Major's Project Report

Zhuoran Zhao zhuoran.zhao@mail.mcgill.ca
Supervisor: Professor Russell Steele russell.steele@mcgill.ca

December 2019

Abstract

As the one of the most effective way to improve the scientific discoveries, sharing research data has been encouraged and demanded by many institutions. One the one hand, data sharing could facilitate many fields such as medical treatment, collaborative research and biostatistics. On the other hand, due to the limited data resource for rare diseases, the data sharing may lead to unique identification of patients. Based on this situation, we tend to provide completely synthetic datasets who maintain fidelity to an underlying data model while also preserve privacy. Professor Steele and his colleagues proposed a DA-MI model which is an improvement for MI model proposed by Professor Rubin in 1993.[1] In this major research, I learned the posterior distribution from Bayesian Inference that used in the paper and implemented the DA-MI model by simulation in Rstan.

1 Introduction

MI model, also known as 'multiply imputed model', mainly draws synthetic dataset from the posterior predictive distribution under proper imputation models to replace target sensitive values. MI model was first introduced by Professor Rubin in "Statistical disclosure limitation" [1] Different from MI model who specifically assumed that there existed the missing data in datasets, our new DA-MI model let us have access to the observed sensitive values. Particulalry, the availability of true values let us create psudo observations by adding data augmentation mask models. And we also know that if imputation model could always predict correctly, although the synthetic datasets and the original datasets would share the same information, the privacy protection is also minimized. Thus the paper's goal was to create multiple imputed datasets to maintain high quality inference outcomes while control the identity disclosure risks as well as preserve the simplicity of inference procedures. In this report, we will start from background information about Bayesian Inference; then introduce the DA-MI approach proposed by paper, also the theory of data utility and disclosure risk. In the end we will talk about the simulation work and results.

2 Background Study

2.1 Bayesian Inference

"Bayesian inference is a method of statistical inference in which Bayes' theorem is used to update the probability for a hypothesis as more evidence or information becomes available."[2] Bayesian inference computes the posterior probability according to Bayes' theorem:

$$P(H|E) = \frac{P(E|H) * P(H)}{P(E)}$$

where H and E represent events. $P(E|H)$ is the conditional probability for event E if given the fact that event H has already occurred. P(H) and P(E) are the marginal probabilities for event H and E. Bayesian inference derives the posterior probability in a similar way, except it uses prior probability and likelihood function. Now in case of Bayesian inference:

$$P(\theta|data) = \frac{P(data|\theta) * P(\theta)}{P(data)}$$

Here $P(\theta)$ is the prior distribution.$P(data|\theta)$ is the likelihood distribution. $P(\theta|data)$ is the posterior distribution. Prior distribution represents the true value of the parameters. Likelihood distribution is the given assumption. Posterior distribution represents the probability of the parameter values based on the knowledge of prior and likelihood. Overall, we can calculate the posterior distribution of parameters using prior beliefs updated with likelihood distribution.

2.2 MCMC

"The estimation of posterior probability distributions using a stochastic process known as Markov Chain Montre Carlo(MCMC)."[3] There are 3 common-used MCMC in contemporary Bayesian Inference, Metropolis Algorithm, Gibbs Sampling, and Hamiltonian Monte Carlo(A.K.A. HMC).

2.2.1 Metropolis Algorithm

One feature of Metropolis Algorithm was we could directly sample from the posterior without assuming a shape. As a general example of MCMC, Metropolis Algorithm was used to draw samples from an unknown and usually complex target distribution. Metropolis-Hastings is a specific implementation of MCMC. Especially it works well in high dimensional spaces as opposed to Gibbs Sampling(which we will introduce later).[4] The idea requires a proposal distribution to help draw samples from posterior distribution. It then uses this distribution to randomly walk in the distrbution space and accept or reject jumps according to how likely the sample is.

2.2.2 Gibbs Sampling

Compared to Metropolis approach, Gibbs sampling could be more efficient with fewer samples approach. Particularly, it gains efficiency by reducing the randomness and exploiting knowledge of the target distribution. Conjugate pairs, known as using particular combinations of prior distributions and likelihood, could give analytical solutions for posterior distribution of an individual parameter that allow Gibbs sampling to make smart jumps around the joint posterior distribution of all parameters. However, there are also some severe limitations to Gibbs Sampling, as conjugate prior sometime may choose a prior that may not be a strong argument from a scientific perspective. Second, Gibbs Sampling is not as efficient when we have larger and more complex parameters.

2.2.3 Hamiltonian Monte Carlo

"Hamiltonian Monte Carlo corresponds to an instance of the Metropolis–Hastings algorithm, with a Hamiltonian dynamics evolution simulated using a time-reversible and volume-preserving numerical integrator (typically the leapfrog integrator) to propose a move to a new point in the state space."[5] HMC is much more computationally costly than Metropolis or Gibbs sampling. However, its proposals are typically much more efficient. It works well than the other 2 algorithms when we have more complex models(As thousands or ten thousands of parameters). Even though HMC is more efficient, there still exist some limitations. HMC requires continuous parameters and it can not glide through a discrete parameter. This implies that HMC is not applicable to situations such as discrete missing data. In practice, HMC need to be turned into a particular model and its data. There is also where the Rstan comes in and initial intention for this research project.

