_0.02_0_0',
'continuous_800_0.5_4_2_0.1_0.02_0_0',
'continuous_900_0.5_4_2_0.1_0.02_0_0',
'continuous_1000_0.5_4_2_0.1_0.02_0_0')



# run results here
dat <- NULL
for (i in 1: length(rows)){
dat <- rbind(dat,
       c(rows[i], get_result(rows[i])))
}
write.csv(dat, file = "ResultCts_n.csv")


dat <- read.csv("ResultCts_n.csv")
dat <- dat[order(dat$Sample.Size),] #sort dat by the variables we need


var=dat$Sample.Size
xname="Sample Size (n)"
yname="Overall Type I Error"
outfile="3.2.1-n"  # by sample size
```

```
pdf(paste0("P:\\Thesis\\R_plots\\", outfile,".pdf"),
    width=8, height=5.5)
plot(c(min(var), max(var)), c(0, max(dat[, 17:22])),type="n" ,
     #     main="Figure 3.1  Type I error performance across different sample sizes",
     xlab=xname, ylab=yname)

axis(1, at = seq(min(var), max(var), by=100), labels = T)

abline (h=0.1, col="Gray23", lwd=2, lty=3)
#critical - s
points(x=var, y=dat$s_Prefer_Full_critical, type="o", col=colP[1],pch=19,lwd=2)
#critical - i
points(x=var, y=dat$i_Prefer_Full_critical, type="o",col=colP[2], pch=2, lty=2, lwd=2)

#alphaVec -s
points(x=var, y=dat$s_Prefer_Full_alphaVec, type="o", col=colP[3], pch=19,lwd=2)
#alphaVec -i
points(x=var, y=dat$i_Prefer_Full_alphaVec, type="o", col=colP[4], pch=2, lty=2, lwd=2)

#Bonfferoni -s
points(x=var, y=dat$s_Prefer_Full_naive, type="o", col=colP[5],pch=19, lwd=2)
points(x=var, y=dat$i_Prefer_Full_naive, type="o", col=colP[6],pch=2, lty=2, lwd=2)

legend("right", c(
  "Unadjusted alpha - Interaction","Unadjusted alpha - Stratification",
  "Critical values - Interaction","Critical values - Stratification",
  "Bonferroni adjusted - Interaction","Bonferroni adjusted - Stratification"),
  xpd=T, horiz=F,
  #inset =location, #c(max(var),max(dat[, 17:22])),
  bty="n",  pch=c(2,19,2,19,2,19), lty=c(2,1,2,1,2,1), lwd=2,
  col=colP[c(4,3,2,1,6,5)],
  cex=1)
dev.off()


######################################################################
##### Figure - Type I error varies w/ alphaF allocation        ######
######################################################################

rows <- c('continuous_500_0.5_4_2_0.1_0_0_0',
'continuous_500_0.5_4_2_0.1_0.01_0_0',
'continuous_500_0.5_4_2_0.1_0.02_0_0',
'continuous_500_0.5_4_2_0.1_0.03_0_0',
'continuous_500_0.5_4_2_0.1_0.04_0_0',
'continuous_500_0.5_4_2_0.1_0.05_0_0',
'continuous_500_0.5_4_2_0.1_0.06_0_0',
'continuous_500_0.5_4_2_0.1_0.07_0_0',
'continuous_500_0.5_4_2_0.1_0.08_0_0',
'continuous_500_0.5_4_2_0.1_0.09_0_0',
'continuous_500_0.5_4_2_0.1_0.095_0_0')

# run results here
dat <- NULL
for (i in 1: length(rows)){
```

```
dat <- rbind(dat,
       c(rows[i], get_result(rows[i])))}
write.csv(dat, file = "ResultCts_a.csv")

dat <- read.csv("ResultCts_a.csv")
dat <- dat[order(dat$alphaF),] #sort dat by the variables we need

var=dat$alphaF
xname="Alpha Allocated to Full Population (alphaF)"
yname="Overall Type I Error"
outfile="3.2.2-alphaF"

pdf(paste0("P:\\Thesis\\R_plots\\", outfile,".pdf"),
    width=8, height=5.5)
plot(c(min(var), max(var)), c(0, max(dat[, 17:22])),type="n" ,
     xlab=xname, ylab=yname)

axis(1, at = seq(min(var), max(var), by=0.01), labels = T)

abline (h=0.1, col="Gray23", lwd=2, lty=3)
#critical - s
points(x=var, y=dat$s_Prefer_Full_critical, type="o", col=colP[1],pch=19,lwd=2)
#critical - i
points(x=var, y=dat$i_Prefer_Full_critical, type="o",col=colP[2], pch=2, lty=2, lwd=2)

