RANDOMIZATION TESTS FOR REGRESSION MODELS IN CLINICAL TRIALS

by

Parwen Parhat

A Dissertation
Submitted to the
Graduate Faculty
of
George Mason University
in Partial Fulfillment of
The Requirements for the Degree
of
Doctor of Philosophy
Statistical Science

Committee:

[illegible]	Dr. William F. Rosenberger, Dissertation Director
[illegible]	Dr. Guoqing Diao, Committee Member
[illegible]	Dr. Jacqueline Hughes-Oliver, Committee Member
[illegible]	Dr. Liansheng Larry Tang, Committee Member
[illegible]	Dr. Pearl Wang, Committee Member
[illegible]	Dr. William F. Rosenberger, Department Chair
[illegible]	Dr. Kenneth S. Ball, Dean, The Volgenau School of Engineering

Date: 24 April 13

Spring Semester 2013
George Mason University
Fairfax, VA

Randomization Tests for Regression Models in Clinical Trials

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy at George Mason University

By

Parwen Parhat
Master of Arts
University of Houston, 2008
Bachelor of Science, Bachelor of Arts
Peking University, 2006

Director: Dr. William F. Rosenberger, Professor
Department of Statistics

Spring Semester 2013
George Mason University
Fairfax, VA

Dedication

To my parents, Parhat Osman and Gülnar Wulam.

Acknowledgments

First and foremost, I would like to express my deep appreciation to my advisor Dr. William F. Rosenberger, for his sage guidance throughout the completion of the thesis. I am forever grateful for his patient support, encouragement and generous sharing of his valuable time to help me make progress. I feel very fortunate to have had the opportunity to work with him.

I am sincerely grateful to Dr. Jacqueline Hughes-Oliver for mentoring me in the practical research of statistics. I thank her clearly guiding me while giving me space to explore and learn in the past two years. I also thank her for serving as my committee member, and for giving me comments and insights to the thesis. I give my gratitude to committee member Dr. Guoqing Diao, for spending many hours to listen my research and offering me resources and suggestions regarding data analysis, coding and simulation. I thank committee members Drs. Larry Tang and Pearl Wang for coordinating their time and giving me support.

I would like to thank Ms. Elizabeth Quigley for her kind and patient administrative assistance. I also thank my colleague Daniel Saxton for proofreading and correcting inaccuracies in the text.

My gratitude goes to Drs. Bent Sorensen and Steven Craig in University of Houston for listening to me and encouraging me to pursue the academic degree where I really have interest and believing in my potential to success.

Very special thanks to my husband Imam Xierali, for giving me constant support, confidence and understanding during every step of the thesis. I also would like to thank my daughters, Ayguzel and Ayperi, for always cheering me up and giving me unlimited love and joy during the most difficult time of my life.

I owe many thanks to my beloved mother Gülnar Wulam, father Parhat Osman, and brother Paruk Parhat. They made innumerable sacrifices to help me complete the degree during the past several years. I appreciate everything they have done for me.

Table of Contents

List of Tables

List of Figures

Abstract

RANDOMIZATION TESTS FOR REGRESSION MODELS IN CLINICAL TRIALS
Parwen Parhat, PhD
George Mason University, 2013
Dissertation Director: Dr. William F. Rosenberger

In this dissertation, we apply randomization tests in the context of regression models to detect treatment effects for clinical trial data. This application allows us to compute randomization tests for a wide variety of outcomes, including covariate adjusted treatment effects, general response functions, survival data, and longitudinal data. We used score residuals as the outcome of the randomization test under the generalized linear model. For the proportional hazards and accelerated linear model, we used the martingales residuals as the outcome of the randomization test. For the generalized linear mixed model, to detect whether there is a time-varying treatment effect, we compute the predicted random slope from the regression as the outcome of the randomization test.

We examine the utility of randomization tests and how they perform compared to the traditional methods. Simulations are applied to compare the size and power of randomization tests and population tests in different scenarios. Results show randomization tests perform better than the traditional test in the following scenarios: when there are necessary covariates omitted in the generalized linear model, when there is misspecification for the distribution of survival time, when there is misspecification of the distribution of the response for longitudinal data, and when there are outliers for the random effects in longitudinal data.

As an important contribution of the thesis, a series of SAS macros are developed to provide easy computation of randomization tests. These macros calculate the p-value of a randomization test directly from the data or under certain regression models for various types of data, given specified randomization procedure and the type of reference set. The macros provide computing convenience for statisticians and clinical trialists in the pharmaceutical industries.

Chapter 1: Introduction

1.1 Background

The randomized clinical trial has become a gold standard as it is a reliable and efficient method to test whether a new treatment is effective (Cook and DeMets, 2008). There are two main reasons for applying randomization in clinical trials. First, it ensures patients are allocated randomly to treatments, so the possibility of selection bias is minimized when compared to other types of controlled trials (Friedman, 1975). Second, it provides a way to test treatment effects based on randomization itself, rather than random sampling which does not occur in clinical trials (Kempthorne, 1977).

Randomization provides the basis for the randomization test. A randomization test is an approach that does not depend on distributional assumptions: it evaluates whether there is a treatment effect by comparing the test value under the null hypothesis to values computed from other possible randomizations. The main focus of this thesis is the application of randomization tests in clinical trials, where the primary outcome is based on a regression model.

To identify whether there is a treatment effect, the traditional inferential approach in statistics is based on assuming a population model. Under the population model, we assume there is a target patient population with a certain probability distribution on the outcome. From that population, we randomly select a sample and assume it is representative of the population (Chow and Liu, 2004). Then we make inferences about the population by performing statistical tests on the sample.

However, clinical trials do not fit into the inferential scheme based on a population model. First, the target population distribution is usually unknown, and second, patients are not randomly selected for most experimental trials. Many studies set different criteria for selecting patients (Chow and Liu, 2004), and therefore it is difficult to derive a valid statistical inference based on a population model.

Sometimes, a population model is invoked to allow standard statistical inference. This assumes patients are first selected from a large unknown population with some sampling strategy, and then are randomly assigned into treatment groups. But there is one critical assumption that is difficult to be tested—the homogeneity of characteristics of patients (Lachin, Matts and Wei, 1988). In fact, it is unnecessary to employ the invoked population model if we use randomization as a basis for statistical inference.

The randomization model is different from the population model in that it aims at evaluating treatment effects without distributional assumptions. There are three main properties of a randomization model in clinical trials:

1. Patients are not randomly selected as a sample and patient responses are considered as fixed under the null hypothesis of no treatment effect but not drawn from a population.

2. The characteristics of patients do not have to be homogeneous.

3. Patients are assigned to treatment groups based on a certain randomization procedure.

A randomization test can be performed in the context of the randomization model. The key idea behind the randomization test is that the difference between treatment groups depends only on the way patients are assigned to each treatment group. The null hypothesis of a randomization test is that patient responses would have been the same no matter which treatment had been assigned. In contrast to the population model, whose reference distribution is from a theoretical probability distribution, the reference distribution of a randomization model is a set that includes all possible randomization sequences. One can obtain the p-value by summing probabilities that the test statistic from the reference distribution is at least as extreme as the observed value.

In this thesis we intend to use the linear rank statistic which is a broad family of statistics for randomization tests (Lehmann, 2006). Define Y_j to be the response variable for the jth patient from a sample where $j = 1, ...n$, assume patients will be assigned into two treatment groups A and B. Let a_{jn} be a score function of Y_j for the jth patient, and T_j be a treatment indicator, whose value is 1 if patient is assigned to treatment A, and 0 if patient is assigned to treatment B. Then the test statistic is given by:

$$S = \sum_{j=1}^{n} (a_{jn} - \bar{a}) T_j. \tag{1.1}$$

Based on different types of data, one might choose different score functions. In the case of binary data, Fisher's exact test could be performed (Gibbons and Chakraborti, 2010); in the case of survival data, Savage scores can be used to calculate the log-rank test statistic; Kalbfleisch and Prentice (2002) provided the exact form of a_{jn} including both uncensored and censored cases.

In this thesis, the following linear rank score functions will be used to calculate test statistics in (1.1):

- *Wilcoxon Scores*: Let R_j be the rank of jth patient's outcome. Then the score function is

$$a_{jn} = R_j. \tag{1.2}$$

This score generates the rank sum statistic for the linear rank test, which is analogous to the Mann-Whitney-Wilcoxon test, which is one of the most powerful tests for detecting location shifts between two groups. This score can be used for many types of data.

- *Savage Scores*: For survival data, there are two forms of the score function depending on whether or not there is censoring (Rosenberger and Lachin, 2002): If there is no censoring, survival times are ordered corresponding to the treatment indicator. The score function is given by:

  $$a_{jn} = 1 - \sum_{k=n+1-j}^{n} \frac{1}{k}. \tag{1.3}$$

  If there is censoring, denote R_m to be the number of patients at risk prior to event time m,

  $$a_{jn} = \begin{cases} 1 - \sum_{m=1}^{j} \frac{1}{R_m} & \text{If not censored;} \\ -\sum_{m=1}^{j} \frac{1}{R_m} & \text{If censored.} \end{cases} \tag{1.4}$$

  The log-rank test based on Savage scores is robust when testing the scale difference between two groups for the exponential distribution (Rosenberger and Lachin, 2002).

- *van der Waerden Scores*: these scores are mostly used for continuous data. It transforms ranks into quantiles of a standard normal distribution. It is computed by

$$a_{jn} = \Phi^{-1}\left(\frac{R_j}{n+1}\right). \tag{1.5}$$

From (1.1), we can conclude that the randomization test statistic depends on the randomization distribution of treatment assignment T, and this distribution is based on the randomization procedure.

Following the previous notation, let p_j be the probability of assigning jth patient to treatment group A, given by

$$p_j = P(T_j = 1|T_1, ..., T_{j-1}). \tag{1.6}$$

In the case of complete randomization, we have $T_1, T_2, ..., T_n$ following a Bernoulli distribution, so that

$$p_j = P(T_j = 1|T_1, ..., T_{j-1}) = \frac{1}{2}. \tag{1.7}$$

In clinical trials, complete randomization is seldom employed, as it has a higher probability of treatment imbalance when the sample size is small or trials are performed over a long period of time with continuous assignments (Chow and Liu, 2004). Restricted randomization procedures are used in the interest of balance. In this project, we review the *permuted block design* (PBD), *biased coin design* (BCD) and *generalized biased coin design* (GBCD), and intend to apply first two types of procedures to regression models.

The PBD is an approach where the entire sample is divided into several blocks and patients are assigned into each block. Usually for balancing purposes, the *random allocation rule* (RAR) or *truncated binomial design* (TBD) is used within a block.

Under the random allocation rule, patients are randomly selected as a subset whose size is equal to the block size from a total known sample size. Half the subset is randomly assigned to treatment group A, and the remainder to group B, and balance is achieved at the end of assignment to the block. Define S to be the number of blocks, and $s = n/S$ to be the number of patients in ith block ($i = 1, ...S$); let $N_A(j)$ be the number of patients assigned to group A after jth assignment, and p_{ij} be the probability of assigning the jth patient at the ith block to treatment A. If we apply the RAR within each block, the probability distribution of T within each block is given as (Rosenberger and Lachin, 2002):

$$p_{ij} = \begin{cases} \frac{1}{2} & j = 1; \\ \frac{\frac{s}{2} - (N_A(j-1) - (i-1)\frac{s}{2})}{s - (j - (i-1)s)} & 1 < j \leq n,\ 1 \leq i \leq S. \end{cases} \tag{1.8}$$

Under the TBD, within each block, patients are randomly selected with probability 1/2 until one of the group sizes equals half the block size. The probability distribution of T within each block is given by (Rosenberger and Lachin, 2002):

$$p_{ij} = \begin{cases} 0 & N_A(j-1) - (i-1)\frac{s}{2} = \frac{s}{2}; \\ \frac{1}{2} & \max\{(N_A(j) - (i-1)\frac{s}{2}), (N_B(j) - (i-1)\frac{s}{2})\} < \frac{s}{2}; \\ 1 & (N_B(j-1) - (i-1)\frac{s}{2}) = \frac{s}{2}. \end{cases} \tag{1.9}$$

The BCD was proposed by Efron (Efron, 1971). Let D_j be the difference between the number of patients of treatment A and B after j patients are allocated, $D_j = N_A(j) - N_B(j)$. Let p be the constant and predetermined probability of balance, usually greater than 1/2. Then we have

$$p_j = \begin{cases} 1-p & D_{j-1} > 0; \\ \frac{1}{2} & D_{j-1} = 0; \\ p & D_{j-1} < 0. \end{cases} \tag{1.10}$$

Smith (1984) introduced a form of the BCD from a mathematical perspective, which can be viewed as a class of generalized biased coin designs (GBCD). Let ρ be a positive parameter which controls the sensitivity of p_j to allocation imbalances. Then p_j is considered as a function of $N_A(j-1)$, $N_B(j-1)$ and ρ :

$$p_j = \frac{N_B(j-1)^\rho}{N_A(j-1)^\rho + N_B(j-1)^\rho}. \tag{1.11}$$

If ρ is 0, the procedure reduces to complete randomization. As ρ increases, the probability of achieving balance increases and the variability of the procedure decreases.

1.2 Randomization Tests

Application of randomization tests for the PBD is straightforward (Rosenberger and Lachin, 2002); however, statistical inference based on the BCD has not been developed until recently because the exact randomization distribution of the BCD was difficult to obtain. Efron (1971) derived the asymptotic distribution for the BCD; Markyaran and Rosenberger (2010) derived the exact distribution of BCD; Plamadeala and Rosenbeger (2012) provided techniques to compute conditional randomization tests under the BCD by using conditional probability theory.

Here we detail two types of tests: based on the difference of reference sets, there are *unconditional* and *conditional* randomization tests. First, assume there are n patients to be assigned into two treatment groups A and B, where n_A is the number of patients receiving treatment A. Then the unconditional reference set given by Ω_{un} contains all possible 2^n randomization sequences. Let S be a well defined statistic and L be the defined randomized sequence. Let S_{obs} be the observed statistic and S_l be the statistic for the lth randomized sequence from the reference set where $l = 1, ...2^n$. Then the p-value (Rosenberger and Lachin, 2002) for the unconditional randomization test is

$$p_{un} = \sum_{l=1}^{2^n} P(|S_l| \geqslant |S_{obs}|) = \sum_{l=1}^{2^n} I(|S_l| \geqslant |S_{obs}|)P(L = l). \tag{1.12}$$

Similarly, define Ω_{con} to be a conditional reference set given $N_A(n) = n_A$, then the p-value for the conditional randomization test is

$$p_{con} = \sum_{l=1}^{\binom{n}{n_A}} P(|S_l| \geqslant |S_{obs}||N_A(n) = n_A) = \sum_{l=1}^{\binom{n}{n_A}} I(|S_l| \geqslant |S_{obs}|)P(L = l|N_A(n) = n_A). \tag{1.13}$$

To compute the unconditional and conditional randomization tests, we need to know the probabilities of assigning patients for certain designs given conditional or unconditional reference sets. Both of them depend on the distribution of $N_A(n)$.

For the biased coin design, Markaryan and Rosenberger (2010) developed the distribution of $N_A(n)$ to compute the unconditional test, and Plamadeala and Rosenberger (2012) provided the algorithm for calculating probabilities of assigning patients for the conditional test based on the conditional distribution of $N_A(n)$ given previous allocations and n_A.

For the unconditional case, for $BCD(p)$, $q = 1 - p$, we have

$$P(N_A(n) = n_A) = \begin{cases} 0.5p^{n_A} \sum_{l=0}^{n_A} \frac{n - n_A - l}{n - n_A + l} \binom{n - n_A + l}{l} q^{n-2n_A+l-1} & 0 \le n_A < \frac{n}{2}; \\ p^{n_A} \sum_{l=0}^{n_A - 1} \frac{n - 2l}{n + 2l} \binom{n_A + l}{l} q^l & n_A = \frac{n}{2}; \\ 0.5p^{n-n_A} \sum_{l=0}^{n-n_A} \frac{n_A - l}{n_A + l} \binom{n_A + l}{l} q^{2n_A-n+l-1} & \frac{n}{2} < n_A \le n. \end{cases} \tag{1.14}$$

Under the conditional case (Plamadeala and Rosenberger, 2012), let ϕ_j be the conditional probability of assigning the $j + 1$th patient to treatment group A given $N_A(n) = n_A$. Here $p_j(m_{j-1})$ is the probability from (1.10) for the jth patient when the number of patients assigned to treatment group A is m_{j-1}. The allocation probability of the jth patient will be

$$\phi_j = \begin{cases} p_j(m_{j-1}) \frac{P(N_A(n)=n_A|N_A(j)=m_{j-1}+1)}{P(N_A(n)=n_A|N_A(j)=m_{j-1})} & 1 < j \le n; \\ \\ p_j \frac{P(N_A(n)=n_A|T_j=1)}{P(N_A(n)=n_A)} & j = 1. \end{cases} \tag{1.15}$$

The exact form of $P(N_A(n) = n_A|N_A(j) = m_j)$ for the BCD is provided by Plamadeala and Rosenberger (2012):

1. If $1 \leq j < n$ and $0 \leq m_j < j/2$, $P(N_A(n) = n_A | N_A(j) = m_j)$ is given by

$$\begin{cases} \binom{n-j}{n_A-m_j} p^{n_A-m_j} q^{n-j-n_A+m_j}, & m_j \leq n_A < j - m_j; \\ 0.5 p^{n_A-m_j} \displaystyle\sum_{l=0}^{n_A+m_j-j} \frac{n-n_A-m_j-l}{n-n_A-m_j+l} \binom{n-n_A-m_j+l}{l} q^{n-2n_A-1+l} & \\ +p^{n_A-m_j} q^{n-j-nA+m_j} (\binom{n-j}{n_A-m_j} - \binom{n-j}{n_A-j+m_j}), & j - m_j \leq n_A < n/2; \\ p^{n_A-m_j} \displaystyle\sum_{l=0}^{n-j-n_A+m_j} \frac{n_A-m_j-l}{n_A-m_j+l} \binom{n_A-m_j+l}{l} q^l, & n_A = n/2; \\ 0.5 p^{n-n_A-m_j} \displaystyle\sum_{l=0}^{n-j-n_A+m_j} \frac{n_A-m_j-l}{n_A-m_j+l} \binom{n_A-m_j+l}{l} q^{2n_A-n-1+l}, & \\ & n/2 < n_A \leq n - j + m_j. \end{cases}$$

2. When $0 \leq j < n$ and $m_j = j/2$,

$$P(N_A(n) = n_A | N_A(j) = m_j) = P(N_A(n-j) = n_A - m_j), \ m_j \leq n_A \leq n - j + m_j.$$

3. When $1 \le j < n$ and $j/2 < m_j \le j$, $P(N_A(n) = n_A | N_A(j) = m_j)$ is

$$\begin{cases} 0.5p^{n_A+m_j-j} \sum_{l=0}^{n_A-m_j} \frac{n-j-n_A+m_j-l}{n-j-n_A+m_j+l} \binom{n-j-n_A+m_j+l}{l} q^{n-2n_A-1+l}, & \\ & m_j \le n_A < n/2; \\ p^{n-j-n_A+m_j} \sum_{l=0}^{n_A-m_j} \frac{n-j-n_A+m_j-l}{n-j-n_A+m_j+l} \binom{n-j-n_A+m_j+l}{l} q^l, & n_A = n/2; \\ 0.5p^{n-j-n_A+m_j} \sum_{l=0}^{n-n_A-m_j} \frac{n_A+m_j-j-l}{n_A+m_j-j+l} \binom{n_A+m_j-j+l}{l} q^{2n_A-n-1+l} & \\ +p^{n-j-n_A+m_j} q^{n_A-m_j} (\binom{n-j}{n_A-m_j} - \binom{n-j}{n_A-j+m_j}), & n/2 < n_A \le n-m_j; \\ \binom{n-j}{n_A-m_j} p^{n-j-n_A+m_j} q^{n_A-m_j}, & n-m_j < n_A \le n-j+m_j. \end{cases}$$

1.3 Randomization Tests Under Regression Models

In many cases, there are also other factors affecting the treatment outcome of clinical trials such as age and gender of patients. One way to analyze treatment effects given those factors is to apply generalized linear regression (McCullagh and Nelder, 1989) while controlling those factors as covariates. Under the population model, the traditional regression will be applied to estimate the effect of those prognostic factors as well as the treatment effect simultaneously.

Let Y be the outcome variable, X the covariate, and T the treatment indicator which is independent of X; and let $h(\cdot)$ be a known function, and η be the parameter representing the linear relationship. The Generalized Linear Model (GLM) is built as follows:

$$\eta = \mu + T\alpha + X\beta; \tag{1.16}$$

$$E(Y|X,T) = h(\eta).$$

Under the population model, the null hypothesis of no treatment effect is $\alpha = 0$. In the case of linear regression, where $h(\eta) = \eta$, the estimate of α will be unbiased whether X is included or not. But for certain non-linear regressions (e.g., Poisson regression with count data), the estimator of α based on the traditional population model may lead to supranominal test size when X is omitted because the estimated variance of the score function for α is biased. Gail, Tan and Piantadosi (1988) proposed to use the randomization distribution of score residuals to estimate the variance; they employed three ways to examine the effect of model misspecification to the population test and the proposed test: the size of the population tests under the false population model, variance estimate ratios between the true variance and the false variance, and the relative efficiency between the proposed test under the false model and the test under the true model. The proposed test is considered to be efficient most of the time.

In this thesis we extend Gail Tan and Piantadosi's strategy (1988) of using randomization distributions of score residuals. Instead of computing the scores under a population model, we apply the randomization test regarding score residuals as fixed outcomes.

The GLM is mainly used for analyzing data with independent observations for fixed effects, and it is not suitable for correlated data. In clinical trials often there are repeated measures for patients based on different times. Random effects could be added to the GLM to account for the correlation; this type of model is called the generalized linear mixed model (GLMM) which includes both fixed effects and random effects .

Consider time as a variable in clinical trials data, then the GLMM is extended from (1.16). Let τ be a time effect factor which is independent of X and T, and let b_0 and b_1 be the random intercept and random slope respectively. The linear predictor under the GLMM has the following form

$$\eta = \mu + T\alpha + X\beta_1 + \tau\beta_2 + T\tau\beta_3 + b_0 + \tau b_1. \tag{1.17}$$

Here β_3 is the time-varying treatment effect indicating whether the treatment effect changes over time for each patient. Under the traditional model, the null hypothesis is $\beta_3 = 0$ and one can estimate β_3 by maximizing the marginal likelihood function. However, from Gail, Tan and Piantadosi's (1988) conclusion, the result of population tests may not be reliable if the model is misspecified as the estimate of the variance of β_3 may not be valid. Our strategy is to apply a randomization test to the estimated random slope $\widehat{b_1}$ from (1.17).

Due to computational difficulties, the randomization test technique has not been widely used and it was not systematically developed compared to traditional methods (Ludbrook and Dudley, 1998). We seek to provide a user-friendly SAS macro package to provide computing convenience for statisticians in pharmaceutical industries and government agencies.

1.4 Structure of the Thesis

We apply randomization tests in the context of the generalized linear mixed model to estimate time-varying treatment effect. The novel methodology develops a test based on the random effect predictors from the regression model. We also seek to develop SAS macro series to incorporate randomization tests, both unconditional and conditional, and incorporate the randomization test via generalized linear models and mixed models as well.

This thesis is organized as follows. In Chapter 2, we detail the algorithm of randomization test under GLM using Monte Carlo simulation and compare the size of randomization tests and population tests under different types of data when the original regression model is misidentified. In Chapter 3, we describe a new method which estimates time-varying treatment effects using a randomization test under the GLMM. In Chapter 4, we examine the operating characteristics of the randomization test for longitudinal data by comparing the size and power to that of the population test. Chapter 5 presents a series of SAS macros which which provides easy computation of randomization tests. We draw conclusions in Chapter 6.

Chapter 2: Randomization Tests Under the Generalized Linear Model

This chapter describes randomization tests in the context of the generalized linear model (GLM). The first two sections provide details for computing p-values of the test by random sampling from the unconditional and conditional reference sets. In the third section we compare the size of the randomization test and population test when a necessary covariate is omitted in the regression model. We provide an application of randomization tests to survival data in the fourth section, where we also investigate the power and type I error rate when the regression model has omitted covariate, the survival time has outliers and the assumption of proportional hazard violated.

2.1 Unconditional Tests Under the Generalized Linear Model

Following the notation from Chapter 1 for the GLM, let X be a covariate for which $E(X) = 0$, an let $T = 1$ if the patient is assigned to treatment group A, -1 if the patient is assigned to treatment group B. Define $\theta(\cdot)$ to be the canonical parameter of an exponential family with first derivative $\theta'(\cdot)$, ϕ as a scale parameter, $d(\cdot)$ and $b(\cdot)$ as functions identifying the distribution of Y, and $R(\cdot)$ is a function that does not contain any unknown parameters. The log likelihood of Y conditional on X and

T is as follows:

$$\ell = (d(\phi))^{-1}[Y\theta(\eta) - b(\theta(\eta)) + R(Y)] + c(Y, \phi). \tag{2.1}$$

The regression model is thus $E(Y|X,T) = h(\eta) = h(\mu + T\alpha + X\beta)$, where $h(\eta) = b'(\theta(\eta))$, where $b'(\cdot)$ is the first derivative. Under the null hypothesis H_0 of the randomization test, there is no difference between treatment groups A and B. Gail, Tan and Piantadosi (1988) proposed to construct randomization tests based on the randomization distribution of the residuals r from the fitted model obtained from maximum likelihood estimation.

Unlike the regular residuals of simple linear model, where $r = y - \widehat{y}$, the residuals under the GLM have different forms such as response residuals, Pearson residuals and deviance residuals (McCullagh and Nelder, 1989, pp. (37-42)). The residual we use here is the score residual (Gail, Tan and Piantadosi, 1988). Suppose ϕ is known, let $\widehat{\eta}_0 = \widehat{\mu}_0 + X\widehat{\beta}_0$, where $\widehat{\mu}_0$ and $\widehat{\beta}_0$ represent maximum likelihood estimator of μ and β under the null hypothesis $H_0 : \alpha = 0$. The score function then is

$$\begin{aligned} U &= \frac{\partial \ell}{\partial \alpha} \\ &= (d(\phi))^{-1} \sum \left(Y \frac{\partial \theta}{\partial \eta} \frac{\partial \eta}{\partial \alpha} - \frac{\partial b}{\partial \theta} \frac{\partial \theta}{\partial \eta} \left. \frac{\partial \eta}{\partial \alpha} \right| \eta = \widehat{\eta}_0 \right) \\ &= (d(\phi))^{-1} \sum T\theta'(\widehat{\eta}_0)(Y - h(\widehat{\eta}_0)). \end{aligned}$$

Under the canonical link, $\theta'(\eta) = 1$.

The residuals r have the following form:

$$r = (d(\phi))^{-1}\theta'(\widehat{\eta}_0)(Y - b'(\theta(\widehat{\eta}_0))) = (d(\phi))^{-1}\theta'(\widehat{\eta}_0)(Y - h(\widehat{\eta}_0)).$$

As T and X are independent, under complete randomization, $E(T) = 0$ and $var(T) = 1$. Given the condition that $E(\partial\ell^2/\partial\alpha\partial\mu) = E(\partial\ell^2/\partial\alpha\partial\beta) = E(\partial\ell^2/\partial\mu\partial\beta) = 0$ for each element of β, the variance of the score function can be computed under the randomization of r and has the form $var(n^{-1/2}U) = n^{-1}\sum r^2$. The test based on this variance is efficient even when the covariates are omitted (Gail, Tan and Piantadosi, 1988). However, the variance is difficult to compute under the restricted randomization procedures which are often applied in clinical trials; instead, we directly perform the randomization test on the residuals. In this case, the treatment effect will be assessed strictly according to the randomization procedure.

For the unconditional test, the p-value is computed by comparing test statistics using all possible randomization assignments from the unconditional reference set to the observed test statistic. A Monte Carlo approach can be employed to obtain an unbiased estimator of the p-value (Zhang and Rosenberger, 2008). Let L be the number of randomization sequences generated using Monte Carlo simulation, where $a_{jn}(r)$ is a rank score based on residuals and observed treatment assignments, $S_{obs} = \sum_{j=1}^{n}(a_{jn}(r) - \bar{a}(r))T_j$, $j = 1, ...n$; let S_l be the test statistic of the residuals given the lth randomization sequence, $l = 1, ...L$. Then given residuals from the GLM, the p-value of the unconditional randomization test is

$$\widehat{p}_{un} \approx \frac{\sum\limits_{l=1}^{L} I(|S_l| \geqslant |S_{obs}|)}{L} \tag{2.2}$$

As the indicator function $I(|S_l| \geqslant |S_{obs}|)$ has a Bernoulli distribution with mean p_{un}, $\widehat{p}_{un}$ is unbiased with mean squared error (MSE) $\widehat{p}_{un}(1-\widehat{p}_{un})/L$. The required number of randomization sequences L is at least 2500 to ensure that the MSE less than 0.0001

(Zhang and Rosenberger, 2008).