2.3 Rstan

As a probabilisitc programming language, Stan was designed to provide efficient computing speed for specifiying statistical models such as Bayesian Inference. It provides full Bayesian Inference through variety of inference procedures. But primarily it used MCMC No-U-Turn sampler, an adaptive method of HMC. Maximum Likelihood Estimates are calculated using optimization methods. We usually call Stan through R by Rstan Package. [6] However, the actual work is done in C++. There are several advantages for using Rstan. Firstly, Stan works fast for large models. Given a very large model, the compliation time is usually minimal compared to running time. Thus the more efficient the algorithm, the faster of the running speed. It is mentionalble that for high dimensions and harder models, NUTS works extremely faster than other general currently available algorithms. Besides, Stan is memory-efficient. It stores the parameters and it stores extra variables for calculating gradient information compared to other techniques.[7] Particularly, in this project, we will take the advantage of Rstan and use it as the tool to implement our posterior distribution for synthetic datasets.

3 DA-MI Approach

Figure 1: A graphical representation of the joint imputation model in the DA-MI method.

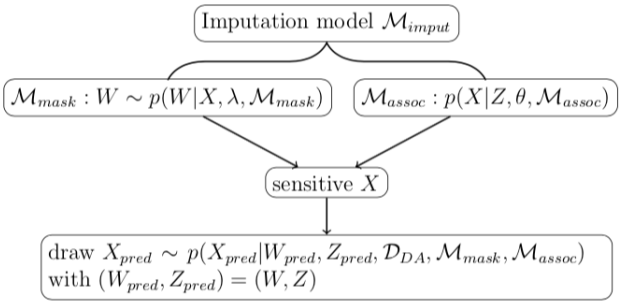

Figure 1: The Figure about the imputation model

Before we introduce the approach, let us define the several terms:

$$D = (X, Z)$$

$$X = (X_1 ... X_n) where X_i = (X_{i1}, ..., X_{ip})$$

$$Z = (Z_1 ... Z_n)$$

Our dataset is composed of 2 parts, X and Z, both of them are fully observed. For the i-th subject ,where i = 1...n, $X_i = (X_i 1, ... X_i p)$ is a vector of key identifying variables and Z_i is the remaining q variables. X_i is procuded is primarily determined by the assumed associations between X and Z. Since we do observe the X-information, the paper proposes the new approach to obtain synthetic copies for the target quasi-identifying variable X via a data-augmentation procedure. Our M_{imput} model consist of two model components: a masking model and an association model. Masking model, denoted by M_{mask}, it links the pseudo-variable W to X with artificially generated observations. Association model, denoted by M_{assco} defines the association between X and the rest of un-perturbed Z.

Under this framework, the observed data now became $D_{DA} = (D, W)$. Now we draw the synthetic copies X_{SYN} from $X_i ...$ by the posterior predictive distribution:

$$p(X_{pred}|Z_{pred}, W_{pred}, D_{DA}, M_{mask}, M_{assoc})$$

$$where (Z_{pred}, W_{pred}) = (Z_i, W_i)$$

Particularly, in this research project, we will do simulation study under the univariate case when $p = 1$. We carry out data-augmented multiple imputation procedure based on a joint imputation model. As indicated there are two components of models, Masking and Association.

3.1 Masking Component

The goal of this model is to balance the preserved X information and level of perturbation for the masking purpose. For each X_i, we produce pseudo-copies, denoted by W_i, and λ contains the tuning parameters:

$$M_{mask} : W_i \; p(W|X_i, \lambda, M_{mask})$$

Depending on the data type of Xi as either continuous or categorical, the model is defined accordingly.

3.1.1 Continuous X

For continuous X, masking model is generally adding the random Gaussian noises with the variance level chosen to control the X-information. In this case each X_i would produce K pseudo-observations, denoted by $W_i = (W_{ik}, k = 1...K)$

$$M_{mask} : W_{ik} = X_i + e_{ik}, k = 1, 2, ...K$$

where e_{ik} is from standard normal, K and σ^2 are key components for controlling the amount of information embedded in X_i would be transferred to W_{ik} Particularly, decreasing K or increasing σ^2, we would increase the level of disclosure protection.

3.1.2 Categorical X

When X_i is a categorical variable. For each X_i, we create a pseudo-observation W_i from a mixture distribution of C components according to the making model:

$$M_{mask} : W_i = \alpha * X_i + u_i$$

where u_i is from i.i.d standard normal and α controls the distance between two adjacent components and determines the information in W_i that can assist the recovery of X_i

3.2 Association Component

For the purpose of publicly releasing a dataset associating with a published study, the association model is commonly chosen to be the primary analytic model. Here we have:

$$M_{assoc} \; p(X|Z, \theta, M_{assoc})$$

, where θ denotes the model parameter.

3.3 Remarks

- Balacing the disclosure risk and quality of inferences of synthetic datasets could be achieved by varying λ. Changing the value of the tuning parameter λ could influence the information of X that preserved in W.
- X will be considered as covariates in the analysis model if X includes quasi-identifiers. Under this framework, we can have $Z^T = (Z_{resp}, Z_{cov}^T)$, as Z_{resp} is a response variable and Z_{cov} is a vector of q-1 coveriates. And we will introduce M_{analy} and M_x model to be the joint of M_{assoc}. M_x is a model that relates X to Z_{cov} and has the parameter ϕ. M_{analy} will have the parameter β.

