Methods for Utility Evaluation #2

MATH 301 Data Confidentiality

Henrik Olsson

February 25, 2020

```
CEdata<- read.csv("CEdata.csv")
head(CEdata)
```

```
##   UrbanRural Income Race Expenditure
## 1          1  98600    1    5972.167
## 2          1  24360    1    5854.500
## 3          1  80200    1    5506.667
## 4          1 150500    1    8968.891
## 5          1 130000    1   10092.833
## 6          1  32836    1    5520.267
```

Read Drechsler (2001) Chapter 6-1, 7-1 in the References folder, and prepare the following results.

```
CEdata$LogExp <- log(CEdata$Expenditure)
CEdata$LogIncome <- log(CEdata$Income)

## create indicator variable for Rural (2)
CEdata$Rural = fastDummies::dummy_cols(CEdata$UrbanRural)[,names(fastDummies::dummy_cols(CEdata$UrbanRu
== ".data_1"]

## create indicator variables for Black (3), Native American (4),
## Asian (5), Pacific Islander (6), and Multi-race (7)
CEdata$Race_Black = fastDummies::dummy_cols(CEdata$Race)[,names(fastDummies::dummy_cols(CEdata$Race)) =
CEdata$Race_NA = fastDummies::dummy_cols(CEdata$Race)[,names(fastDummies::dummy_cols(CEdata$Race)) == "
CEdata$Race_Asian = fastDummies::dummy_cols(CEdata$Race)[,names(fastDummies::dummy_cols(CEdata$Race)) =
CEdata$Race_PI = fastDummies::dummy_cols(CEdata$Race)[,names(fastDummies::dummy_cols(CEdata$Race)) == "
CEdata$Race_M = fastDummies::dummy_cols(CEdata$Race)[,names(fastDummies::dummy_cols(CEdata$Race)) == ".
```

```
## JAGS script
modelString <-"
model {
## sampling
for (i in 1:N){
y[i] ~ dnorm(beta0 + beta1*x_income[i] + beta2*x_rural[i] +
beta3*x_race_B[i] + beta4*x_race_N[i] +
beta5*x_race_A[i] + beta6*x_race_P[i] +
beta7*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)
beta7 ~ dnorm(mu7, g7)
invsigma2 ~ dgamma(a, b)
sigma <- sqrt(pow(invsigma2, -1))
}"
```

```r
y = as.vector(CEdata$LogExp)
x_income = as.vector(CEdata$LogIncome)
x_rural = as.vector(CEdata$Rural)
x_race_B = as.vector(CEdata$Race_Black)
x_race_N = as.vector(CEdata$Race_NA)
x_race_A = as.vector(CEdata$Race_Asian)
x_race_P = as.vector(CEdata$Race_PI)
x_race_M = as.vector(CEdata$Race_M)
N = length(y) # Compute the number of observations


## Pass the data and hyperparameter values to JAGS
the_data <- list("y" = y, "x_income" = x_income,
"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, "mu7" = 0, "g7" = 1,
"a" = 1, "b" = 1)
```

```r
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))
}
```

```r
## Run the JAGS code for this model:
posterior_MLR <- run.jags(modelString,
n.chains = 1,
data = the_data,
monitor = c("beta0", "beta1", "beta2",
"beta3", "beta4", "beta5",
"beta6", "beta7", "sigma"),
adapt = 1000,
burnin = 5000,
sample = 5000,
thin = 1,
inits = initsfunction)
```

```
## Loading required namespace: rjags
```

```
## Compiling rjags model...
## Calling the simulation using the rjags method...
## Note: the model did not require adaptation
```