2.2 Conditional Tests Under the Generalized Linear Model

Plamadeala and Rosenberger (2012) describe a method for computation of conditional randomization tests that reduces the computational complexity of the problem by generating Monte Carlo sequences directly from the conditional reference set. For the conditional test, n_A is observed from the data and $N_A(k)$ is the number of patients assigned to treatment A at the kth randomized sequence, $k = 1, ...K$. Here K has to be large enough so that we can choose all possible sequences given $N_A(k) = n_A$. Then the conditional p-value (Zhang and Rosenberger, 2008) is estimated as

$$\widehat{p}_{con} \approx \frac{\sum_{k=1}^{K} I(|S_k| \geqslant |S_{obs}|, N_A(k) = n_A)}{\sum_{k=1}^{K} I(N_A(k) = n_A)}.$$

The required number of sequences K depends on the randomization procedure and sample size; using Monte Carlo sampling from the unconditional reference set would be computationally infeasible if the sample size is large and the treatment assignments are highly imbalanced (Plamadeala and Rosenberger, 2012).

If we generate sequences directly from the conditional reference set, let L_{con} be the number of randomization sequences required, the p-value of the conditional test can be estimated as

$$\widehat{p}_{con} \approx \frac{\sum_{l=1}^{L_{con}} I(|S_l| \geqslant |S_{obs}|)}{L_{con}}. \tag{2.3}$$

The p-value estimator is unbiased with MSE $= \widehat{p}_{con}(1-\widehat{p}_{con})/L_{con}$. Under a constraint on the size of the MSE or on the precision of the p-value estimator, Plamadeala and Rosenberger (2012) determine that L_{con} should be in the range of $2,500$ to less than $20,000$, which is computationally feasible in SAS or R software.

Now let $p_j(m_j) = P(T_j = 1|N_A(j-1) = m_{j-1})$. Then the rule

$$\phi_j = \begin{cases} p_j(m_{j-1}) \dfrac{P\left(N_A(n) = n_A|N_A(j) = m_j\right)}{P\left(N_A(n) = n_A|N_A(j-1) = m_{j-1}\right)}, & 1 < j \leq n, \\ \\ p_j \dfrac{P\left(N_A(n) = n_A|T_j = 1\right)}{P\left(N_A(n) = n_A\right)}, & j = 1, \end{cases}$$

is used to sample directly from the conditional reference set.

To compute both p-values of conditional and unconditional tests, it is important to clarify the algorithm for generating randomization sequences. For the unconditional test, randomization tests for complete randomization, PBD, BCD were reviewed in Chapter 1, along with the method of generating randomization sequences for conditional tests under the BCD.

The procedure ϕ_j for the conditional test under complete randomization is the same as the random allocation rule (Rosenberger and Lachin, 2002), where n_A is the predetermined number of patients assigned to group A.

Here we detail the randomization method for conditional tests under the PBD. The balance of patients in the two groups depends on the relationship between block size and total patients. If the number of patients can be evenly divided by block size given that both are even numbers, then perfect balance will be achieved. In this case the conditional test will be equivalent to the unconditional test, as N_A is initially fixed to be n_A. On the other hand, if the last block is not fully filled, the conditional test will be different from the unconditional test.

- Let p_{ij} be the probability of assigning jth patient to treatment group A at ith block, $N_A(j)$ be the number of patients assigned to treatment group A at jth assignment, and s be the block size, $i = 1, ...S$ and $j = 1, ...n$. If we use RAR in each block, for the conditional test we have:

  If $i < S$,

$$p_{i,j} = \begin{cases} \frac{1}{2} & j = 1; \\ \frac{\frac{s}{2}-(N_A(j-1)-(i-1)\frac{s}{2})}{s-(j-(i-1)*s)} & 1 < j < n, \ 1 \leq i \leq S-1. \end{cases} \tag{2.4}$$

  If $i = S$,

$$p_{i,j} = \begin{cases} 0 & N_A(j-1) = n_A; \\ \frac{\frac{s}{2}-(N_A(j-1)-(i-1)\frac{s}{2})}{s-(j-(i-1)*s)} & 0 < n_A - N_A(j-1) < n-j+1; \\ 1 & n_A - N_A(j-1) = n-j+1. \end{cases} \tag{2.5}$$

- Under the TBD, the probability of assigning patients to group A is given by:

If $i < S$,

$$p_{ij} = \begin{cases} 0 & N_A(j-1) = \frac{s}{2}; \\ \frac{1}{2} & \max\{(N_A(j-1) - (i)\frac{s}{2}), (N_B(j-1) - (i-1)\frac{s}{2})\} < \frac{s}{2}; \\ 1 & N_B(j-1) = \frac{s}{2}. \end{cases} \tag{2.6}$$

If $i = S$,

$$p_{ij} = \begin{cases} 0 & N_A(j-1) = n_A; \\ \frac{1}{2} & 0 < n_A - N_A(j) < n - j + 1; \\ 1 & n_A - N_A(j-1) = n - j + 1. \end{cases} \tag{2.7}$$

2.3 Monte Carlo Simulation for the Randomization Test Under the Generalized Linear Model

In this section, we compare the sizes of both the population test and randomization test when a necessary covariate is omitted under the GLM. Assume the true model is $\eta = \mu + T\alpha + X\beta$, where $P(X = 1) = P(X = -1) = 0.5$, $\alpha = 0$, and sample size n is 100. We consider the following distributions: normal, Poisson multiplicative, Poisson additive, exponential multiplicative, exponential reciprocal, exponential additive, Bernoulli logistic and Bernoulli additive. For the Bernoulli additive distribution, we have $\mu = 0.5$, $\beta = 0.25$, and for other additive distributions, we have $\mu = 1$, $\beta = 0.5$. The rest of the distributions have $\mu = 0$ and $\beta = 0.5$. We consider the misspecified model is $\eta^* = \mu^*$ where X is omitted. All tests are two-sided with significance level 0.05. For all conditional randomization tests, we have $n_A = 46$.

The following steps are used for computing the population test:

1. Generate data under the true model where $\eta = \mu + X\beta$ with sample size n and obtain the MLE $\widehat{\mu}^*$ and $\widehat{\eta}^*$ using the misspecified model $\eta^* = \mu^*$.

2. Compute the score test statistic $Z^* = U(\widehat{V}n)^{-\frac{1}{2}}$ given $\widehat{\eta}^*$, where the score function is computed by $U = \partial\ell/\partial\alpha$, and the estimation of the variance is computed by $\widehat{V} = -n^{-1}\sum(\partial^2\ell/\partial\alpha^2)$ given the condition that $E(\partial\ell^2/\partial\alpha\partial\mu) = E(\partial\ell^2/\partial\alpha\partial\beta) = 0$.

3. Replicate 1-2 m times, and compute the proportion of times $|Z^*| \geq 1.96$, this is the size of the population test.

For the randomization test, we use the following algorithm:

- Compute the critical value S_{cr}.

  1. Generate one pass data with initial parameter settings under the true model, and obtain $\widehat{\eta}$ under the misspecified model, compute the score residuals based on (2.3).

  2. Generate L sequences of treatment designs from the unconditional reference set, and merge each sequence to the residuals obtained from 1.

  3. Compute the linear rank statistic for each sequence (for normally distributed data we use van der Waerden scores and for other types of data we use Wilcoxon scores).

4. Order the L test statistics and find the critical value under the significance level, we consider this statistic as S_{cr}.

- Compute the size of the test.

  1. Similar to finding the S_{cr}, generate L randomization sequences, and merge each sequence to the residuals obtained from regression of the one pass data.
  2. Compute the linear rank statistic S for each sequence.
  3. Replicate 1-2 m times and find the proportion of times $|S| \geq |S_{cr}|$.

The following simulation results are obtained using SAS/STAT software with version 9.3. Table 2.1 shows simulations for the population tests at the significance level of 0.05. Except for the Bernoulli additive distribution and the Bernoulli logistic distribution, the size of the test under all other types of distribution exceeds 0.05, especially for exponential distribution families where the size is close to 0.1. This indicates that if there is misidentification of covariates, the assessment of treatment effect will be affected if we use the population method.

Tables 2.2-2.8 show the unconditional and conditional randomization tests under various randomization procedures. Figure 2.1 shows the comparison between the traditional test and randomization tests under the misspecified model. As the treatment effect is evaluated based on randomization only, the size of the randomization test remains around 0.05 for all types of distributions. This indicates that the randomization test is efficient in assessing the treatment effect even when the covariate of the regression is omitted because the test is not dependent on other covariates but depends on the randomization procedure and design only.

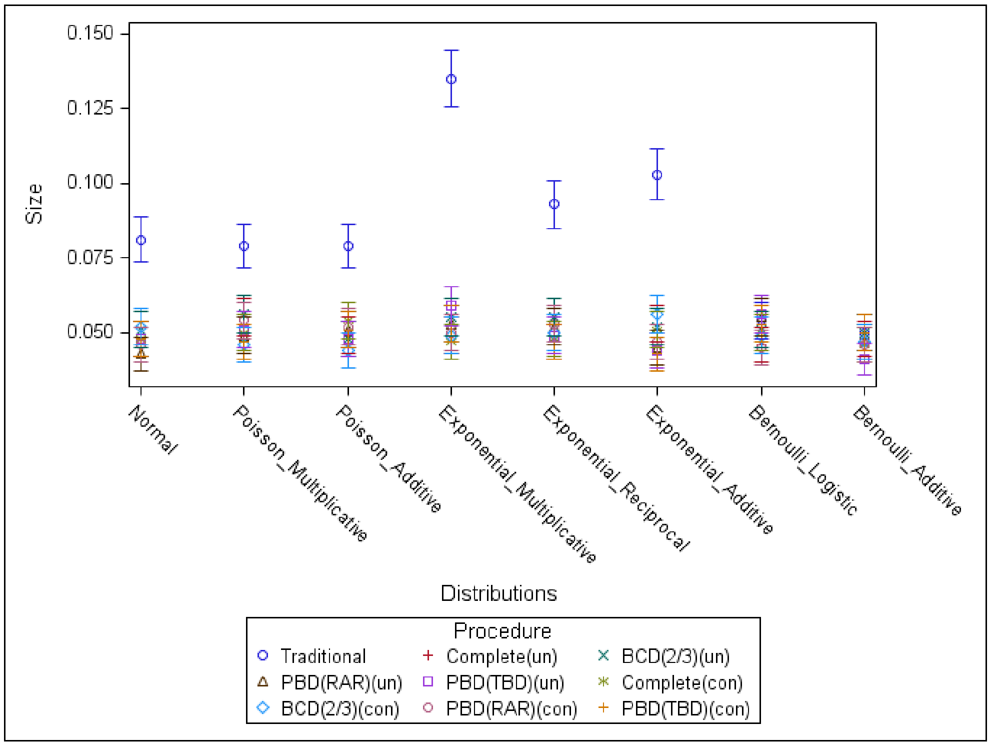

Figure 2.1: Size of the population test and randomization tests under GLM when needed covariate is omitted.

Table 2.1: *Size of the population test when the needed covariate is omitted ($m = 5000$).*

Distribution	$\theta(\eta)$	$h(\eta)$	Size (95% CI)
Normal	η	η	0.081 (0.073, 0.089)
Poisson Multiplicative	η	e^{η}	0.079 (0.071, 0.086)
Poisson Additive	$\log(\eta)$	η	0.079 (0.071, 0.086)
Exponential Multiplicative	$-e^{\eta}$	$e^{-\eta}$	0.135 (0.126, 0.144)
Exponential Reciprocal	$-\eta$	η^{-1}	0.093 (0.085, 0.101)
Exponential Additive	$-\eta^{-1}$	η	0.103 (0.095, 0.101)
Bernoulli Logistic	η	$e^{\eta}(1+e^{\eta})^{-1}$	0.054 (0.048, 0.060)
Bernoulli Additive	$\log(\eta/(1-\eta))$	η	0.050 (0.044, 0.056)

2.4 Randomization Tests for Survival Data Under the Regression Model

In this section, we first introduce approaches for survival analysis, then we give algorithms for computing randomization tests under two commonly used survival models. We give two examples by computing p-values using traditional tests and randomization tests. Finally we simulate survival data and compare the size and power between these two tests under three scenarios: first when the necessary covariate is omitted, second is when there are outliers for the survival time, and third is when the regression model is misspecified.

Table 2.2: *Size of the unconditional randomization test for complete randomization,* $L = 2500$*,* $m = 5000$*.*

Distribution	$\theta(\eta)$	$h(\eta)$	Size (95% CI)
Normal	η	η	0.048 (0.042, 0.054)
Poisson Multiplicative	η	e^{η}	0.055 (0.049, 0.061)
Poisson Additive	$\log(\eta)$	η	0.049 (0.043, 0.055)
Exponential Multiplicative	$-e^{\eta}$	$e^{-\eta}$	0.049 (0.043, 0.055)
Exponential Reciprocal	$-\eta$	η^{-1}	0.047 (0.041, 0.053)
Exponential Additive	$-\eta^{-1}$	η	0.053 (0.047, 0.059)
Bernoulli Logistic	η	$e^{\eta}(1+e^{\eta})^{-1}$	0.046 (0.040, 0.052)
Bernoulli Additive	$\log(\eta/(1-\eta))$	η	0.048 (0.042, 0.054)

Table 2.3: *Size of the unconditional randomization test for BCD(2/3), $L = 2500$, $m = 5000$.*

Distribution	$\theta(\eta)$	$h(\eta)$	Size (95% CI)
Normal	η	η	0.051 (0.045, 0.057)
Poisson Multiplicative	η	e^{η}	0.056 (0.050, 0.062)
Poisson Additive	$\log(\eta)$	η	0.048 (0.042, 0.054)
Exponential Multiplicative	$-e^{\eta}$	$e^{-\eta}$	0.055 (0.049, 0.061)
Exponential Reciprocal	$-\eta$	η^{-1}	0.055 (0.049, 0.061)
Exponential Additive	$-\eta^{-1}$	η	0.052 (0.046, 0.058)
Bernoulli Logistic	η	$e^{\eta}(1+e^{\eta})^{-1}$	0.051 (0.045, 0.057)
Bernoulli Additive	$\log(\eta/(1-\eta))$	η	0.050 (0.044, 0.056)

Table 2.4: *Size of the unconditional randomization test for PBD under RAR, $L = 2500$, $m = 5000$.*

Distribution	$\theta(\eta)$	$h(\eta)$	Size(95% CI)
Normal	η	η	0.043 (0.037, 0.049)
Poisson Multiplicative	η	e^{η}	0.049 (0.043, 0.055)
Poisson Additive	$\log(\eta)$	η	0.051 (0.045, 0.057)
Exponential Multiplicative	$-e^{\eta}$	$e^{-\eta}$	0.053 (0.047, 0.059)
Exponential Reciprocal	$-\eta$	η^{-1}	0.052 (0.046, 0.058)
Exponential Additive	$-\eta^{-1}$	η	0.045 (0.039, 0.050)
Bernoulli Logistic	η	$e^{\eta}(1+e^{\eta})^{-1}$	0.055 (0.049, 0.061)
Bernoulli Additive	$\log(\eta/(1-\eta))$	η	0.047 (0.041, 0.053)

Table 2.5: *Size of the unconditional randomization test for PBD under TBD, $L = 2500$, $m = 5000$.*

Distribution	$\theta(\eta)$	$h(\eta)$	Size(95% CI)
Normal	η	η	0.048 (0.042, 0.054)
Poisson Multiplicative	η	e^{η}	0.051 (0.045, 0.057)
Poisson Additive	$\log(\eta)$	η	0.048 (0.042, 0.054)
Exponential Multiplicative	$-e^{\eta}$	$e^{-\eta}$	0.059 (0.052, 0.066)
Exponential Reciprocal	$-\eta$	η^{-1}	0.049 (0.043, 0.055)
Exponential Additive	$-\eta^{-1}$	η	0.044 (0.038, 0.050)
Bernoulli Logistic	η	$e^{\eta}(1+e^{\eta})^{-1}$	0.056 (0.050, 0.062)
Bernoulli Additive	$\log\left(\eta/(1-\eta)\right)$	η	0.041 (0.036, 0.046)

Table 2.6: *Size of the conditional randomization test for complete randomization,* $L_{con} = 2500$*,* $m = 5000$*,* $n_A = 46$*.*

Distribution	$\theta(\eta)$	$h(\eta)$	Size (95% CI)
Normal	η	η	0.048 (0.042, 0.054)
Poisson Multiplicative	η	e^{η}	0.050 (0.044, 0.056)
Poisson Additive	$\log(\eta)$	η	0.054 (0.048, 0.060)
Exponential Multiplicative	$-e^{\eta}$	$e^{-\eta}$	0.047 (0.041, 0.053)
Exponential Reciprocal	$-\eta$	η^{-1}	0.048 (0.042, 0.054)
Exponential Additive	$-\eta^{-1}$	η	0.051 (0.045, 0.057)
Bernoulli Logistic	η	$e^{\eta}(1+e^{\eta})^{-1}$	0.050 (0.044, 0.056)
Bernoulli Additive	$\log(\eta/(1-\eta))$	η	0.046 (0.040, 0.052)

Table 2.7: *Size of the conditional randomization test for BCD (2/3), $L_{con} = 2500$, $m = 5000$, $n_A = 46$.*

Distribution	$\theta(\eta)$	$h(\eta)$	Size (95% CI)
Normal	η	η	0.052 (0.046, 0.058)
Poisson Multiplicative	η	e^{η}	0.046 (0.040, 0.052)
Poisson Additive	$\log(\eta)$	η	0.044 (0.038, 0.050)
Exponential Multiplicative	$-e^{\eta}$	$e^{-\eta}$	0.049 (0.043, 0.055)
Exponential Reciprocal	$-\eta$	η^{-1}	0.050 (0.044, 0.056)
Exponential Additive	$-\eta^{-1}$	η	0.056 (0.050, 0.062)
Bernoulli Logistic	η	$e^{\eta}(1+e^{\eta})^{-1}$	0.049 (0.043, 0.055)
Bernoulli Additive	$\log(\eta/(1-\eta))$	η	0.047 (0.041, 0.053)

Table 2.8: *Size of the conditional randomization test for PBD under RAR, $L_{con} = 2500$, $m = 5000$, $n_A = 46$.*

Distribution	$\theta(\eta)$	$h(\eta)$	Size (95% CI)
Normal	η	η	0.046 (0.040, 0.052)
Poisson Multiplicative	η	e^{η}	0.054 (0.048, 0.060)
Poisson Additive	$\log(\eta)$	η	0.052 (0.046, 0.058)
Exponential Multiplicative	$-e^{\eta}$	$e^{-\eta}$	0.050 (0.044, 0.056)
Exponential Reciprocal	$-\eta$	η^{-1}	0.053 (0.047, 0.059)
Exponential Additive	$-\eta^{-1}$	η	0.047 (0.041, 0.053)
Bernoulli Logistic	η	$e^{\eta}(1+e^{\eta})^{-1}$	0.045 (0.039, 0.051)
Bernoulli Additive	$\log(\eta/(1-\eta))$	η	0.046 (0.040, 0.052)

Table 2.9: *Size of the conditional randomization tests for PBD under TBD, $L_{con} = 2500$, $m = 5000$, $n_A = 46$.*

Distribution	$\theta(\eta)$	$h(\eta)$	Size (95% CI)
Normal	η	η	0.048 (0.042, 0.054)
Poisson Multiplicative	η	e^{η}	0.047 (0.041, 0.053)
Poisson Additive	$\log(\eta)$	η	0.051 (0.045, 0.057)
Exponential Multiplicative	$-e^{\eta}$	$e^{-\eta}$	0.053 (0.047, 0.059)
Exponential Reciprocal	$-\eta$	η^{-1}	0.047 (0.041, 0.053)
Exponential Additive	$-\eta^{-1}$	η	0.043 (0.037, 0.049)
Bernoulli Logistic	η	$e^{\eta}(1+e^{\eta})^{-1}$	0.053 (0.047, 0.059)
Bernoulli Additive	$\log(\eta/(1-\eta))$	η	0.050 (0.044, 0.056)

2.4.1 Approaches to Survival Analysis

Survival analysis is a class of statistical procedures that is applied to the data where the outcome variable is the time to the occurrence of events. Assume in a clinical study a patient with treatment T and covariate X has failure time W. Because of potential right censoring, we observe $Y = \min(W, C)$ and $\Delta = I(W \leq C)$, where C is the censoring time. The censoring indicator Δ takes value 1 if we observe the failure time and takes value 0 otherwise.

We define two functions as follows: let t be a specified time, then the survival function is the probability that W is longer than t:

$$S(t) = P(W > t).$$

The hazard function is given by

$$\lambda(t) = \lim_{\delta \to 0} \frac{P(t \leq W < t + \delta | W \geq t)}{\delta}.$$

Regression models are often used if there are several covariates to be examined and the data fit certain assumptions. We consider two survival models in this thesis, namely, the Cox proportional hazards (PH) model and the accelerated failure time (AFT) model. The PH model gives the expression of the hazard function as a product of two components: a baseline hazard function which contains information of observed time only, and a component which contains information on covariates only. Let $\lambda_0(\cdot)$ be an unspecified baseline hazard, and $\Lambda_0(\cdot)$ be the corresponding baseline cumulative hazard function. Given the linear predictor $\eta = T\alpha + X\beta$, the hazard function and survival function at time t for the PH model are :

$$\lambda(t|T,X) = \lambda_0(t)\exp(\eta), \tag{2.8}$$

and

$$S(t|T,X) = \exp(-\Lambda_0(t)\exp(\eta)). \tag{2.9}$$

We maximize the partial likelihood function based on n independently and identically distributed observations $\{(Y_i, \Delta_i, T_i, X_i),\ i = 1, ...n\}$ to obtain the maximum likelihood estimators of (α, β), denoted by $(\hat{\alpha}, \hat{\beta})$. Note that the partial likelihood function is the profile likelihood function of (α, β), in which the unspecified baseline hazard is profiled out. Inference can be carried out by applying the Wald, score or likelihood ratio tests.

The key assumption of the PH model is that the hazard ratio, the hazard function of one patient divided by the hazard of another patient is constant over time. One can use $\log(-\log)$ of the empirically estimated survival curves (e.g. Kaplan-Meier estimator) to test the proportional hazard assumption; if these curves are parallel over the time, then the PH model is appropriate.

The AFT model is used when the failure time W is assumed to have a certain distribution, and the impact of the covariates are proportional with respect to the time t. Common distributions under the AFT model include exponential, Weibull, lognormal and log-logistic. Let $S_0(t)$ be the baseline survival, $i = 1, ...n$. The common distributions under the AFT model include exponential, Weibull, lognormal and log-logistic. The log form of the distribution parameter is usually represented by the regression parameters and covariates. Therefore, the AFT model specifies that the conditional survival function of W given T and X takes the form

$$S(t|T,X) = S_0(t/\exp(\eta)), \tag{2.10}$$

where η is the linear predictor defined above.

For a given distribution, if the hazard function satisfies both assumptions of the PH model and the AFT model, such as the Weibull distribution, then these two models are equivalent; the difference is only the way we parameterize the regression and distribution parameters. However, for some distributions like the log-logistic, only the AFT model can be accommodated. The AFT model for the log-logistic distribution is also called the proportional odds model, as its odds ratio is constant over time but not hazard ratio. Under the proportional odds model, the log of the variable W follows a logistic distribution. Let ρ be a scale parameter and $\eta = \mu + T\alpha + X\beta$; then the survival function and hazard function are given by

$$S(t|T,X) = \frac{1}{1+\exp(\eta)t^{\rho}} \tag{2.11}$$

and

$$\lambda(t|T,X) = \frac{\exp(\eta)\rho t^{\rho-1}}{1+\exp(\eta)t^{\rho}}. \tag{2.12}$$

The assumption of the proportional odds model can be examined by plotting $\log((1-S(t))/S(t))$ versus $\log(t)$ using Kaplan-Meier estimators.

Now we detail the algorithm for randomization tests under the PH model and AFT models. Following the strategy for performing randomization tests under the GLM, here we propose to use residuals from those regressions as the outcome variable. It can be shown $E(\Delta|T,X) = \Lambda(Y|T,X)$ (Breslow, 1978), where $\Lambda(t|T,X)$ is the conditional hazard function of W given T and X. The martingale residuals under the PH model and the AFT model are given by $\Delta - \widehat{\Lambda_0}(Y)\exp(\hat{\eta})$ and $\Delta - (-\log \widetilde{S}_0(Y/\exp(\widetilde{\eta}))$, where $(\widehat{\Lambda}_0, \widehat{\eta})$ and $(\widetilde{S}_0, \widetilde{\eta})$ are the maximum likelihood estimators of (Λ_0, η) and (S_0, η), respectively.

To compute the p-value under a specified procedure, we use following steps:

1. Perform the regression (either PH or AFT model) by assuming $\alpha = 0$, obtain the residuals from the regression.

2. Compute the linear rank test statistic with corresponding treatment assignments; this is the observed test statistic.

3. Generate L randomization sequences, merge each one to the residuals from step 1, compute the linear rank test statistic for each sequence. The proportion of times that the test statistics greater than the observed statistic will be the estimated p-value.

2.4.2 Examples of Applying Randomization Test Under the Proportional Hazards Model

In this section, we provide two examples of comparing results for randomization tests under the PH model and the traditional model-based approach ignoring the randomization procedure.

Example 1. The Remission Data

The data for the first example is the remission data (Freireich et al., 1963), which is from a study that assessed the impact of a maintenance therapy 6-Mercaptopurine for prolonging the duration of steroid-induced remission to patients under the age of 20 with acute leukemia. Patients in remission are assigned into either treatment group with 6-Mercaptopurine therapy or placebo group. There were 92 patients who entered the study, and the study was stopped after analysis of first 42 patients (21 pairs). The outcome variable of this data is the remission survival time of patients, the independent covariates include the log form of white blood count (WBC), which has been shown that the initial count of blood cells is negatively related to the survival of leukemia patients, the censoring indicator (1 indicates censoring and 0 indicates failure), and the treatment indicator (1 indicates the patient is in the treatment group and 0 is in the placebo group) (Kleinbaum and Klein, 2005).

For the traditional population-based test, we assume patients are randomly assigned to the treatment group and placebo group. We fit the PH model based on (2.8) and (2.9). Under the null hypothesis, $\alpha = 0$. where X is the log of WBC, T is the treatment indicator. Using the PHREG procedure in SAS, we obtain the MLE $\widehat{\alpha}$, the p-value for the test is computed based on comparing a Wald test statistic to critical values from chi squared distribution.

For the randomization test, we apply the algorithm detailed in Section 2.4.1. We assume the PBD with block size 2 is used as the randomization procedure, and RAR is applied within each block.

Example 2. The Ovarian Cancer Data

The second example is from a clinical trial of a single chemotherapy vs. combined chemotherapy for patients with ovarian cancer (Edmonson et al., 1979). This trial

Table 2.10: *Comparison of p-values between two tests.*

Data	Traditional Test	Randomization Test
Remission	0.0020	0.0010
Ovarian Cancer	0.0632	0.2202

consisted of 26 female patients with minimal residual disease and who experienced surgery of removing tumors greater than 2 cm in diameters. Patients were randomized into treatment groups after the surgery. The outcome variable is the survival time in days following the treatment assignment; other variables in the data include age of patients, the censoring indicator, treatment indicator (1 is the combined therapy, and 0 is single therapy), the extent of the residual disease (1 is incomplete, 2 is complete), and the performance (1 is good and 2 is poor).

For the traditional test, we fit the AFT model based on (2.11) and (2.12) under the Weibull distribution (Collett, 2003). Here, we only include age in X as other covariances do not have significant impact on the survival time, T is the treatment indicator. We obtain the p-value using the LIFEREG procedure in SAS.

For the randomization test, we also apply the algorithm detailed in Section 2.4.1. We assume conditional complete randomization is applied for the treatment assignment.

Results

Table 2.10 shows the results from two tests. For the remission data, both the traditional test and the randomization test indicate there is a significant treatment effect, while for the ovarian cancer data, both tests indicate there is no significant different between the two treatment groups.

2.4.3 Simulation of the Randomization Tests Under Misspecified Survival Models

To compare the efficiency of the randomization and parametric test, we consider three different cases of simulated survival data.

In the first set of simulation, we generate the failure time from

$$\lambda(t|T, X) = \exp(T\alpha + X\beta),$$

where $T = 1$ for the treatment group and $T = 0$ for the control group, and X is a Bernoulli random variable with success probability 1/2. we compare the size and power of parametric and randomization tests when X is omitted under the PH model.

In the second simulation, we replace 10 percent of the survival times generated from the above model by standard lognormal outliers and analyze data under the PH model.

Int the third simulation, we generate failure time data from the proportional odds model

$$S(t|T, X) = \frac{1}{1 + \exp(3 + T\alpha + X\beta)t}, \tag{2.13}$$

and analyze data under the PH model.

In all simulations, we set $\beta = 2$ and the treatment effect α is set to be 0 under the null hypothesis and varies from 1 to 3 under the alternative. The censoring time is from an exponential distribution with mean 1 and varies from 5 percent to 20 percent under all scenarios. For each simulation setting, we generate 5000 replications each with 60 subjects. The hypothesis testing results are based on two-sided significance level of 0.05. For all conditional randomization tests, we use $n_A = 30$.