4 Theory of Analyzing the result

In order to evaluate the data analytic utility, paper used two commonly used analysis-specific measures proposed by Snoke et al.(2018)[8] Basiclly, in our research project, we will also use these two techniques to evaluate our simulation results. Also, we will use one technique to measure our disclosure risk.

4.1 95% Confidence Overlap Measure

We will calculate the percentage overlap of CIs of true values and synthetic data based on following formula:

$$0.5[\frac{min(U_{orig}, U_{syn}) - max(L_{orig}, L_{syn})}{U_{orig} - L_{orig}} + \frac{min(U_{orig}, U_{syn}) - max(L_{orig}, L_{syn})}{U_{syn} - L_{syn}}]$$

where U and L represents the upper and lower bounds of the CI respectively. "orgi" and "syn" represents the orignial and synthetic datasets respectively. Higher data utility will correspond to higher positive vlue of interval overlap measure. And negative value suggests no overlap between two CIs. Particularly we will apply this formula to all four parameters to our simulated results.

4.2 Standardized Difference

We will use following formula to calculate the standardized difference between the estimates using the original and synthetic datasets:

$$\frac{|\hat{\beta}_{orig} - \hat{\beta}_{syn}|}{SE(\hat{\beta}_{orig})}$$

where $\hat{\beta_{orig}}$ and $\hat{\beta_{syn}}$ are the estimated coefficients obtained using the original and synthetic datasets respectively. $SE(\hat{\beta}_{orig})$ is the estimated standard error of the coefficient using the original dataset.

4.3 Disclosure Risk Measures

We use the disclosure risk measures from measurement proposed by Reiter (2005a).[9] Recall n is the sample size in the target sensitive dataset, and as we already assumed we could access to our original dataset. We define original dataset as the target t where t_i consists of all the variables we have.(Both categorical and continuous ones) And the synthetic dataset is $D_{pub} = {D_{pub}}^1, {D_{pub}}^2, ...{D_{pub}}^M$ that the M sub-datasets we released to users. Now for each record t_i, we would like to find the corresponding synthetic records. For the categorical identifier is that of same value, for the continuous is within a small range d. Now define:

$$Pr(J^{ti} = l|t_i, {D_{pub}}^m) = 1/r$$

J^{ti} is represented for a random variable for the corresponding record t_i and l is the index of lth synthetic dataset ${D_{pub}}^m$. This is saying that the probability of ith target t_i is matched to the lth data in synthetic dataset is $1/r$. And r represents the total records that matched on t_i. After we get this probability across the M synthetic datasets, we take the average of M and get $Pr(J^{ti} = l|t_i, D_{pub}) = 1/r$. Further,

$$c_i = i^* : Pr(J^{ti} = i^*|t_i, D_{pub}) = argmax_l Pr(J^{ti} = l|t_i, D_{pub})$$

c_i means the total number of records with the highest match probability for the target record t_i. Now we have c_i and let T_i be the indicator when the correct match. Now define I_i to be the indicator:

$$I_i = \begin{cases} 1 & c_i * T_i = 1 \\ 0 & otherwise \end{cases} \tag{1}$$

Finally, the disclosure risk, aka true match risk is

$$\pi_{DR} = n^{-1} \sum_{i=1}^{n} I_i$$

5 Simulation

5.1 Data

The key part of this major's project is the simulation study. We will introduce how to generate the variables one by one. Let n be 500, fix $\beta_0 = 1$, $\beta_{x1} = 1$, $\beta_{x2} = 1$, $\beta_z = 1$. Let $i = 1, 2, ..., n$ Overall, we let k equals to 15, α equals 1.

$$U_{i1}, U_{i2} : Uniform(-1, 1), \sigma = 0.5$$

$$Z_{cov} : Uniform(-2, 2)$$

$$X_{1i} : Normal({Z_{cov}}^2 + U_{i1}, 1)$$

$$X_{2i} : Normal({Z_{cov}}^2 + U_{i1}, 1)$$

$$Z_{resp}, i : Bernoulli(p = \Phi^{-1}(1 + X_{1i} + x_{2i} + Z_{cov}, i)$$

For each subject i, $i = 1, ..., n$, we let Z_{resp}, i be a binary outcome of interest in the analysis model which is a probit regression model that relates Z_{resp}, i to three coveriates X_{1i}, X_{2i} and Z_{cov}, i. Then we let Z_{cov} to be uniformly distributed from -2 to 2, X_1, X_2 to be normally distributed with mean ${Z_{cov}}^2 + U_{i1}$ and variance 1. And finally U_{i1}, U_{i2} to be marginally follow unifrom (-1, 1) with a correlation of 0.5.

5.2 Model

5.2.1 Mask Model

For the mask models, since we have both X1 and X2 as continuous variables, we could generate pseudo-obervations as generate $W_{1,ik}$ and $W_{2,ik}$ by adding independent random errors with variance σ^2 to X_{1i} and X_{2i}.