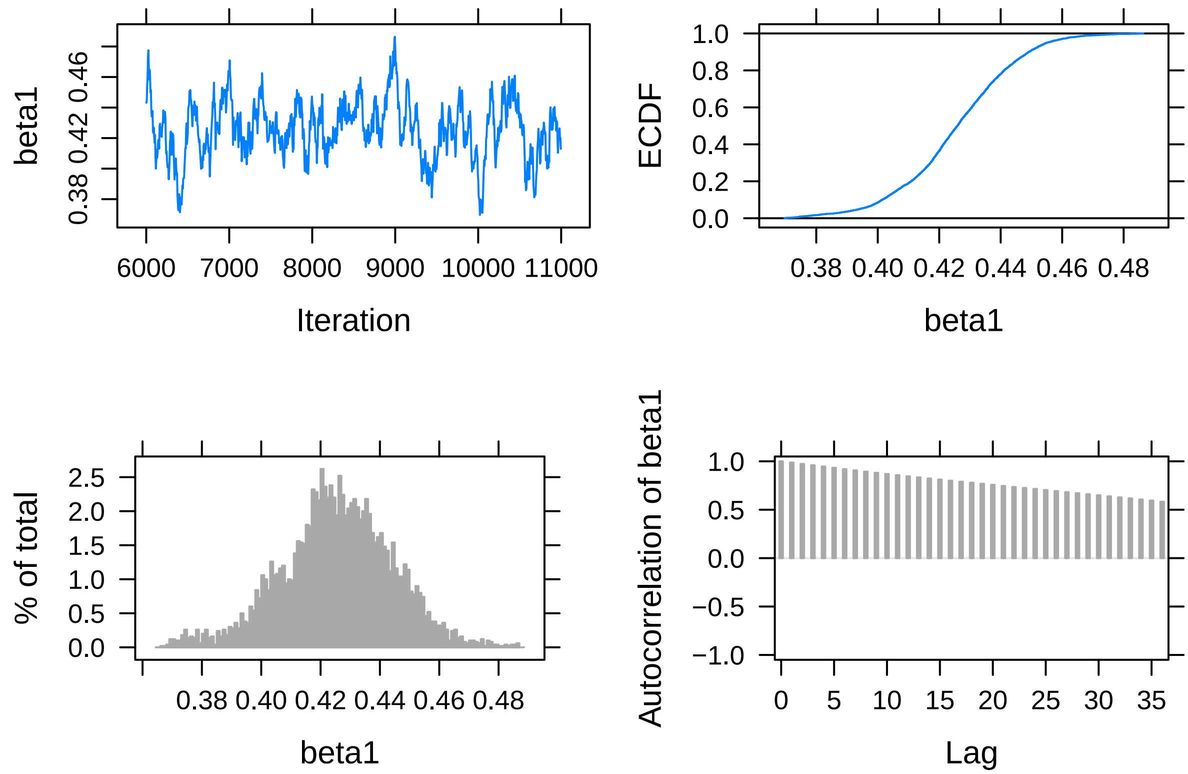
```
## Burning in the model for 5000 iterations...
## Running the model for 5000 iterations...
## Simulation complete
## Calculating summary statistics...

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

## Finished running the simulation
```

```r
## JAGS output
summary(posterior_MLR)
```

```
##           Lower95      Median     Upper95        Mean         SD Mode
## beta0  3.54580441  4.00080795  4.47511996  4.02342538 0.22801936   NA
## beta1  0.38724270  0.42589245  0.46270522  0.42560569 0.01888028   NA
## beta2  0.07556203  0.27567761  0.49356550  0.27739517 0.10683751   NA
## beta3 -0.33286463 -0.19636237 -0.05011323 -0.19589145 0.07350734   NA
## beta4 -0.49856355  0.01200176  0.52491192  0.01108006 0.26200777   NA
## beta5 -0.07838912  0.15751196  0.38365788  0.15652442 0.11925047   NA
## beta6 -0.47113608  0.08692820  0.60972710  0.08885212 0.28043794   NA
## beta7 -0.31549244  0.04217888  0.37819450  0.04125956 0.17844949   NA
## sigma  0.69161484  0.72115468  0.75539675  0.72161423 0.01621386   NA
##              MCerr MC%ofSD SSeff        AC.10 psrf
## beta0 0.0438707646    19.2    27  0.894522661   NA
## beta1 0.0032115062    17.0    35  0.868794694   NA
## beta2 0.0093583674     8.8   130  0.595800945   NA
## beta3 0.0012361528     1.7  3536 -0.000743349   NA
## beta4 0.0037053495     1.4  5000 -0.008922402   NA
## beta5 0.0017885674     1.5  4445 -0.018979087   NA
## beta6 0.0039659914     1.4  5000  0.007764935   NA
## beta7 0.0026293845     1.5  4606  0.017692787   NA
## sigma 0.0002292986     1.4  5000  0.010989815   NA
```

```r
plot(posterior_MLR, vars = "beta1")
```

```
## Generating plots...
```




```r
## Saving posterior parameter draws
post <- as.mcmc(posterior_MLR)

## Generating one set of sythetic data
synthesize <- function(X, index, n){
  mean_Y <- post[index, "beta0"] +  X$x_income * post[index, "beta1"] +  X$x_rural * post[index, "beta2
  synthetic_Y <- rnorm(n, mean_Y, post[index, "sigma"])
  data.frame(X$x_income, synthetic_Y)
}
```