We apply the following steps to compute the size and power of the traditional tests for three scenarios:

- Compute the size and power of the traditional test.

    1. Generate data set under a specific hypothetical value of α, for the third scenario, we generate data based on proportional odds model.
    2. Obtain p-values for $\widehat{\alpha}$ when we use the misspecified models. For the first scenario, we perform the regression without β, for the second and third scenario, we perform the regression using the PH model.
    3. Replicating step 1-2 m times, the proportion of times that the p-value is less than 0.05 is the estimated size or power of the test.

- Compute the size and power of the randomization test:

    1. We follow Section 2.3 to compute the size of the test, but use the martingale residuals obtained from the regression model.
    2. We using following steps to compute the power of the test:
        (a) For all scenarios, generate data under the alternative hypothesis, obtain the martingale residuals under $\alpha = 0$ and misspecified models.
        (b) Compute the linear rank test statistic using Wilcoxon scores based on residuals and the corresponding treatment assignment, this is the observed test statistic S_{obs}.
        (c) Generate L randomization sequences, merge residuals with each sequence. We obtain L rank test statistic S, the proportion of times S greater than the observed one will be the estimated p-value.
        (d) Replicate 1-3 for m times, compute the proportion of times $p \leq 0.05$; this is the estimated power of the test.

Table 2.11: *Size and power of the traditional and randomization tests under the true PH model ($L = 2500$, $m = 5000$).*

Test	Size (95% CI)	Power ($\alpha = 1$) (95% CI)	Power ($\alpha = 1.5$) (95% CI)
Traditional	0.0496	0.8208	0.9952
	(0.0436,0.0556)	(0.8101, 0.8314)	(0.9933, 0.9971)
Randomization (Complete unconditional)	0.0373	0.5158	0.8796
	(0.0320,0.0426)	(0.5019,0.5297)	(0.8706, 0.8886)
Randomization (Complete conditional)	0.0520	0.6074	0.9350
	(0.0458,0.0581)	(0.5939, 0.6209)	(0.9282, 0.9418)
Randomization (BCD(2/3) unconditional)	0.0443	0.6032	0.9304
	(0.0386,0.0500)	(0.5896, 0.6168)	(0.9233, 0.9374)
Randomization (BCD(2/3) conditional)	0.0420	0.5828	0.9184
	(0.0364,0.0476)	(0.5691, 0.5964)	(0.9108, 0.9260)

Tables 2.11-2.15, Figure 2.2-2.4 give the simulation results for the size and power under different situations. The size of all the tests is close to the nominal significance level, except when there are lognormal outliers, in which case the size of the test is significantly inflated. The power of the traditional test is superior under the true models for all cases, and it dropped significantly when the covariate is omitted under the proportional hazards model with $\alpha = 1$ and when there are contaminants for the survival time under the PH model with $\alpha = 2$ and $\alpha = 3$. These finding are consistent with those reported in Lakagos and Schoenfeld (1984) and Morgan, Lakagos and Schoenfeld (1986). When the true model is the proportional odds model and examined as the PH model, the power of the traditional test is only slightly affected, the possible reason is that sometimes the odds ratio is similar to the hazard ratio.

For the randomization test, the size of the test is below the nominal level for all cases. The power of the tests is not significantly affected when there are omitted covariates. All the test outperform the traditional test when there are lognormal contaminants(Table 2.13). This confirms that on some occasions the randomization test might be a good alternative to the traditional test if there is misidentification for the modeling in survival analysis.

Table 2.12: *Size and power of the traditional and randomization tests with omitted covariates under the PH model ($L = 2500$, $m = 5000$).*

Test	Size (95% CI)	Power ($\alpha = 1$) (95% CI)	Power ($\alpha = 1.5$) (95% CI)
Traditional	0.0504	0.5788	0.9174
	(0.0443,0.0564)	(0.5651, 0.5925)	(0.9098, 0.9250)
Randomization (Complete unconditional)	0.0418	0.4200	0.7812
	(0.0363,0.0474)	(0.4063, 0.4337)	(0.7697, 0.7927)
Randomization (Complete conditional)	0.0446	0.5708	0.8596
	(0.0389,0.0503)	(0.5571, 0.5841)	(0.8490, 0.8692)
Randomization (BCD(2/3) unconditional)	0.0380	0.4904	0.8522
	(0.0327,0.0432)	(0.4765, 0.5043)	(0.8424, 0.8620)
Randomization (BCD(2/3) conditional)	0.0394	0.4558	0.8200
	(0.0340,0.0448)	(0.4420, 0.4696)	(0.8094, 0.8306)

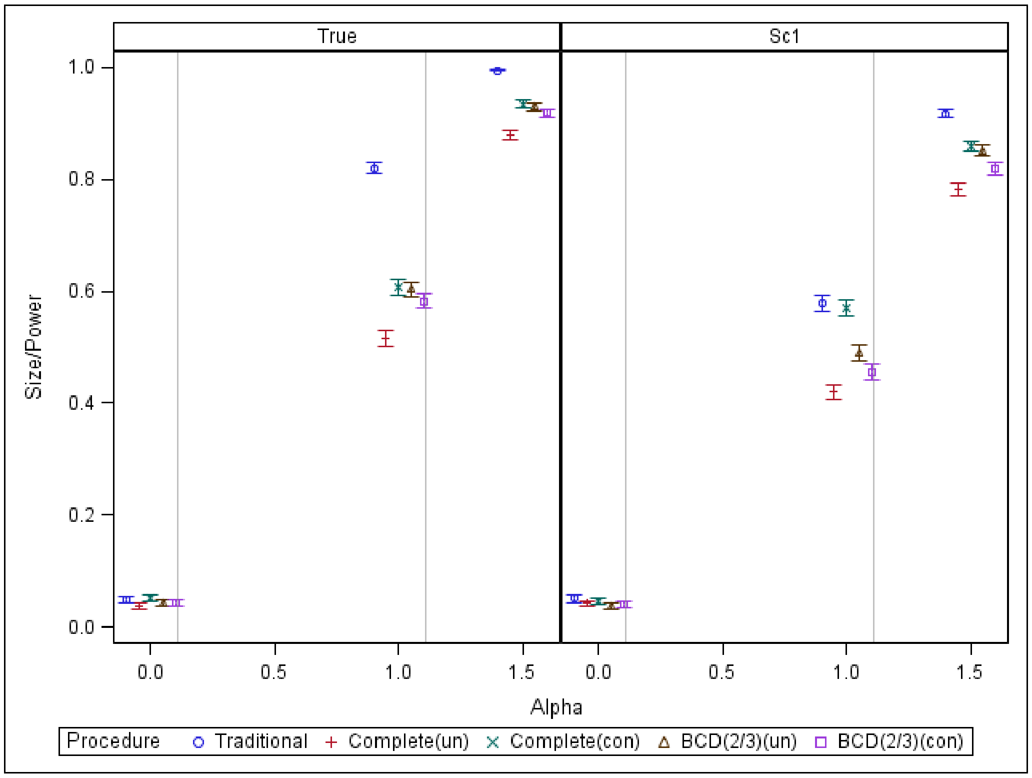

Figure 2.2: Size and Power of the traditional and randomization tests under the PH model with missing covariate.

Table 2.13: *Size and power of the traditional and randomization tests with lognormal contaminants under the PH model ($L = 2500$, $m = 5000$).*

Test	Size (95% CI)	Power ($\alpha = 2$) (95% CI)	Power ($\alpha = 3$) (95% CI)
Traditional	0.1608	0.3018	0.5358
	(0.1506,0.1710)	(0.2891, 0.3145)	(0.5220, 0.5496)
Randomization (Complete unconditional)	0.0403	0.5002	0.7948
	(0.0348,0.0458)	(0.4863, 0.5141)	(0.7836, 0.8060)
Randomization (Complete conditional)	0.0396	0.5774	0.8418
	(0.0342,0.0450)	(0.5577, 0.5851)	(0.8317, 0.8519)
Randomization (BCD(2/3) unconditional)	0.0388	0.5184	0.8012
	(0.0334,0.0451)	(0.5046, 0.5322)	(0.7901, 0.8122)
Randomization (BCD(2/3) conditional)	0.0386	0.4542	0.7526
	(0.0332,0.0439)	(0.4404, 0.4640)	(0.7406, 0.7645)

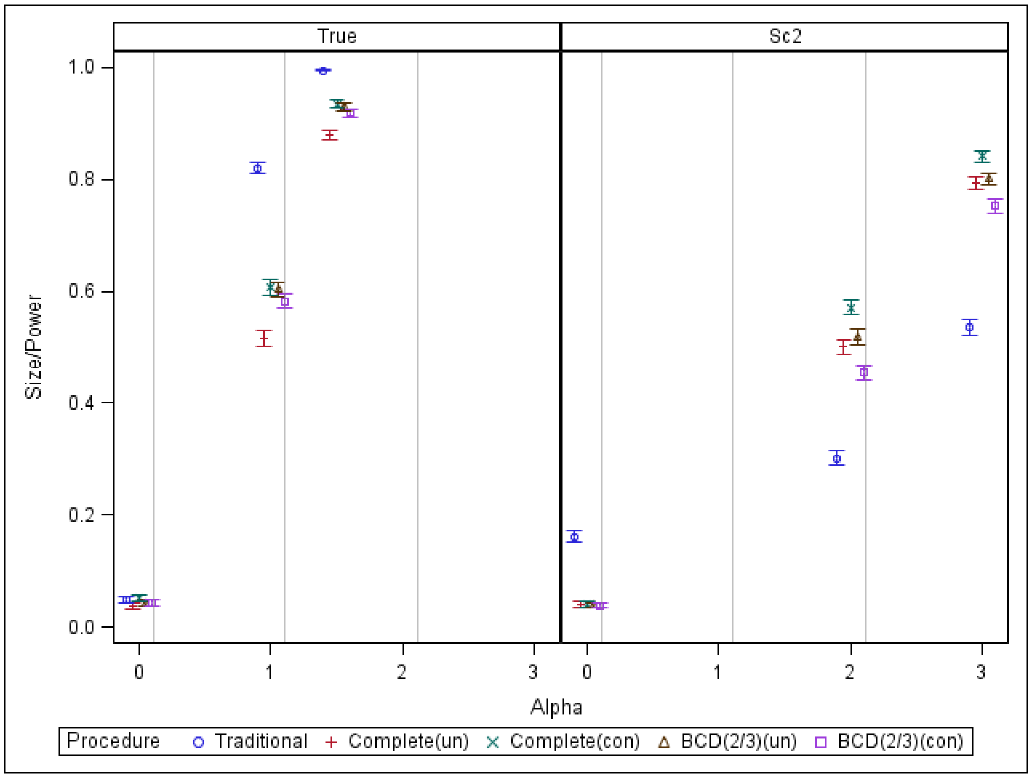

Figure 2.3: Size and Power of the traditional and randomization tests under the PH model with 10 % lognormal outliers for the response.

Table 2.14: *Size and power of the traditional and randomization tests under the true proportional odds model ($L = 2500$, $m = 5000$).*

Test	Size (95% CI)	Power ($\alpha = 1.5$) (95% CI)	Power ($\alpha = 2$) (95% CI)
Traditional	0.0582	0.8426	0.9730
	(0.0571,0.0647)	(0.8325, 0.8527)	(0.9685, 0.9775)
Randomization (Complete unconditional)	0.0410	0.7592	0.9352
	(0.0355,0.0465)	(0.7473, 0.7711)	(0.9284, 0.9420)
Randomization (Complete conditional)	0.0449	0.7702	0.9430
	(0.0392,0.0506)	(0.7585, 0.7819)	(0.9366, 0.9494)
Randomization (BCD(2/3) unconditional)	0.0469	0.7698	0.9446
	(0.0410,0.0528)	(0.7581, 0.7815)	(0.9383, 0.9509)
Randomization (BCD(2/3) conditional)	0.0414	0.7696	0.9374
	(0.0359,0.0469)	(0.7579, 0.7813)	(0.9307, 0.9441)

Table 2.15: *Size and power of the traditional and randomization tests under the mis-specified modeling assumption ($L = 2500$, $m = 5000$).*

Test	Size (95% CI)	Power ($\alpha = 1.5$) (95% CI)	Power ($\alpha = 2$) (95% CI)
Traditional	0.0610	0.7352	0.9196
	(0.0544,0.0676)	(0.7230, 0.7474)	(0.9121, 0.9271)
Randomization (Complete unconditional)	0.0438	0.7472	0.9306
	(0.0381,0.0495)	(0.7352, 0.7592)	(0.9236, 0.9376)
Randomization (Complete conditional)	0.0455	0.7482	0.9330
	(0.0397,0.0513)	(0.7362, 0.7602)	(0.9261, 0.9399)
Randomization (BCD(2/3) unconditional)	0.0413	0.7516	0.9366
	(0.0358,0.0469)	(0.7396, 0.7696)	(0.9298, 0.9433)
Randomization (BCD(2/3) conditional)	0.0406	0.7480	0.9212
	(0.0351,0.0460)	(0.7360, 0.7600)	(0.9137, 0.9287)

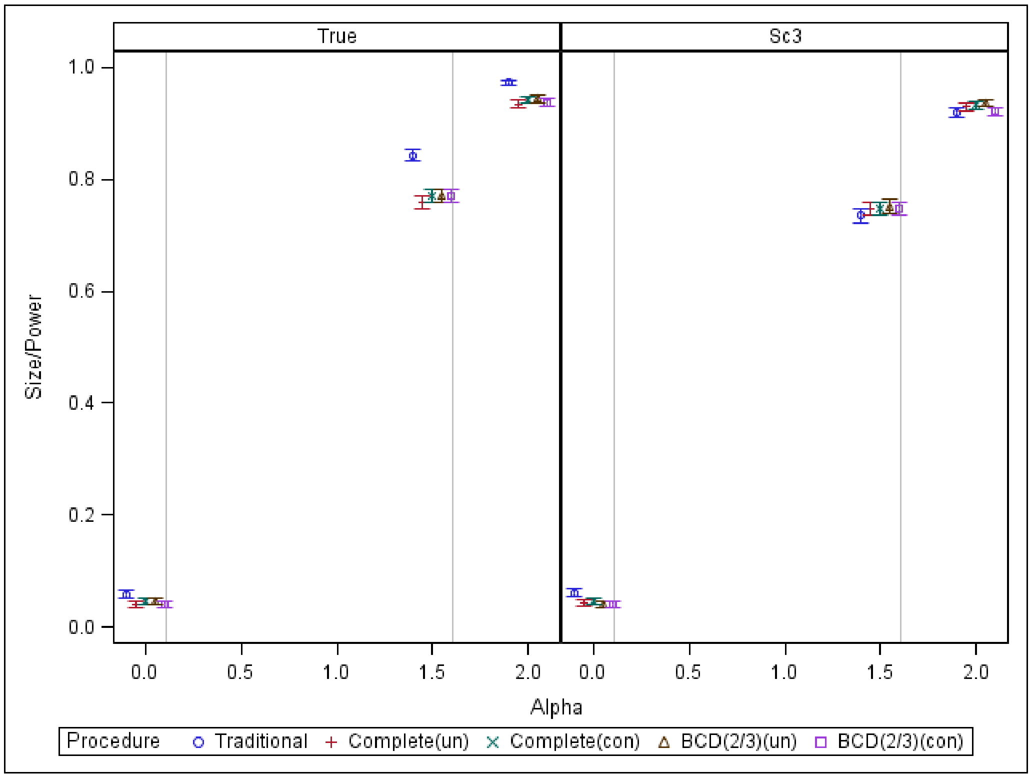

Figure 2.4: Size and Power of the traditional and randomization tests with misspecified modeling assumption.

Chapter 3: Randomization Tests Under the Generalized Linear Mixed Model

In this chapter we provide the algorithm for computing randomization tests for the generalized linear mixed model (GLMM). In particular, we investigate the treatment effect variation over the repeated measures, such as whether a treatment has time varying effect for a patient. In the first section we review inference based on traditional maximum likelihood estimation. In the second section, we use the predicted random slope from the GLMM and perform randomization tests. We provide details of estimating p-values for both unconditional and conditional randomization tests.

3.1 Traditional Inference Under the Generalized Linear Mixed Model

The GLMM is an extension of the GLM for longitudinal data or clustered data which incorporates random effects. The random effects allow for correlation between the repeated observations within each subject and variation between subjects. The followings are characteristics of the GLMM:

1. Let Y_{ij} be the response variable for jth measurement occasion of ith subject, where $i = 1, ...n$, $j = 1, ...n_i$. Conditional on a $q \times 1$ vector of random effects $\mathbf{b}_i$, $\mathbf{Y}_i = (Y_{i,1}, ...Y_{i,n_i})$ are independent and each belongs to the exponential family.

2. Let $\mathbf{X}_i$ be a $n_i \times p$ matrix, $\mathbf{Z}_i$ be a $n_i \times q$ design matrix, μ be a constant intercept, $T_i = 1$ if patient i is assigned to group A, 0 if patient i is assigned to group B. Let $h(\cdot)$ be a known function, $v(\cdot)$ be a variance function and ϕ be a scale parameter. Then we have

$$\eta_i = \mu + T_i\alpha + \mathbf{X}_i\beta + \mathbf{Z}_i\mathbf{b}_i, \tag{3.1}$$

   where

$$E(Y_{ij}|X_{ij}, T_i, \mathbf{b}_i) = h(\eta_{ij}) \tag{3.2}$$

   and

$$var(Y_{ij}|X_{ij}, T_i, \mathbf{b}_i) = v(h(\eta_{ij}))\phi. \tag{3.3}$$

   In the above equations, μ, α, β are considered as fixed-effects parameters, X_{ij} and η_{ij} denote the covatiates and linear predictor of the jth occasion for the ith subject.

3. The random effects are usually assumed to be multivariate normal with mean zero and a $q \times q$ variance-covariance matrix $\mathbf{G}$.

As it is of interest to relate the treatment effect to the repeated measures, let $\mathbf{X}_i = (X_{1i}, \tau_{ij}, \tau_{ij} \times T_i)$, $\mathbf{Z}_i = (1, \tau)$, $\mathbf{b}_i = (b_{0i}, b_{1i})'$, where X_{1i} is some covariate, τ_{ij} is the jth time point for the ith patient, and b_{0i} and b_{1i} are the random intercept and the random slope for the ith patient, respectively. Then the linear predictor can be written as

$$\eta_{ij} = \mu + T_i\alpha + X_{1i}\beta_1 + \tau_{ij}\beta_2 + \tau_{ij}T_i\beta_3 + b_{0i} + \tau_{ij}b_{1i}. \tag{3.4}$$

Under the null hypothesis of the population test, there is no varying treatment via repeated measures, and we have $\beta_3 = 0$. As the random effects are not observed, we can estimate $(\beta, \mathbf{G})$ based on the likelihood function, given by

$$L(\beta, \phi, \mathbf{G}) = \prod_{i=1}^{n} \int f_Y(\mathbf{Y}_i|\mathbf{b}_i) f_b(\mathbf{b}_i) d\mathbf{b}_i \tag{3.5}$$

where f_Y is the conditional density function for the response variables and f_b is the density function of the random effects. This likelihood function is obtained by integrating out $\mathbf{b}_i$; it depends on the variance-covariance of $\mathbf{b}_i$ but not the unobserved $\mathbf{b}_i$ (Fitzmaurice, Laird and Ware 2004). Numerical integration techniques have to be used to obtain the solution of the likelihood function. There is also no simple solution for the variance of $\widehat{\beta}$ which strongly depends on the variance-covariance of the random effects. For the population tests, the test statistic $Q = \widehat{\beta}_3/\widehat{SE}(\widehat{\beta}_3)$ approximately follows a t-distribution given estimated $\mathbf{G}$ and ϕ. The robustness of the test for β_3 then depends on the accuracy of the specified variance-covariance structure of the random effects, the distribution of the response, and whether all other assumptions are valid.

Many studies have evaluated how inference for the fixed effects is affected if there is misspecification for the GLMM. For balanced and complete data, in general the estimation of β may have negligible bias if the distribution of the random effects is misspecified (Fitzmaurice, Laird and Ware 2004, Chen, Zhang and Davidian 2002, Agresti, Caffo and Ohman-Strickland 2004). However, misidentification of the random effects distribution can bias the estimators of the variance-covariance parameters. Litière, Alonso and Molenberghs (2007) found that the type I error rate rarely exceeds the significance level under various misspecifications of the random effects, but power could be seriously affected, depending on the shape and the variance of the underlying random effects distribution. Jacqmin-Gadda et al. (2007) investigated the consequences to the fixed effects when the error distribution is misspecified for the special case of the GLMM where the response are normally distributed. They concluded that the inference is robust when errors are non-normal or heteroscedastic, except if there is correlation between the error variance and covariates in the model. The estimators of the fixed effects are found to be slightly biased if the errors are correlated.

3.2 Unconditional Tests and Conditional Tests Under the Generalized Linear Mixed Model

To avoid the distributional and variance-covariance assumptions inherent in the traditional population-based inference procedures, we propose to use a randomization test to detect the change of the treatment effects with repeated measures considering the predicted random slope as the fixed outcome.

Given maximum likelihood estimators of β, ϕ and $\mathbf{G}$, the random effects $\mathbf{b_i}$ can be predicted as follows:

$$\widehat{\mathbf{b}}_i = E(\mathbf{b_i} | \mathbf{Y_i}, \widehat{\beta}, \widehat{\phi}, \widehat{\mathbf{G}}). \tag{3.6}$$

This is the conditional mean of $\mathbf{b_i}$ from the ith subject, the estimated fixed effects and variance parameters.

Under the null hypothesis of the randomization test, for each subject there is no difference in the repeated measure changes regardless of which treatment is assigned. The algorithm for estimating p-values for unconditional tests and conditional tests under the GLMM is similar to the algorithm under the GLM:

1. Following (3.4), for a given data set, perform the regression analysis assuming

$\beta_3 = 0$ and obtain the estimates of β, ϕ and $\mathbf{G}$.

2. Predict the random slope b_{1i} based on (3.6) given the response and $\widehat{\beta}, \widehat{\phi}, \widehat{\mathbf{G}}$ for each subject i.

3. Compute the rank test statistic given $\widehat{b}_{1i}$ and corresponding treatment assignments; this is the observed test statistic S_{obs}.

4. Generate L randomization sequences. For unconditional tests, these sequences are from unconditional reference set; for conditional tests, these sequences are from conditional reference set. Compute the rank tests statistic S given $\widehat{b}_{1i}$ for each sequence.

5. Given the reference set, the p-value is estimated by using (2.2) or (2.3).

Chapter 4: Application of Randomization Tests to Longitudinal Data

We propose to apply both unconditional and conditional randomization tests to a longitudinal study where data are collected repeatedly over time. This is the first time to our knowledge that the randomization test has been applied to the longitudinal data to test the time-varying treatment effect.

In the first section, we provide an example of applying the randomization test to longitudinal clinical trial data. In the second section, we examine four different scenarios for misspecification of the GLMM and provide simulation results for both traditional population tests and randomization tests. For the randomization test, randomization procedures include complete randomization and BCD for both unconditional and conditional tests.

4.1 An Example of Application of the Randomization Test to Longitudinal Clinical Trial Data

In this section, we provide an example comparing the result from randomization tests under the linear mixed model and the traditional test for detecting the time-varying treatment effect.

The AIDS CD4 Counts Data

The data set is from a randomized, double-blind and controlled clinical trial of AIDS patients (Henry et al. 1998). The purpose of this study is to compare the survival benefits of using different HIV-1 inhibitors therapies. The benefit is measured by CD4 counts, which is a type of white blood cell that fights infection. There were 1309 patients randomized into four groups with different reverse transcriptase inhibitors therapies. Measurements of CD4 counts were recorded based on 8 weeks intervals. The number of measurements ranges from 1 to 9. Since there are missing values due to skipped visits and dropouts, we only consider the records of 729 patients with the first four visits. We combine groups 2, 3 and 4 and set treatment value as 1, and for group 1, treatment value is 0 (Fitzmaurice, Laird and Ware 2004). The response variable is the log form of the CD4 counts $\log(CD4 + 1)$; covariates include age, gender, treatment and visit.

For the traditional test, we assume patients are randomly assigned to the two treatment groups. We fit the model from (4.1), assuming the conditional distribution of the response is normal. Under the null hypothesis, $\beta_3 = 0$. Using the GLIMMIX procedure, we obtain $\widehat{\beta}_3$ using restricted maximum likelihood, and compute the p-value based on a t statistic.

For the randomization test, we apply the algorithm for p-values in GLMM provided in Section 3.2. As only 197 out of 729 patients were in the treatment group with value 1, we apply the conditional test based on complete randomization and BCD(2/3).

Results

All tests demonstrate a significant time-varying treatment effect. Under the traditional model, the p-value for a t test of $\widehat{\beta}_3$ is less than 0.0001; under the randomization model, the p-value for the conditional complete randomization test is 0.0002 and less than 0.0001 for the conditional BCD(2/3).

4.2 Simulation of the Randomization Tests for Longitudinal Data

4.2.1 Parameter Settings

The simulation is performed assuming an identity function $h(\cdot)$, which is the classic linear mixed model, where $\mathbf{Y_i}$ has a normal distribution conditional on the random effects. We have $\mathbf{Y_i} = \eta_\mathbf{i} + \mathbf{e_i}$, where the error term for the ith patient $\mathbf{e_i}$ is assumed to be normally distributed with mean zero and a $n_i \times n_i$ covariance matrix $\mathbf{R_i}$. Then $var(\mathbf{Y_i}) = \mathbf{Z_i G Z_i}' + \mathbf{R_i}$, and $var(\mathbf{Y_i}|\mathbf{b_i}) = \mathbf{R_i}$. We use sample size $n = 100$. For each patient, the repeated time τ varies from 1 to 5; for other parameters we have $\mu = -3$, $\alpha = 2$, $\beta_1 = 2$, $\beta_2 = 1.5$. The random effects have the multivariate normal distribution with $E(b_{0i}) = E(b_{1i}) = 0$, $var(b_{0i}) = 4$, $var(b_{1i}) = 4$, and $cov(b_{0i}, b_{1i}) = 0$. The error term has the standard normal distribution where $E(\mathbf{e_i}) = 0$ and $\mathbf{R_i}$ is an identity matrix. Under the null hypothesis, there is no time-varing treatment effect, i.e., $\beta_3 = 0$. All tests are two-sided with significance level 0.05. For all conditional randomization tests, we use $n_A = 46$.

There are four scenarios considered as follows:

1. The conditional normal distribution of $\mathbf{Y_i}$ is misspecified; by doing so, we replace 10 percent of the data with $\mathbf{Y_i^*} = -\eta_\mathbf{i} + \mathbf{e_i}$ and analyze data assuming the response having a normal distribution.

2. The variance-covariance matrix of the error term is misspecified. Here, instead of generating standard normal distributed errors, the variance-covariance matrix $\mathbf{R_i^*}$ has a first-order autoregressive structure, with $\rho = 0.5$. And we analyze data assuming the error term has a standard normal distribution.

3. The distribution of $\mathbf{b_i}$ is misspecified. We replace 20 percent of b_{1i}'s with Cauchy distributed random variables. We analyze the data assuming the $\mathbf{b_i}$'s are normally and independently distributed.

4. The variance-covariance matrix of the random effects is misspecified. Instead of using the original variance-covariance structure, we use $var(b^*_{0i}) = 4$, $var(b^*_{1i}) = 4$, and the variance-covariance matrix $\mathbf{G^*_i}$ has a first-order autoregressive structure with $\rho = 0.8$. We analyze the data assuming b_{0i} and b_{1i} are independent.

To compute the size of the population test, the following steps are used:

1. Generate one pass data under the null hypothesis which is $\beta_3 = 0$, and obtain the maximum likelihood estimate of β_3.

2. Compute the t test statistic $t = \widehat{\beta}_3/\widehat{SE}(\widehat{\beta}_3)$. Then the p-value is $P(|t| \geq t_c)$ where t_c is the critical value of t distribution given the degree of freedom and significance level, with 0.05 as the two-sided significance level.

3. Replicate 1-2 m times, and compute the proportion of times $p \leq 0.05$. This is the simulated size of the population test.

The procedure for computing the power of the population test is similar to computing the type I error rate; the difference is that we first generate data under the alternative hypothesis where $\beta_3 = 1$ or $\beta_3 = 1.5$. After that we obtain p-values for the test statistic using same algorithm. After replicating these two steps for m times, the proportion of times $p \leq 0.05$ is the power of the test.

To estimate the size of the randomization test, we use the following algorithm:

- Compute the critical value S_{cr} of the test:

   1. Generate one pass data under the case that there is no time-varying treatment effect ($\beta_3 = 0$), and obtain the maximum likelihood estimates of μ, β_1, β_2, ϕ and $\mathbf{G}$; predict $\widehat{b}_{1i}$ based on (3.6).
   2. Generate L randomization sequences, and merge each sequence with the predicted $\widehat{b}_{1i}$ obtained from 1.
   3. Compute the linear rank statistic for each sequence; here we use van der Waerden scores for normally distributed data.
   4. Order the L test statistics and find the critical value S_{cr} under the specified significance level.