5.2.2 Analysis Model

$$p(Z_{resp,i}|X_{1i}, X_{2i}, Z_{cov,i}, \beta, M_{analy}) = \pi_i^{Z_{resp,i}} * (1 - \pi_i)^{1 - Z_{resp,i}}$$

where

$$\pi_i = \Phi(\beta_0 + \beta_{x1} * X_{1i} + beta_{x2} * X_{2i} + beta_z * Z_{cov,i})$$

We are using the association from analysis model for both truth value and synthetics' values' corresponding parameters. By the R package GLM.

5.2.3 Mx

$p(X_{i1}, X_{i2}|Z_{cov,i}, \phi, Mx) = p(X_{i1} = |Z_{cov,i}, \phi_1, Mx_1) * p(X_{i2} = |Z_{cov,i}, \phi_2, Mx_2)$

where Mx_1 and Mx_2 are, respectively, 2 Normal-distributed model, with mean of X_{1i} and X_{2i}as a simple linear function of $Z_{cov,i}$.

6 Results

We implemented the above simulation in R and model in Rstan. Within each 500 synthetic datasets result, we take the subset of 5 datasets(100th, 200th, 300th, 400th and 500th) and take the generalized linear model results of these 5 results to get the corresponding synthetic parameters. And we also take the generalized linear model of the true values. We compared these parameters by 95% overlap rate and standardized difference for data utility. Then we use the disclosure risk to measure the percentage of risk to be uniquely identified. Particularly, we repeat our simulation for 25 times. Notice, I attached all the graphs and code at the end of this report.

6.1 X1, X2, Zcov, Intercept Parameters

Using the association between Z_{resp} and X_1, X_2, Z_{cov}, we use the package generalized linear model from r to fit our synthetic dataset and we compare each of the results to the original value. With comparison to the original datasets, we could see all of these four parameters contained the original parameters for the generalized linear model. And it could be also seen that normally our synthetic datasets' parameters have a larger range compared to the original datasets'. Besides, it is noticeable that there exists one outlier for intercept and X2, our synthetic model also recognize this value. The boxplot of this part could be found in Appendix 1-1. The scatterplot of this part could be found in Appendix 1-2.

6.2 Standardized Difference

Similar to the calculation procedure in 95% overlap rate. With alpha equals to 1, all betas equal to 1, we have the average standardized difference for all parameters. Also, as indicated in the graph and summary table, overall we have a similar range for X1, X2 and Z_{cov} except intercept may have a higher minimum value. All these three parameters have the average around 0.6, but intercept has 0.9. Then, compared to the paper, we could see we reached a very close result. The graph is attached in Appendix 2.

6.3 95% Overlap Rate

Instead of taking the relative percentage of overlap rate as paper did, we just take this value. We calculate this value coefficient by coefficient and dataset by dataset, then use these measurements to produce an average. We could see that overall trend of this result is very close to what we get in standard difference. Excpet, this time we have Z_{cov} that behaves differently from the other three. From the summary table we could see that the minimum value for overlap are all negative but very close to 0 and the maximum is around 0.7 except Z_{cov} . The graph is attched in Appendix 3.

6.4 Disclosure Risk

Here in our simulation study, we choose d to be 0.1. And each t_i is composed of ith record of X_1 and X_2. We could see from the result that the overall disclosure risk is pretty small with even the maximum 0.008 and average 0.00392, this indicates that the data is well protected and is less likely to find the match for each target true data record. The graph is attached in Appendix 4.

7 Summary

From our results, we could know that the overall disclosure rate is very low and our data utility is still reasonable. This result indicate that our model is very good for both data protection and minimal disclosure risk. Besides, consider for each simulation we are taking 5 out of synthetic datasets, and we did this simulation for 25 times, we could say this result is very stable under the large simulaiton. Overall, in this project, I started from reading the book to learn basic bayesian inference knowledge to self-implement the posterior distribution for producing synthetic datasets with the help of Professor Steele in Rstan. Not only I learned how bayesian inference could be applied to multiple fields for prediction, but also I got an insight on how Rstan particularly useful for implementing models for posterior distributions. Throughout this project, I also developed the interest for research for data privacy and protection. In the end, I want to thank Professor Steele for all the help.

References

[1] Rubin (1993) Statistical disclosure limitation
Journal of Official Statistics 9(2), 461-468

[2] Wikipedia: Bayesian Inference
Retrived from *https://en.wikipedia.org/wiki/Bayesian_inference*

[3] Richard McElreath (2016) A Bayesian course with examples in R and Stan
Boca Raton : CRC Press/Taylor Francis Group

[4] Joseph Moukarzel (2018, November 3).
Markov Chain Monte Carlo and Metropolis Hastings, in python
Retrieved from *https://towardsdatascience.com/from-scratch-bayesian-inference-markov-chain-monte-carlo-and-metropolis-hastings-in-python-ef21a29e25a*

[5] Wikipedia:Hamiltonian Monte Carlo
Retrived from *https://en.wikipedia.org/wiki/Hamiltonian_Monte_Carlo*

[6] Introduction to Stan
Retrived from *r-bloggers.com/an-introduction-to-stan-with-r/*

[7] Selling Rstan
Retrived from *https://discourse.mc-stan.org/t/selling-stan/3693/2*

