Assignment #03

Yitong Wu

February 18, 2020

(a) Use your own synthesis model to synthesize m = 1 synthetic dataset for the CE sample.

```
library(readr)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(ggplot2)
library(fastDummies)
knitr::opts_chunk$set(echo = TRUE)
def.chunk.hook  <- knitr::knit_hooks$get("chunk")
knitr::knit_hooks$set(chunk = function(x, options) {
  x <- def.chunk.hook(x, options)
  ifelse(options$size != "normalsize", paste0("\\", options$size,"\n\n", x, "\n\n \\normalsize"), x)
})
require(runjags)
```

```
## Loading required package: runjags
```

```
require(coda)
```

```
## Loading required package: coda
```

```
data <- read.csv("CEdata.csv")
data$Rural = fastDummies::dummy_cols(data$UrbanRural)[,names(fastDummies::dummy_cols(data$UrbanRural)) =
data$Race_Black = fastDummies::dummy_cols(data$Race)[,names(fastDummies::dummy_cols(data$Race)) == ".da
data$Race_NA = fastDummies::dummy_cols(data$Race)[,names(fastDummies::dummy_cols(data$Race)) == ".data_
data$Race_Asian = fastDummies::dummy_cols(data$Race)[,names(fastDummies::dummy_cols(data$Race)) == ".da
data$Race_PI = fastDummies::dummy_cols(data$Race)[,names(fastDummies::dummy_cols(data$Race)) == ".data_
data$Race_M = fastDummies::dummy_cols(data$Race)[,names(fastDummies::dummy_cols(data$Race)) == ".data_6
data$logInc <- log(data$Income)
```

```
modelString <-"
model {
## sampling
for (i in 1:N){
y[i] ~ dnorm(beta0 + beta1*x_rural[i] +
beta2*x_race_B[i] + beta3*x_race_N[i] +
beta4*x_race_A[i] + beta5*x_race_P[i] + beta6*x_race_M[i], invsigma2)
}
## priors
beta0 ~ dnorm(mu0, g0)
beta1 ~ dnorm(mu1, g1)
beta2 ~ dnorm(mu2, g2)
beta3 ~ dnorm(mu3, g3)
beta4 ~ dnorm(mu4, g4)
beta5 ~ dnorm(mu5, g5)
beta6 ~ dnorm(mu6, g6)
invsigma2 ~ dgamma(a, b)
sigma <- sqrt(pow(invsigma2, -1))
}
"
```

```
y = as.vector(data$logInc)
x_rural = as.vector(data$Rural)
x_race_B = as.vector(data$Race_Black)
x_race_N = as.vector(data$Race_NA)
x_race_A = as.vector(data$Race_Asian)
x_race_P = as.vector(data$Race_PI)
x_race_M = as.vector(data$Race_M)
N = length(y)
```

```
the_data <- list("y" = y,
                 "x_rural" = x_rural, "x_race_B" = x_race_B,
                 "x_race_N" = x_race_N, "x_race_A" = x_race_A,
                 "x_race_P" = x_race_P, "x_race_M" = x_race_M,
                 "N" = N,
                 "mu0" = 0, "g0" = 1, "mu1" = 0, "g1" = 1,
                 "mu2" = 0, "g2" = 1, "mu3" = 0, "g3" = 1,
                 "mu4" = 0, "g4" = 1, "mu5" = 0, "g5" = 1,
                 "mu6" = 0, "g6" = 1,
                 "a" = 1, "b" = 1)
```

```
initsfunction <- function(chain){
  .RNG.seed <- c(1,2)[chain]
  .RNG.name <- c("base::Super-Duper",
                 "base::Wichmann-Hill")[chain]
  return(list(.RNG.seed=.RNG.seed,
              .RNG.name=.RNG.name))
}
```

```
posterior_MLR <- run.jags(modelString,
                      n.chains = 1,
                      data = the_data,
                      monitor = c("beta0", "beta1", "beta2",
                                  "beta3", "beta4", "beta5",
                                  "beta6", "sigma"),
                      adapt = 1000,
                      burnin = 5000,
                      sample = 5000,
                      thin = 1,
                      inits = initsfunction)
```

```
## Warning: Convergence cannot be assessed with only 1 chain
```

```
post <- as.mcmc(posterior_MLR)
```