#alphaVec -s
points(x=var, y=dat$s_Prefer_Full_alphaVec, type="o", col=colP[3], pch=19,lwd=2)
#alphaVec -i
points(x=var, y=dat$i_Prefer_Full_alphaVec, type="o", col=colP[4], pch=2, lty=2, lwd=2)

#Bonfferoni -s
points(x=var, y=dat$s_Prefer_Full_naive, type="o", col=colP[5],pch=19, lwd=2)
points(x=var, y=dat$i_Prefer_Full_naive, type="o", col=colP[6],pch=2, lty=2, lwd=2)

legend("left", c(
  "Unadjusted alpha - Interaction","Unadjusted alpha - Stratification",
  "Critical values - Interaction","Critical values - Stratification",
  "Bonferroni adjusted - Interaction","Bonferroni adjusted - Stratification"),
  xpd=T, horiz=F,
  bty="n",  pch=c(2,19,2,19,2,19), lty=c(2,1,2,1,2,1), lwd=2,
  col=colP[c(4,3,2,1,6,5)],
  cex=1)
dev.off()

######################################################################
#####       Figure - Prevalence 0.2 vs. 0.5                      ######
######################################################################
rows <- c(
'continuous_500_0.2_2_2_0.1_0.02_0_0',
'continuous_500_0.2_3_2_0.1_0.02_0_0',
'continuous_500_0.2_4_2_0.1_0.02_0_0',
'continuous_500_0.2_5_2_0.1_0.02_0_0',
'continuous_500_0.2_6_2_0.1_0.02_0_0',
# 'continuous_500_0.2_7_2_0.1_0.02_0_0',
# 'continuous_500_0.2_8_2_0.1_0.02_0_0',
```

```
# 'continuous_500_0.2_9_2_0.1_0.02_0_0',
# 'continuous_500_0.2_10_2_0.1_0.02_0_0',

'continuous_500_0.5_2_2_0.1_0.02_0_0',
'continuous_500_0.5_3_2_0.1_0.02_0_0',
'continuous_500_0.5_4_2_0.1_0.02_0_0',
'continuous_500_0.5_5_2_0.1_0.02_0_0',
'continuous_500_0.5_6_2_0.1_0.02_0_0',
'continuous_500_0.5_7_2_0.1_0.02_0_0',
'continuous_500_0.5_8_2_0.1_0.02_0_0',
'continuous_500_0.5_9_2_0.1_0.02_0_0',
'continuous_500_0.5_10_2_0.1_0.02_0_0')



dat <- NULL
for (i in 1: length(rows)){
dat <- rbind(dat,
       c(rows[i], get_result(rows[i])))
}
write.csv(dat, file = "ResultCts_prevalence.csv")


dat1 <- data.frame(read.csv("ResultCts_prevalence.csv"))
dat1 <- dat1[order(dat1$Prevalence, dat1$k),] #sort dat by the variables we need

#prev=0.5
dat <- dat1[which(dat1$Prevalence==0.5),]

var=dat$k
xname="k"
yname="Overall Type I Error"
outfile="3.2.3-prev-0.5"

pdf(paste0("P:\\Thesis\\R_plots\\", outfile,".pdf"))
plot(c(min(var), max(var)), c(0, max(dat[, 17:22])),type="n" ,
     #     main="Figure 3.1  Type I error performance across different sample sizes",
     xlab=xname, ylab=yname)

axis(1, at = seq(min(var), max(var), by=1), labels = T)
#axis(2, at = c( 0, 0.07, 0.08, 0.09, 0.1,0.11, 0.12, 0.13), labels=T)

abline (h=0.1, col="Gray23", lwd=2, lty=3)
#critical - s
points(x=var, y=dat$s_Prefer_Full_critical, type="o", col=colP[1],pch=19,lwd=2)
#critical - i
points(x=var, y=dat$i_Prefer_Full_critical, type="o",col=colP[2], pch=2, lty=2, lwd=2)

#alphaVec -s
points(x=var, y=dat$s_Prefer_Full_alphaVec, type="o", col=colP[3], pch=19,lwd=2)
#alphaVec -i
points(x=var, y=dat$i_Prefer_Full_alphaVec, type="o", col=colP[4], pch=2, lty=2, lwd=2)