[8] Snoke et al (2018) General and specific measures for synthetic data
Journal of the Royal Statistical Society: Series A(Statistics in Society) 181(3), 663-688

[9] Reiter, J.P (2005a) Estimating risks of identification disclosure in micro-data
Journal of the American Statistical Association 100(472), 1103-1112

A Appendix 1-1

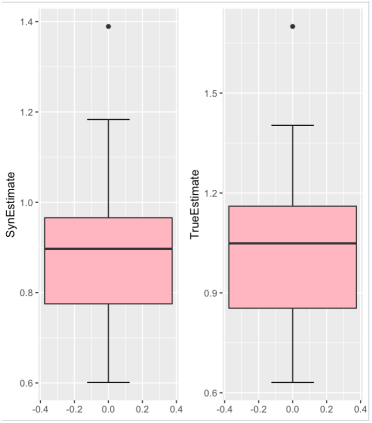

Figure 2: Boxplot for X1

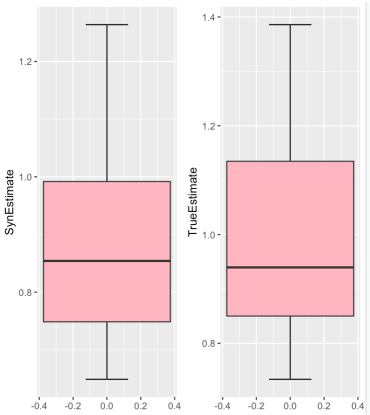

Figure 3: Boxplot for X2

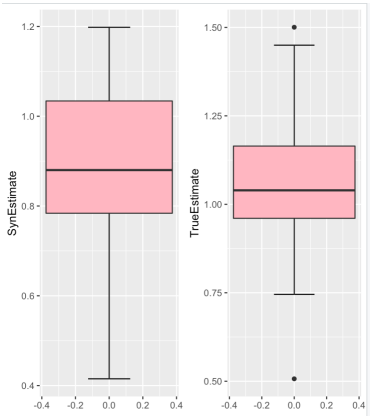

Figure 4: Boxplot for intercept

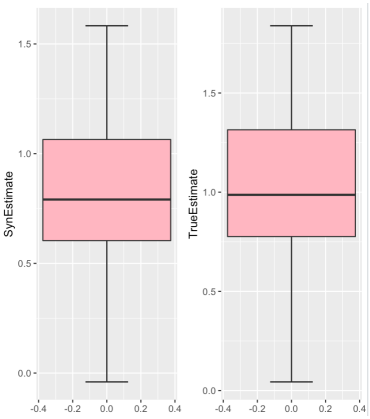

Figure 5: Boxplot for Zcov

B Appendix 1-2

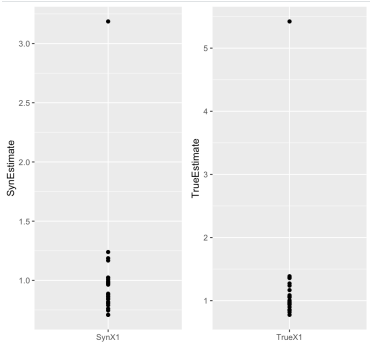

Figure 6: Scatterplot for X1

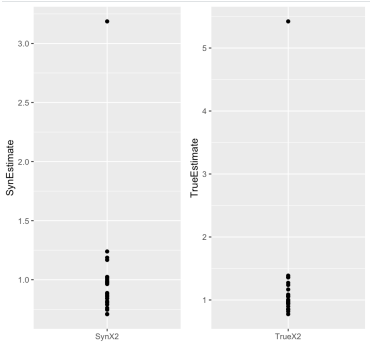

Figure 7: Scatterplot for X2

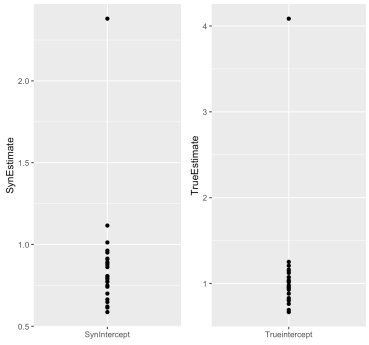

Figure 8: Scatterplot for intercept

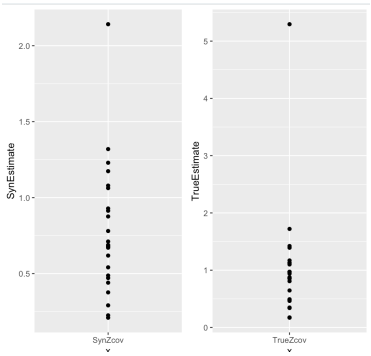

Figure 9: Scatterplot for Zcov

C Appendix 2

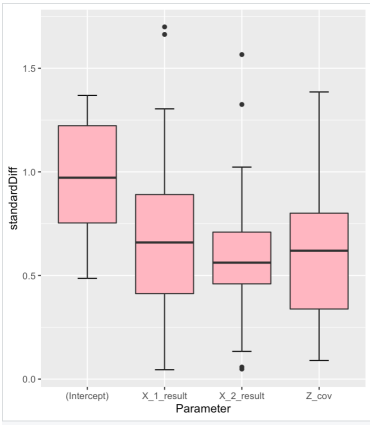

Figure 10: Boxplot for standard difference

```
# A tibble: 4 x 4
  Parameter   Avg_SD Max_SD Min_SD
  <chr>        <dbl>  <dbl>  <dbl>
1 (Intercept)  0.953   1.37 0.486
2 X_1_result   0.681   1.70 0.0451
3 X_2_result   0.605   1.57 0.0484
4 Z_cov        0.601   1.39 0.0896
```