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.

```r
set.seed(123)
m <- 20
n <- dim(CEdata)[1]
synthetic_m <- vector("list",m)
new <- data.frame(x_income, x_rural, x_race_B, x_race_N, x_race_A, x_race_P, x_race_M)
for (l in 1:m){
  synthetic_one <- synthesize(new, 4980+l, n)
  names(synthetic_one) <- c("OrigLogIncome", "SynLogIncome")
  synthetic_m[[l]] <- synthetic_one
}
```

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.

```r
## Estimates the mean, median, mode, variance, and range of synthetic log Income, as well as regression
mean <- c()
```

```r
median <- c()
mode <- c()
variance <- c()
range <- c()

for (l in 1:m){
  mean[l] = mean(synthetic_m[[l]]$SynLogIncome)
  median[l] = median(synthetic_m[[l]]$SynLogIncome)
  mode[l] = mode(synthetic_m[[l]]$SynLogIncome)
  variance[l] = var(synthetic_m[[l]]$SynLogIncome)
  range[l] = range(synthetic_m[[l]]$SynLogIncome)
  print(lm(CEdata$LogExp ~ synthetic_m[[l]]$SynLogIncome))
}
```

```
## Warning in range[l] <- range(synthetic_m[[l]]$SynLogIncome): number of
## items to replace is not a multiple of replacement length
```

```
##
## Call:
## lm(formula = CEdata$LogExp ~ synthetic_m[[l]]$SynLogIncome)
##
## Coefficients:
##                   (Intercept)  synthetic_m[[l]]$SynLogIncome
##                        6.0914                         0.3067
```

```
## Warning in range[l] <- range(synthetic_m[[l]]$SynLogIncome): number of
## items to replace is not a multiple of replacement length
```

```
##
## Call:
## lm(formula = CEdata$LogExp ~ synthetic_m[[l]]$SynLogIncome)
##
## Coefficients:
##                   (Intercept)  synthetic_m[[l]]$SynLogIncome
##                        5.8886                         0.3283
```

```
## Warning in range[l] <- range(synthetic_m[[l]]$SynLogIncome): number of
## items to replace is not a multiple of replacement length
```

```
##
## Call:
## lm(formula = CEdata$LogExp ~ synthetic_m[[l]]$SynLogIncome)
##
## Coefficients:
##                   (Intercept)  synthetic_m[[l]]$SynLogIncome
##                        5.9803                         0.3197
```

```
## Warning in range[l] <- range(synthetic_m[[l]]$SynLogIncome): number of
## items to replace is not a multiple of replacement length
```

```
##
## Call:
## lm(formula = CEdata$LogExp ~ synthetic_m[[l]]$SynLogIncome)
##
## Coefficients:
##                   (Intercept)  synthetic_m[[l]]$SynLogIncome
##                        5.9853                         0.3181
```

```
## Warning in range[l] <- range(synthetic_m[[l]]$SynLogIncome): number of
## items to replace is not a multiple of replacement length

##
## Call:
## lm(formula = CEdata$LogExp ~ synthetic_m[[l]]$SynLogIncome)
##
## Coefficients:
##                  (Intercept)  synthetic_m[[l]]$SynLogIncome
##                       5.8097                         0.3397

## Warning in range[l] <- range(synthetic_m[[l]]$SynLogIncome): number of
## items to replace is not a multiple of replacement length

##
## Call:
## lm(formula = CEdata$LogExp ~ synthetic_m[[l]]$SynLogIncome)
##
## Coefficients:
##                  (Intercept)  synthetic_m[[l]]$SynLogIncome
##                       5.7294                         0.3476

## Warning in range[l] <- range(synthetic_m[[l]]$SynLogIncome): number of
## items to replace is not a multiple of replacement length

##
## Call:
## lm(formula = CEdata$LogExp ~ synthetic_m[[l]]$SynLogIncome)
##
## Coefficients:
##                  (Intercept)  synthetic_m[[l]]$SynLogIncome
##                       5.7798                         0.3426

## Warning in range[l] <- range(synthetic_m[[l]]$SynLogIncome): number of
## items to replace is not a multiple of replacement length

##
## Call:
## lm(formula = CEdata$LogExp ~ synthetic_m[[l]]$SynLogIncome)
##
## Coefficients:
##                  (Intercept)  synthetic_m[[l]]$SynLogIncome
##                       5.6578                         0.3572

## Warning in range[l] <- range(synthetic_m[[l]]$SynLogIncome): number of
## items to replace is not a multiple of replacement length

##
## Call:
## lm(formula = CEdata$LogExp ~ synthetic_m[[l]]$SynLogIncome)
##
## Coefficients:
##                  (Intercept)  synthetic_m[[l]]$SynLogIncome
##                       5.8630                         0.3322

## Warning in range[l] <- range(synthetic_m[[l]]$SynLogIncome): number of
## items to replace is not a multiple of replacement length

##
```