#Bonfferoni -s
points(x=var, y=dat$s_Prefer_Full_naive, type="o", col=colP[5],pch=19, lwd=2)
```

```
points(x=var, y=dat$i_Prefer_Full_naive, type="o", col=colP[6],pch=2, lty=2, lwd=2)

legend("right", c(
  "Unadjusted alpha - Interaction","Unadjusted alpha - Stratification",
  "Critical values - Interaction","Critical values - Stratification",
  "Bonferroni adjusted - Interaction","Bonferroni adjusted - Stratification"),
  xpd=T, horiz=F,
  #inset =location, #c(max(var),max(dat[, 17:22])),
  bty="n",  pch=c(2,19,2,19,2,19), lty=c(2,1,2,1,2,1), lwd=2,
  col=colP[c(4,3,2,1,6,5)],
  cex=1)
dev.off()

#prev=0.2
dat <- dat1[which(dat1$Prevalence==0.2),]

var=dat$k
xname="k"
yname="Overall Type I Error"
outfile="3.2.3-prev-0.2.pdf"

pdf(paste0("P:\\Thesis\\R_plots\\", outfile))
plot(c(min(var), max(var)), c(0, max(dat[, 17:22])),type="n" ,
     #     main="Figure 3.1  Type I error performance across different sample sizes",
     xlab=xname, ylab=yname)

axis(1, at = seq(min(var), max(var), by=1), labels = T)
#axis(2, at = c( 0, 0.07, 0.08, 0.09, 0.1,0.11, 0.12, 0.13), labels=T)

abline (h=0.1, col="Gray23", lwd=2, lty=3)
#critical - s
points(x=var, y=dat$s_Prefer_Full_critical, type="o", col=colP[1],pch=19,lwd=2)
#critical - i
points(x=var, y=dat$i_Prefer_Full_critical, type="o",col=colP[2], pch=2, lty=2, lwd=2)

#alphaVec -s
points(x=var, y=dat$s_Prefer_Full_alphaVec, type="o", col=colP[3], pch=19,lwd=2)
#alphaVec -i
points(x=var, y=dat$i_Prefer_Full_alphaVec, type="o", col=colP[4], pch=2, lty=2, lwd=2)

#Bonfferoni -s
points(x=var, y=dat$s_Prefer_Full_naive, type="o", col=colP[5],pch=19, lwd=2)
points(x=var, y=dat$i_Prefer_Full_naive, type="o", col=colP[6],pch=2, lty=2, lwd=2)

legend("right", c(
  "Unadjusted alpha - Interaction","Unadjusted alpha - Stratification",
  "Critical values - Interaction","Critical values - Stratification",
  "Bonferroni adjusted - Interaction","Bonferroni adjusted - Stratification"),
  xpd=T, horiz=F,
  #inset =location, #c(max(var),max(dat[, 17:22])),
  bty="n",  pch=c(2,19,2,19,2,19), lty=c(2,1,2,1,2,1), lwd=2,
  col=colP[c(4,3,2,1,6,5)],
  cex=1)
dev.off()
```

```
#########################################################################
#####  Figure - Type I error various combinations of K and L   ######
#########################################################################