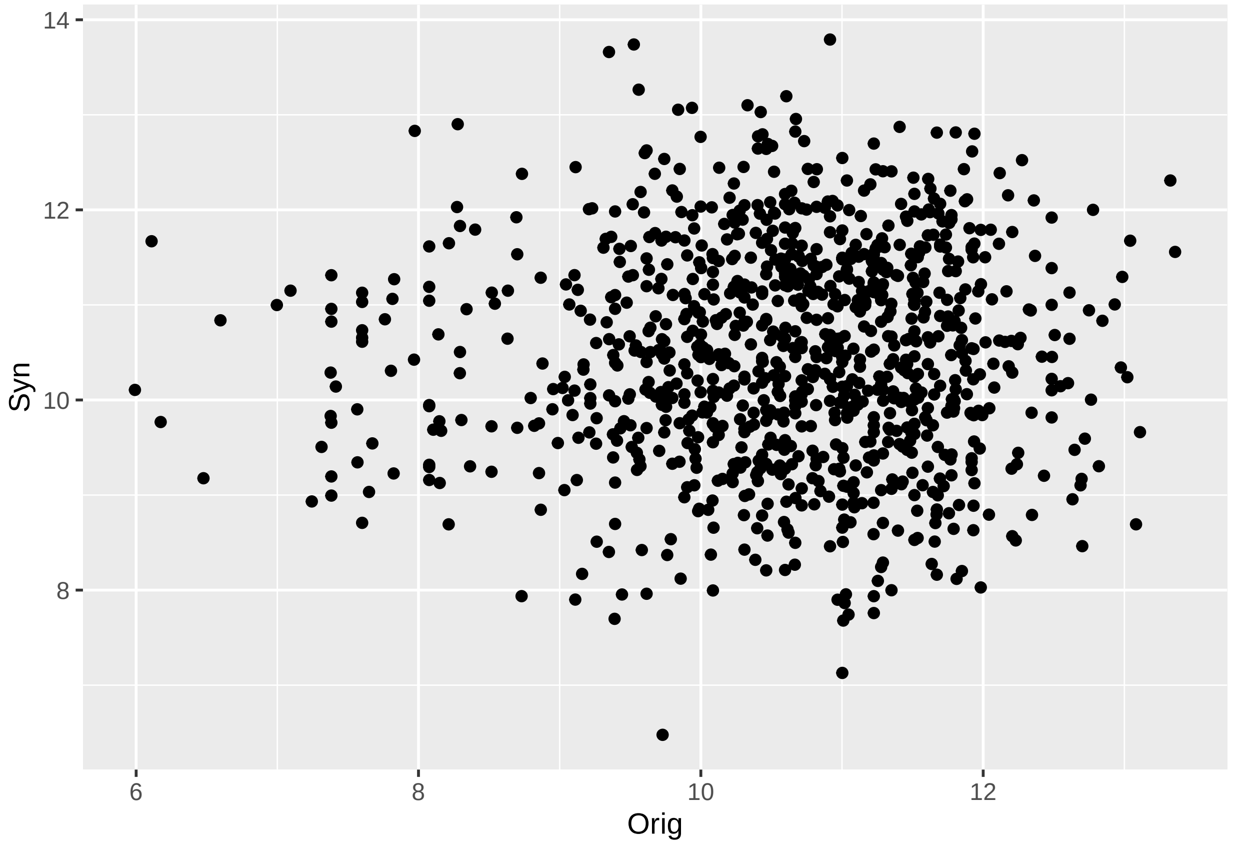
```
synthesize <- function(X_rural, X_RB, X_RN, X_RA, X_RP, X_RM, index, n){
  mean_Y <- post[index, "beta0"] +  X_rural * post[index, "beta1"] + X_RB * post[index, "beta2"] + X_RN * post[index, "beta3"] +
  synthetic_Y <- rnorm(n, mean_Y, post[index, "sigma"])
  data.frame(synthetic_Y, X_rural, X_RB, X_RN, X_RA, X_RP, X_RM)
}
```

```
n <- dim(data)[1]
Syndata <- synthesize(data$Rural, data$Race_Black, data$Race_NA, data$Race_Asian, data$Race_PI, data$Ra
names(Syndata) <- c("logInc", "Rural", "RB", "RN", "RA", "RP", "RM")
```