- Find the size of the test:

   1. Similar to finding the critical value, we need to generate L randomization sequences, and merge each sequence to the residuals obtained from regression of the one pass data;
   2. Compute the linear rank statistic S for each sequence;
   3. Replicate 1-2 m times and find the average proportion of times $|S| \geq |S_{cr}|$; this is the estimated size of the test.

To compute the power of the test, we follow following steps:

1. Generate one pass data under the alternative hypothesis ($\beta_3 = 1$ or $\beta_3 = 1.5$), and obtain maximum likelihood estimates of μ, β_1, β_2, ϕ and $\mathbf{G}$ assuming $\beta_3 = 0$; predict the random slope $\widehat{b}_{1i}$'s based on (3.6).
2. Compute the linear rank test statistic using van der Waerden scores based on predicted random slopes and the corresponding treatment assignment from the data, this is the observed test statistic S_{obs}.

3. Generate L randomization sequences, merge $\widehat{b}_{1i}$'s with each sequence, and obtain L rank test statistics. The proportion of times that statistics greater than S_{obs} will be the estimated p-value.

4. Replicate 1-3 for m times, compute the proportion of times $p \leq 0.05$; this will be the power of the test.

4.2.2 Simulation Results

Figure 4.1 represents the histogram of the responses under the correct model. In Figure 4.2, the upper graph shows the distribution of the response of 90 percent data under the true model, and the lower graph shows the distribution of the 10 percent contamination. The lower graph Figure 4.3 shows the distribution of the 20% Cauchy outliers of random effects.

Tables 4.1-4.5, Figure 4.4-4.7 give the simulation results for the size and power under different situations. The results indicate that first, if the model is fitted correctly, and both the population test and randomization tests give robust performance. If the variance-covariance of error term is misspecified such as Scenario 2, neither of the tests is strongly affected. If the variance-covariance of random effects is misspecified such as Scenario 3, the size of both population and randomization tests remained at the significance level, but there is substantial power loss for both tests. However, if there are distributional assumptions misidentified such as Scenario 1 and 3, because the randomization test does not rely on distribution of the data, it provides higher power than the population tests.

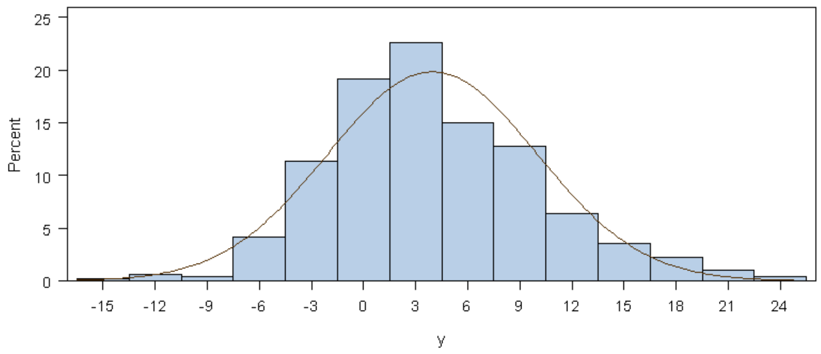

Figure 4.1: The distribution of the response under the true model

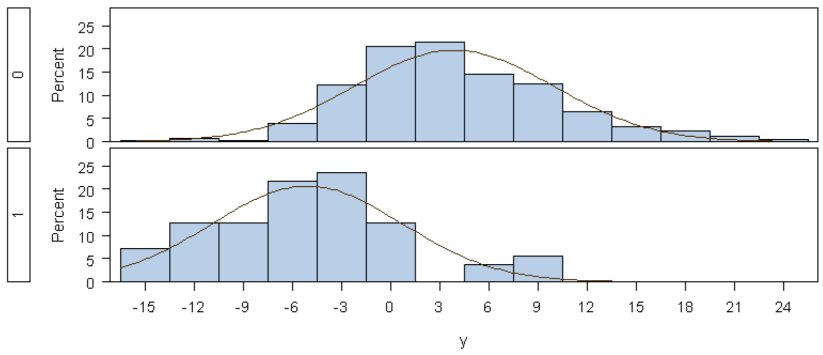

Figure 4.2: Comparison between the distributions of response under the true model and the response with 10% contamination

Table 4.1: *Size and power of the population and randomization tests under the correct GLMM ($L = 2500, m = 5000$).*

Test	Size (95% CI)	Power ($\beta_3 = 1$) (95% CI)	Power ($\beta_3 = 1.5$) (95% CI)
Population	0.0564	0.6968	0.9582
	(0.0500,0.0628)	(0.6841, 0.7095)	(0.9527, 0.9637)
Randomization (complete)	0.0440	0.6878	0.9558
	(0.0383,0.0497)	(0.6750, 0.7006)	(0.9501, 0.9615)
Randomization (conditional)	0.0426	0.6904	0.9546
	(0.0370,0.0482)	(0.6776, 0.7032)	(0.9488, 0.9604)
Randomization (BCD(2/3) unconditional)	0.0445	0.6918	0.9572
	(0.0387,0.0502)	(0.6790, 0.7046)	(0.9516, 0.9628)
Randomization (BCD(2/3) conditional)	0.0387	0.6824	0.9490
	(0.0333,0.0440)	(0.6695, 0.6953)	(0.9429, 0.9551)

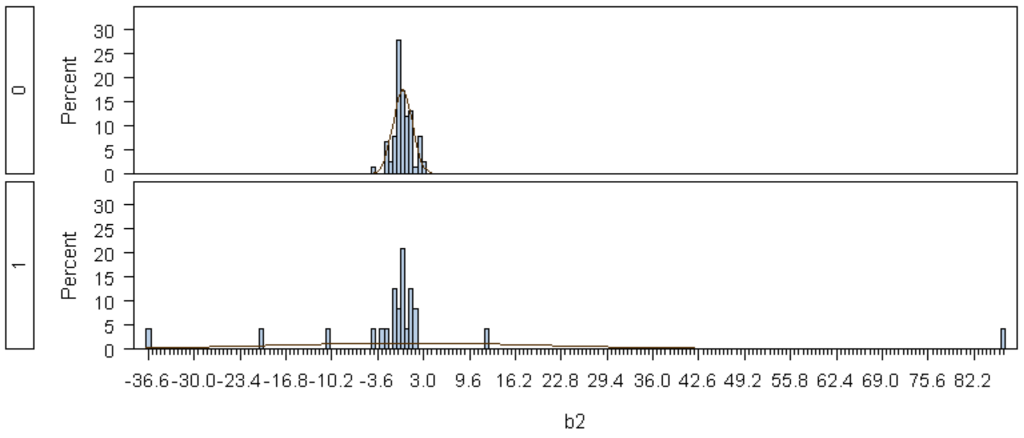

Figure 4.3: Comparison between the distributions of the random slope under the true model and the random slope with 20% of outliers

Table 4.2: *Size and power of the population and randomization tests under GLMM with misspecified response distribution ($L = 2500, m = 5000$).*

Test	Size (95% CI)	Power ($\beta_3 = 1$) (95% CI)	Power ($\beta_3 = 1.5$) (95% CI)
Population	0.0596	0.3738	0.6220
	(0.0530,0.0662)	(0.3604, 0.3872)	(0.6086, 0.6354)
Randomization (complete)	0.0550	0.4438	0.7470
	(0.0487,0.0613)	(0.4300, 0.4576)	(0.7349, 0.7591)
Randomization (conditional)	0.0390	0.4472	0.7476
	(0.0336,0.0443)	(0.4334, 0.4610)	(0.7356, 0.7596)
Randomization (BCD(2/3) unconditional)	0.0444	0.4346	0.7600
	(0.0387,0.0501)	(0.4209, 0.4483)	(0.7482, 0.7718)
Randomization (BCD(2/3) conditional)	0.0459	0.4264	0.7434
	(0.0401,0.0517)	(0.4127, 0.4401)	(0.7313, 0.7550)

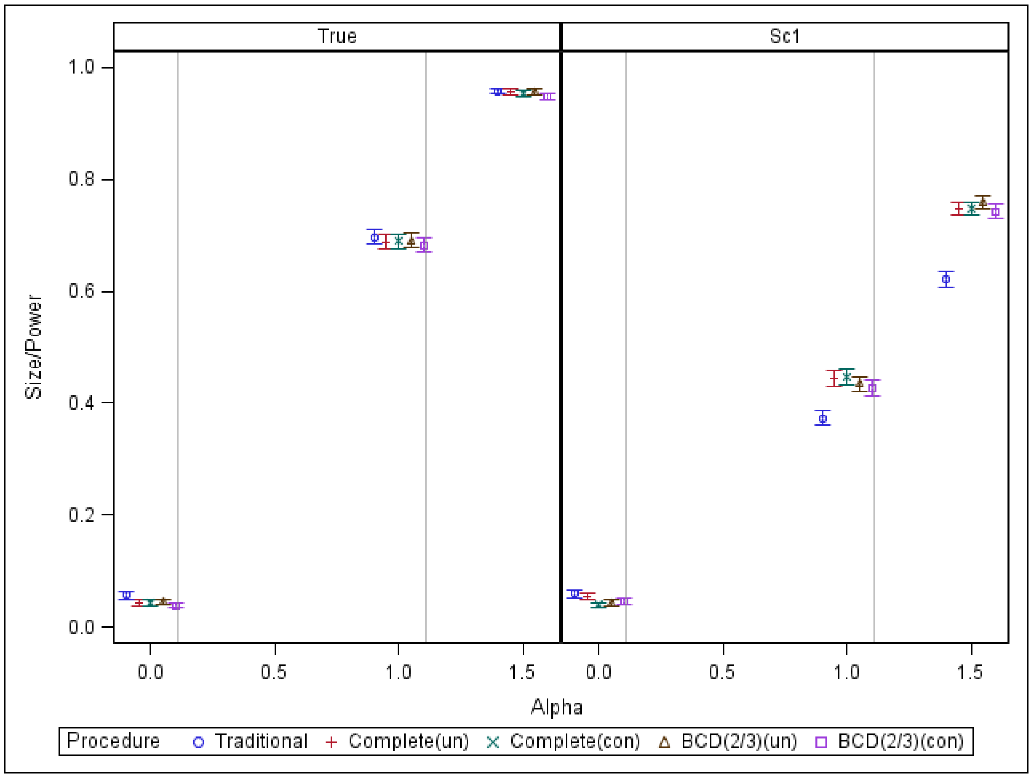

Figure 4.4: Size and Power of the population and randomization tests under GLMM with misspecified response distribution.

Table 4.3: *Size and power of the population and randomization tests under GLMM with misspecified variance-covariance structure for error term ($L = 2500, m = 5000$).*

Test	Size (95% CI)	Power ($\beta_3 = 1$) (95% CI)	Power ($\beta_3 = 1.5$) (95% CI)
Population	0.0558	0.6888	0.9552
	(0.0494,0.0622)	(0.6760, 0.7016)	(0.9495, 0.9609)
Randomization (complete)	0.0423	0.6732	0.9488
	(0.0367,0.0479)	(0.6602, 0.6862)	(0.9427, 0.9549)
Randomization (conditional)	0.0418	0.6802	0.9514
	(0.0363,0.0473)	(0.6673, 0.6931)	(0.9454, 0.9574)
Randomization (BCD(2/3) unconditional)	0.0478	0.6762	0.9530
	(0.0419,0.0537)	(0.6632, 0.6891)	(0.9471, 0.9589)
Randomization (BCD(2/3) conditional)	0.0337	0.6634	0.9438
	(0.0287,0.0387)	(0.6503, 0.6765)	(0.9374, 0.9502)

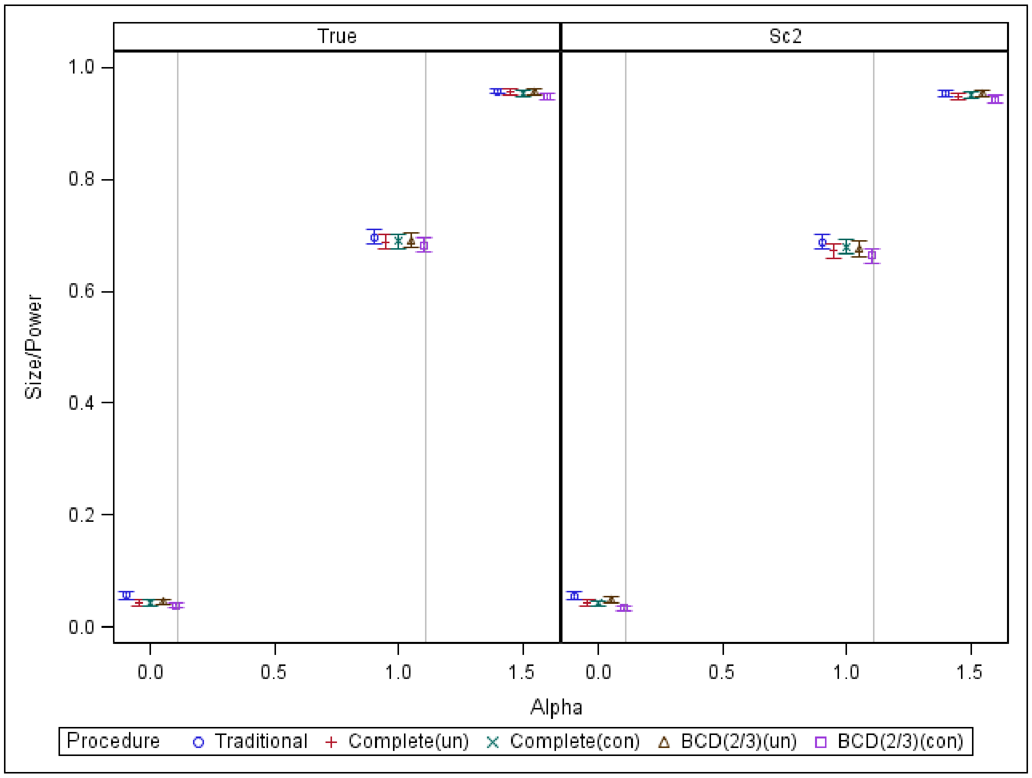

Figure 4.5: Size and Power of the population and randomization tests under GLMM with misspecified variance-covariance structure for error term.

Table 4.4: *Size and power of the population and randomization tests under GLMM with misspecified distribution for random effects ($L = 2500, m = 5000$).*

Test	Size (95% CI)	Power ($\beta_3 = 1$) (95% CI)	Power ($\beta_3 = 1.5$) (95% CI)
Population	0.0288	0.2190	0.3792
	(0.0242,0.0334)	(0.2075, 0.2305)	(0.3658, 0.3926)
Randomization (complete)	0.0451	0.5232	0.8488
	(0.0393,0.0509)	(0.5094, 0.5370)	(0.8389, 0.8587)
Randomization (conditional)	0.0440	0.5246	0.8598
	(0.0383,0.0497)	(0.5108, 0.5384)	(0.8502, 0.8694)
Randomization (BCD(2/3) unconditional)	0.0387	0.5460	0.8686
	(0.0334,0.0440)	(0.5322, 0.5598)	(0.8592, 0.8780)
Randomization (BCD(2/3) conditional)	0.0465	0.5224	0.8588
	(0.0407,0.0523)	(0.5086, 0.5362)	(0.8491, 0.8685)

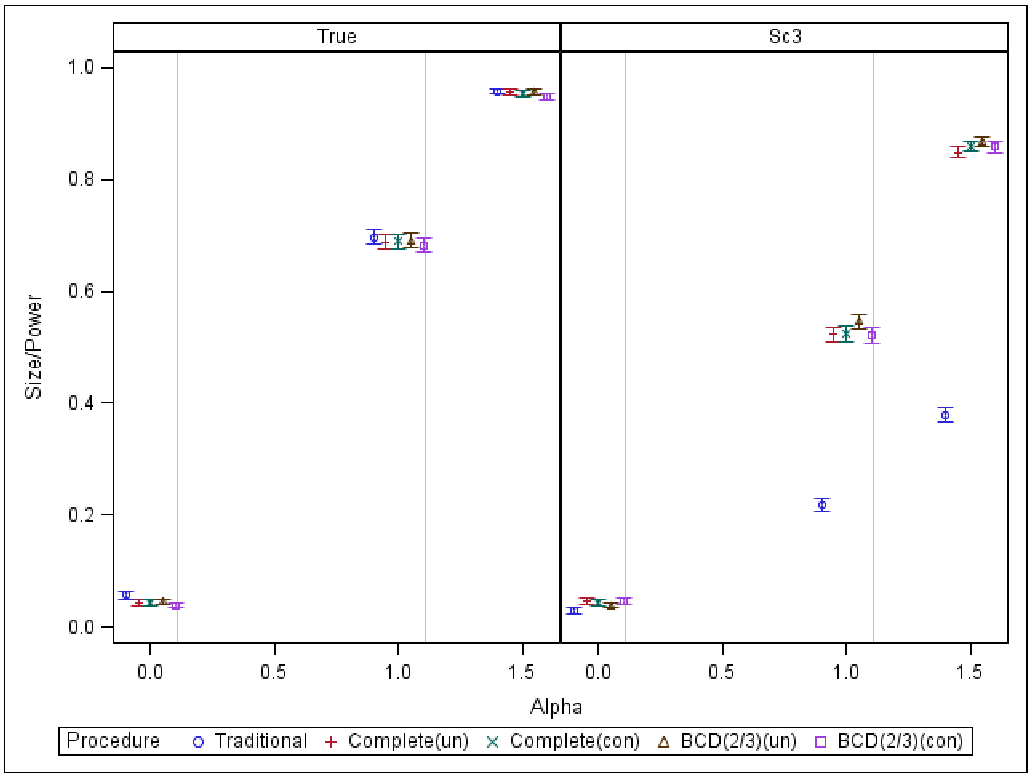

Figure 4.6: Size and Power of the population and randomization tests under GLMM with misspecified distribution for random effects.

Table 4.5: *Size and power of the population and randomization tests under GLMM with misspecified variance-covariance structure for random effects ($L = 2500, m = 5000$).*

Test	Size (95% CI)	Power ($\beta_3 = 1$) (95% CI)	Power ($\beta_3 = 1.5$) (95% CI)
Population	0.0456	0.3884	0.7282
	(0.0398,0.0514)	(0.3749, 0.4019)	(0.7159,0.7405)
Randomization (complete)	0.0395	0.3722	0.7134
	(0.0341,0.0445)	(0.3588, 0.3856)	(0.7009, 0.7259)
Randomization (conditional)	0.0403	0.3824	0.7158
	(0.0348,0.0458)	(0.3689, 0.3856)	(0.7033, 0.7283)
Randomization (BCD(2/3) unconditional)	0.0375	0.3696	0.7182
	(0.0323,0.0428)	(0.3562, 0.3830)	(0.7057, 0.7307)
Randomization (BCD(2/3) conditional)	0.0445	0.3748	0.7206
	(0.0388,0.0502)	(0.3614, 0.3882)	(0.7082, 0.7330)

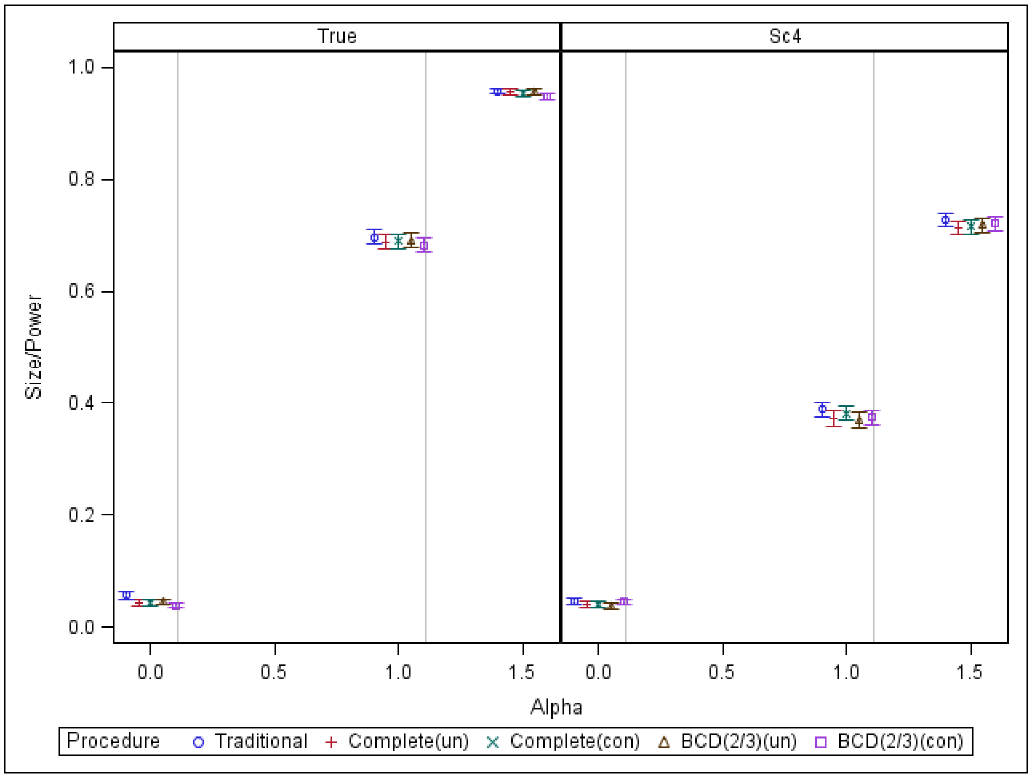

Figure 4.7: Size and Power of the population and randomization tests under GLMM with misspecified variance-covariance structure for random effects.

Chapter 5: Macro Series for Randomization Tests

In this chapter we describe details of a series of SAS macros which compute p-values for conditional and unconditional randomization tests. These macros will provide convenient use of randomization tests in a variety of clinical trial data. These macros are given in Appendices A-D.

In the first section we provide an overview of the macros, in the second section we introduce the algorithm and describe input for each macro. In the third section we will give examples using each macro. We discuss computer resources and computing time in the third section.

5.1 Overview of the Macros

We now describe a series of SAS macros for randomization-based inference: RANDIRECT, RANGLM, RANGLMM and RANSURVIVAL. The macro RANDIRECT, found in Appendix A, is used when a simple randomization test is performed on binary or continuous outcome with no covariate information. The macro RANGLM, found in Appendix B, is used when it is necessary to build a generalized linear model. The macro RANGLMM, found in Appendix C, is used when the data are longitudinal trial data and the aim is to detect a time-varying treatment effect if the repeated measure is time. The macro RANSURVIVAL, found in Appendix D, is used when the outcomes are time to event data or survival modeling is required.

Each macro consists two parts: the first part extracts information from the given data set and gathers or generates the needed outcome for the randomization test; the second part uses Monte Carlo methods generate randomization sequences under a specified randomization procedure and test statistic, and obtains p-value of the test. The output of each macro will contain information about the size of the data, the randomization procedure, whether the test is one-sided or two-sided, and the p-value of the test.

In the first part of RANGLM and RANGLMM, the GLIMMIX procedure is called to perform the appropriate regression and obtain corresponding residuals or random slope predictors, while in RANSURVIVAL, there are two types of regression model– the proportional hazard (PH) model and the accelerated failure time (AFT) model. The PHREG procedure is called for the PH model, and the LIFEREG procedure is called for the AFT model. In the second part of all macros, where randomization tests are applied, SAS Interactive Matrix Language (IML) is used, as the matrix computation helps to reduce computing time.

There are a number of similar features for these four macros. First, all input data have to be complete, as there is no consideration of missing values in the macro. Second, all macros will compute both unconditional and conditional tests upon request. Third, all macros can be performed for randomization procedures for complete randomization, Efron's biased coin design (EBCD), and the permuted block design (PBD) with truncated binomial design (TBD) or the PBD with random allocation rule (RAR). Fourth, the number of randomization sequences can be set to a desired number. Finally, the testing procedure can be replicated many times to simulate the size or power of the test if desired.

There are also special features and requirements for the input parameters of these macros. For the RANDIRECT macro, the input data is required to contain the treatment indicator and outcome variable. For the RANGLM macro, besides outcome and treatment variables, the input data is required to contain information for the generalized linear regression, such as prognostic factors if there are any, and there are options for input parameters regarding regression, such as the distribution and link function. For the RANGLMM macro, the input data has to contain repeated measures and subject numbers, the input parameters are also required for the corresponding generalized linear mixed model, such as the type of covariance matrix associated for the random effects. For the RANSURVIVAL macro, users have options to choose whether to use a regression model (PH and AFT) or not, and the input data could either be a simple data set containing treatment, censoring and failure time, or a data containing additional covariates. If the PH model or the AFT model is specified, there are options for additional input parameters regarding the regression such as name of the covariates, the distribution and link function.

5.2 Algorithms for the Macros

The following sections introduce details of each macro.

5.2.1 Macro RANDIRECT (Appendix A)

RANDIRECT is used to perform a simple randomization test given the treatment and outcome of a clinic trial data. In this case, there is no need to apply any regression to the data. The macro allows the users to compute the p-value of the test under specified conditions. The following is a description of what RANDIRECT does:

1. PROC SQL is used to compute the size of the data and the number of subjects assigned to the treatment 1 (for the conditional test) if necessary. A new data set is generated with the outcome and treatment variables based on the given

data.

2. The macro TEST will be called to compute the randomization test and p-value with the specified procedure and linear rank statistic. The macro TEST is programmed using SAS/IML and it contains three steps:

    - Compute the observed test statistic S_{obs} with specified score given the new data set from 1.
    - Generate randomization sequences under a specified randomization procedure and type of test statistic. For each sequence, compute the test statistic S, and compare it to the observed test statistic.
    - Repeat the second step for a specified number of times. For a two-sided test, the proportion of times $|S| \geq S_{obs}$ will be the p-value of the test. One can also choose upper or lower bound one-sided p-values.

The input variables for RANDIRECT are:

INDATA:	Data set to be analyzed. This data set has to have at least two variables–the response variable, either continuous or categorical, and a treatment indicator variable. Both variables are numeric.
RESPONSE:	Name of the response or outcome variable from the data above to be used for score computation in randomization test.
TRT:	Name of the variable that defines treatment. Must be a binary variable with values 0 or 1.
TYPE:	Name of the type of the reference sets (UNCONDITIONAL or COND ITIONAL). The default value is UNCONDITIONAL.

RANPRO:	Randomization procedure (COMPLETE, EBCD, PBD). The default value is COMPLETE. When the EBCD is specified, p is required for the bias probability. When PBD is specified, the blocksize and the randomization procedure used in each block (TBD, RAR) are needed.
NUMREPS:	Number of time the inference procedures is replicated. Results can be used to simulate the size and power of the test. The default value is 1.
NUMSEQ:	Number of sequences to be used for randomization. The default and minimum value is 2500.
SCORE:	Score function to be used in the randomization test, one can choose either WILCOXON or VAN DER WAERDEN.
SEED:	Seed to be used for generating randomization sequences. The default value is 1.
P:	The bias probability used for the EBCD procedure.
BLKSIZE :	The block size for the PBD procedure.
PROINBLK:	Randomization procedure to be used for the PBD: TBD or RAR.
PVALUESIDE:	Specify whether it is a two-sided or one-sided p-value. One can choose one of the UPPER, LOWER and TWO SIDED three options. TWO SIDED is the default.

The output will be the simple p-value of the test specified with the randomization procedure and test type. The sample code and output will be provided in Section 5.2.

5.2.2 Macro RANGLM (Appendix B)

RANGLM is used when the data are to be analyzed using a generalized linear model (GLM). The following are the steps performed by RANGLM:

1. We apply a generalized linear model to the response variable with covariates, and obtain residuals for each subject. This step is performed by the GLIMMIX procedure in SAS with the specified distribution and link function. As there is no treatment difference between two groups under the null hypothesis, no treatment variable is involved in the regression.

2. We create a new data with the residuals and corresponding treatments for each subject. The residuals are the outcome variable for the randomization test. We also compute the size of the data and the number of assignments to treatment group if conditional test is requested.

3. Similar to the RANDIRECT macro, we call the TEST macro to perform randomization tests for the new outcome data and obtain the p-value.

Most of the input parameters for the RANGLM macro are similar to the RANDIRECT macro. We list the following parameters which are unique to RANGLM:

INDATA:	Data set to be analyzed. This data set has to include the response variable which is used to perform the GLM regression, the treatment indicator, and other covariates to be used in the regression. The response and treatment variable are numeric, covariates need not to be numeric, but if there are categorical covariates, they have to be specified in CLASS option.
METHOD:	The estimation method to be used. Options include: residual psudo-likelihood based on random effects solution (RSPL), maximum psudo-

	likelihood based on random effects solution (MSPL), residual psudo-likelihood based on mean of the random effects (RMPL), maximum psudo-likelihood based on mean of the random effects (MMPL), maximum likelihood with Laplace approximation (LAPLACE), and maximum likelihood with adaptive quadrature (QUAD). The quadrature nodes can also specified within parentheses after in QUAD option (e.g. QUAD(QPOINTS=)).
RESPONSE:	Name of response or outcome variable from the data to be used for the GLM.
COV:	Name of variables to be used as covariates for the GLM regression in the MODEL statement of GLIMMIX.
CLASS:	Name of variables to be used in the CLASS statement of the GLIMMIX procedure.
DIST:	Distribution of the data to be used in GLM. Default value is NORMAL. Distribution options include: BETA, BINARY, BINOMIAL, EXPONENTIAL, GAMMA, NORMAL, GEOMETRIC, INVGAUSS, LOGNORMAL, MUTINOMIAL, NEGBINOMIAL, POISSON and T-CENTRAL.
LINK :	Link function to be specified for the GLM regression. The default value is identity. Link function options include: CUMCLL, CUMLOGIT, CUMLOGLOG, CUMPROBIT, CLOGLOG, GLOGIT, IDENTITY, LOG, LOGIT, LOGLOG, PROBIT, POWER, POWERMINUS2 and RECIPROCAL.