rows <- c(
'continuous_500_0.5_2_1_0.1_0.02_0_0', #L=1
'continuous_500_0.5_3_1_0.1_0.02_0_0',
'continuous_500_0.5_4_1_0.1_0.02_0_0',
'continuous_500_0.5_5_1_0.1_0.02_0_0',
'continuous_500_0.5_6_1_0.1_0.02_0_0',
'continuous_500_0.5_7_1_0.1_0.02_0_0',
'continuous_500_0.5_8_1_0.1_0.02_0_0',
'continuous_500_0.5_9_1_0.1_0.02_0_0',
'continuous_500_0.5_10_1_0.1_0.02_0_0',
'continuous_500_0.5_11_1_0.1_0.02_0_0',
'continuous_500_0.5_12_1_0.1_0.02_0_0',
'continuous_500_0.5_13_1_0.1_0.02_0_0',
'continuous_500_0.5_14_1_0.1_0.02_0_0',
'continuous_500_0.5_15_1_0.1_0.02_0_0',
'continuous_500_0.5_16_1_0.1_0.02_0_0',
'continuous_500_0.5_17_1_0.1_0.02_0_0',
'continuous_500_0.5_18_1_0.1_0.02_0_0',
'continuous_500_0.5_19_1_0.1_0.02_0_0',
'continuous_500_0.5_20_1_0.1_0.02_0_0',
# 'continuous_500_0.5_21_1_0.1_0.02_0_0',
# 'continuous_500_0.5_22_1_0.1_0.02_0_0',
# 'continuous_500_0.5_23_1_0.1_0.02_0_0',
# 'continuous_500_0.5_24_1_0.1_0.02_0_0',
'continuous_500_0.5_25_1_0.1_0.02_0_0',
# 'continuous_500_0.5_26_1_0.1_0.02_0_0',
# 'continuous_500_0.5_27_1_0.1_0.02_0_0',
# 'continuous_500_0.5_28_1_0.1_0.02_0_0',
# 'continuous_500_0.5_29_1_0.1_0.02_0_0',
'continuous_500_0.5_30_1_0.1_0.02_0_0',
# 'continuous_500_0.5_31_1_0.1_0.02_0_0',
# 'continuous_500_0.5_32_1_0.1_0.02_0_0',
# 'continuous_500_0.5_33_1_0.1_0.02_0_0',
# 'continuous_500_0.5_34_1_0.1_0.02_0_0',
'continuous_500_0.5_35_1_0.1_0.02_0_0',
# 'continuous_500_0.5_36_1_0.1_0.02_0_0',
# 'continuous_500_0.5_37_1_0.1_0.02_0_0',
# 'continuous_500_0.5_38_1_0.1_0.02_0_0',
# 'continuous_500_0.5_39_1_0.1_0.02_0_0',
'continuous_500_0.5_40_1_0.1_0.02_0_0',
# 'continuous_500_0.5_41_1_0.1_0.02_0_0',
# 'continuous_500_0.5_42_1_0.1_0.02_0_0',
# 'continuous_500_0.5_43_1_0.1_0.02_0_0',
# 'continuous_500_0.5_44_1_0.1_0.02_0_0',
'continuous_500_0.5_45_1_0.1_0.02_0_0',
# 'continuous_500_0.5_46_1_0.1_0.02_0_0',
# 'continuous_500_0.5_47_1_0.1_0.02_0_0',
# 'continuous_500_0.5_48_1_0.1_0.02_0_0',
```

```
# 'continuous_500_0.5_49_1_0.1_0.02_0_0',
'continuous_500_0.5_50_1_0.1_0.02_0_0',


'continuous_500_0.5_2_2_0.1_0.02_0_0',  #L=2
'continuous_500_0.5_3_2_0.1_0.02_0_0',
'continuous_500_0.5_4_2_0.1_0.02_0_0',
'continuous_500_0.5_5_2_0.1_0.02_0_0',
'continuous_500_0.5_6_2_0.1_0.02_0_0',
'continuous_500_0.5_7_2_0.1_0.02_0_0',
'continuous_500_0.5_8_2_0.1_0.02_0_0',
'continuous_500_0.5_9_2_0.1_0.02_0_0',
'continuous_500_0.5_10_2_0.1_0.02_0_0',
'continuous_500_0.5_11_2_0.1_0.02_0_0',
'continuous_500_0.5_12_2_0.1_0.02_0_0',
'continuous_500_0.5_13_2_0.1_0.02_0_0',
'continuous_500_0.5_14_2_0.1_0.02_0_0',
'continuous_500_0.5_15_2_0.1_0.02_0_0',
'continuous_500_0.5_16_2_0.1_0.02_0_0',
'continuous_500_0.5_17_2_0.1_0.02_0_0',
'continuous_500_0.5_18_2_0.1_0.02_0_0',
'continuous_500_0.5_19_2_0.1_0.02_0_0',
'continuous_500_0.5_20_2_0.1_0.02_0_0',
'continuous_500_0.5_25_2_0.1_0.02_0_0',
'continuous_500_0.5_30_2_0.1_0.02_0_0',
'continuous_500_0.5_35_2_0.1_0.02_0_0',
'continuous_500_0.5_40_2_0.1_0.02_0_0',
'continuous_500_0.5_45_2_0.1_0.02_0_0',

'continuous_500_0.5_3_3_0.1_0.02_0_0', #L=3
'continuous_500_0.5_4_3_0.1_0.02_0_0',
'continuous_500_0.5_5_3_0.1_0.02_0_0',
'continuous_500_0.5_6_3_0.1_0.02_0_0',
'continuous_500_0.5_7_3_0.1_0.02_0_0',
'continuous_500_0.5_8_3_0.1_0.02_0_0',
'continuous_500_0.5_9_3_0.1_0.02_0_0',
'continuous_500_0.5_10_3_0.1_0.02_0_0'
,
'continuous_500_0.5_4_4_0.1_0.02_0_0', #L=4
'continuous_500_0.5_5_4_0.1_0.02_0_0',
'continuous_500_0.5_6_4_0.1_0.02_0_0',
'continuous_500_0.5_7_4_0.1_0.02_0_0',
'continuous_500_0.5_8_4_0.1_0.02_0_0',
'continuous_500_0.5_9_4_0.1_0.02_0_0'
,