(b) Make a scatter plot of the synthesized log(Income) against the original log(Income), and see what you find.

```
Synthesis <- cbind(data$logInc, Syndata$logInc)
Synthesis <- data.frame(Synthesis)
names(Synthesis) <- c("Orig", "Syn")
```

```
ggplot(Synthesis, aes(x=Orig, y=Syn)) + geom_point()
```



(b) Compare the mean and median of log(Income), in the original dataset and the confidential dataset. Are they close to each other?

```
mean(Synthesis$Orig)
```

```
## [1] 10.59507
```

```
median(Synthesis$Orig)
```

```
## [1] 10.70574
```

```
mean(Synthesis$Syn)
```

```
## [1] 10.48264
```

```
median(Synthesis$Syn)
```

```
## [1] 10.47355
```

(c) Compare the point estimate of the regression coefficients of log(Income) on log(Expenditure), in the original dataset and the confidential dataset. Are they close to each other?

```
Synthesis <- cbind(Synthesis, log(data$Expenditure))
names(Synthesis) <- c("Orig", "Syn", "Ex")
model_orig <- lm(Orig ~ Ex, data=Synthesis)
```

```
model_syn <- lm(Syn ~ Ex, data=Synthesis)
summary(model_orig)
```

```
##
## Call:
## lm(formula = Orig ~ Ex, data = Synthesis)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.9086 -0.4371  0.1069  0.6110  2.8823
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.11123    0.30801   13.35   <2e-16 ***
## Ex           0.73814    0.03489   21.15   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.958 on 992 degrees of freedom
## Multiple R-squared:  0.3109, Adjusted R-squared:  0.3102
## F-statistic: 447.5 on 1 and 992 DF,  p-value: < 2.2e-16
```

```
summary(model_syn)
```

```
##
## Call:
## lm(formula = Syn ~ Ex, data = Synthesis)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.0097 -0.7894 -0.0123  0.8537  3.3229
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 10.79697    0.36640  29.468   <2e-16 ***
## Ex          -0.03578    0.04151  -0.862    0.389
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.14 on 992 degrees of freedom
## Multiple R-squared:  0.0007486,  Adjusted R-squared:  -0.0002587
## F-statistic: 0.7432 on 1 and 992 DF,  p-value: 0.3888
```

(d) Evaluate the global utility measures of your synthesized log(Income) from your Bayesian synthesis model for the CE sample.

```
Oridata <- cbind(data$logInc, data$Rural,data$Race_Black,data$Race_NA,data$Race_Asian,data$Race_PI,data
Oridata <- data.frame(Oridata)
names(Oridata) <- c("logInc", "Rural", "RB", "RN", "RA", "RP", "RM")
merged_data <- rbind(Oridata, Syndata)
merged_data$T <- c(rep(0,994),rep(1,994))
```

I. Propensity Score Measure

```
log_reg <- glm(T ~ logInc + as.factor(Rural) + as.factor(RB) + as.factor(RN) + as.factor(RA) + as.facto
summary(log_reg)
```

```
##
## Call:
## glm(formula = T ~ logInc + as.factor(Rural) + as.factor(RB) +
##     as.factor(RN) + as.factor(RA) + as.factor(RP) + as.factor(RM),
##     family = "binomial", data = merged_data)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.34866  -1.17247  -0.02424   1.17506   1.29890
##
## Coefficients:
##                    Estimate Std. Error z value Pr(>|z|)
## (Intercept)         0.93377    0.42512   2.196   0.0281 *
## logInc             -0.08809    0.03981  -2.213   0.0269 *
## as.factor(Rural)1 -0.01195    0.20501  -0.058   0.9535
## as.factor(RB)1    -0.03967    0.14572  -0.272   0.7854
## as.factor(RN)1    -0.09123    0.54092  -0.169   0.8661
## as.factor(RA)1     0.02868    0.23266   0.123   0.9019
## as.factor(RP)1    -0.02488    0.58003  -0.043   0.9658
## as.factor(RM)1    -0.04222    0.34759  -0.121   0.9033
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2756  on 1987  degrees of freedom
## Residual deviance: 2751  on 1980  degrees of freedom
## AIC: 2767
##
## Number of Fisher Scoring iterations: 3
```

```
pred <- predict(log_reg, data=merged_data)
probs <- exp(pred)/(1+exp(pred))
Up <- 1/(2*n)*sum((probs-1/2)^2)
Up
```

```
## [1] 0.0006173112
```