The output will gives the p-value for the specified test based on scores of residuals obtained from the regression.

5.2.3 Macro RANGLMM (Appendix C)

The macro RANGLMM is used when a mixed model is needed, such as when the clinical trial data are longitudinal data. While RANGLM uses the residuals of the model, RANGLMM uses predictors of random slopes as the outcome. Other computing steps are similar to RANGLM:

1. We apply a mixed model to the response variable with covariates, given a distribution and link function. For the random effect, we assume there will be a random intercept and random slope. Users can specify the type of covariance matrix separately for random effects and residuals. We obtain the predictors for the random slope from the output for each subject. This step is also performed using the GLIMMIX procedure in SAS. The treatment variable might be considered as one of the fixed effects in the model.

2. We create a new data set with predicted random slope and corresponding treatment assignment for each subject, and count the size of the data and the number of assignments to treatment 1 if necessary.

3. Similar to previous macros, we call the TEST macro to perform the randomization test to the data and obtain the *p*-value.

The following are input parameters that are unique to RANGLMM. Other input variables are the same as previously introduced macros.

INDATA:	Data set to be analyzed. It has to include the response variable which is used to perform the mixed model regression, the subject variable which specifies subjects of the data such as patient ID, the repeated measure, a treatment indicator, and other covariates if any. The response variable, subject variable, repeated measure and treatment indicator are numeric; covariates can be either numeric or categorical.
RESPONSE:	Name of the response or outcome variable from the data to be used for the mixed model.
GTYPE:	Specifies the covariance structure of the random effects. By default, the covariance structure is UN. The options of GTYPE include: ANTE(1), AR(1), ARH(1), ARMA(1,1), CHOL, CS, CSH, FA, FA0, HF, LIN, PSPLINE, RSMOOTH, SIMPLE, SP(EXP), SP(GAU), SP(MAT), SP(POW), SP(POWA), SP(SPH), TOEP, TOEPH, UN, UNR, and VC.
RTYPE:	Specifies the covariance structure of residuals. The default value is the identity matrix. The choices for RTYPE are the same as GTYPE.
SUB :	Name of the variable that specifies subjects of the data.
REPEAT:	Name of the variable that specifies repeated measures of the data, such as time of repeated treatment for the same subject.

The output will give the p-value for the specified test based on scores of the random slope predictors obtained from the GLMM.

5.2.4 Macro RANSURVIVAL (Appendix D)

RANSURVIVAL is a macro especially programmed for applying the randomization tests to survival data. Users can apply this macro either directly compute the p-values of the test, or apply regression models and use the residuals of the regression and perform randomization test. For the parametric models, one can either choose the PH model or the AFT model. The procedures are as follows:

1. We generate a new data set with the outcome, censoring information, and treatment variable based on the given data set.

    - If there is no model specified, the new data set is just an extraction from the original data.
    - If the PH model is specified, we first compute the martingale residuals using PHREG procedure, and use these residuals as outcome for the randomization test.
    - If the AFT model is specified, we use LIFEREG procedure to apply the regression and compute the martingale residuals as the outcome. One can choose whether to use the log transform, whether to include intercept in the regression.

2. We call another macro named STEST which computes the randomization test and gives p-values of the test. Similar to macro TEST, the macro STEST is also programmed using SAS/IML, the procedure of computing p-values is also similar. The difference is that the score function for the data without modeling is set to be SAVAGE, while for the PH or the AFT model, the score function is WILCOXON.

We include input parameters that are unique to RANSURVIVAL as follows. Other inputs are similar to previous macros.

INDATA:	Data set to be analyzed. It has the following variables: the response variable which is used to perform the randomization test, the censoring indicator if any, a treatment indicator variable, and other covariates to be included in the regression.
RESPONSE:	Name of the response variable to be used for regression if model is specified or randomization test if no model is specified.
CENSOR:	Variable that defines censoring status. We consider value 0 as censored, and 1 as uncensored. By default, there is no censoring.
MODEL:	Types of model to be used for data. Values include NONE, PH, AFT. By default, there is no modeling.
COV :	Name of the covariates to be used for regression if model is PH or AFT.
DIST:	Variable that specifies the distribution type for failure time if the AFT model is used. Options include EXPONENTIAL, GAMMA, LLOGISTIC, LNORMAL, LOGISTIC, NORMAL and WEIBULL. The default distribution is a type 1 extreme-value distribution to the log of the response.
NOLOG :	For the AFT model, it requests whether to use log transformation or not. By default, there is log transformation.
NOINT :	For the AFT model, it requests whether intercept is included in the regression. By default, the intercept is included.

The output will give the *p*-value for the specified test and models.

5.3 Examples

In this section, we give examples using the series of macro to perform randomization tests on different types of data, with different types of tests and randomization procedures.

5.3.1 Example of RANDIRECT

We provide an example from the Diabetes Complications and Control Trial, using the data from Table 7.4 of Rosenberger and Lachin (2002, p. 112). The data set contains three variables: the level of cholesterol of the patient (chol), treatment indicator (tx) and the number of observations. The following is the statement we submit to perform a conditional complete randomization test, we specify the number of sequences to be 10000, the test statistic to be van der Waerden:

```
%randirect(indata=one,
           response =chol,
           trt= tx,
           type=conditional,
           ranpro=complete,
           numseq=10000,
           score='van der waerden').
```

The output of the macro is shown in Figure 5.1.

5.3.2 Example of RANGLM

We provide an example from a placebo-controlled clinical trial of 59 epileptics, using the data from Table 2 in Thall and Vail (1990, p. 664). In this study, patients are

UNCONDITIONAL Test Result for COMPLETE Randomization

Obs	n	Scores	Alternative	P_value
1	50	VAN DER WAERDEN	TWO SIDED	0.6313

Figure 5.1: Output of RANDIRECT Macro

randomized into treatment group (standard anti-epileptic chemotherapy) and placebo group, and the duration of the trial is 8 weeks. The seizure count was recorded prior the trial, and at 2-weeks intervals during the trial. The data set contains the following variables: patient ID, treatment indicator, age, the baseline seizure count prior the trial, and the seizure counts from week 2 to week 8. Here we only consider the treatment effect in week 8. The following is the statement we submit to perform a unconditional PBD randomization test with blocksize 4 and TBD within each block, after a GLM regression with Poisson distribution with log link. We specify the number of sequences to be 20000, and the score function to be Wilcoxon:

```
%ranglm(indata=b,
        response =week8,
        cov=age,
        trt= treatment,
        dist=poisson,
        link=log,
        type=conditional ,
        ranpro=pbd ,
        numseq=20000,
        score='wilcoxon',
        seed=12345,
        blksize=4,
        proinblk=tbd
              ) ;
```

The output of the macro is shown in Figure 5.2.

CONDITIONAL Test Result for PBD with TBD Randomization

Obs	n	n1	Scores	Alternative	P_value
1	59	31	WILCOXON	TWO SIDED	0.2545

Figure 5.2: Output of RANGLM Macro

5.3.3 Example of RANGLMM

We continue with the previous seizure data example. There are some structural changes made for the data. A new response variable is created by combining all seizure count from base week to week 8. The repeated measure visit is created by listing the number of visits for each subject. The revised data set contains following variables: Y (response), ID (subject), VISIT (repeated measure), TREATMENT, AGE (covariate). We first apply a mixed model under the Poisson distribution, then perform a conditional EBCD randomization test with $p = 2/3$. We specify the number of randomization sequences to be 20000, and the score function to be van der Waerden. The following are the submitted statements:

```
%ranglmm(indata=x,
         response = y,
         cov= age trt,
         class=id,
         dist=poisson,
         link=log,
         gtype=ar(1),
         trt= trt,
         sub=id,
         repeat=visit,
         type=conditional ,
         ranpro=ebcd,
         numseq=20000,
         score='van der waerden',
         seed=12345,
         p=2/3
              ) ;
```

The output of the macro is shown in Figure 5.3.

5.3.4 Example of RANSURVIVAL

We provide an example from a lung cancer clinical trial of 137 patients, described by Kalbfleisch and Prentice (1980, pp. 223-224). In this study, males with advanced inoperable lung cancer were randomized into a treatment group (a test chemotherapy) and a control group (a standard therapy). The purpose of this study is to compare the survival time of patients between two therapies. In this data set, the outcome

CONDITIONAL Test Result for EBCD(p=2/3) Randomization

Obs	size	n1	score	side	p_value
1	59	31	VAN DER WAERDEN	TWOSIDED	0.0219

Figure 5.3: Output of RANGLM Macro

variable is SURVIVALTIME, censoring indicator variable is STATUS, treatment indicator is TREATMENT. There are other explanatory variables including: types of tumor cell (adeno, large, small, or squamous), prior (prior therapy: 0 = no, 10 = yes), age, duration (months from diagnosis to randomization), and performance (Karnofsky performance scale). Here we first consider a Weibull AFT model, where SURVIVALTIME is the response, PERFORMANCE, DURATION and STATUS are covariates. We apply the unconditional EBCD randomization test with 10000 randomization sequences, and request a one-sided p-value. The following is the submitted code:

```
%ransurvival(indata=vet,
             response=survivaltime,
             censor=status,
             model='aft',
             cov=age performance duration,
             dist=weibull,
             noint=1,
             trt= trt,
             type=unconditional,
             ranpro=ebcd,
             numseq=10000,
             p=2/3,
             pvaluside='upper'
           ) ;
```

The output of the macro is shown in Figure 5.4.

UNCONDITIONAL Test Result for EBCD (p=2/3) Randomization

Obs	n	Scores	Alternative	P_value
1	137	WILCOXON	UPPER	0.3036

Figure 5.4: Output of RANSURVIVAL Macro

5.4 Computer Resources and Computing Time Requirement

As previously mentioned, though it is powerful tool to assess treatment effects in clinical trials, the randomization test is not widely used because it requires extensive computation. While more time may be needed to compute randomization tests compared to traditional tests, because of the fast development of computer technology, the issue should not be a significant concern.

The major advantage of the series of macros is it reduces the run time of the test by using the IML procedure, instead of generating data sets or using data step loops.

To implement a randomization test, one has to generate a large number of randomization sequences based on the randomization procedure, merge with the original data, compute the tests statistic and replicate the whole process many times. Calling loops for generating randomization sequences or generating datasets to store the sequences would use too much computer memory and CPU time so that it slows down the computation of p-values. In the macro, as IML is used, a matrix is mainly used for generating the randomization sequences, and the test statistic is computed instantly without saving to a data set. The p-value is also computed in IML; therefore, in our series of macros, the run time of the macro to compute the randomization test is mainly the execution time of IML. For most of the randomization procedures, IML gives results in a reasonable amount of time. Moreover, the ODS option is applied in RANGLM RANGLMM and RANSURVIVAL to suppress unnecessary output, and this also helps to reduce the total time of performing the randomization test.

High-speed computing is another useful resource to improve the efficiency of computing the randomization test. All the simulation and macro computation in this thesis is performed in SAS 9.3 by using a DELL desktop computer. The processor of the computer is Intel(R) Core(TM) i5 with speed of 3.2GHz, the hard drive is 500GB and the internal memory is 8GB, the system is Windows 7 64-bit operating system. For a simple unconditional complete randomization test with 1000 Monte Carlo replications, the run time is about 6 minutes. While for the conditional EBCD test with 10 Monte Carlo replications, the run time is about 1.5 minutes.

Table 5.1 and Table 5.2 show the computing time of four macros with different cases. It indicates the run time is highly depending the complexity of generating the randomization sequences. As the unconditional test has less restriction than conditional tests, the run time is generally short. For the conditional test, EBCD

Table 5.1: *The computing time for the macros given different types of the tests, procedures and replications with number of sequences 2500.*

Macro	Procedure	Unconditional Tests		Conditional Tests	
		10 Reps	100 Reps	10 Reps	100 Reps
RANDIRECT	Complete	1.30 sec	9.43 sec	1.29 sec	12.94 sec
RANGLM	EBCD	2.18 sec	22.30 sec	1 min 22 sec	14 min 12 sec
RANGLMM	PBD (TBD)	5.33 sec	54.26 sec	5.81 sec	57.79 sec
RANSURVIVAL	PBD (RAR)	6.67 sec	1 min 1 sec	7.26 sec	1 min 29 sec

Table 5.2: *The computing time for the macros given different types of the tests, procedures and randomization sequences with one replication.*

Macro	Procedure	Unconditional Tests		Conditional Tests	
		100000 Seq	1000000 Seq	100000 Seq	1000000 Seq
RANDIRECT	Complete	3.75 sec	37.78 sec	5.27 sec	52.18 sec
RANGLM	EBCD	8.98 sec	1 min 34 sec	5 min 40 sec	59 min 38 sec
RANGLMM	PBD (TBD)	13.13 sec	2 min 12 sec	14.13 sec	2 min 20 sec
RANSURVIVAL	PBD (RAR)	28.49 sec	4 min 32 sec	31.13 sec	5 min 14 sec

takes the longest time. Based on the conditional distribution development for E-BCD (Plamadeala and Rosenberger, 2012), the formula of computing the probability of treatment assignment is complicated. However, it reduces the time to generate massive numerous of unconditional sequences.

Chapter 6: Conclusions

We have applied the randomization test, given unconditional and conditional reference sets, for data based on the generalized linear model, two survival models, and the generalized linear mixed model. This application provides a viable option to detect the treatment effect in clinical trials when the assumptions for the traditional inference are violated when there are misspecifications in the population models. It accommodates different randomization procedures including complete randomization, Efron's biased coin design and the permuted block design. All of these procedures are implemented in a user-friendly series of SAS macros. These SAS macros are flexible and comprehensive, allowing the clinical trialist to compute a randomization test for different data structure when restricted randomization is used.

For the generalized linear model, we provide an algorithm for computing p-values using Gail, Wieand, and Piantadosi's strategy (1988), where the score residuals, from the population model are used as the outcome of the randomization test. For Efron's biased coin design, we compute the conditional test by generating randomization sequences directly from the conditional reference set (Plamadeala and Rosenberger, 2012). Simulation results show the size of the randomization test maintains the nominal level when necessary covariates are omitted, while the size of the population test is inflated for response distributions including the normal, Poisson, and exponential.

We also developed randomization tests for survival data by using martingale residuals from the proportional hazard model and the accelerated failure model. Simulations indicate that in the case of misspecification of distribution of the survival time, randomization tests have better power and size than the traditional test.

For the generalized linear mixed model, we have shown how to perform the randomization test to detect a time-varying treatment effect for the longitudinal data. The outcome used for the test is the predictor of the random slope for each subject from the generalized linear mixed regression model. We explore by simulation the size or power of the test when the distribution of the error term or the response is misspecified. Our simulation results show the randomization test should be considered as an alternative to the traditional test in the following cases: when there are misspecifications of the distribution of the response, and when there are outliers for the random effects. In the case of misspecification of the covariance structure of the error term or random effects, there is insignificant impact to the size and power of the population test.

We have also developed a series of SAS macros for computing the randomization test for different types of data. This package consists of four macros: RANDIRECT, RANSURVIVAL, RANGLM and RANGLMM. Clinical trialists can apply these macros to compute p-values of the randomization test with or without modeling, given a specified restricted randomization procedure and the type of reference set. Interactive Matrix Language (IML) is used in the macro for computing the p-value of the test. It helps to reduce the run time, and all macros work well for large number of Monte Carlo sequences. This series of macros will provide convenience for researchers who intend to apply the randomization test.

There are several interesting open research topics that can be addressed for the randomization test in the future. First, we did not consider any data with missing values in our research and it is still an open problem for how to deal with data containing missing values. Rosenberger and Lachin (2002) proposed to use a stratum-specified rank score by putting complete and missing records into two strata. For the complete records, the score values are functions of the outcome while for the missing records, scores are undefined. If the missing data are independent, one can apply the randomization test by taking a subset of the data with complete information. Kennes, Hilgers and Heussen (2012) examined two methods of performing randomization tests on data with missing values, one is to keep records with missing values and assigning 0 to missing observations and imputing the lowest rank; the other one is to eliminate the records with missing values. They applied a randomization test based on the random allocation rule and simple linear rank sum test statistic. Their simulation results do not show significant differences between the two methods.

Second, there are other randomization procedures that could be included. These designs include the class of generalized biased coin designs (Smith, 1984), stratified designs and response-adaptive randomization procedures. The exact conditional distribution of the generalized biased coin design is still unknown so computing the conditional test for this procedure using conditional probabilities is still an open problem. Simon and Simon (2011) described an unconditional randomization test for response-adaptive and covariate-adaptive randomization procedures, where the sample size is finite, and type I error rate is well controlled. A new macro could be developed using a similar strategy for adaptive designs as well as the stratified design if the computing complexity is solved.

There are limitations of this research. The data and modeling in our simulations are based on specific distributions, covariates and parameters; they are not exhaustive in terms of every possible situation encountered in clinical trials.