'continuous_500_0.5_5_5_0.1_0.02_0_0', #L=5
'continuous_500_0.5_6_5_0.1_0.02_0_0'
#'continuous_500_0.5_7_5_0.1_0.02_0_0'


)

# run results here
dat <- NULL
```

```
for (i in 1: length(rows)){
dat <- rbind(dat,
       c(rows[i], get_result(rows[i])) )
}
write.csv(dat, file = "ResultCts_k_L.csv")

dat <- read.csv("ResultCts_k_L.csv")
dat <- dat[order(dat$k, dat$ell),] #sort dat by the variables we need

dat2<-dat
dat2$s_Prefer_Full_critical2 <- 25*(dat2$s_Prefer_Full_critical-0.1)+dat2$ell
dat2$i_Prefer_Full_critical2 <- 25*(dat2$i_Prefer_Full_critical-0.1)+dat2$ell
ml <- max(dat2$ell)

pdf("P:\\Thesis\\R_plots\\3.2.4-k_L.pdf",
width=8, height=5.5)
par(mar=c(5, 4, 1, 6) + 0.1) #extra room for axis
plot(c(min(dat2$k), max(dat2$k)), c(0.5, 4.5),type="n" ,
 xlab="Number of Covariates (k)",
 ylab="Max Number of Covariates in SHAPES (L)",
 #main="Stratification Models",
 xaxt='n',yaxt='n')
axis(1, at = c(1, seq(5,max(dat2$k), 5)), labels = T)
axis(2, at = c( 1:ml), labels=T)

# lables on right side
mtext("Overall Type I error",side=4,line=4)
axis(4,
 at=rep(c(1:ml),each=3)+rep(c(-0.25, 0, 0.25), ml),
 labels=rep (c(0.09, 0.1, 0.11),ml),
 las=1, tck=-0.01)
# points for each L
for (i in 1:ml){
points(x=dat2[which(dat2$ell==i),]$k,
y=dat2[which(dat2$ell==i),]$s_Prefer_Full_critical2,
     type="o", lwd=1, col=colA[i], pch=19)
points(x=dat2[which(dat2$ell==i),]$k,
y=dat2[which(dat2$ell==i),]$i_Prefer_Full_critical2,
     type="o", lwd=1, col=colB[i], pch=2, lty=2)
# overall Type I error range lines
abline (h=i, col="Gray23", lwd=1, lty=4)
abline (h=i+0.25, col="Gray23", lwd=1, lty=3)
abline (h=i-0.25, col="Gray23", lwd=1, lty=3)
}
legend("topright",pch=c(1,2), lty=c(1,2),lwd=1,cex=0.9,
   c("Stratification Models", "Interaction Models"))
dev.off()

dat[,c(5:8, 14:15)]



#####################################################################
#####           Figure - Correlated Covariates                  ######
```

```
#####################################################################
rows <- c('continuous_500_0.5_4_4_0.1_0.02_2_0.1',
  'continuous_500_0.5_4_4_0.1_0.02_2_0.2',
  'continuous_500_0.5_4_4_0.1_0.02_2_0.3',
  'continuous_500_0.5_4_4_0.1_0.02_2_0.4',
  'continuous_500_0.5_4_4_0.1_0.02_2_0.5',
  'continuous_500_0.5_4_4_0.1_0.02_2_0.6',
  'continuous_500_0.5_4_4_0.1_0.02_2_0.7',
  'continuous_500_0.5_4_4_0.1_0.02_2_0.8',
  'continuous_500_0.5_4_4_0.1_0.02_2_0.9',
  'continuous_500_0.5_4_4_0.1_0.02_3_0.1',
  'continuous_500_0.5_4_4_0.1_0.02_3_0.2',
  'continuous_500_0.5_4_4_0.1_0.02_3_0.3',
  'continuous_500_0.5_4_4_0.1_0.02_3_0.4',
  'continuous_500_0.5_4_4_0.1_0.02_3_0.5',
  'continuous_500_0.5_4_4_0.1_0.02_3_0.6',
  # 'continuous_500_0.5_4_4_0.1_0.02_3_0.7',
  # 'continuous_500_0.5_4_4_0.1_0.02_3_0.8',
  # 'continuous_500_0.5_4_4_0.1_0.02_3_0.9',
  'continuous_500_0.5_4_4_0.1_0.02_4_0.1',
  'continuous_500_0.5_4_4_0.1_0.02_4_0.2',
  'continuous_500_0.5_4_4_0.1_0.02_4_0.3',
  'continuous_500_0.5_4_4_0.1_0.02_4_0.4',
  'continuous_500_0.5_4_4_0.1_0.02_4_0.5',
  'continuous_500_0.5_4_4_0.1_0.02_4_0.6'
  #,
  # 'continuous_500_0.5_4_4_0.1_0.02_4_0.7',
  # 'continuous_500_0.5_4_4_0.1_0.02_4_0.8',
  # 'continuous_500_0.5_4_4_0.1_0.02_4_0.9'
)