```
## Call:
## lm(formula = CEdata$LogExp ~ synthetic_m[[l]]$SynLogIncome)
##
## Coefficients:
##                   (Intercept)  synthetic_m[[l]]$SynLogIncome
##                        6.2432                         0.2902

## Warning in range[l] <- range(synthetic_m[[l]]$SynLogIncome): number of
## items to replace is not a multiple of replacement length

##
## Call:
## lm(formula = CEdata$LogExp ~ synthetic_m[[l]]$SynLogIncome)
##
## Coefficients:
##                   (Intercept)  synthetic_m[[l]]$SynLogIncome
##                        5.6420                         0.3571

## Warning in range[l] <- range(synthetic_m[[l]]$SynLogIncome): number of
## items to replace is not a multiple of replacement length

##
## Call:
## lm(formula = CEdata$LogExp ~ synthetic_m[[l]]$SynLogIncome)
##
## Coefficients:
##                   (Intercept)  synthetic_m[[l]]$SynLogIncome
##                        6.1351                         0.3012

## Warning in range[l] <- range(synthetic_m[[l]]$SynLogIncome): number of
## items to replace is not a multiple of replacement length

##
## Call:
## lm(formula = CEdata$LogExp ~ synthetic_m[[l]]$SynLogIncome)
##
## Coefficients:
##                   (Intercept)  synthetic_m[[l]]$SynLogIncome
##                        5.7580                         0.3443

## Warning in range[l] <- range(synthetic_m[[l]]$SynLogIncome): number of
## items to replace is not a multiple of replacement length

##
## Call:
## lm(formula = CEdata$LogExp ~ synthetic_m[[l]]$SynLogIncome)
##
## Coefficients:
##                   (Intercept)  synthetic_m[[l]]$SynLogIncome
##                        5.8873                         0.3292

## Warning in range[l] <- range(synthetic_m[[l]]$SynLogIncome): number of
## items to replace is not a multiple of replacement length

##
## Call:
## lm(formula = CEdata$LogExp ~ synthetic_m[[l]]$SynLogIncome)
##
## Coefficients:
```

```
##                   (Intercept)  synthetic_m[[l]]$SynLogIncome
##                        5.8961                         0.3299
## Warning in range[l] <- range(synthetic_m[[l]]$SynLogIncome): number of
## items to replace is not a multiple of replacement length
##
## Call:
## lm(formula = CEdata$LogExp ~ synthetic_m[[l]]$SynLogIncome)
##
## Coefficients:
##                   (Intercept)  synthetic_m[[l]]$SynLogIncome
##                        5.9462                         0.3231
## Warning in range[l] <- range(synthetic_m[[l]]$SynLogIncome): number of
## items to replace is not a multiple of replacement length
##
## Call:
## lm(formula = CEdata$LogExp ~ synthetic_m[[l]]$SynLogIncome)
##
## Coefficients:
##                   (Intercept)  synthetic_m[[l]]$SynLogIncome
##                         5.912                          0.329
## Warning in range[l] <- range(synthetic_m[[l]]$SynLogIncome): number of
## items to replace is not a multiple of replacement length
##
## Call:
## lm(formula = CEdata$LogExp ~ synthetic_m[[l]]$SynLogIncome)
##
## Coefficients:
##                   (Intercept)  synthetic_m[[l]]$SynLogIncome
##                        6.0511                         0.3134
## Warning in range[l] <- range(synthetic_m[[l]]$SynLogIncome): number of
## items to replace is not a multiple of replacement length
##
## Call:
## lm(formula = CEdata$LogExp ~ synthetic_m[[l]]$SynLogIncome)
##
## Coefficients:
##                   (Intercept)  synthetic_m[[l]]$SynLogIncome
##                        5.8061                         0.3418
## Warning in range[l] <- range(synthetic_m[[l]]$SynLogIncome): number of
## items to replace is not a multiple of replacement length
##
## Call:
## lm(formula = CEdata$LogExp ~ synthetic_m[[l]]$SynLogIncome)
##
## Coefficients:
##                   (Intercept)  synthetic_m[[l]]$SynLogIncome
##                        6.1301                         0.3027
```

```r
synthetic_data <- synthetic_m[[1]]
```