Figure 11: Summary for standard difference

D Appendix 3

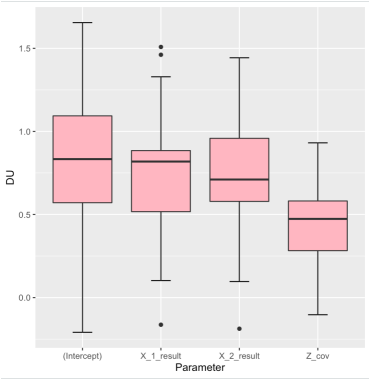

Figure 12: Boxplot for overlap

```
# A tibble: 4 x 4
  Parameter   Avg_DU Max_DU Min_DU
  <chr>        <dbl>  <dbl>  <dbl>
1 (Intercept)  0.840  1.65  -0.209
2 X_1_result   0.724  1.51  -0.163
3 X_2_result   0.727  1.44  -0.187
4 Z_cov        0.442  0.931 -0.103
```

Figure 13: Summary for overlap

E Appendix 4

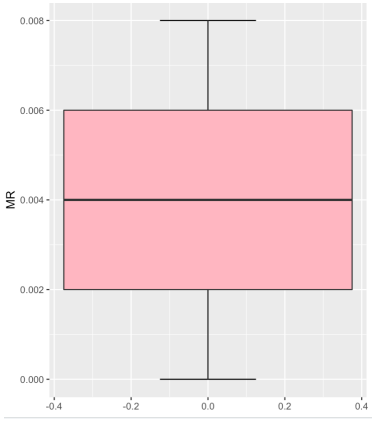

Figure 14: Boxplot for overlap

	Avg_MR	Max_DU	Min_DU
1	0.00392	0.008	0

Figure 15: Summary for overlap

F Rstan Code

```
data {
  int N;
  int K;
  real Z_cov[N];
  int Z_resp[N];
  real W1[N, K];
  real W2[N, K];
  real X1[N];
  real X2[N];
  real alpha;
}

parameters {
  real phi1[2];
  real phi2[3];
  real beta0;
  real betaX1;
  real betaX2;
  real betaZ;
  real X1_new[N];
  real X2_star_new[N];
}

model {
  for (n in 1:N)
    X1[n] ~ normal(Z_cov[n]*phi1[1] + phi1[2], 1);

  for (n in 1:N)
    X2[n] ~ normal(Z_cov[n]*phi1[1] + phi1[2], 1);

  for (n in 1:N)
    X1_new[n] ~ normal(Z_cov[n]*phi1[1] + phi1[2], 1);

  for (n in 1:N)
    X2_star_new[n] ~ normal(Z_cov[n]*phi1[1] + phi1[2], 1);

  for (n in 1:N)
    Z_resp[n] ~ bernoulli(Phi(beta0 + betaX1*X1[n] + betaX2*X2[n] + betaZ*Z_cov[n]));

  for (n in 1:N)
    for (k in 1: K)
      W1[n,k] ~ normal(X1_new[n],1);

 for (n in 1:N)
    for (k in 1: K)
      W2[n,k] ~ normal(X2_star_new[n],1);
}

```

Figure 16: Stan Code

G R Code for generating models

```r
library(mitools)
library(tidyverse)
#************************************Simulation Function*************************************************************#
# r as target correlation between U1 and U2
# n as the total number of simulation
# k as the number of observations for each X(as data augmentation part)
# var as the variance for error added to models(chosen from {1, 3, 5})
setwd("~Desktop/FALL2019/MATH410")
simulation <- function(iteration){
  r <- 0.5                          # Target (Spearman) correlation
  n <- 500                          # Number of samples

  # Generate U
  gen.gauss.cop <- function(r, n){
    rho <- 2 * sin(r * pi/6)          # Pearson correlation
    P <- toeplitz(c(1, rho))          # Correlation matrix
    d <- nrow(P)                      # Dimension
    U <- pnorm(matrix(rnorm(n*d), ncol = d) %*% chol(P))
    return(U)
  }

  # U1 and U2 are uniformly correlated
  U <- gen.gauss.cop(r, n)
  U_1<-U[,1]
  U_2<-U[,2]

  # Z_cov has uniform distribution with (-2, 2)
  Z_cov <- runif(n, -2, 2)

  # Generate X1, X2
  miu <- Z_cov^2 + U_1
  X1 <- rnorm(n, miu, 1)
  X2 <- rnorm(n, miu, 1)

  # Z_resp is from bernoulli
  l <- 1 + X1 + X2 + Z_cov
  p_z <- pnorm(l)                   # calculate the cdf of l
  Z_resp <- rbinom(n, 1, p_z)

  # Mask Model
  k = 15                            # number of oberservation for each X fixed in paper
  var = 1                           # variance for error added to model {1, 3, 5}
  X1p = t(rep(1, k) %*% t.default(X1))
  W1 = X1p + matrix(rnorm(500*k, 0, var), ncol=k)

  X2p = t(rep(1, k) %*% t.default(X2))
  W2 = X2p + matrix(rnorm(500*k, 0, var), ncol=k)