dat <- NULL
for (i in 1: length(rows)){
dat <- rbind(dat,
       c(rows[i], get_result(rows[i])))
}
write.csv(dat, file = "ResultCts_correlated_4_4.csv")
dat <- read.csv("ResultCts_correlated_4_4.csv")

dat  <-  dat[order(dat$Correlated.covariates,  dat$Correlation),]  #sort  dat  by  the
variables we need

dat2<-dat
dat2$s_Prefer_Full_critical2            <-            25*(dat2$s_Prefer_Full_critical-
0.1)+dat2$Correlated.covariates
dat2$i_Prefer_Full_critical2            <-            25*(dat2$i_Prefer_Full_critical-
0.1)+dat2$Correlated.covariates
ml <- max(dat2$Correlated.covariates)

pdf(paste0("P:\\Thesis\\R_plots\\", "3.2.5-corr.pdf"),
    width=8, height=5.5)
par(mar=c(5, 4, 1, 6) + 0.1) #extra room for axis
plot(c(min(dat2$Correlation), max(dat2$Correlation)), c(1.5, 4.5),type="n" ,
     xlab="Correlation between Covariates",
     ylab="Number of Correlated Covariates",
```

```
     #main="Stratification Models",
     xaxt='n',yaxt='n')
axis(1, at = seq(min(dat2$Correlation), max(dat2$Correlation), by=0.1), labels = T)
axis(2, at = c( 2:ml), labels=T)
# lables on right side
mtext("Overall Type I error",side=4,line=4)
# 95% CI is +-0.006 instead of +-0.01
axis(4,
     at=rep(c(2,3,4),each=3)+rep(c(-0.25*0.6, 0, 0.25*0.6), ml-1),
     labels=rep (c(0.094, 0.1, 0.106),ml-1),
     las=1, tck=-0.01)
for (i in 2:ml){
  points(x=dat2[which(dat2$Correlated.covariates==i),]$Correlation,
         y=dat2[which(dat2$Correlated.covariates==i),]$s_Prefer_Full_critical2,
         type="o", lwd=1, col=colA[i], pch=19)
  points(x=dat2[which(dat2$Correlated.covariates==i),]$Correlation,
         y=dat2[which(dat2$Correlated.covariates==i),]$i_Prefer_Full_critical2,
         type="o", lwd=1, col=colB[i], pch=2, lty=2)
  abline (h=i, col="Gray23", lwd=1, lty=4)
  abline (h=i+0.25*0.6, col="Gray23", lwd=1, lty=3)
  abline (h=i-0.25*0.6, col="Gray23", lwd=1, lty=3)
}
legend("topright",pch=c(19,2), lty=c(1,2),lwd=1,cex=0.9,
       c("Stratification Models", "Interaction Models"))
dev.off()
```

A.5 Run_Sheet.xlsx

Below is an example of the structure of Run_Sheet.xlsx in organizing the simulations.

runthisrow	binary	n	prev	k	ell	alphaT	alphaF	kc	corr	nNull	nSim	cluster	folder_name
1	binary	500	0.5	4	2	0.1	0.02	0	0	1000	200	5	binary_500_0.5_4_2_0.1_0.02_0_0
2	binary	500	0.5	4	2	0.1	0.05	0	0	1000	200	5	binary_500_0.5_4_2_0.1_0.05_0_0
3	binary	500	0.5	4	2	0.1	0.08	0	0	1000	200	5	binary_500_0.5_4_2_0.1_0.08_0_0
4	binary	500	0.5	4	2	0.1	0.02	0	0	1000	200	5	binary_500_0.5_4_2_0.1_0.02_0_0
5	binary	500	0.5	4	3	0.1	0.02	0	0	1000	200	5	binary_500_0.5_4_3_0.1_0.02_0_0
6	binary	500	0.5	4	4	0.1	0.02	0	0	1000	200	5	binary_500_0.5_4_4_0.1_0.02_0_0
7	binary	500	0.5	3	2	0.1	0.02	0	0	1000	200	5	binary_500_0.5_3_2_0.1_0.02_0_0
8	binary	500	0.5	4	2	0.1	0.02	0	0	1000	200	5	binary_500_0.5_4_2_0.1_0.02_0_0
9	binary	500	0.5	5	2	0.1	0.02	0	0	200	200	25	binary_500_0.5_5_2_0.1_0.02_0_0
10	binary	500	0.5	6	2	0.1	0.02	0	0	200	200	25	binary_500_0.5_6_2_0.1_0.02_0_0
11	binary	500	0.5	7	2	0.1	0.02	0	0	200	200	25	binary_500_0.5_7_2_0.1_0.02_0_0
…	…												
611	continuous	500	0.5	4	4	0.1	0.02	0	0	200	200	25	continuous_500_0.5_4_4_0.1_0.02_0_0
612	continuous	500	0.5	5	4	0.1	0.02	0	0	200	200	25	continuous_500_0.5_5_4_0.1_0.02_0_0
613	continuous	500	0.5	6	4	0.1	0.02	0	0	200	200	25	continuous_500_0.5_6_4_0.1_0.02_0_0
614	continuous	500	0.5	7	4	0.1	0.02	0	0	200	200	25	continuous_500_0.5_7_4_0.1_0.02_0_0
615	continuous	500	0.5	8	4	0.1	0.02	0	0	200	200	25	continuous_500_0.5_8_4_0.1_0.02_0_0
616	continuous	500	0.5	9	4	0.1	0.02	0	0	200	200	25	continuous_500_0.5_9_4_0.1_0.02_0_0
…	…												

A.6 Submitting jobs on cluster

A shell script names bash.sh and contains:

```
#!/bin/bash
R CMD BATCH  "--args $row $cluster" R_run.R B_${row}_${cluster}.Rout
```

On Bayes server, use qsub for submission and passing arguments to shell script:

```
qsub -v row=1 -v cluster=11 -q  normal.q -cwd bash.sh
qsub -v row=1 -v cluster=12 -q  normal.q -cwd bash.sh
qsub -v row=1 -v cluster=13 -q  normal.q -cwd bash.sh
qsub -v row=1 -v cluster=14 -q  normal.q -cwd bash.sh
qsub -v row=1 -v cluster=15 -q  normal.q -cwd bash.sh
```

Appendix B.1

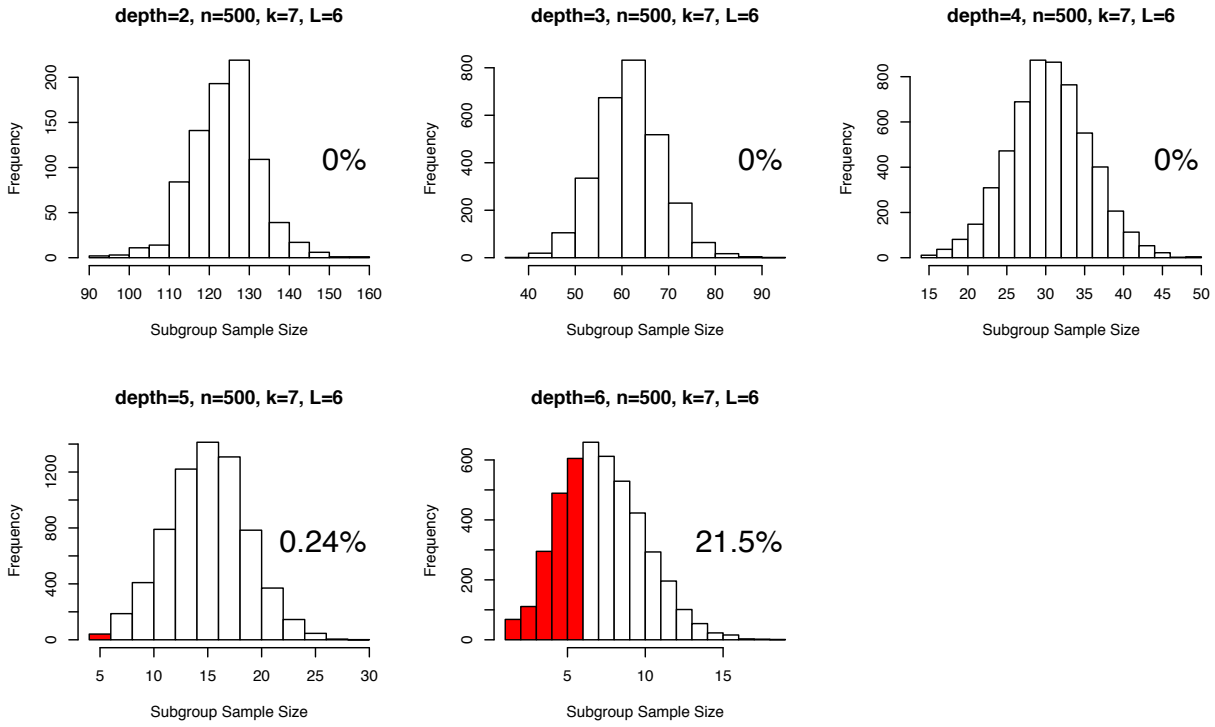