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.

```r
## Univariate estimands
## We need the following for inferences for scalar Q
qbar_m = sum(mean)/m
b_m = sum(mean - qbar_m)^2/(m-1)
ubar_m = sum(variance)/m

## Use qbar_m to estimate Q and the following to estimate the variance of qbar_m
T_m = (1 + (m^-1))*b_m-ubar_m
```

Fully Synthetic Data

```r
## Synthesized log income mean
u_mean = var(mean)
qbar_mean = sum(mean)/m
b_mean = sum(mean - qbar_mean)^2/(m-1)
ubar_mean = sum(u_mean)/m
T_mean = (1 + (m^-1))*b_mean - ubar_mean

qbar_mean
```

```
## [1] 8.771576
```

```r
T_mean
```

```
## [1] -4.030788e-05
```

```r
## Synthesized log income median
u_median = var(median)
qbar_median = sum(median)/m
b_median = sum(median - qbar_median)^2/(m-1)
ubar_median = sum(u_median)/m
T_median = (1 + (m^-1))*b_median - ubar_median

qbar_median
```

```
## [1] 8.788588
```

```r
T_median
```

```
## [1] -7.619068e-05
```

```r
## Synthesized log income variance
u_var = var(variance)
qbar_var = sum(variance)/m
b_var = sum(variance - qbar_var)^2/(m-1)
ubar_var = sum(u_var)/m
T_var = (1 + (m^-1))*b_var - ubar_var

qbar_var
```

```
## [1] 0.7826023
```

```
T_var
```

```
## [1] -4.130823e-05
```

Partially Synthetic Data

```
## Use qbar_m to estimate Q and the following to estimate the variance of qbar_m
T_p = (b_m/m) + ubar_m
```

```
## Synthesized log income mean
u_mean = var(mean)
qbar_mean = sum(mean)/m
b_mean = sum(mean - qbar_mean)^2/(m-1)
ubar_mean = sum(u_mean)/m
T_meanp = (b_mean/m) + ubar_mean

qbar_mean
```

```
## [1] 8.771576
```

```
T_meanp
```

```
## [1] 4.030788e-05
```

```
## Synthesized log income median
u_median = var(median)
qbar_median = sum(median)/m
b_median = sum(median - qbar_median)^2/(m-1)
ubar_median = sum(u_median)/m
T_medianp = (b_median/m) + ubar_median

qbar_median
```

```
## [1] 8.788588
```

```
T_medianp
```

```
## [1] 7.619068e-05
```

```
## Synthesized log income variance
u_var = var(variance)
qbar_var = sum(variance)/m
b_var = sum(variance - qbar_var)^2/(m-1)
ubar_var = sum(u_var)/m
T_varp = (b_var/m) + ubar_var

qbar_var
```

```
## [1] 0.7826023
```

```
T_varp
```

```
## [1] 4.130823e-05
```

I am not completely sure how to replicate the results of Drechsler to create final point estimates and confidence intervals in the partially and fully synthetic data

Interval Overlap Measure

```r
L_s = quantile(synthetic_data$SynLogIncome, 0.025)
U_s = quantile(synthetic_data$SynLogIncome, 0.975)

L_o = quantile(synthetic_data$OrigLogIncome, 0.025)
U_o = quantile(synthetic_data$OrigLogIncome, 0.975)

L_i = max(L_s, L_o)
U_i = min(U_s, U_o)

I = (U_i - L_i) / (2 * (U_o - L_o)) + (U_i - L_i)/ (2 * (U_s - L_s))
I
```

```
##    97.5%
## 0.710088
```

Since the interval overlap measure is close to 1 then we have a relatively high utility.