Appendix A: Macro RANDIRECT

```
/********************************************************************
Macro Name:  RANDIRECT
PROGRAM: Macro for randomization tests
PURPOSE: Macro that allows to easily compute p-values of randomization
         tests for various data.
CREATED BY: Parwen Parhat, George Mason University
DATE CREATED: January 2013
********************************************************************
Inputs:

INDATA       Data set to be analyzed.  This data set has to have at least
             two variables--the response variable, either continuous or
             categorical, and a treatment indicator variable. Both
             variables are numeric.

Response     Name of the response or outcome variable from the data above
             to be used for score computation in randomization test.

TRT          Name of the variable that defines treatment. Must be a binary
             variable with values 0 or 1.

Type         Name of the type of the randomization tests (UNCONDITIONAL or
             CONDITIONAL). The default value is UNCONDITIONAL.

RANPRO       Randomization procedure (COMPLETE, EBCD, PBD). The default
             value is COMPLETE. When the EBCD is specified, p is required
             for the biased probability. When PBD is specified, the
             blocksize and the randomization procedure (TBD, RAR) used in
```

```
            each block are needed.

NUMREPS     Number of time the inference procedures is replicated. Results
            can be used to simulate the size and power of the test. The
            default value is 1.

NUMSEQ      Number of sequences to be used for randomization. The default
            and minimum value is 2500.

SCORE       Score function to be used in the randomization test, one can
            choose either 'WILCOXON' or 'VAN DER WAERDEN'.

SEED        Seed to be used for generating randomization sequences. The
            default value is 1.

P           The biased probability used for the EBCD procedure.

BLKSIZE     The block size for the PBD procedure.

PROINBLK    Randomization procedure to be used for the PBD: TBD or RAR.

PVALUESIDE  Specify whether it is a two-sided or one-sided p-value. One
            can choose one of the 'UPPER', 'LOWER' and 'TWO SIDED' three
            options. 'TWO SIDED' is the default.
**************************************************************************
**************************************************************************
**************************************************************************/;
%macro randirect(indata=,
             response =,
             trt= ,
             type= ,
             ranpro= ,
             numreps=,
             numseq=,
             score=,
             seed=,
             p=,
             blksize=,
             proinblk=,
             pvalueside=
             );

/*Warnings of inappropriate use of the program*/
%if %length(&indata)=0  %then %do;
      %put ALERT ALERT ALERT: INDATA  must be specified;
      %goto stopmac;
%end;
%if %length(&trt)=0  %then %do;
      %put ALERT ALERT ALERT: TRT  must be specified;
      %goto stopmac;
%end;
%if (%length(&type) ne 0 ) and ( %upcase(&type) ne UNCONDITIONAL) and
( %upcase(&type) ne CONDITIONAL) %then %do;
      %put ALERT ALERT ALERT: TYPE  must be UNCONDITIONAL  or CONDITIONAL;
      %goto stopmac;
%end;

%if %index(COMPLETE EBCD PBD, %upcase(&ranpro))=0 %then %do;
 %if (%length(&ranpro) ne 0) %then %do;
      %put ALERT ALERT ALERT: RANPRO  must be one of COMPLETE, EBCD, PBD;
      %goto stopmac;
 %end;
```

```
%end;

%if %sysevalf(&numreps<1) and  (%length(&numreps) ne 0)  %then %do;
      %put ALERT ALERT ALERT: NUMREPS  must be at least 1;
      %goto stopmac;
%end;

%if %sysevalf(&numseq<2500) and  (%length(&numseq) ne 0)  %then %do;
      %put ALERT ALERT ALERT: NUMSEQ  must be at least 2500;
      %goto stopmac;
%end;

%if %index(‘WILCOXON’ ‘VAN DER WAERDEN’, %upcase(&score))=0  %then %do;
 %if (%length(&score) ne 0) %then %do;
      %put ALERT ALERT ALERT: SCORE  must be one of WILCOXON,
      VAN DER WAERDEN;
      %goto stopmac;
 %end;
%end;

%if %sysevalf(&seed<0) and %length(&seed) ne 0 %then %do;
      %put ALERT ALERT ALERT: SEED  must be at least 1;
      %goto stopmac;
%end;

%if (%upcase(&ranpro)=EBCD) and  (%length(&p) = 0) %then %do;
      %put ALERT ALERT ALERT: P  must be at specified with EBCD procedure;
      %goto stopmac;
%end;

%if (%upcase(&ranpro)=EBCD) %then %do;
```

```
 %if %sysevalf(&p<0) or  %sysevalf(&p>1)  %then %do;
      %put ALERT ALERT ALERT: P  must between 0 and 1;
      %goto stopmac;
 %end;
%end;

%if (%upcase(&ranpro)=PBD) and (%length( &blksize) = 0) %then %do;
      %put ALERT ALERT ALERT: BLKSIZE  must be at specified
      with PBD procedure;
      %goto stopmac;
%end;



%if (%upcase(&ranpro)=PBD) and  (%length(&proinblk) = 0 ) %then %do;
      %put ALERT ALERT ALERT: PROINBLK  must be at specified
      with PBD procedure;
      %goto stopmac;
%end;

%if %index(EBCD, %upcase(&ranpro))=0 %then %do;
 %if (%length(&p) ne 0) %then %do;
      %put ALERT ALERT ALERT: P  must be at specified with EBCD procedure;
      %goto stopmac;
 %end;
%end;



%if %index(PBD, %upcase(&ranpro))=0 and %length(&blksize) ne 0 %then %do;
      %put ALERT ALERT ALERT: BLKSIZE must be at specified with
      PBD procedure;
      %goto stopmac;
```

```
%end;

%if %index(PBD,%upcase(&ranpro))=0 and %length(&proinblk) ne 0 %then %do;
      %put ALERT ALERT ALERT: PROINBLK must be at specified
      with PBD procedure;
      %goto stopmac;
%end;

%if %index(‘UPPER’ ‘LOWER’ ‘TWO SIDED’, %upcase(&pvalueside))=0
and %length(&pvalueside) ne 0 %then %do;
       %put ALERT ALERT ALERT: PVALUSIDE  must be one of UPPER, LOWER,
       TWO SIDED;
       %goto stopmac;
%end;

/*Compute the number of obs*/
%global num;
proc sql noprint;
select count(*) into: num from &indata;
quit;

%if %upcase(&type)=CONDITIONAL %then %do;
%global n1;
proc sql noprint;
select sum(&trt) into: n1 from &indata;
quit;
%end;

/*Create data to be used to compute the test*/
data a;
set &indata;
```

```
y=&response;
trt=&trt;
run;

%if %length(&type)=0 %then %let type=UNCONDITIONAL;
%if %length(&ranpro)=0 %then %let ranpro=COMPLETE;
%if %length(&numreps)=0 %then %let numreps=1;
%if %length(&numseq)=0 %then %let numseq=2500;
%if %length(&seed)=0 %then %let seed=1;
%if %length(&pvalueside)=0 %then %let pvalueside = 'TWO SIDED';

/*compute  p values*/
%if %length(&proinblk)=0 %then %do;
%test(data=a, testtype= %upcase(&type), testranpro= %upcase(&ranpro)) ;
%end;
%else %if %length(&proinblk) ne 0 %then %do;
%test(data=a, testtype= %upcase(&type), testranpro= %upcase(&ranpro),
testproinblk=%upcase(&proinblk)) ;
%end;
%stopmac:;
%mend;



%macro test(data=, testtype= , testranpro= , testproinblk=);

/*Compute the test for unconditional complete randomization*/
%if %upcase(&testtype)=UNCONDITIONAL and
%upcase(&testranpro)=COMPLETE %then %do;
proc iml;
seed = &seed;
```

```
n=&num;
m=&numreps;
s=&numseq;
snew=j(s,1,.);
l=j(s,1,.);
lsum=j(m,1,.);
tnew=j(n,1,.);
use &data;
read all  var{trt} into trt ;
read all var{y} into y ;
close &data;
do i=1 to m;
do j=1 to s;
do k=1 to n;
c=ranuni(&seed+m);
p=0.5 ;if c<p then tnew[k]=1 ; else tnew[k]=0;
end;
if upcase(&score)='VAN DER WAERDEN' then  aj=probit(rank(y)/(n+1));
else if         upcase(&score)='WILCOXON'
then aj=ranktie(y);
score=aj-sum(aj)/n;
sobs=trt'*score;
snew[j]=tnew'*score;
if upcase(&pvalueside)='UPPER' then do;
if snew[j]>abs(sobs) then l[j]=1; else l[j]=0;
end;
if      upcase(&pvalueside)='LOWER' then do;
if snew[j]<-abs(sobs)  then l[j]=1; else l[j]=0;


end;
if upcase(&pvalueside)='TWO SIDED' then do;
if snew[j]>abs(sobs) |snew[j]<-abs(sobs)  then l[j]=1; else l[j]=0;
end;
end;
lsum[i]=sum(l);
end;
pvalue=sum(lsum)/(m*s);
xx= pvalue;
cname = { "P-value" };
create x from xx [colname=cname];
append from xx;
quit;
data x2;
n=&num;
Scores=upcase(&score);
Alternative=upcase(&pvalueside);
set x;
run;

proc print data=x2;
title "&testtype Test Result for &testranpro Randomization";
run;
%goto stopmactest;
%end;



/*Compute test for conditional complete randomization*/
%if %upcase(&testtype)=CONDITIONAL and
%upcase(&testranpro)=COMPLETE %then %do;
proc iml;
seed = &seed;
n1=&n1;
```

```
n=&num;
m=&numreps;
s=&numseq;
snew=j(s,1,.);
l=j(s,1,.);
lsum=j(m,1,.);
tnew=j(n,1,.);
use &data;
read all  var{trt} into trt ;
read all var{y} into y ;
close &data;
do i=1 to m;
do j=1 to s;
mj=0;
do k=1 to n;
c=ranuni(&seed+m);
p=(n1-mj)/(n-k+1);if c<p then do; tnew[k]=1 ;mj=mj+1; end;
else do;  tnew[k]=0; mj=mj+0;end;
end;
if upcase(&score)=‘VAN DER WAERDEN’ then aj=probit(rank(y)/(n+1));
else if         upcase(&score)=‘WILCOXON’
then aj=ranktie(y);
score=aj-sum(aj)/n;
sobs=trt’*score;
snew[j]=tnew’*score;
if upcase(&pvalueside)=‘UPPER’ then do;
if snew[j]>abs(sobs) then l[j]=1; else l[j]=0;
end;
if      upcase(&pvalueside)=‘LOWER’ then do;
if snew[j]<-abs(sobs)  then l[j]=1; else l[j]=0;
```

```
end;
if upcase(&pvalueside)=‘TWO SIDED’ then do;
if snew[j]>abs(sobs) |snew[j]<-abs(sobs)  then l[j]=1; else l[j]=0;
end;
end;
lsum[i]=sum(l);
end;
pvalue=sum(lsum)/(m*s);
xx= pvalue;
cname = { "P-value" };
create x from xx [colname=cname];
append from xx;
quit;
data x2;
n=&num;
n1=&n1;
Scores=upcase(&score);
Alternative=upcase(&pvalueside);
set x;
run;

proc print data=x2;
title "&testtype Test Result for &testranpro Randomization";
run;
%goto stopmactest;

%end;


/*Compute the test for unconditonal EBCD*/
%if %upcase(&testtype)=UNCONDITIONAL and
```

```
%upcase(&testranpro)=EBCD %then %do;
proc iml;
start efronunp(p,n1,n2);
if n1=n2 then
pp=0.5;
else if n1<n2 then
pp=p;
else if n1>n2 then pp=1-p;
return (pp);
finish;
store module=efronunp;
quit;

proc iml;
load module=efronunp;
seed = &seed;
n=&num;
m=&numreps;
s=&numseq;
p=&p;
snew=j(s,1,.);
l=j(s,1,.);
lsum=j(m,1,.);
tnew=j(n,1,.);
use &data;
read all  var{trt} into trt ;
read all var{y} into y ;
close &data;
do i=1 to m;
do j=1 to s;
n1=0;n2=0;
```

```
do k=1 to n;
c=ranuni(&seed+m);
pp=efronunp(&p,n1,n2);if c<pp then do;tnew[k]=1 ; n1=n1+1;n2=n2+0; end;
else do ;tnew[k]=0; n1=n1+0;n2=n2+1;end;
end;
if upcase(&score)=‘VAN DER WAERDEN’ then aj=probit(rank(y)/(n+1));
else if         upcase(&score)=‘WILCOXON’ then aj=ranktie(y);
score=aj-sum(aj)/n;
sobs=trt’*score;
snew[j]=tnew’*score;
if upcase(&pvalueside)=‘UPPER’ then do;
if snew[j]>abs(sobs) then l[j]=1; else l[j]=0;
end;
if      upcase(&pvalueside)=‘LOWER’ then do;
if snew[j]<-abs(sobs)  then l[j]=1; else l[j]=0;


end;
if upcase(&pvalueside)=‘TWO SIDED’ then do;
if snew[j]>abs(sobs) |snew[j]<-abs(sobs)  then l[j]=1; else l[j]=0;
end;
end;
lsum[i]=sum(l);
end;
pvalue=sum(lsum)/(m*s);
xx= pvalue;
cname = { "P-value" };
create x from xx [colname=cname];
append from xx;
quit;
data x2;
n=&num;
```

```
Scores=upcase(&score);
Alternative=upcase(&pvalueside);
set x;
run;

proc print data=x2;
title "&testtype Test Result for &testranpro(p=&p) Randomization";
run;
%goto stopmactest;

%end;

/*Compute the test for conditonal EBCD*/
%if %upcase(&testtype)=CONDITIONAL and
%upcase(&testranpro)=EBCD %then %do;
proc iml;
seed=12345;
start efronun (p,n,n1);
if n1=n/2 then do;
   xc=j(n1,1,.);
   do l=0 to n1-1;
   xc[l+1]=((n-2*l)/(n+2*l)) *(comb(n1+l,l)*(1-p)**l);
   end;
pp=(p**n1)*sum(xc);
end;

else if n1<n/2 then do;
     xc1=j(n1+1,1,.);
     do l=0 to n1;
     xc1[l+1]=((n-n1-l)/(n-n1+l)) *(comb(n-n1+l,l)*(1-p)**(n-2*n1+l-1));
     end;
```

```
pp=0.5*(p**n1)*sum(xc1);
end;

else if n1>n/2 then do;
     n2=n-n1;
     xc2=j(n2+1,1,.);
     do l=0 to n2;
     xc2[l+1]=((n1-l)/(n1+l)) *(comb(n1+l,l)*(1-p)**(2*n1-n+l-1));
     end;
pp=0.5*(p**n2)*sum(xc2);
end;

return (pp);
finish;
store module=efronun;

start efroncon(p,n,n1,j,mj);
if j>=1 & j<n & mj>=0 & mj<j/2 then do;
   if mj<=n1 & j-mj>n1 then
   pp=comb(n-j,n1-mj)*(p**(n1-mj))*((1-p)**(n-j-n1+mj));

   else if j-mj<=n1 & n1<n/2 then do;
         xc=j(n1+mj-j+1,1,.);
         do l=0 to n1+mj-j;
              xc[l+1]=(n-n1-mj-l)/(n-n1-mj+l)*comb(n-n1-mj+l,l)
                   *((1-p)**(n-2*n1-1+l));
              end;
   pp=0.5*(p**(n1-mj))*sum(xc)+(p**(n1-mj))*((1-p)**(n-j-n1+mj))
   *(comb(n-j,n1-mj)-comb(n-j,n1-j+mj));
   end;
   else if n1=n/2 then do;
       xc1=j(n-j-n1+mj+1,1,.);
       do l=0 to n-j-n1+mj;
       xc1[l+1]=(n1-mj-l)/(n1-mj+l)*comb(n1-mj+l,l)*((1-p)**l);
       end;
   pp= (p**(n1-mj))*sum(xc1);
   end;
   else if n1>n/2 & j-mj<=n-n1 then do;
       xc2=j(n-j-n1+mj+1,1,.);
           do l=0 to n-j-n1+mj;
       xc2[l+1]= (n1-mj-l)/(n1-mj+l)*comb(n1-mj+l,l)*((1-p)**(2*n1-n-1+l));
       end;
   pp=0.5*(p**(n-n1-mj))*sum(xc2);
   end;
end;

else if j>=0 & j<n & mj=j/2 & mj<=n1 & j-mj<=n-n1
then pp=efronun(p,n-j,n1-mj);

else if j>=1 & j<n & mj>j/2 & mj<=j then do;
     if n1>=mj & n1<n/2 then do;
         xc3=j(n1-mj+1,1,.);
         do l=0 to n1-mj;
         xc3[l+1]=(n-j-n1+mj-l)/(n-j-n1+mj+l)*
         comb(n-j-n1+mj+l,l)*((1-p)**(n-2*n1-1+l));
         end;
     pp=0.5*(p**(n1+mj-j))*sum(xc3);
     end;

     else if n1=n/2 then do;
      xc4= j(n1-mj+1,1,.);
```

```
          do l=0 to n1-mj;
          xc4[l+1]=(n-j-n1+mj-l)/(n-j-n1+mj+l)*comb(n-j-n1+mj+l,l)
                  *((1-p)**(l));
          end;
      pp=(p**(n-j-n1+mj))*sum(xc4);
      end;


      else if n1>n/2 & mj<=n-n1 then do;
      xc5= j(n-n1-mj+1,1,.);
          do l=0 to n-n1-mj;
          xc5[l+1]=(n1+mj-j-l)/(n1+mj-j+l)*comb(n1+mj-j+l,l)*
                  ((1-p)**(2*n1-n-1+l));
          end;
      pp=0.5*(p**(n-j-n1+mj))*sum(xc5)+(p**(n-j-n1+mj))*((1-p)**(n1-mj))
      *(comb(n-j,n1-mj)-comb(n-j,n1-j+mj));
      end;


      else if mj>n-n1 & j-mj<=n-n1 then


      pp=comb(n-j,n1-mj)*(p**(n-j-n1+mj))*(1-p)**(n1-mj);


end;


else if mj=n1 & j=n then


pp=1;


else if mj>n1 & j<n then
pp=0;


return (pp);
```

```
finish;
store module=efroncon;
quit;



proc iml;
load module=efronun;
load module=efroncon;
seed = &seed;
n1=&n1;
p=&p;
n=&num;
m=&numreps;
s=&numseq;
snew=j(s,1,.);
l=j(s,1,.);
lsum=j(m,1,.);
tnew=j(n,1,.);
use &data;
read all  var{trt} into trt ;
read all var{y} into y ;
close &data;
do i=1 to m;
do j=1 to s;
mj=0;
do k=0 to n-1;
f=k+1;
c=ranuni(&seed+m);
if mj<n1 then do;
       if k=0 then
       pp=0.5*efroncon(p,n,n1,1,1)/efronun(p,n,n1);
       else if k>=1 then do;
       if mj=(k+1)/2 then
       pp= 0.5*efroncon(p,n,n1,k+1,mj+1)/efroncon(p,n,n1,k,mj) ;
       else if mj>(k+1)/2 then
       pp=(1-p)*efroncon(p,n,n1,k+1,mj+1)/efroncon(p,n,n1,k,mj) ;
       else if mj<(k+1)/2 then
       pp=p*efroncon(p,n,n1,k+1,mj+1)/efroncon(p,n,n1,k,mj);
       end;
         end;
     else pp=0;
      if c<=pp then do;
      tnew[f]=1;
      mj=mj+1;
      end;
      else do; tnew[f]=0; mj=mj+0;end; end;
if upcase(&score)='VAN DER WAERDEN' then aj=probit(rank(y)/(n+1));
else if upcase(&score)='WILCOXON'
then aj=ranktie(y);
score=aj-sum(aj)/n;
sobs=trt'*score;
snew[j]=tnew'*score;
if upcase(&pvalueside)='UPPER' then do;
if snew[j]>abs(sobs) then l[j]=1; else l[j]=0;
end;
if      upcase(&pvalueside)='LOWER' then do;
if snew[j]<-abs(sobs)  then l[j]=1; else l[j]=0;


end;
if upcase(&pvalueside)='TWO SIDED' then do;
if snew[j]>abs(sobs) |snew[j]<-abs(sobs)  then l[j]=1; else l[j]=0;
end;
```

```
end;
lsum[i]=sum(l);
end;
pvalue=sum(lsum)/(m*s);
xx= pvalue;
cname = { "P-value" };
create x from xx [colname=cname];
append from xx;
quit;
data x2;
n=&num;
n1=&n1;
Scores=upcase(&score);
Alternative=upcase(&pvalueside);
set x;
run;

proc print data=x2;
title "&testtype Test Result for &testranpro(p=&p) Randomization";
run;
%goto stopmactest;

%end;

/*Compute the test for unconditonal PBD*/
/*Truncated binomial*/
%if %upcase(&testtype)=UNCONDITIONAL and %upcase(&testranpro)=PBD
and  %upcase(&testproinblk)=TBD %then %do;

proc iml;
seed = &seed;
```

```
n=&num;
m=&numreps;
s=&numseq;
blocksize=&blksize ;
snew=j(s,1,.);
ll=j(s,1,.);
lsum=j(m,1,.);
tnew=j(n,1,.);
use &data;
read all  var{trt} into trt ;
read all var{y} into y ;
close &data;
do i=1 to m;
do j=1 to s;
if mod(n,blocksize)=0 then l=n/blocksize;
else if mod(n, blocksize)>0 then l=int(n/blocksize)+1;
do k=1 to l;
 mj=0; nj=0;
 do kk= 0 to blocksize-1 until (k=l & kk=n-(l-1)*blocksize-1);
 id=kk+(k-1)*blocksize+1;
  if mj=blocksize/2 then p=0;
        else if max(mj,nj)<blocksize/2 then p=0.5;
        else if nj=blocksize/2 then p=1;
c=ranuni(&seed+m);
if c<p then do;tnew[id]=1 ; nj=nj+0;    mj=mj+1; end;
else do ;tnew[id]=0; nj=nj+1;   mj=mj+0;end;
end; end;
if upcase(&score)='VAN DER WAERDEN' then  aj=probit(rank(y)/(n+1));
else if upcase(&score)='WILCOXON'
then aj=ranktie(y);
score=aj-sum(aj)/n;
```

```
sobs=trt'*score;
snew[j]=tnew'*score;
if upcase(&pvalueside)='UPPER' then do;
if snew[j]>abs(sobs) then ll[j]=1; else ll[j]=0;
end;
if      upcase(&pvalueside)='LOWER' then do;
if snew[j]<-abs(sobs)  then ll[j]=1; else ll[j]=0;


end;
if upcase(&pvalueside)='TWO SIDED' then do;
if snew[j]>abs(sobs) |snew[j]<-abs(sobs)  then ll[j]=1; else ll[j]=0;
end;
end;
lsum[i]=sum(ll);
end;
pvalue=sum(lsum)/(m*s);
xx= pvalue;
cname = { "P-value" };
create x from xx [colname=cname];
append from xx;
quit;
data x2;
n=&num;
Scores=upcase(&score);
Alternative=upcase(&pvalueside);
set x;
run;

proc print data=x2;
title "&testtype Test Result for &testranpro with &testproinblk
Randomization";
```

```
run;
%goto stopmactest;

%end;

/*Random allocation Rule*/
%if %upcase(&testtype)=UNCONDITIONAL and %upcase(&testranpro)=PBD
and  %upcase(&testproinblk)=RAR  %then %do;

proc iml;
seed = &seed;
n=&num;
m=&numreps;
s=&numseq;
blocksize=&blksize ;
snew=j(s,1,.);
ll=j(s,1,.);
lsum=j(m,1,.);
tnew=j(n,1,.);
use &data;
read all  var{trt} into trt ;
read all var{y} into y ;
close &data;
do i=1 to m;
do j=1 to s;
if mod(n,blocksize)=0 then l=n/blocksize;
else if mod(n, blocksize)>0 then l=int(n/blocksize)+1;
do k=1 to l;
 mj=0;
 do kk= 0 to blocksize-1 until (k=l & kk=n-(l-1)*blocksize-1);
id=kk+(k-1)*blocksize+1;
```

```
c=ranuni(&seed+m);
 p=(blocksize/2-mj)/(blocksize-kk);
 if c<p then do; tnew[id]=1 ;   mj=mj+1; end;
   else do ;tnew[id]=0; mj=mj+0; end;
  end;
  end;
if upcase(&score)=‘VAN DER WAERDEN’ then  aj=probit(rank(y)/(n+1));
else if upcase(&score)=‘WILCOXON’
then aj=ranktie(y);
score=aj-sum(aj)/n;
sobs=trt’*score;
snew[j]=tnew’*score;
if upcase(&pvalueside)=‘UPPER’ then do;
if snew[j]>abs(sobs) then ll[j]=1; else ll[j]=0;
end;
if      upcase(&pvalueside)=‘LOWER’ then do;
if snew[j]<-abs(sobs)  then ll[j]=1; else ll[j]=0;


end;
if upcase(&pvalueside)=‘TWO SIDED’ then do;
if snew[j]>abs(sobs) |snew[j]<-abs(sobs)  then ll[j]=1; else ll[j]=0;
end;
end;
lsum[i]=sum(ll);
end;
pvalue=sum(lsum)/(m*s);
xx= pvalue;
cname = { "P-value" };
create x from xx [colname=cname];
append from xx;
quit;
```

```
data x2;
n=&num;
Scores=upcase(&score);
Alternative=upcase(&pvalueside);
set x;
run;

proc print data=x2;
title "&testtype Test Result for &testranpro with &testproinblk
Randomization";
run;
%goto stopmactest;

%end;



/*Conditional PBD*/
/*TBD*/
%if %upcase(&testtype)=CONDITIONAL and %upcase(&testranpro)=PBD
and  %upcase(&testproinblk)=TBD  %then %do;

proc iml;
seed = &seed;
n1=&n1;
n=&num;
m=&numreps;
s=&numseq;
blocksize=&blksize ;
snew=j(s,1,.);
ll=j(s,1,.);
```

```
lsum=j(m,1,.);
tnew=j(n,1,.);
use &data;
read all  var{trt} into trt ;
read all var{y} into y ;
close &data;
do i=1 to m;
do j=1 to s;
if mod(n,blocksize)=0 then l=n/blocksize;
else if mod(n, blocksize)>0 then l=int(n/blocksize)+1;
n11=0;
do k=1 to l;
 mj=0; nj=0;
 do kk= 0 to blocksize-1 until (k=l & kk=n-(l-1)*blocksize-1);
 id=kk+(k-1)*blocksize+1;
 if n11<n1 then do;
    if mj=blocksize/2 then p=0;
        else if max(mj,nj)<blocksize/2 then p=0.5;
        else if nj=blocksize/2 then p=1; end;
        else p=0;
c=ranuni(&seed+m);
if c<p then do;tnew[id]=1 ; nj=nj+0;    mj=mj+1;n11=n11+1; end;
else do ;tnew[id]=0; nj=nj+1;   mj=mj+0;n11=n11+0;end;
end; end;
if upcase(&score)='VAN DER WAERDEN' then  aj=probit(rank(y)/(n+1));
else if         upcase(&score)='WILCOXON' then aj=ranktie(y);
score=aj-sum(aj)/n;
sobs=trt'*score;
snew[j]=tnew'*score;
if upcase(&pvalueside)='UPPER' then do;
if snew[j]>abs(sobs) then ll[j]=1; else ll[j]=0;
```

```
end;
if      upcase(&pvalueside)='LOWER' then do;
if snew[j]<-abs(sobs)  then ll[j]=1; else ll[j]=0;


end;
if upcase(&pvalueside)='TWO SIDED' then do;
if snew[j]>abs(sobs) |snew[j]<-abs(sobs)  then ll[j]=1; else ll[j]=0;
end;
end;
lsum[i]=sum(ll);
end;
pvalue=sum(lsum)/(m*s);
xx= pvalue;
cname = { "P-value" };
create x from xx [colname=cname];
append from xx;
quit;
data x2;
n=&num;
n1=&n1;
Scores=upcase(&score);
Alternative=upcase(&pvalueside);
set x;
run;


proc print data=x2;
title "&testtype Test Result for &testranpro with &testproinblk
Randomization";
run;
%goto stopmactest;
```

```
%end;


/*Random allocation Rule*/
%if %upcase(&testtype)=CONDITIONAL and %upcase(&testranpro)=PBD and
 %upcase(&testproinblk)=RAR  %then %do;


proc iml;
seed = &seed;
n=&num;
m=&numreps;
s=&numseq;
blocksize=&blksize ;
n1=&n1;
snew=j(s,1,.);
ll=j(s,1,.);
lsum=j(m,1,.);
tnew=j(n,1,.);
use &data;
read all  var{trt} into trt ;
read all var{y} into y ;
close &data;
do i=1 to m;
do j=1 to s;
if mod(n,blocksize)=0 then l=n/blocksize;
else if mod(n, blocksize)>0 then l=int(n/blocksize)+1;
n11=0;
do k=1 to l;
 mj=0;
 do kk= 0 to blocksize-1 until (k=l & kk=n-(l-1)*blocksize-1);
id=kk+(k-1)*blocksize+1;
 if n11<n1 then p=(blocksize/2-mj)/(blocksize-kk); else p=0;
 c=ranuni(&seed+m);
 if c<p then do; tnew[id]=1 ;   mj=mj+1; n11=n11+1;end;
   else do ;tnew[id]=0; mj=mj+0; n11=n11+0;end;
  end;
  end;
if upcase(&score)='VAN DER WAERDEN' then  aj=probit(rank(y)/(n+1));
else if upcase(&score)='WILCOXON' then aj=ranktie(y);
score=aj-sum(aj)/n;
sobs=trt'*score;
snew[j]=tnew'*score;
if upcase(&pvalueside)='UPPER' then do;
if snew[j]>abs(sobs) then ll[j]=1; else ll[j]=0;
end;
if      upcase(&pvalueside)='LOWER' then do;
if snew[j]<-abs(sobs)  then ll[j]=1; else ll[j]=0;


end;
if upcase(&pvalueside)='TWO SIDED' then do;
if snew[j]>abs(sobs) |snew[j]<-abs(sobs)  then ll[j]=1; else ll[j]=0;
end;
end;
lsum[i]=sum(ll);
end;


pvalue=sum(lsum)/(m*s);
xx= pvalue;
cname = { "P-value" };
create x from xx [colname=cname];
append from xx;
quit;
data x2;
```

```
n=&num;
n1=&n1;
Scores=upcase(&score);
Alternative=upcase(&pvalueside);
set x;
run;

proc print data=x2;
title "&testtype Test Result for &testranpro with &testproinblk
Randomization";
run;
%goto stopmactest;

%end;
%stopmactest:;
%mend;
```

Appendix B: Macro RANGLM

```
/*********************************************************************
Name: RANGLM

PROGRAM: Macro for randomization tests

PURPOSE: Macro that allows to easily compute p-values of randomization

         tests under the GLM.

*********************************************************************
Inputs:


INDATA         Data set to be analyzed.  This data set has to include the

               response variable which is for the GLM regression, the

               treatment indicator, and other covariates to be used in the

               regression. The response and treatment variable are numeric,

               covariates need not to be numeric, but if there are

               categorical covariates, they have to be specified in CLASS

               option.


METHOD         The estimation method to be used. Options include: Residual

               psudo-likelihood based on random effects solution (RSPL),

               maximum psudo-likelihood based on random effects solution

               (MSPL), residual psudo-likelihood based on mean of the random

               effects (RMPL), maximum psudo-likelihood based on mean of the

               random effects (MMPL), maximum likelihood with  Laplace

               approximation (LAPLACE), and maximum likelihood with adaptive

               quadrature (QUAD). The quadrature nodes can also specified

               within parentheses in QUAD option (e.g. QUAD(QPOINTS= ).


RESPONSE       Name of response or outcome variable from the data to be used

               for the GLM regression.
```

COV	Name of variables to be used as covariates for GLM regression in the MODEL statement of GLIMMIX.
CLASS	Name of variables to be used in the Class statement of the GLIMMIX procedure.
DIST	Distribution of the data to be used in GLM regression. Default value is Normal. Distribution options include: BETA, BINARY, BINOMIAL, EXPONENTIAL, GAMMA, NORMAL, GEOMETRIC, INVGAUSS, LOGNORMAL, MUTINOMIAL, NEGBINOMIAL, POISSON and TCENTRAL.
LINK	Link function to be specified for the GLM regression. The default value is identity. Link function options include: CUMCLL, CUMLOGIT, CUMLOGLOG, CUMPROBIT, CLOGLOG, GLOGIT, IDENTITY, LOG, LOGIT, LOGLOG, PROBIT, POWER, POWERMINUS2 and RECIPROCAL.
TRT	Name of the variable that defines treatment. Must be a binary variable with values 0 or 1.
Type	Name of the type of the randomization tests (UNCONDITIONAL or CONDITIONAL). The default value is UNCONDITIONAL.
RANPRO	Randomization procedure (COMPLETE, EBCD, PBD). The default value is COMPLETE. When the EBCD is specified, p is required for the biased probability. When PBD is specified, the blocksize and the randomization procedure used in each block (TBD, RAR) are needed.

```
NUMREPS       Number of time the inference procedures is replicated.
              Results can be used to simulate the size and power of the
              test. The default value is 1.

NUMSEQ        Number of sequences to be used for randomization. The default
              and minimum value is 2500.

SCORE         Score function to be used in the randomization test, one can
              choose either 'WILCOXON' or 'VAN DER WAERDEN'.

SEED          Seed to be used for generating randomization sequences. The
              default  value is 1.

P             The biased probability used for the EBCD procedure.

BLKSIZE       The block size for the PBD procedure.

PROINBLK      Randomization procedure to be used for the PBD: TBD or RAR.

PVALUESIDE    Specify whether it is a two-sided or one-sided p-value. One
              can choose one of the 'UPPER', 'LOWER' and 'TWO SIDED' three
              options. 'TWO SIDED' is the default.
***************************************************************************
***************************************************************************
*************************************************************************/;
%macro ranglm(indata=,
              method=,
              response =,
              cov=,
              class=,
              trt= ,
              dist=,
              link=,
              type= ,
              ranpro= ,
              numreps=,
              numseq=,
              score=,
              seed=,
              p=,
              blksize=,
              proinblk=,
              pvalueside=
               ) ;
/*Warnings of inappropriate use of the program*/
%if %length(&indata)=0  %then %do;
      %put ALERT ALERT ALERT: INDATA  must be specified;
      %goto stopmac;
   %end;

%if %length(&trt)=0  %then %do;
      %put ALERT ALERT ALERT: TRT  must be specified;
      %goto stopmac;
   %end;



%if (%length(&type) ne 0 ) and ( %upcase(&type) ne UNCONDITIONAL) and
(%upcase(&type) ne CONDITIONAL) %then %do;
      %put ALERT ALERT ALERT: TYPE  must be UNCONDITIONAL  or CONDITIONAL;
      %goto stopmac;
%end;
```

```
%if %index(BETA BIANRY BINOMIAL EXPONENTIAL GAMMA NORMAL GEOMETRIC INVGAUSS
LOGNORMAL MUTINOMIAL NEGBINOMIAL POISSON TCENTRAL,  %upcase(&dist )) =0 and
(%length(&dist) ne 0) %then %do;
      %put ALERT ALERT ALERT: Dist must be one of BETA, BIANRY, BINOMIAL,
      EXPONENTIAL, GAMMA, NORMAL, GEOMETRIC, INVGAUSS, LOGNORMAL,
      MUTINOMIAL, NEGBINOMIAL, POISSON, TCENTRAL;
      %goto stopmac;
 %end;

%if %index(CUMCLL CUMLOGIT CUMLOGLOG CUMPROBIT CLOGLOG GLOGIT IDENTITY LOG
LOGIT LOGLOG PROBIT POWER POWERMINUS2 RECIPROCAL,  %upcase(&link )) =0 and
(%length(&link) ne 0) %then %do;
      %put ALERT ALERT ALERT:LINK must be one of the following funciton :
      CUMCLL, CUMLOGIT, CUMLOGLOG, CUMPROBIT, CLOGLOG, GLOGIT, IDENTITY,
      LOG, LOGIT, LOGLOG, PROBIT, POWER, POWERMINUS2, RECIPROCAL;
      %goto stopmac;
%end;

%if %index(COMPLETE EBCD PBD, %upcase(&ranpro))=0 and
(%length(&ranpro) ne 0) %then %do;
      %put ALERT ALERT ALERT: RANPRO  must be one of COMPLETE, EBCD, PBD;
      %goto stopmac;
%end;

%if %sysevalf(&numreps<1) and  (%length(&numreps) ne 0)  %then %do;
      %put ALERT ALERT ALERT: NUMREPS  must be at least 1;
      %goto stopmac;
%end;

%if %sysevalf(&numseq<2500) and  (%length(&numseq) ne 0)  %then %do;
      %put ALERT ALERT ALERT: NUMSEQ  must be at least 2500;
      %goto stopmac;
%end;

%if %index(‘WILCOXON’ ‘VAN DER WAERDEN’, %upcase(&score))=0 and
      (%length(&score) ne 0) %then %do;
      %put ALERT ALERT ALERT: SCORE  must be one of WILCOXON,
      VAN DER WAERDEN;
      %goto stopmac;
%end;

%if %sysevalf(&seed<0) and %length(&seed) ne 0 %then %do;
      %put ALERT ALERT ALERT: SEED  must be at least 1;
      %goto stopmac;
%end;



%if (%upcase(&ranpro)=EBCD) and  %length(&p) = 0 %then %do;
      %put ALERT ALERT ALERT: P  must be at specified with
      EBCD procedure;
      %goto stopmac;
%end;

%if (%upcase(&ranpro)=EBCD) %then %do;
%if %sysevalf(&p<0) or  %sysevalf(&p>1)  %then %do;
      %put ALERT ALERT ALERT: P  must between 0 and 1;
      %goto stopmac;
%end;
%end;

%if (%upcase(&ranpro)=PBD) and (%length( &blksize) = 0) %then %do;
```

```
      %put ALERT ALERT ALERT: BLKSIZE  must be at specified with
      PBD procedure;
      %goto stopmac;
%end;

%if (%upcase(&ranpro)=PBD) and  (%length(&proinblk) = 0 ) %then %do;
      %put ALERT ALERT ALERT: PROINBLK  must be at specified with
      PBD procedure;
      %goto stopmac;
%end;



%if %index(EBCD, %upcase(&ranpro))=0 and  %length(&p) ne 0 %then %do;
%put ALERT ALERT ALERT: P  must be at specified with  EBCD procedure;
      %goto stopmac;
%end;



%if %index(PBD,%upcase(&ranpro))=0      and %length(&blksize) ne 0
      %then %do;
      %put ALERT ALERT ALERT: BLKSIZE  must be at specified with
      PBD procedure;
      %goto stopmac;
%end;

%if %index(PBD,%upcase(&ranpro))=0      and %length(&proinblk) ne 0
      %then %do;
      %put ALERT ALERT ALERT: PROINBLK  must be at specified with
      PBD procedure;
      %goto stopmac;
%end;
```

```
%if %index('UPPER' 'LOWER' 'TWO SIDED', %upcase(&pvalueside))=0
and %length(&pvalueside) ne 0 %then %do;
      %put ALERT ALERT ALERT: PVALUSIDE  must be one of UPPER, LOWER,
      TWO SIDED;
      %goto stopmac;
          %end;
%global num;

/*Compute the size of the data*/
proc sql noprint;
select count(*) into: num from &indata;
quit;

/*put default values for which not be specified*/
%if %length(&type)=0 %then %let type=UNCONDITIONAL;
%if %length(&method)=0 %then %let method=RSPL;
%if %length(&ranpro)=0 %then %let ranpro=COMPLETE;
%if %length(&numreps)=0 %then %let numreps=1;
%if %length(&numseq)=0 %then %let numseq=2500;
%if %length(&seed)=0 %then %let seed=1;
%if %length(&dist)=0 %then %let dist=normal;
%if %length(&link)=0 %then %let link=identity;
%if %length(&pvalueside)=0 %then %let pvalueside = 'TWO SIDED';

/*Compute n1 for conditional tests*/
%if %upcase(&type)=CONDITIONAL %then %do;
%global n1;
proc sql noprint;
select sum(&trt) into: n1 from &indata;
```

```
quit;
%end;

/*Create data to be used to compute the test*/
%if %length(&class)=0 %then %do;
ods select none;

/*Compute the residuals*/
proc glimmix data=&indata method=&method ;
model &response= &cov /dist=&dist link=&link;
output out=residual residual=r;
run;
ods select all;
%end;
%else %do;
ods select none;



/*Compute the residuals*/
proc glimmix data=&indata method=&method ;
class &class;
model &response= &cov /dist=&dist link=&link;
output out=residual residual=r;
run;
ods select all;
%end;
data a;
set residual;
y=r;
trt=&trt;
run;
```

```
/*Compute  p values*/
%if %length(&proinblk)=0 %then %do;
%test(data=a, testtype= %upcase(&type), testranpro= %upcase(&ranpro)) ;
%end;
%else %if %length(&proinblk) ne 0 %then %do;
%test(data=a, testtype= %upcase(&type), testranpro= %upcase(&ranpro),
testproinblk=%upcase(&proinblk)) ;
%end;
%stopmac:;
%mend;
```

Appendix C: Macro RANGLMM

```
/************************************************************************
Name: RANGLMM
PROGRAM: Macro of randomization tests
PURPOSE: Macro that allows to easily compute p-values of randomization
         tests under the GLMM.
*************************************************************************
Inputs:

INDATA        Data set to be analyzed. It has to include the response
              variable which is used to perform the mixed model regression,
              the subject variable which specifies subjects of the data
              such as patient ID, the repeated measure, a treatment indicator,
              and other covariates if any.

METHOD        The estimation method to be used. Options include: Residual
              psudo-likelihood based on random effects solution (RSPL),
              maximum psudo-likelihood based on random effects solution
              (MSPL), residual psudo-likelihood based on mean of the random
              effects (RMPL), maximum psudo-likelihood based on mean of the
              random effects (MMPL), maximum likelihood with Laplace
              approximation (LAPLACE), and maximum likelihood with adaptive
              quadrature (QUAD).The quadrature nodes can also specified
              within parentheses after in QUAD option (e.g.
              QUAD(QPOINTS= )).

RESPONSE      Name of the response or outcome variable from the data to be
              used for the mixed model.
```

COV	Name of variables to be used as covariates for GLM regression in the MODEL statement of GLIMMIX.
CLASS	Name of variables to be used in the Class statement of the GLIMMIX procedure.
DIST	Distribution of the data to be used in GLM regression. Default is Normal Distribution. Options include: BETA, BINARY, BINOMIAL, EXPONENTIAL, GAMMA, NORMAL, GEOMETRIC, INVGAUSS, LOGNORMAL, MUTINOMIAL, NEGBINOMIAL, POISSON and TCENTRAL.
LINK	Link function to be specified for the GLM regression. The default value is identity. Link function options include: CUMCLL, CUMLOGIT, CUMLOGLOG, CUMPROBIT, CLOGLOG, GLOGIT, IDENTITY, LOG, LOGIT, LOGLOG, PROBIT, POWER, POWERMINUS2 and RECIPROCAL.
GTYPE	Specifies the covariance structure of the random effects. By default, the covariance structure is unstructured. The options of GTYPE include: ANTE(1), AR(1), ARH(1), ARMA(1,1), CHOL, CS, CSH, FA, FA0, HF, LIN, PSPLINE, RSMOOTH, SIMPLE SP(EXP), SP(GAU), SP(MAT), SP(POW), SP(POWA),SP(SPH), TOEP, TOEPH, UN, UNR, and VC.
RTYPE	Specifies the covariance structure of residuals. The default value is the identity matrix. The choices for RTYPE are the same as GTYPE.
TRT	Name of the variable that defines treatment. Must be a binary variable with values 0 or 1.