II. Cluster Analysis Measure

```
clusters <- hclust(dist(merged_data[,1:2]), method='average')
```

```
ClusterAnalysis <- function(G){
  clusterCut <- cutree(clusters, G)
  cluster_T <- as.data.frame(cbind(clusterCut, merged_data$T))
  names(cluster_T) <- c("cluster", "T")
  table(cluster_T)
  n_gS <- table(cluster_T)[,1]
  n_g <- rowSums(table(cluster_T))
  w_g <- n_g/(2*n)
  Uc <- (1/G)*sum(w_g*(n_gS/n_g-1/2)^2)
  Uc
}
```

```
ClusterAnalysis(3)
```

```
## [1] 0.0001895369
```

```
ClusterAnalysis(5)
```

```
## [1] 0.0006810602
```

```
ClusterAnalysis(10)
```

```
## [1] 0.0004248599
```

```
ClusterAnalysis(15)
```

```
## [1] 0.0003931866
```

```
ClusterAnalysis(20)
```

```
## [1] 0.0003456766
```

III. Empirical CDF Measure

```
ecdf_ori <- ecdf(Oridata[,"logInc"])
ecdf_syn <- ecdf(Syndata[,"logInc"])
percentile_ori <- ecdf_ori(merged_data[,"logInc"])
percentile_syn <- ecdf_syn(merged_data[,"logInc"])
```

```
ecdf_diff <- percentile_ori-percentile_syn
Um <- max(abs(ecdf_diff))
Um
```

```
## [1] 0.1096579
```

```
Ua <- mean(ecdf_diff^2)
Ua
```

```
## [1] 0.003470492
```

Drechsler (2001)

i. Generate m = 20 synthetic datasets given your synthesis model for the CE sample. If you are using set.seed(), make sure that you do not generate the same synthetic data for each m = 20.

```
Syn_sets <- NULL
n <- dim(data)[1]
Syn_sets <- cbind(data$Rural,data$Race_Black, data$Race_NA, data$Race_Asian, data$Race_PI, data$Race_M)
Syn_sets <- data.frame(Syn_sets)
names(Syn_sets) <- c("Rural", "RB", "RN", "RA", "RP", "RM")
synthesize <- function(X_rural, X_RB, X_RN, X_RA, X_RP, X_RM, index, n){
  mean_Y <- post[index, "beta0"] +  X_rural * post[index, "beta1"] + X_RB * post[index, "beta2"] + X_RN
  synthetic_Y <- rnorm(n, mean_Y, post[index, "sigma"])
  data.frame(synthetic_Y)
}
```

```
for (i in 1:20) {
  Syndata <- synthesize(data$Rural, data$Race_Black, data$Race_NA, data$Race_Asian, data$Race_PI, data$
  Syn_sets <- cbind(Syn_sets, Syndata$synthetic_Y)
  names(Syn_sets[,6+i]) <- c(paste("syn",as.character(i)))
}
```