```

Figure 17: RCode Part1

```r
  # Stan model
  library(rstan)
  rstan_options(auto_write = TRUE)
  model_Mx <- stan_model('Test_3.stan')
  fit_1 <- sampling(model_Mx, list(N=n, K=k, Z_cov=Z_cov,Z_resp = Z_resp,
                                   W1 = W1, W2=W2, X1=X1, X2=X2,alpha = 1))

  # For loop to create 5 synthetic datasets from draws 100,200,300,400,500 from sampler
  list_of_draws <- extract(fit_1)
  X1data_list <- list(c(list_of_draws$X1_new[100,]), c(list_of_draws$X1_new[200,]), c(list_of_draws$X1_new[300,]),
                      c(list_of_draws$X1_new[400,]), c(list_of_draws$X1_new[500,]))
  X2data_list <- list(c(list_of_draws$X2_star_new[100,]), c(list_of_draws$X2_star_new[200,]),
                      c(list_of_draws$X2_star_new[300,]),c(list_of_draws$X2_star_new[400,]),
                      c(list_of_draws$X2_star_new[500,]))
  mylist<-list()
  for(i in 1:5){
    X_1_result <- X1data_list[[i]]
    X_2_result <- X2data_list[[i]]
    mylist[[i]]<- glm(Z_resp~Z_cov+X_1_result+X_2_result+1, family=binomial(link="probit"))
  }

  # True values' result
  trueresult <- as_tibble(coef(summary(glm(Z_resp~Z_cov+X1+X2, family=binomial(link="probit")))), rownames="Parameter")%>%
    rename(TrueEstimate = Estimate, TrueSD = `Std. Error` ) %>% select(TrueEstimate, TrueSD)
  trueresult <- transform(trueresult, TrueCILow = TrueEstimate - 1.96*TrueSD)
  trueresult <- transform(trueresult, TrueCIHigh = TrueEstimate + 1.96*TrueSD)

  # Synthetic datasets' result
  sampleresult <- as_tibble(summary(MIcombine(mylist)),rownames="Parameter") %>% rename(SynEstimate = results, SynSD= se,
                                                                                        SynLower = `(lower`, SynUpper = `upper)`)
  sampleresult <- transform(sampleresult, SynCILow = SynEstimate - 1.96*SynSD)
  sampleresult <- transform(sampleresult, SynCIHigh = SynEstimate + 1.96*SynSD)

  finalresult <- bind_cols(sampleresult, trueresult)

  # Calculate the standard difference
  finalresult <- transform(finalresult, standardDiff = abs(TrueEstimate - SynEstimate)/TrueSD)

  # Calculate data utility
  data_utility <- select(finalresult, Parameter, SynCILow, SynCIHigh, TrueCILow, TrueCIHigh)

  data_utility <- data_utility %>% mutate(DU = (min(TrueCIHigh, SynCIHigh) - max(TrueCILow, SynCILow))/(TrueCIHigh-TrueCILow) +
                          (min(TrueCIHigh, SynCIHigh) - max(TrueCILow, SynCILow))/(SynCIHigh-SynCILow))
  finalresult <- finalresult %>% mutate(DU= data_utility$DU)

  match_rate <- match_rate_fn(X1, X2, X1data_list, X2data_list)
  finalresult <- finalresult %>% mutate(MR = match_rate)
  # Generate final result by adding iterations
  finalresult <- finalresult %>% mutate(Iter=iteration)
}
```

Figure 18: RCode Part2

```r
#Match Rate function
match_rate_fn <- function(X1, X2, X1data_list, X2data_list){
  match_array <- array(rep(0, 500*500*5), dim=c(500, 500, 5))
  d <- 0.1
  # initalize the true range array X[1][1] represents the lower bound for X1's 1st record,
  # X[1][4] represents the upper bound for X2's 1st record
  true_array <- array(rep(0, 500*4), dim=c(500, 4))
  X1_array <- array(X1, dim = c(500,1))
  X2_array <- array(X2, dim = c(500,1))
  for (i in 1:500){
    true_array[i,1] = X1_array[i] - d
    true_array[i,2] = X1_array[i] + d
    true_array[i,3] = X2_array[i] - d
    true_array[i,4] = X2_array[i] + d
  }

  for (i in 1:5){
    X1_syn = X1data_list[[i]]
    X2_syn = X2data_list[[i]]
    for (j in 1:500){
      X1_j <- X1_syn[j]
      X2_j <- X2_syn[j]
      count <- 0
      for (k in 1:500){
        if(true_array[k,1] <= X1_j  && X1_j<= true_array[k,2] && true_array[k,3] <= X2_j && X2_j<= true_array[k, 4]){
          match_array[j, k, i] <- 1
        }
      }
    }
  }

  for (i in 1:5){
    for (j in 1:500){
      sum <- sum(match_array[j,,i])
      for (k in 1:500){
        if(match_array[j,k,i] ==1){
          match_array[j,k,i] <-1/sum
        }
      }
    }
  }
  twod_matching <- array(rep(0,500*500), dim=c(500, 500))
  for (j in 1:500){
    for(k in 1:500){
      twod_matching[j,k] = mean(match_array[j,k,])
    }
  }
```