SUB	Name of the variable that specifies subjects of the data.
REPEAT	Name of the variable that specifies repeated measures of the data.
TRT	Name of the variable that defines treatment. Must be a binary variable with values 0 or 1.
Type	Name of the type of the randomization tests (UNCONDITIONAL or CONDITIONAL). The default value is UNCONDITIONAL.
RANPRO	Randomization procedure (COMPLETE, EBCD, PBD). The default value is COMPLETE. When the EBCD is specified, p is required for the biased probability. When PBD is specified, the blocksize and e randomization procedure used in each block (TBD, RAR) are needed.
NUMREPS	Number of time the inference procedures is replicated. Results can be used to simulate the size and power of the test. The default value is 1.
NUMSEQ	Number of sequences to be used for randomization. The default and minimum value is 2500.
SCORE	Score function to be used in the randomization test, one can choose either 'WILCOXON' or 'VAN DER WAERDEN'.
SEED	Seed to be used for generating randomization sequences. The default value is 1.

```
P              The biased probability used for the EBCD procedure.

BLKSIZE        The block size for the PBD procedure.

PROINBLK       Randomization procedure to be used for the PBD: TBD or RAR.

PVALUESIDE     Specify whether it is a two-sided or one-sided p-value. One
               can choose one of the 'UPPER', 'LOWER' and 'TWO SIDED' three
               options. 'TWO SIDED' is the default.

**********************************************************************
**********************************************************************
**********************************************************************/;

%macro ranglmm(indata=,
               method=,
               response =,
               cov=,
               class= ,
               dist=,
               link=,
               gtype=,
               rtype=,
               trt= ,
               sub=,
               repeat=,
               type= ,
               ranpro= ,
               numreps=,
               numseq=,
               score=,
               seed=,
               p=,
               blksize=,
               proinblk=,
               pvalueside=
               ) ;
/*Warnings of inappropriate use of the program*/
%if %length(&indata)=0  %then %do;
      %put ALERT ALERT ALERT: INDATA  must be specified;
      %goto stopmac;
   %end;

%if %length(&trt)=0  %then %do;
      %put ALERT ALERT ALERT: TRT  must be specified;
      %goto stopmac;
   %end;

%if %length(&sub)=0  %then %do;
      %put ALERT ALERT ALERT: SUB must be specified;
      %goto stopmac;
   %end;

%if %length(&repeat)=0  %then %do;
      %put ALERT ALERT ALERT: REPEAT must be specified;
      %goto stopmac;
   %end;
%if %length(&class)=0  %then %do;
      %put ALERT ALERT ALERT: CLASS must be specified;
      %goto stopmac;
%end;
```

```
%if %index(ANTE(1) AR(1) ARH(1) ARMA(1,1) CHOL CS CSH FA FA0 HF LIN
PSPLINE RSMOOTH SIMPLE SP(EXP) SP(GAU) SP(MAT)SP(POW) SP(POWA) SP(SPH)
TOEP  TOEPH UN UNR VC,   %upcase(&gtype))=0 and (%length(&gtype) ne 0)
%then %do;
      %put ALERT ALERT ALERT: Dist must be one of ANTE(1) AR(1) ARH(1)
      ARMA(1,1) CHOL CS CSH FA FA0 HF LIN PSPLINE RSMOOTH SIMPLE SP(EXP)
      SP(GAU) SP(MAT) SP(POW) SP(POWA) SP(SPH) TOEP  TOEPH UN UNR VC;
      %goto stopmac;
%end;

%if %index(ANTE(1) AR(1) ARH(1) ARMA(1,1) CHOL CS CSH FA FA0 HF LIN
PSPLINE RSMOOTH SIMPLE SP(EXP) SP(GAU) SP(MAT)SP(POW) SP(POWA) SP(SPH)
TOEP  TOEPH UN UNR VC,   %upcase(&rtype))=0 and (%length(&rtype) ne 0)
%then %do;
      %put ALERT ALERT ALERT: Dist must be one of ANTE(1) AR(1) ARH(1)
      ARMA(1,1) CHOL CS CSH FA FA0 HF LIN PSPLINE RSMOOTH SIMPLE SP(EXP)
      SP(GAU) SP(MAT) SP(POW) SP(POWA) SP(SPH) TOEP  TOEPH UN UNR VC;
      %goto stopmac;
%end;

%if (%length(&type) ne 0 ) and ( %upcase(&type) ne UNCONDITIONAL) and
(%upcase(&type) ne CONDITIONAL) %then %do;
      %put ALERT ALERT ALERT: TYPE  must be UNCONDITIONAL  or CONDITIONAL ;
      %goto stopmac;
%end;

%if %index(BETA BINARY BINOMIAL EXPONENTIAL GAMMA NORMAL GEOMETRIC INVGAUSS
LOGNORMAL MUTINOMIAL NEGBINOMIAL POISSON TCENTRAL,  %upcase(&dist )) =0 and
(%length(&dist) ne 0) %then %do;
       %put ALERT ALERT ALERT: Dist must be one of BETA, BINARY, BINOMIAL,
        EXPONENTIAL, GAMMA, NORMAL, GEOMETRIC, INVGAUSS, LOGNORMAL,
        MUTINOMIAL, NEGBINOMIAL, POISSON, TCENTRAL;
       %goto stopmac;
%end;

%if %index(CUMCLL CUMLOGIT CUMLOGLOG CUMPROBIT CLOGLOG GLOGIT IDENTITY LOG
LOGIT LOGLOG PROBIT POWER POWERMINUS2 RECIPROCAL,  %upcase(&link )) =0 and
(%length(&link) ne 0) %then %do;
        %put ALERT ALERT ALERT:LINK must be one of the following funciton:
        CUMCLL, CUMLOGIT, CUMLOGLOG, CUMPROBIT, CLOGLOG, GLOGIT, IDENTITY,
        LOG, LOGIT, LOGLOG, PROBIT, POWER, POWERMINUS2, RECIPROCAL;
      %goto stopmac;
%end;

%if %index(COMPLETE EBCD PBD, %upcase(&ranpro))=0 and
(%length(&ranpro) ne 0) %then %do;
        %put ALERT ALERT ALERT: RANPRO  must be one of COMPLETE, EBCD, PBD;
        %goto stopmac;
%end;

%if %sysevalf(&numreps<1) and  (%length(&numreps) ne 0)  %then %do;
        %put ALERT ALERT ALERT: NUMREPS  must be at least 1;
        %goto stopmac;
 %end;

%if %sysevalf(&numseq<2500) and  (%length(&numseq) ne 0)  %then %do;
      %put ALERT ALERT ALERT: NUMSEQ  must be at least 2500;
      %goto stopmac;
%end;

%if %index('WILCOXON' 'VAN DER WAERDEN', %upcase(&score))=0 and
(%length(&score) ne 0) %then %do;
```

```
      %put ALERT ALERT ALERT: SCORE  must be one of WILCOXON,
      VAN DER WAERDEN;
      %goto stopmac;
%end;

%if %sysevalf(&seed<0) and %length(&seed) ne 0 %then %do;
%put ALERT ALERT ALERT: SEED  must be at least 1;
      %goto stopmac;
%end;



%if (%upcase(&ranpro)=EBCD) and  %length(&p) = 0 %then %do;
      %put ALERT ALERT ALERT: P  must be at specified with EBCD procedure;
      %goto stopmac;
%end;

%if (%upcase(&ranpro)=EBCD) %then %do;
%if %sysevalf(&p<0) or  %sysevalf(&p>1)  %then %do;
      %put ALERT ALERT ALERT: P  must between 0 and 1;
      %goto stopmac;
%end;
%end;

%if (%upcase(&ranpro)=PBD) and (%length( &blksize) = 0) %then %do;
      %put ALERT ALERT ALERT: BLKSIZE  must be at specified with
      PBD procedure;
      %goto stopmac;
%end;

%if (%upcase(&ranpro)=PBD) and  (%length(&proinblk) = 0 ) %then %do;
      %put ALERT ALERT ALERT: PROINBLK  must be at specified with
      PBD procedure;
      %goto stopmac;
%end;



%if %index(EBCD, %upcase(&ranpro))=0 and  %length(&p) ne 0 %then %do;
      %put ALERT ALERT ALERT: P  must be at specified with  EBCD procedure;
      %goto stopmac;
%end;



%if %index(PBD,%upcase(&ranpro))=0      and %length(&blksize) ne 0
      %then %do;
      %put ALERT ALERT ALERT: BLKSIZE  must be at specified with
      PBD procedure;
      %goto stopmac;
%end;

%if %index(PBD,%upcase(&ranpro))=0      and %length(&proinblk) ne 0
      %then %do;
      %put ALERT ALERT ALERT: PROINBLK  must be at specified with
      PBD procedure;
      %goto stopmac;
%end;

%if %index(‘UPPER’ ‘LOWER’ ‘TWO SIDED’, %upcase(&pvalueside))=0
and %length(&pvalueside) ne 0 %then %do;
      %put ALERT ALERT ALERT: PVALUSIDE  must be one of UPPER, LOWER,
      TWO SIDED;
      %goto stopmac;
%end;
```

```
/*Put default values for which not be specified*/
%if %length(&type)=0 %then %let type=UNCONDITIONAL;
%if %length(&ranpro)=0 %then %let ranpro=COMPLETE;
%if %length(&numreps)=0 %then %let numreps=1;
%if %length(&numseq)=0 %then %let numseq=2500;
%if %length(&seed)=0 %then %let seed=1;
%if %length(&method)=0 %then %let method=RSPL;
%if %length(&dist)=0 %then %let dist=normal;
%if %length(&link)=0 %then %let link=identity;
%if %length(&gtype)=0 %then %let gtype=un;
%if %length(&pvalueside)=0 %then %let pvalueside = 'TWO SIDED';

%global num;

/*Compute the number of the subjects*/
data rt1 (keep=&trt &sub);
   set &indata ;
   proc sort
   nodup;
   by &sub;
run;
proc sql noprint;
select count(*) into: num from rt1;
quit;


%if %upcase(&type)=CONDITIONAL %then %do;
%global n1;
proc sql noprint;
select count(distinct &sub) into: n1 from &indata where &trt=1;;
```

```
quit;
%end;

/*Create data to be used to compute the test*/
/*Compute the residuals*/
%if %length(&rtype) = 0 %then %do;
ods select SolutionR;
proc glimmix data=&indata method=&method;
 class &class ;
model &response= &cov /dist=&dist link=&link ;
  random int &repeat/subject=&sub type=&gtype  solution  ;
   ods output SolutionR=rt;
run;
%end;

%if %length(&rtype) ne 0 %then %do;
ods select SolutionR;
proc glimmix data=&indata method=&method;
 class &class ;
model &response= &cov /dist=&dist link=&link ;
  random int &repeat/subject=&sub type=&gtype  solution  ;
  random _residual_/type=&rtype;
   ods output SolutionR=rt;
run;
%end;

data rt2;
set rt;
if effect =‘Intercept’ then delete;
y=estimate;
run;
```

```
data a;
set rt1;set rt2;
trt=&trt;
run;

/*Compute  p values with previously loaded TEST macro*/
%if %length(&proinblk)=0 %then %do;
%test(data=a, testtype= %upcase(&type), testranpro= %upcase(&ranpro)) ;
%end;
%else %if %length(&proinblk) ne 0 %then %do;
%test(data=a, testtype= %upcase(&type), testranpro= %upcase(&ranpro),
testproinblk=%upcase(&proinblk)) ;
%end;
%stopmac:;
%mend;
```

Appendix D: Macro RANSURVIVAL

```
/*********************************************************************
Name: RANSURVIVAL
PROGRAM: Macro for randomization tests of survival data
PURPOSE: Macro that allows to easily compute p-values of randomization
         tests for survival data
**********************************************************************
Inputs:

INDATA        Data set to be analyzed. It has the following variables: the
              response variable which is used to perform the randomization
              test, the censoring indicator if any, a treatment indicator
              variable, and other covariates to be included in the
              regression.

RESPONSE      Name of the response variable to be used for regression
              if model is specified or randomization test if no model is
              specified.

CENSOR        Variable that defines censoring status. We consider value 0
              as censored, and 1 as uncensored. By default, there is no
              censoring.

MODEL         Types of model to be used for data. Values include NONE, PH,
              AFT. By default, there is no modeling.

COV           Name of the covariates to be used for regression if model is
              PH or AFT.
```