ii. Estimate a few analysis-specific utility measures, e.g. the mean and median of a continuous synthetic variable, the regression analysis coefficients, for each synthetic dataset.

```
Syn_sets <- cbind(Syn_sets, data$logInc, log(data$Expenditure))
names(Syn_sets) <- c("Rural", "RB", "RN", "RA", "RP", "RM","m1","m2","m3","m4","m5","m6","m7","m8",
                     "m9","m10","m11","m12","m13","m14","m15","m16","m17","m18","m19","m20","ori","ex")
mean <- c()
median <- c()
coeff <- c()
pVal <- c()
variance <- c()
for (i in 1:21) {
  Syn_sets$target <- Syn_sets[,i+6]
  mean<- append(mean, mean(Syn_sets$target))
  median <- append(median, median(Syn_sets$target))
  lr <- lm(target ~ ex, data=Syn_sets)
  coe <- summary(lr)$coefficients[2, 1]
  coeff <- append(coeff, coe)
  pvalue <- summary(lr)$coefficients[2, 4]
  pVal <- append(pVal, pvalue)
  variance <- append(variance, var(Syn_sets$target))
}
Analysis <- cbind(mean, median, coeff, pVal, variance)
Analysis <- data.frame(Analysis)
names(Analysis) <- c("mean","median","coeff","pVal", "variance")
Analysis
```

```
##        mean   median        coeff         pVal variance
## 1  10.51355 10.53726 -0.003141683 9.420850e-01 1.407583
## 2  10.59510 10.57208  0.107992662 8.583629e-03 1.275252
## 3  10.59988 10.62107  0.066399481 1.221123e-01 1.390165
## 4  10.51870 10.48205  0.034256341 4.334615e-01 1.440107
## 5  10.61464 10.59914  0.092065792 2.593074e-02 1.289319
## 6  10.63823 10.62265  0.097133955 2.443097e-02 1.406019
## 7  10.55714 10.52877 -0.006796671 8.750428e-01 1.406035
## 8  10.56683 10.54801 -0.025857912 5.529836e-01 1.430004
## 9  10.66747 10.70796  0.061097291 1.450718e-01 1.324671
## 10 10.52346 10.52383  0.050915946 2.319512e-01 1.366617
## 11 10.55542 10.53428  0.104186609 1.358523e-02 1.345442
## 12 10.59726 10.58756  0.027299539 5.352915e-01 1.459815
## 13 10.51637 10.48682  0.079748258 6.395211e-02 1.397287
## 14 10.60760 10.61100 -0.001927762 9.631433e-01 1.309966
## 15 10.54066 10.52085  0.063251254 1.344894e-01 1.345905
## 16 10.58647 10.59106  0.091814505 3.120686e-02 1.370222
## 17 10.51784 10.53160  0.026529956 5.382356e-01 1.398744
## 18 10.60544 10.57778  0.089748411 3.137324e-02 1.311820
## 19 10.56639 10.56163  0.101849275 9.158309e-03 1.153558
## 20 10.54352 10.52183  0.059238110 1.680769e-01 1.391479
## 21 10.59507 10.70574  0.738140359 2.849973e-82 1.330535
```

iii. Use the combining rules in Drechsler 2001 Chapter 6-1 (for fully synthetic data) and / or Drechsler 2001 Chapter 7-1 (for partially synthetic data) and create your final point estimate and confidence interval of the analysis-specific utility measures you calculated in Item ii above.

For mean:

```
qm <- sum(Analysis[1:20,1])/20
bm <- sum((Analysis[1:20,1]-qm)^2)/(20-1)
```

```r
um <- sum(Analysis[1:20,5])/20
```

```r
qm
```

```
## [1] 10.5716
```

```r
bm
```

```
## [1] 0.001939386
```

```r
um
```

```
## [1] 1.361001
```

```r
qm+1.645*bm
```

```
## [1] 10.57479
```

```r
qm-1.645*bm
```

```
## [1] 10.56841
```

The final point estimate is then 10.59, with a 90% confidence interval of [10.588,10.594].

Drechsler, J. and Reiter, J. P. (2009)

i. Calcuate the corresponding interval overlap measure for each of the analysis-specific utility measures you have done in Item 2.ii above.

```r
var_syn <- sum(Analysis[1:20,5])/20
var_ori <- sum(Analysis[21,5])
mean_syn <- sum(Analysis[1:20,1])/20
mean_ori <- sum(Analysis[21,1])
```

```r
mean_syn + 1.645*var_syn
```

```
## [1] 12.81044
```

```r
mean_syn - 1.645*var_syn
```

```
## [1] 8.332753
```

```r
mean_ori + 1.645*var_ori
```

```
## [1] 12.7838
```

```r
mean_ori - 1.645*var_ori
```

```
## [1] 8.406342
```

```r
(12.81-8.37)/(2*(12.78-8.41))+(12.81-8.37)/(2*(12.81-8.37))
```

```
## [1] 1.008009
```