Figure B.1.1. Distribution of the sample size on each depth. The full population is 500 with 7 covariates and allow up to 6 covariates in SHAPES subgroup. Only the subgroups with sample size less than 250 are plotted in this histogram.

Appendix B.2

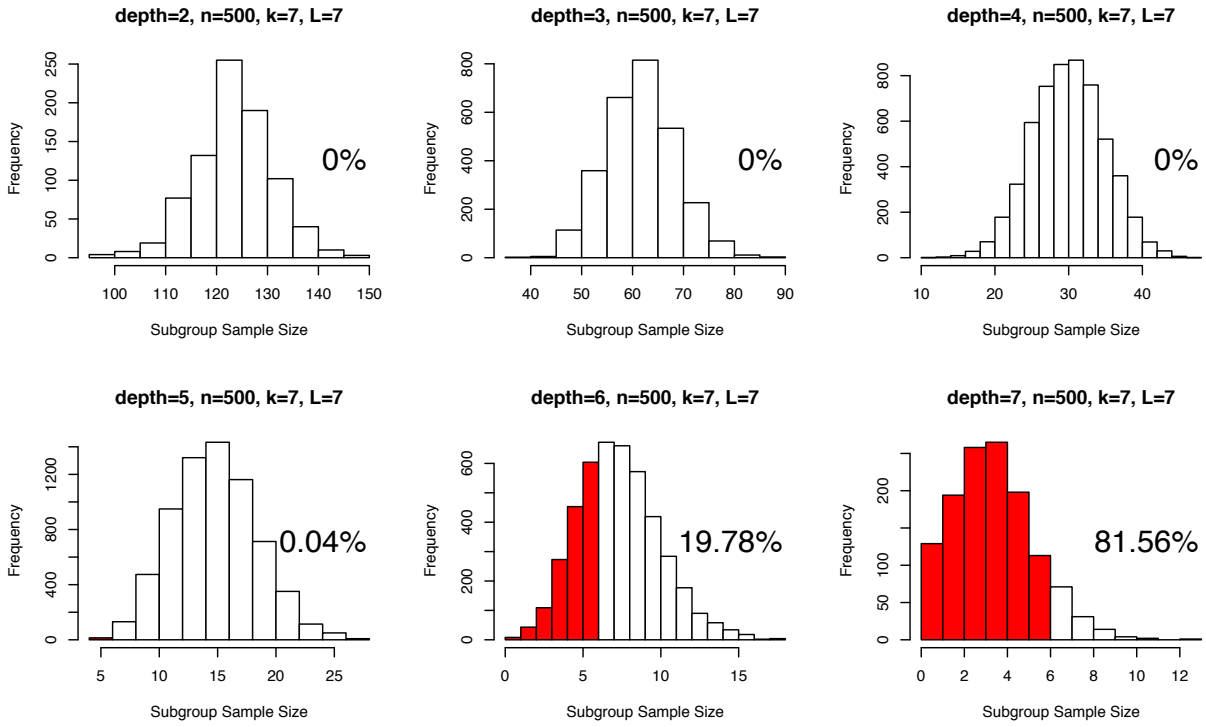


Figure B.2.1. Distribution of the sample size on each depth. The full population is 500 with 7 covariates and allow up to 7 covariates in SHAPES subgroup. Only the subgroups with sample size less than 250 are plotted in this histogram.