DIST	Variable that specifies the distribution type for failure time if the AFT model is used. Options include EXPONENTIAL, GAMMA, LLOGISTIC, LNORMAL, LOGISTIC, NORMAL and WEIBULL. The default distribution is a type 1 extreme-value distribution to the log of the response.
NOLOG	For the AFT model, it requests whether to use log transformation or not. By default, there is log transformation.
NOINT	For the AFT model, it requests whether intercept is included in the regression. By default, the intercept is included.
TRT	Name of the variable that defines treatment. Must be a binary variable with values 0 or 1.
Type	Name of the type of the randomization tests (UNCONDITIONAL or CONDITIONAL). The default value is UNCONDITIONAL.
RANPRO	Randomization procedure (COMPLETE, EBCD, PBD). The default value is COMPLETE. When the EBCD is specified, p is required for the biased probability. When PBD is specified, the blocksize and the randomization procedure used in each block (TBD, RAR) are needed.
NUMREPS	Number of time the inference procedures is replicated. Results can be used to simulate the size and power of the

```
              test. The default value is 1.

NUMSEQ        Number of sequences to be used for randomization. The default
              and minimum value is 2500.

SCORE         Score function to be used in the randomization test, one can
              choose either 'WILCOXON' or 'VAN DER WAERDEN'.

SEED          Seed to be used for generating randomization sequences. The
              default value is 1.

P             The biased probability used for the EBCD procedure.

BLKSIZE       The block size for the PBD procedure.

PROINBLK      Randomization procedure to be used for the PBD: TBD or RAR.

PVALUESIDE    Specify whether it is a two-sided or one-sided p-value. One
              can choose one of the 'UPPER', 'LOWER' and 'TWO SIDED'
              three options. 'TWO SIDED' is the default.

*************************************************************************
*************************************************************************
*************************************************************************/;


%macro ransurvival(indata=,
                   response=,
                   censor=,
                   model=,
                   cov=,
                   dist=,
                   nolog=,
                   noint=,
                   trt= ,
                   type= ,
                   ranpro= ,
                   numreps=,
                   numseq=,
                   seed=,
                   p=,
                   blksize=,
                   proinblk=,
                   pvalueside=
                   ) ;

/*Warnings of inappropriate use of the program*/
%if %length(&indata)=0  %then %do;
      %put ALERT ALERT ALERT: INDATA  must be specified;
      %goto stopmac;
   %end;

%if %length(&trt)=0  %then %do;
      %put ALERT ALERT ALERT: TRT  must be specified;
      %goto stopmac;
   %end;

%if %length(&model) ne 0 and %index('AFT' 'PH',  %upcase(&model ))=0
     %then %do;
     %put ALERT ALERT ALERT: If MODEL is specified, it  must be  AFT or PH;
      %goto stopmac;
   %end;
```

```
%if (%length(&type) ne 0 ) and ( %upcase(&type) ne UNCONDITIONAL) and
( %upcase(&type) ne CONDITIONAL) %then %do;
%put ALERT ALERT ALERT: TYPE  must be UNCONDITIONAL  or CONDITIONAL ;
      %goto stopmac;
%end;

%if %upcase(&model)=‘AFT’ %then %do;
%if %index(EXPONENTIAL GAMMA LLOGISTIC  LNORMAL LOGISTIC NORMAL WEIBULL,
  %upcase(&dist )) =0 and (%length(&dist) ne 0)%then %do;
       %put ALERT ALERT ALERT: Dist must be one of EXPONENTIAL, GAMMA,
       LLOGISTIC, LNORMAL, LOGISTIC, NORMAL, WEIBULL;
      %goto stopmac;
 %end;
%end;



%if %index(COMPLETE EBCD PBD, %upcase(&ranpro))=0 and (%length(&ranpro)
 ne 0) %then %do;
      %put ALERT ALERT ALERT: RANPRO  must be one of COMPLETE, EBCD, PBD;
      %goto stopmac;
%end;

%if %sysevalf(&numreps<1) and  (%length(&numreps) ne 0)  %then %do;
      %put ALERT ALERT ALERT: NUMREPS  must be at least 1;
      %goto stopmac;
 %end;

%if %sysevalf(&numseq<2500) and  (%length(&numseq) ne 0)  %then %do;
      %put ALERT ALERT ALERT: NUMSEQ  must be at least 2500;
      %goto stopmac;
 %end;
```

```
%if %sysevalf(&seed<0) and %length(&seed) ne 0 %then %do;
      %put ALERT ALERT ALERT: SEED  must be at least 1;
      %goto stopmac;
 %end;



%if (%upcase(&ranpro)=EBCD) and  %length(&p) = 0 %then %do;
      %put ALERT ALERT ALERT: P  must be at specified with EBCD procedure;
      %goto stopmac;
%end;

%if (%upcase(&ranpro)=EBCD) %then %do;
%if %sysevalf(&p<0) or  %sysevalf(&p>1)  %then %do;
 %put ALERT ALERT ALERT: P  must between 0 and 1;
      %goto stopmac;
%end;
%end;

%if (%upcase(&ranpro)=PBD) and (%length( &blksize) = 0) %then %do;
      %put ALERT ALERT ALERT: BLKSIZE  must be at specified with
      PBD procedure;
      %goto stopmac;
%end;

%if (%upcase(&ranpro)=PBD) and  (%length(&proinblk) = 0 ) %then %do;
      %put ALERT ALERT ALERT: PROINBLK  must be at specified with
      PBD procedure;
      %goto stopmac;
```

```
%end;



%if %index(EBCD, %upcase(&ranpro))=0 and  %length(&p) ne 0 %then %do;
      %put ALERT ALERT ALERT: P  must be at specified with  EBCD procedure;
      %goto stopmac;
%end;



%if %index(PBD,%upcase(&ranpro))=0      and %length(&blksize) ne 0
      %then %do;
      %put ALERT ALERT ALERT: BLKSIZE  must be at specified with
      PBD procedure;
      %goto stopmac;
%end;

%if %index(PBD,%upcase(&ranpro))=0      and %length(&proinblk) ne 0
     %then %do;
     %put ALERT ALERT ALERT: PROINBLK  must be at specified with
     PBD procedure;
      %goto stopmac;
%end;



%if %index('UPPER' 'LOWER' 'TWO SIDED', %upcase(&pvalueside))=0
and %length(&pvalueside) ne 0 %then %do;
%put ALERT ALERT ALERT: PVALUSIDE  must be one of UPPER, LOWER, TWO SIDED;
      %goto stopmac;
%end;

%global num;
```

```
/*Compute the size of the data*/
proc sql noprint;
select count(*) into: num from &indata;
quit;

%if %upcase(&type)=CONDITIONAL %then %do;
%global n1;
proc sql noprint;
select sum(&trt) into: n1 from &indata;
quit;
%end;

/*Put default values for which not be specified*/
%if %length(&type)=0 %then %let type=UNCONDITIONAL;
%if %length(&ranpro)=0 %then %let ranpro=COMPLETE;
%if %length(&numreps)=0 %then %let numreps=1;
%if %length(&numseq)=0 %then %let numseq=2500;
%if %length(&seed)=0 %then %let seed=1;
%if %upcase(&model)=AFT and  %length(&dist)=0 %then %let dist=weibull;
%if %length(&pvalueside)=0 %then %let pvalueside = ‘TWO SIDED’;
%if  %length(&noint)=0 %then %let noint=0;
%if  %length(&nolog)=0 %then %let nolog=0;
%if %length(&model)=0  %then %let model=‘NONE’;
%global Scores;
%if %upcase(&model)=‘NONE’ %then %let Scores=‘SAVAGE’;
%else %let Scores=‘WILCOXON’;

/*Create data to be used to compute the test*/
/*No censoring*/
%if %length(&censor)=0 %then %do;
%if %upcase(&model)=‘NONE’ %then %do;
```

```
proc rank data =&indata out=a1 savage;
var &response;
ranks r;
run;
data a;
set a1;
trt=&trt;
censor=1;
run;
%end;
%if %upcase(&model)=‘PH’ %then %do;
proc phreg data=&indata noprint;
   model &response= &cov;
  output out=r1 resmart=r ;
  run;

data a;
set r1;
censor=1;
trt=&trt;
run;
%end;


%if %upcase(&model)=‘AFT’ %then %do;
%if &nolog=0 %then %do;
%if  &noint=0  %then %do;
   proc lifereg data=&indata noprint;
   model &response= &cov /dist=&dist;
  output out=r1 crresidual=cr ;
  run;
data a;
```

```
set r1;
censor=1;
  r = censor - cr;
trt=&trt;
  run;
%end;
%if  &noint=1  %then %do;
proc lifereg data=&indata noprint;
   model &response= &cov /dist=&dist noint;
  output out=r1 crresidual=cr ;
  run;
data a;
set r1;
censor=1;
  r = censor - cr;
trt=&trt;
  run;
%end;
%end;
%if &nolog=1 %then %do;
%if  &noint=0  %then %do;
   proc lifereg data=&indata noprint;
   model &response= &cov /dist=&dist nolog;
 output out=r1 crresidual=cr ;
  run;
data a;
set r1;
censor=1;
  r = censor - cr;
  trt=&trt;
  run;
```

```
%end;
%if  &noint=1 %then %do;
   proc lifereg data=&indata noprint;
   model &response= &cov /dist=&dist nolog noint;
 output out=r1 crresidual=cr ;
  run;
data a;
set r1;
censor=1;
  r = censor - cr;
  run;
%end;
%end;
%end;
%end;

/*Censoring*/
%if %length(&censor) ne 0 %then %do;
%if %upcase(&model)=‘NONE’ %then %do;
proc rank data=&indata out=a1 savage;
var &response;
ranks r;
run;
data a;
set a1;
trt=&trt;
r=&response;
censor=&censor;
run;
%end;
```

```
%if %upcase(&model)=‘PH’ %then %do;
proc phreg data=&indata noprint;
   model &response*&censor(0)= &cov ;
  output out=r1 resmart=r ;
  run;
data a;
set r1;
censor=&censor;
trt=&trt;
%end;

%if %upcase(&model)=‘AFT’ %then %do;
%if &nolog=0  %then %do;
%if  &noint=0  %then %do;
   proc lifereg data=&indata noprint;
   model &response*&censor(0)= &cov /dist=&dist;
  output out=r1 cresidual=cr ;
  run;
data a;
set r1;
censor=&censor;
  r = censor - cr;
trt=&trt;
  run;
%end;
%if  &noint=1  %then %do;

proc lifereg data=&indata noprint;
model &response*&censor(0)= &cov /dist=&dist noint;
output out=r1 cresidual=cr ;
run;
```

```
data a;
set r1;
censor=&censor;
  r = censor - cr;
trt=&trt;
  run;
%end;
%end;
%if &nolog=1 %then %do;
%if  &noint=0  %then %do;
   proc lifereg data=&indata noprint;
   model &response*&censor(0)= &cov /dist=&dist nolog;
 output out=r1 cresidual=cr ;
  run;
data a;
set r1;
censor=&censor;
  r = censor - cr;
trt=&trt;
  run;
%end;
%if  &noint=1 %then %do;
   proc lifereg data=&indata noprint;
   model &response*&censor(0)= &cov /dist=&dist nolog noint;
 output out=r1 cresidual=cr ;
  run;
data a;
set r1;
censor=&censor;
  r = censor - cr;
trt=&trt;
```

```
  run;
%end;
%end;
%end;
%end;

/*Compute  p values*/
%if %length(&proinblk)=0 %then %do;
%stest(data=a, testtype= %upcase(&type), testranpro= %upcase(&ranpro)) ;
%end;
%else %if %length(&proinblk) ne 0 %then %do;
%stest(data=a, testtype= %upcase(&type), testranpro= %upcase(&ranpro),
testproinblk=%upcase(&proinblk)) ;
%end;
%stopmac:;
%mend;



%macro stest(data=, testtype= , testranpro= , testproinblk=);
/*Compute unconditional complete randomization*/
%if %upcase(&testtype)=UNCONDITIONAL and %upcase(&testranpro)=COMPLETE
%then %do;
proc iml;
seed = &seed;
n=&num;
m=&numreps;
s=&numseq;
xj=j(n,1,.);
snew=j(s,1,.);
l=j(s,1,.);
lsum=j(m,1,.);
```

```
tnew=j(n,1,.);
use &data;
read all  var{trt} into trt ;
read all var{r} into r ;
read all var{censor} into d;
close &data;
do i=1 to m;
do j=1 to s;
do k=1 to n;
c=ranuni(&seed+m);
p=0.5 ;if c<p then tnew[k]=1 ; else tnew[k]=0;
xj[k]=n-k+1;
end;
if upcase(&model)='NONE' then aj=d-(r+1);
else if upcase(&model)='AFT' | upcase(&model)='PH' then aj=ranktie(r);
score=aj-sum(aj)/n;
sobs=trt'*score;
snew[j]=tnew'*score;
if upcase(&pvalueside)='UPPER' then do;
if snew[j]>abs(sobs) then l[j]=1; else l[j]=0;
end;
if      upcase(&pvalueside)='LOWER' then do;
if snew[j]<-abs(sobs)  then l[j]=1; else l[j]=0;


end;
if upcase(&pvalueside)='TWO SIDED' then do;
if snew[j]>abs(sobs) |snew[j]<-abs(sobs)  then l[j]=1; else l[j]=0;
end;
end;
lsum[i]=sum(l);
end;
```

```
pvalue=sum(lsum)/(m*s);
xx= pvalue;
cname = { "P-value" };
create x from xx [colname=cname];
append from xx;
quit;
data x2;
n=&num;
if upcase(&model)='NONE' then Scores='SAVAGE'; else Scores='WILCOXON';
Alternative=upcase(&pvalueside);
set x;
run;

proc print data=x2;
title "&testtype Test Result for &testranpro Randomization";
run;

 %goto stopmactest;
%end;

/*Compute test for conditional complete randomization*/
%if %upcase(&testtype)=CONDITIONAL and %upcase(&testranpro)=COMPLETE
%then %do;
proc iml;
seed = &seed;
n1=&n1;
n=&num;
m=&numreps;
s=&numseq;
snew=j(s,1,.);
l=j(s,1,.);
```

```
lsum=j(m,1,.);

tnew=j(n,1,.);

xj=j(n,1,.);

use &data;

read all  var{trt} into trt ;

read all var{r} into r ;

read all var{censor} into d;

close &data;

do i=1 to m;

do j=1 to s;

mj=0;

do k=1 to n;

c=ranuni(&seed+m);

p=(n1-mj)/(n-k+1);if c<p then do; tnew[k]=1 ;mj=mj+1; end;

else do;  tnew[k]=0; mj=mj+0;end;

xj[k]=n-k+1;

end;

if upcase(&model)='NONE' then aj=d-(r+1);

else if upcase(&model)='AFT' | upcase(&model)='PH' then aj=ranktie(r);

score=aj-sum(aj)/n;

sobs=trt'*score;

snew[j]=tnew'*score;

if upcase(&pvalueside)='UPPER' then do;

if snew[j]>abs(sobs) then l[j]=1; else l[j]=0;

end;

if      upcase(&pvalueside)='LOWER' then do;

if snew[j]<-abs(sobs)  then l[j]=1; else l[j]=0;


end;

if upcase(&pvalueside)='TWO SIDED' then do;

if snew[j]>abs(sobs) |snew[j]<-abs(sobs)  then l[j]=1; else l[j]=0;
```

```
end;
end;
lsum[i]=sum(l);
end;
pvalue=sum(lsum)/(m*s);
xx= pvalue;
cname = { "P-value" };
create x from xx [colname=cname];
append from xx;
quit;
data x2;
n=&num;
n1=&n1;
Scores=&Scores;
Alternative=upcase(&pvalueside);set x;
run;

proc print data=x2;
title "&testtype Test Result for &testranpro  Randomization";
run;

%goto stopmactest;
%end;



/*Compute the test for unconditonal EBCD*/
%if %upcase(&testtype)=UNCONDITIONAL and %upcase(&testranpro)=EBCD
%then %do;
proc iml;
start efronunp(p,n1,n2);
if n1=n2 then
```

```
pp=0.5;
else if n1<n2 then
pp=p;
else if n1>n2 then pp=1-p;
return (pp);
finish;
store module=efronunp;
quit;

proc iml;
load module=efronunp;
seed = &seed;
n=&num;
m=&numreps;
s=&numseq;
p=&p;
snew=j(s,1,.);
l=j(s,1,.);
lsum=j(m,1,.);
tnew=j(n,1,.);
xj=j(n,1,.);
use &data;
read all  var{trt} into trt ;
read all var{r} into r ;
read all var{censor} into d;
close &data;
do i=1 to m;
do j=1 to s;
n1=0;n2=0;
do k=1 to n;
c=ranuni(&seed+m);
```

```
pp=efronunp(&p,n1,n2);if c<pp then do;tnew[k]=1 ; n1=n1+1;n2=n2+0; end;
else do ;tnew[k]=0; n1=n1+0;n2=n2+1;end;
xj[k]=n-k+1;
end;
if upcase(&model)='NONE' then aj=d-(r+1);
else if upcase(&model)='AFT' | upcase(&model)='PH' then aj=ranktie(r);
score=aj-sum(aj)/n;
sobs=trt'*score;
snew[j]=tnew'*score;
if upcase(&pvalueside)='UPPER' then do;
if snew[j]>abs(sobs) then l[j]=1; else l[j]=0;
end;
if      upcase(&pvalueside)='LOWER' then do;
if snew[j]<-abs(sobs)  then l[j]=1; else l[j]=0;


end;
if upcase(&pvalueside)='TWO SIDED' then do;
if snew[j]>abs(sobs) |snew[j]<-abs(sobs)  then l[j]=1; else l[j]=0;
end;
end;
lsum[i]=sum(l);
end;
pvalue=sum(lsum)/(m*s);
xx= pvalue;
cname = { "P-value" };
create x from xx [colname=cname];
append from xx;
quit;
data x2;
n=&num;
Scores=&Scores;
```

```
Alternative=upcase(&pvalueside);set x;
run;

proc print data=x2;
title "&testtype Test Result for &testranpro (p=&p) Randomization";
run;

%goto stopmactest;
%end;


/*Compute the test for conditonal EBCD*/
%if %upcase(&testtype)=CONDITIONAL and %upcase(&testranpro)=EBCD %then %do;
proc iml;
seed=12345;
start efronun (p,n,n1);
if n1=n/2 then do;
   xc=j(n1,1,.);
   do l=0 to n1-1;
   xc[l+1]=((n-2*l)/(n+2*l)) *(comb(n1+l,l)*(1-p)**l);
   end;
pp=(p**n1)*sum(xc);
end;

else if n1<n/2 then do;
     xc1=j(n1+1,1,.);
     do l=0 to n1;
     xc1[l+1]=((n-n1-l)/(n-n1+l)) *(comb(n-n1+l,l)*(1-p)**(n-2*n1+l-1));
     end;
pp=0.5*(p**n1)*sum(xc1);
end;
```

```
else if n1>n/2 then do;
     n2=n-n1;
     xc2=j(n2+1,1,.);
     do l=0 to n2;
     xc2[l+1]=((n1-l)/(n1+l)) *(comb(n1+l,l)*(1-p)**(2*n1-n+l-1));
     end;
pp=0.5*(p**n2)*sum(xc2);
end;

return (pp);
finish;
store module=efronun;

start efroncon(p,n,n1,j,mj);
if j>=1 & j<n & mj>=0 & mj<j/2 then do;
   if mj<=n1 & j-mj>n1 then
   pp=comb(n-j,n1-mj)*(p**(n1-mj))*((1-p)**(n-j-n1+mj));

   else if j-mj<=n1 & n1<n/2 then do;
        xc=j(n1+mj-j+1,1,.);
        do l=0 to n1+mj-j;
            xc[l+1]=(n-n1-mj-l)/(n-n1-mj+l)*comb(n-n1-mj+l,l)*
                ((1-p)**(n-2*n1-1+l));
            end;
   pp=0.5*(p**(n1-mj))*sum(xc)+(p**(n1-mj))*((1-p)**(n-j-n1+mj))
   *(comb(n-j,n1-mj) -comb(n-j,n1-j+mj));
   end;

   else if n1=n/2 then do;
       xc1=j(n-j-n1+mj+1,1,.);
```

```
        do l=0 to n-j-n1+mj;
        xc1[l+1]=(n1-mj-l)/(n1-mj+l)*comb(n1-mj+l,l)*((1-p)**l);
        end;
    pp= (p**(n1-mj))*sum(xc1);
    end;
    else if n1>n/2 & j-mj<=n-n1 then do;
        xc2=j(n-j-n1+mj+1,1,.);
            do l=0 to n-j-n1+mj;
        xc2[l+1]= (n1-mj-l)/(n1-mj+l)*comb(n1-mj+l,l)*((1-p)**(2*n1-n-1+l));
        end;
    pp=0.5*(p**(n-n1-mj))*sum(xc2);
    end;
end;


else if j>=0 & j<n & mj=j/2 & mj<=n1 & j-mj<=n-n1 then
pp=efronun(p,n-j,n1-mj);


else if j>=1 & j<n & mj>j/2 & mj<=j then do;
     if n1>=mj & n1<n/2 then do;
         xc3=j(n1-mj+1,1,.);
         do l=0 to n1-mj;
         xc3[l+1]=(n-j-n1+mj-l)/(n-j-n1+mj+l)*comb(n-j-n1+mj+l,l)*
         ((1-p)**(n-2*n1-1+l));
         end;
     pp=0.5*(p**(n1+mj-j))*sum(xc3);
     end;

     else if n1=n/2 then do;
      xc4= j(n1-mj+1,1,.);
          do l=0 to n1-mj;
          xc4[l+1]=(n-j-n1+mj-l)/(n-j-n1+mj+l)*comb(n-j-n1+mj+l,l)
```

```
                   *((1-p)**(l));
          end;
      pp=(p**(n-j-n1+mj))*sum(xc4);
      end;

      else if n1>n/2 & mj<=n-n1 then do;
      xc5= j(n-n1-mj+1,1,.);
          do l=0 to n-n1-mj;
          xc5[l+1]=(n1+mj-j-l)/(n1+mj-j+l)*comb(n1+mj-j+l,l)*
                   ((1-p)**(2*n1-n-1+l));
          end;
      pp=0.5*(p**(n-j-n1+mj))*sum(xc5)+(p**(n-j-n1+mj))*((1-p)**(n1-mj))
         *(comb(n-j,n1-mj)-comb(n-j,n1-j+mj));
      end;

      else if mj>n-n1 & j-mj<=n-n1 then

      pp=comb(n-j,n1-mj)*(p**(n-j-n1+mj))*(1-p)**(n1-mj);

end;

else if mj=n1 & j=n then

pp=1;

else if mj>n1 & j<n then
pp=0;

return (pp);
finish;
store module=efroncon;
```

```
quit;



proc iml;
load module=efronun;
load module=efroncon;
seed = &seed;
n1=&n1;
p=&p;
n=&num;
m=&numreps;
s=&numseq;
snew=j(s,1,.);
l=j(s,1,.);
lsum=j(m,1,.);
tnew=j(n,1,.);
xj=j(n,1,.);

use &data;
read all  var{trt} into trt ;
read all var{r} into r ;
read all var{censor} into d;
close &data;
do i=1 to m;
do j=1 to s;
mj=0;
do k=0 to n-1;
f=k+1;
c=ranuni(&seed+m);
if mj<n1 then do;
       if k=0 then
       pp=0.5*efroncon(p,n,n1,1,1)/efronun(p,n,n1);
       else if k>=1 then do;
       if mj=(k+1)/2 then
       pp= 0.5*efroncon(p,n,n1,k+1,mj+1)/efroncon(p,n,n1,k,mj) ;
       else if mj>(k+1)/2 then
       pp=(1-p)*efroncon(p,n,n1,k+1,mj+1)/efroncon(p,n,n1,k,mj) ;
       else if mj<(k+1)/2 then
       pp=p*efroncon(p,n,n1,k+1,mj+1)/efroncon(p,n,n1,k,mj);
       end;
         end;
     else pp=0;
      if c<=pp then do;
      tnew[f]=1;
      mj=mj+1;
      end;
      else do; tnew[f]=0; mj=mj+0;end;
xj[f]=n-f+1;
end;
if upcase(&model)='NONE' then aj=d-(r+1);
else if upcase(&model)='AFT' | upcase(&model)='PH' then aj=ranktie(r);
score=aj-sum(aj)/n;
sobs=trt'*score;
snew[j]=tnew'*score;
if upcase(&pvalueside)='UPPER' then do;
if snew[j]>abs(sobs) then l[j]=1; else l[j]=0;
end;
if      upcase(&pvalueside)='LOWER' then do;
if snew[j]<-abs(sobs)  then l[j]=1; else l[j]=0;


end;
if upcase(&pvalueside)='TWO SIDED' then do;
if snew[j]>abs(sobs) |snew[j]<-abs(sobs)  then l[j]=1; else l[j]=0;
end;
end;
lsum[i]=sum(l);
end;
pvalue=sum(lsum)/(m*s);
xx= pvalue;
cname = { "P-value" };
create x from xx [colname=cname];
append from xx;
quit;
data x2;
n=&num;
n1=&n1;
Scores=&Scores;
Alternative=upcase(&pvalueside);set x;
run;

proc print data=x2;
title "&testtype Test Result for &testranpro (p=&p) Randomization";
run;

%goto stopmactest;
%end;

/*Compute the test for unconditonal PBD*/
/*Truncated binomial*/
%if %upcase(&testtype)=UNCONDITIONAL and %upcase(&testranpro)=PBD
and  %upcase(&testproinblk)=TBD
%then %do;
```

```
proc iml;
seed = &seed;
n=&num;
m=&numreps;
s=&numseq;
blocksize=&blksize ;
snew=j(s,1,.);
ll=j(s,1,.);
lsum=j(m,1,.);
tnew=j(n,1,.);
xj=j(n,1,.);

use &data;
read all  var{trt} into trt ;
read all var{r} into r ;
read all var{censor} into d ;
close &data;
do i=1 to m;
do j=1 to s;
if mod(n,blocksize)=0 then l=n/blocksize;
else if mod(n, blocksize)>0 then l=int(n/blocksize)+1;
do k=1 to l;
 mj=0; nj=0;
 do kk= 0 to blocksize-1 until (k=l & kk=n-(l-1)*blocksize-1);
 id=kk+(k-1)*blocksize+1;
  if mj=blocksize/2 then p=0;
        else if max(mj,nj)<blocksize/2 then p=0.5;
        else if nj=blocksize/2 then p=1;
c=ranuni(&seed+m);
if c<p then do;tnew[id]=1 ; nj=nj+0;    mj=mj+1; end;
else do ;tnew[id]=0; nj=nj+1;   mj=mj+0;end; xj[id]=n-id+1;
```

```
end; end;


if upcase(&model)='NONE' then aj=d-(r+1);
else if upcase(&model)='AFT' | upcase(&model)='PH' then aj=ranktie(r);
score=aj-sum(aj)/n;
sobs=trt'*score;
snew[j]=tnew'*score;
if upcase(&pvalueside)='UPPER' then do;
if snew[j]>abs(sobs) then ll[j]=1; else ll[j]=0;
end;
if      upcase(&pvalueside)='LOWER' then do;
if snew[j]<-abs(sobs)  then ll[j]=1; else ll[j]=0;


end;
if upcase(&pvalueside)='TWO SIDED' then do;
if snew[j]>abs(sobs) |snew[j]<-abs(sobs)  then ll[j]=1; else ll[j]=0;
end;
end;
lsum[i]=sum(ll);
end;
pvalue=sum(lsum)/(m*s);
xx= pvalue;
cname = { "P-value" };
create x from xx [colname=cname];
append from xx;
quit;
data x2;
n=&num;


Scores=&Scores;
Alternative=upcase(&pvalueside);set x;
```

```
run;

proc print data=x2;
title "&testtype Test Result for &testranpro with &testproinblk
Randomization";
run;

%goto stopmactest;
%end;



/*Random allocation Rule*/
%if %upcase(&testtype)=UNCONDITIONAL and %upcase(&testranpro)=PBD and
%upcase(&testproinblk)=RAR
%then %do;

proc iml;
seed = &seed;
n=&num;
m=&numreps;
s=&numseq;
blocksize=&blksize ;
snew=j(s,1,.);
ll=j(s,1,.);
lsum=j(m,1,.);
tnew=j(n,1,.);
xj=j(n,1,.);

use &data;
read all  var{trt} into trt ;
read all var{r} into r ;
read all var{censor} into d ;
close &data;
do i=1 to m;
do j=1 to s;
if mod(n,blocksize)=0 then l=n/blocksize;
else if mod(n, blocksize)>0 then l=int(n/blocksize)+1;
do k=1 to l;
 mj=0;
 do kk= 0 to blocksize-1 until (k=l & kk=n-(l-1)*blocksize-1);
id=kk+(k-1)*blocksize+1;
c=ranuni(&seed+m);
 p=(blocksize/2-mj)/(blocksize-kk);
 if c<p then do; tnew[id]=1 ;   mj=mj+1; end;
   else do ;tnew[id]=0; mj=mj+0; end;
   xj[id]=n-id+1;
  end;
  end;
if upcase(&model)='NONE' then aj=d-(r+1);
else if upcase(&model)='AFT' | upcase(&model)='PH' then aj=ranktie(r);
score=aj-sum(aj)/n;
sobs=trt'*score;
snew[j]=tnew'*score;
if upcase(&pvalueside)='UPPER' then do;
if snew[j]>abs(sobs) then ll[j]=1; else ll[j]=0;
end;
if       upcase(&pvalueside)='LOWER' then do;
if snew[j]<-abs(sobs)  then ll[j]=1; else ll[j]=0;


end;
if upcase(&pvalueside)='TWO SIDED' then do;
if snew[j]>abs(sobs) |snew[j]<-abs(sobs)  then ll[j]=1; else ll[j]=0;
```

```
end;
end;
lsum[i]=sum(ll);
end;
pvalue=sum(lsum)/(m*s);
xx= pvalue;
cname = { "P-value" };
create x from xx [colname=cname];
append from xx;
quit;
data x2;
n=&num;
Scores=&Scores;
Alternative=upcase(&pvalueside);set x;
run;

proc print data=x2;
title "&testtype Test Result for &testranpro with &testproinblk
Randomization";
run;

%goto stopmactest;
%end;



/*Conditional PBD*/
/*TBD*/
%if %upcase(&testtype)=CONDITIONAL and %upcase(&testranpro)=PBD and
%upcase(&proinblk)=TBD  %then %do;

proc iml;
```

```
seed = &seed;
n1=&n1;
n=&num;
m=&numreps;
s=&numseq;
blocksize=&blksize ;
snew=j(s,1,.);
ll=j(s,1,.);
lsum=j(m,1,.);
tnew=j(n,1,.);
xj=j(n,1,.);
use &data;
read all  var{trt} into trt ;
read all var{r} into r ;
read all var{censor} into d;
close &data;
do i=1 to m;
do j=1 to s;
if mod(n,blocksize)=0 then l=n/blocksize;
else if mod(n, blocksize)>0 then l=int(n/blocksize)+1;
n11=0;
do k=1 to l;
 mj=0; nj=0;
 do kk= 0 to blocksize-1 until (k=l & kk=n-(l-1)*blocksize-1);
 id=kk+(k-1)*blocksize+1;
 if n11<n1 then do;
    if mj=blocksize/2 then p=0;
        else if max(mj,nj)<blocksize/2 then p=0.5;
        else if nj=blocksize/2 then p=1; end;
        else p=0;
c=ranuni(&seed+m);
```

```
if c<p then do;tnew[id]=1 ; nj=nj+0;    mj=mj+1;n11=n11+1; end;
else do ;tnew[id]=0; nj=nj+1;   mj=mj+0;n11=n11+0;end;
xj[id]=n-id+1;
end; end;
if upcase(&model)='NONE' then aj=d-(r+1);
else if upcase(&model)='AFT' | upcase(&model)='PH' then aj=ranktie(r);
score=aj-sum(aj)/n;
sobs=trt'*score;
snew[j]=tnew'*score;
if upcase(&pvalueside)='UPPER' then do;
if snew[j]>abs(sobs) then ll[j]=1; else ll[j]=0;
end;
if      upcase(&pvalueside)='LOWER' then do;
if snew[j]<-abs(sobs)  then ll[j]=1; else ll[j]=0;


end;
if upcase(&pvalueside)='TWO SIDED' then do;
if snew[j]>abs(sobs) |snew[j]<-abs(sobs)  then ll[j]=1; else ll[j]=0;
end;
end;
lsum[i]=sum(ll);
end;
pvalue=sum(lsum)/(m*s);
xx= pvalue;
cname = { "P-value" };
create x from xx [colname=cname];
append from xx;
quit;
data x2;
n=&num;
n1=&n1;
```

```
Scores=&Scores;
Alternative=upcase(&pvalueside);set x;
run;

proc print data=x2;
title "&testtype Test Result for &testranpro with &testproinblk
Randomization";
run;

%goto stopmactest;
%end;


/*Random allocation Rule*/
%if %upcase(&testtype)=CONDITIONAL and %upcase(&testranpro)=PBD and
%upcase(&proinblk)=RAR  %then %do;

proc iml;
seed = &seed;
n=&num;
m=&numreps;
s=&numseq;
blocksize=&blksize ;
n1=&n1;
snew=j(s,1,.);
ll=j(s,1,.);
lsum=j(m,1,.);
tnew=j(n,1,.);
xj=j(n,1,.);
use &data;
read all  var{trt} into trt ;
read all var{r} into r ;
read all var{censor} into d ;
close &data;
do i=1 to m;
do j=1 to s;
if mod(n,blocksize)=0 then l=n/blocksize;
else if mod(n, blocksize)>0 then l=int(n/blocksize)+1;
n11=0;
do k=1 to l;
 mj=0;
 do kk= 0 to blocksize-1 until (k=l & kk=n-(l-1)*blocksize-1);
id=kk+(k-1)*blocksize+1;
 if n11<n1 then p=(blocksize/2-mj)/(blocksize-kk); else p=0;
 c=ranuni(&seed+m);
 if c<p then do; tnew[id]=1 ;   mj=mj+1; n11=n11+1;end;
   else do ;tnew[id]=0; mj=mj+0; n11=n11+0;end;
 xj[id]=n-id+1;
  end;
  end;
if upcase(&model)='NONE' then aj=d-(r+1);
else if upcase(&model)='AFT' | upcase(&model)='PH' then aj=ranktie(r);
score=aj-sum(aj)/n;
sobs=trt'*score;
snew[j]=tnew'*score;
if upcase(&pvalueside)='UPPER' then do;
if snew[j]>abs(sobs) then ll[j]=1; else ll[j]=0;
end;
if      upcase(&pvalueside)='LOWER' then do;
if snew[j]<-abs(sobs)  then ll[j]=1; else ll[j]=0;


end;
```

```
if upcase(&pvalueside)=‘TWO SIDED’ then do;
if snew[j]>abs(sobs) |snew[j]<-abs(sobs)  then ll[j]=1; else ll[j]=0;
end;
end;
lsum[i]=sum(ll);
end;
pvalue=sum(lsum)/(m*s);
xx= pvalue;
cname = { "P-value" };
create x from xx [colname=cname];
append from xx;
quit;
data x2;
n=&num;
n1=&n1;
Scores=&Scores;
Alternative=upcase(&pvalueside);set x;
run;

proc print data=x2;
title "&testtype Test Result for &testranpro with &testproinblk
Randomization";
run;
 %goto stopmactest;
%end;
%stopmactest:
%mend;
```

Bibliography

Bibliography

[1] Agresti, A., Caffo, B., Ohman-Strickland, P. (2004). Examples in which misspecification of a random effects distribution reduces efficiency and possible remedies. *Biostatistics* **47** 639-653.

[2] Breslow, N. (1978). The proportional hazards model: applications in epidemiology. *Communications in Statistics* **A7** 315-332.

[3] Chen, J., Zhang, D., Davidian, D. (2002). A Monte Carlo EM algorithm for generalized linear mixed models with flexible random effects distribution. *Biostatistics* **3** 347-360.

[4] Chow, S-C., Liu, J-P. (2004). *Design and Analysis of Clinical Trials*. New York: John Wiley and Sons.

[5] Collett, D. (2003). *Modeling Survival Data in Medical Research*. Boca Raton: CRC Press.

[6] Cook, T. D., DeMets, D. L. (2008). *Introduction to Statistical Methods for Clinical Trials*. Boca Raton: CRC Press.

[7] Edmonson, J. H., Fleming, T.R., Decker, T. G., Malkasian, G. D., Jorgensen, E. O., Jefferies, J. A., Webb, M. J., Kvols, L. K. (1979). Different chemotherapeutic sensitivities and host factors affecting prognosis in advanced ovarian carcinoma versus minimal residual disease. *Cancer Treatment Reports* **63** 641-647.

[8] Efron, B. (1971). Forcing a sequential experiment to be balanced. *Biometrika* **69** 61-67.

[9] Fitzmaurice, G., Laird, N., Ware, J. (2004). *Applied Longitudinal Analysis.* New York: John Wiley and Sons.

[10] Friedman, A. S. (1975). Interaction of drug therapy with marital therapy in depressive patients. *Archives of General Psychiatry* **32** 619-637.

[11] Freireich, E. J., Gehan, E., Frei, E. F., Schroeder, L. R., Wolman, I. J., Anbari, R., Burgert, E. O., Mills, S. D., Pinkel, D., Selaway,

O. S., Moon, J. H., Gendel, B. R., Spurr, C. L., Storrs, R., Haurani, F., Hoogstraten, B., Stanley, L. (1963). The effect of 6-mercaptopurine on the duration of steroid-induced remissions in acute leukemia: a model for evaluation of other potentially useful therapy. *Blood* **21** 699-716.

[12] Jacqmin-Gadda, H., Sibillot, S., Proust, C., Molina, J.-M., Thiébaut, R. (2007). Robustness of the linear mixed model to misspecified error distribution. *Computational Statistics and Data Analysis* **51** 5142-5154.

[13] Gail, M. H., Tan, W. Y., Piantadosi, S. (1988). Tests for no treatment effect in randomized clinical trials. *Biometrika* **75** 57-64.

[14] Gail, M. H., Wieand, S., Piantadosi, S. (1984). Biased estimates of treatment effect in randomized experiments with nonlinear regressions and omitted covariates. *Biometrika* **71** 431-444.

[15] Gibbons, J D., Chakraborti, S. (2010). *Nonparametric Statistical Inference*. Boca Raton: CRC Press.

[16] Henry, K., Erice, A., Tierney, C., Balfour, H. H., Fischl, M. A., Kmack, A., Liou, S. H., Kenton, A., Hirsch, M. S., Phair, J., Martinez, A., Kahn J. O. (1998). A randomized, controlled, double-blind study comparing the survival benefit of four different reverse transcriptase inhibitor therapies (three-drug, two-drug, and alternating drug) for the treatment of advanced AIDS. *Journal of Acquired Immune Deficiency Syndromes and Human Retrovirology* **19** 339-349.

[17] Kalbfleisch, J. D., Prentice, R. L. (1980). *Statistical Analysis of Failure Time Data*. New York: John Wiley and Sons.

[18] Kempthorne, O. (1977). Why randomize? *Journal of Statistical Planning and Inference.* **1** 1-25.

[19] Kennes, L. N., Hilgers, R-D., Heussen, N. (2012). Choice of the reference set in a randomization test based on linear ranks in the presence of missing values. *Communications in Statistics-Simulation and Computation* **47** 1051-1061.

[20] Kleinbaum, D. G., Klein, M. (2005). *Survival Analysis: A Self-Learning Text*. London: Springer.

[21] Lachin, J. M. (1988). Statistical properties of randomization in clinical trials. *Controlled Clinical Trials* **9** 289-311.

[22] Lachin, J. M., Matts, J. P., Wei, L. J. (1988). Randomization in clinical trials:conclusions and recommendations. *Controlled Clinical Trials* **9** 365-374.

[23] Lagakos, S. W., Schoenfeld, D. A. (1984). Properties of proportional-hazards score tests under misspecified regression models. *Biometrics* **40** 1037-1048.

[24] Lehmann, E. L. (2006). *Nonparametrics: Statistical Methods Based on Ranks.* New York: Springer.

[25] Litiére, S., Alonso, A., Molenberghs, G. (2007). Type I and type II error under random-effects misspecification in generalized linear mixed models. *Biometrics* **63** 1038-1044.

[26] Ludbrook, J., Dudley, H. (1998). Why permutation tests are superior to t and F tests in biomedical research. *The American Statistician* **52** 127-132.

[27] Markaryan, T., Rosenberger, W. F. (2010). Exact properties of Efron's biased coin randomization procedure. *Annals of Statistics* **38** 1546-1567.

[28] McCullagh, P., Nelder, J. A. (1989). *Generalized Linear Models*. London: Chapman and Hall.

[29] Morgan, T. M., Lagakos, S. W., Schoenfeld, D. A. (1986). Omitting covariates from the proportional hazards model. *Biometrics* **42** 993-995.

[30] Plamadeala, V., Rosenberger, W. F. (2012). Sequential monitoring with conditional randomization tests. *Annals of Statistics* **40** 30-44.

[31] Rosenberger, W. F., Lachin, J. M. (2002). *Randomization in Clinical Trials: Theory and Practice*. New York: John Wiley and Sons.

[32] SAS Institute Inc. (2009). *SAS/STAT 9.2 User's Guide*. Cary, NC. *http://www.support.sas.com.*

[33] Simon, R., Simon, N. R. (2011). Using randomization tests to preserve type I error with response adaptive and covariate adaptive randomization. *Statistics and Probability Letters* **81** 767-772.

[34] Smith, R. L. (1984). Sequential treatment allocation using biased coin designs. *Journal of the Royal Statistical Society B* **46** 519-543.

[35] Thall, P.F., Vail, S. C. (1990). Some covariance models for longitudinal count data with overdispersion. *Biometrics* **46** 657-671.

[36] Zhang, Y., Rosenberger, W. F. (2008). Sequential monitoring with conditional randomization tests: generalized biased coin design. *Sequential Analysis* **27** 234-253.

Curriculum Vitae

Parwen Parhat graduated from Peking University with a B.A in Economics and a B.S. in Statistics (double major) in 2006. She received her Masters degree in Economics from the University of Houston in 2008, and then in 2009, she came to George Mason University to study Statistics. In 2011, she worked as a research assistant in the Robert Graham Center for policy studies in family medicine and primary care of the American Academy of Family Physicians, conducted statistical analysis of large health care data, participated in a grant project studying the social accountability of Graduate Medical Education, and helped in manuscript preparations which resulted in two peer-reviewed publications. From 2011 to 2013, as a research assistant for Dr.Hughes-Oliver, she worked on reviewing and evaluating major existing classification measures, and performed data analysis with various statistical classification techniques.