Figure 19: RCode Part3

```r
  result <- rep(0, 500)
  match_rate <-0
  for (i in 1:500){
    max_prob <- max(twod_matching[i,])
    index_len <- length(which(twod_matching[i,]== max(twod_matching[i,])))
    result[i] <- max_prob*index_len
    if(result[i] == 1){
      match_rate <- match_rate +1
    }
  }
  match_rate <- match_rate/500
}
```

Figure 20: RCode Part4

H R code for analyzing result

```r
# Analyze the result
# Part 1 Data Utility- Overlap
results %>% group_by(Parameter)%>%
  summarise(Avg_DU = mean(DU), Max_DU = max(DU), Min_DU = min(DU))
ggplot(results, aes(x=Parameter, y=DU)) + stat_boxplot(geom = 'errorbar', width = 0.25) +
  geom_boxplot(fill ='lightpink')

# Part 2 Data Utility- Standard Difference
results %>% group_by(Parameter) %>%
  summarise(Avg_SD = mean(standardDiff), Max_SD=max(standardDiff), Min_SD = min(standardDiff))
ggplot(results, aes(x=Parameter, y=standardDiff)) + stat_boxplot(geom = 'errorbar', width = 0.25) +
  geom_boxplot(fill ='lightpink')
```

Figure 21: RCode Part3

```r
# Part 3 Analyzing the Parameters
X1_res <- results %>% filter(Parameter == "X_1_result")
X2_res <- results %>% filter(Parameter == "X_2_result")
interc <- results %>% filter(Parameter == "(Intercept)")
zcov_res <- results %>% filter(Parameter == "Z_cov")

p1 <- ggplot(X1_res, aes(y = SynEstimate)) + stat_boxplot(geom = 'errorbar', width = 0.25) + geom_boxplot(fill ='lightpink')
p2 <- ggplot(X1_res, aes(y = TrueEstimate)) + stat_boxplot(geom = 'errorbar', width = 0.25) + geom_boxplot(fill ='lightpink')
grid.arrange(p1, p2, nrow=1)

p3 <- ggplot(X2_res, aes(y = SynEstimate)) + stat_boxplot(geom = 'errorbar', width = 0.25) + geom_boxplot(fill ='lightpink')
p4 <- ggplot(X2_res, aes(y = TrueEstimate)) + stat_boxplot(geom = 'errorbar', width = 0.25) + geom_boxplot(fill ='lightpink')
grid.arrange(p3, p4, nrow=1)

p5 <- ggplot(interc, aes(y = SynEstimate)) + stat_boxplot(geom = 'errorbar', width = 0.25) + geom_boxplot(fill ='lightpink')
p6 <- ggplot(interc, aes(y = TrueEstimate)) + stat_boxplot(geom = 'errorbar', width = 0.25) + geom_boxplot(fill ='lightpink')
grid.arrange(p5, p6, nrow=1)

p7 <- ggplot(zcov_res, aes(y = SynEstimate)) + stat_boxplot(geom = 'errorbar', width = 0.25) + geom_boxplot(fill ='lightpink')
p8 <- ggplot(zcov_res, aes(y = TrueEstimate)) + stat_boxplot(geom = 'errorbar', width = 0.25) + geom_boxplot(fill ='lightpink')
grid.arrange(p7, p8, nrow=1)

p9 <- ggplot(X1_res, aes(x="SynX1", y=SynEstimate)) + geom_point()
p10 <- ggplot(X1_res, aes(x="TrueX1", y=TrueEstimate)) + geom_point()
grid.arrange(p9, p10, nrow=1)

p11 <- ggplot(X2_res, aes(x="SynX2", y=SynEstimate)) + geom_point()
p12 <- ggplot(X2_res, aes(x="TrueX2", y=TrueEstimate)) + geom_point()
grid.arrange(p11, p12, nrow=1)

p12 <- ggplot(interc, aes(x="SynIntercept", y=SynEstimate)) + geom_point()
p13 <- ggplot(interc, aes(x="Trueintercept", y=TrueEstimate)) + geom_point()
grid.arrange(p12, p13, nrow=1)

p14 <- ggplot(zcov_res, aes(x="SynZcov", y=SynEstimate)) + geom_point()
p15 <- ggplot(zcov_res, aes(x="TrueZcov", y=TrueEstimate)) + geom_point()
grid.arrange(p14, p15, nrow=1)

# Part 4 Analyzing the True Match Rate
subresults <- results %>% filter(Parameter == 'Z_cov')
ggplot(subresults, aes(y=MR)) + stat_boxplot(geom = 'errorbar', width = 0.25) +
  geom_boxplot(fill ='lightpink')
subresults %>%
  summarise(Avg_MR = mean(MR), Max_DU = max(MR), Min_DU = min(MR))
```

Figure 22: RCode Part4