Lab 3

Isaac Kleisle-Murphy

February 8, 2020

```r
suppressMessages(require(dplyr))
suppressMessages(require(ggplot2))
suppressMessages(require(runjags))
suppressMessages(require(coda))
suppressMessages(require(tidyr))
setwd("~/Documents/Swat_2020/Data_Privacy/Data-Confidentiality/datasets")

CEsample = read.csv("CEsample.csv")%>%
  mutate(logInc = log(TotalIncomeLastYear),
         logExp = log(TotalExpLastQ))
```

1.)

Completed

2.)

As noted in my previous lab writeup, `logInc` appears to follow a normal distribution, in which there is a linear relationship with `logExp`. However, as I noted, the "positioning" (i.e. mode) of this normal density appears to differ when split by both `Race` and `UrbanRural`; relatedly, the "width" (i.e.) variance of this normal distribution only appears to differ when split by `Race` (the variances are roughly equal when split by `UrbanRural`).

The linear relationship, in conjunction with the slightly-differing, albeit related normal distributions, suggest that a hierarchical linear regression is the logical way to build upon the simple linear regression from class. Specifically, we model for $Y_i = log(Income\ for\ respondent\ i), X_i = log(Expenditure\ for\ respondent\ i)$:

$$Y_i \sim N(\beta_{0,r_i,u_i} + \beta_{1,r_i,u_i} \cdot X_i, \sigma^2_{r_i}),$$

with shared/groupwise priors (note that r_i denotes respondent i's race, while u_i denotes respondent i's Urban/Rural response):

$$\beta_{0,r_i,u_i} \sim N(b_0, \tau_0^2), \beta_{1,r_i,u_i} \sim N(b_1, \tau_1^2), 1/\sigma_r^2 \sim Gamma(1,1),$$

and hyperpriors:

$$b_0, b_1 \sim^{iid} N(0,1) 1/\tau_0^2, 1/\tau_1^2 \sim^{iid} Gamma(1,1).$$

My hope was that this hierarchical structure, ie the shared priors and hyperpriors, would promote both information sharing across groups, while also letting the "uniqueness" of each group persist.

Next, I drew posterior samples in JAGs.

```
the_data = list("logInc" = CEsample$logInc,
                "logExp" = CEsample$logExp,
                "r" = CEsample$Race,
                "R" = max(CEsample$Race),
                "u" = CEsample$UrbanRural,
                "U" = max(CEsample$UrbanRural),
                "N" = nrow(CEsample),
                "mu_b0" = 0,
                "mu_b1" = 0,
                "prec_b0" = 1,
                "prec_b1" = 1
                )


the_formula = "

model{

#model
for (i in 1:N){
  logInc[i] ~ dnorm(B_0[r[i], u[i]] + B_1[r[i], u[i]]*logExp[i], inv_sigma_sq[r[i]])
}

#priors
for (rc in 1:R){
  for (ur in 1:U){
    B_0[rc, ur] ~ dnorm(b_0, tau_sq_0)
    B_1[rc, ur] ~ dnorm(b_1, tau_sq_1)
  }
  inv_sigma_sq[rc] ~ dgamma(1,1)
  sigma[rc] <- sqrt(pow(inv_sigma_sq[rc], -1))
}

#hyperpriors
b_0 ~ dnorm(mu_b0,prec_b0)
b_1 ~ dnorm(mu_b1,prec_b1)
tau_sq_0 ~ dgamma(1,1)
tau_sq_1 ~ dgamma(1,1)
}

"




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_jags  <- run.jags(the_formula,
                            n.chains = 2,
                            data = the_data,
                            monitor = c("B_0", "B_1", "sigma"),
                            adapt = 1000,
                            burnin = 5000,
                            sample = 2500,
                            thin = 100,
                            inits = initsfunction)
```

```
## Calling the simulation...
## Welcome to JAGS 4.3.0 on Tue Feb 11 14:19:24 2020
## JAGS is free software and comes with ABSOLUTELY NO WARRANTY
## Loading module: basemod: ok
## Loading module: bugs: ok
## . . Reading data file data.txt
## . Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 994
##    Unobserved stochastic nodes: 34
##    Total graph size: 6019
## . Reading parameter file inits1.txt
## . Reading parameter file inits2.txt
## . Initializing model
## . Adaptation skipped: model is not in adaptive mode.
## . Updating 5000
## -------------------------------------------------| 5000
## ************************************************** 100%
## . . . . Updating 250000
## -------------------------------------------------| 250000
## ************************************************** 100%
## . . . . . Updating 0
## . Deleting model
## .
## Note: the model did not require adaptation
## Simulation complete.  Reading coda files...
## Coda files loaded successfully
## Calculating summary statistics...
## Calculating the Gelman-Rubin statistic for 30 variables....
## Finished running the simulation
```

I'll spare you the printing of all the trace plots, but you have my word that they converge.

```
#plot(posterior_jags)
```

Next, I extract the samples for posterior predictive/synthetic sampling.

```
#extract, store in dictionary for quicker retrieval
posterior_df = data.frame(as.mcmc(posterior_jags))
```

```
## Warning in as.mcmc.runjags(posterior_jags): Combining the 2 mcmc chains together
```

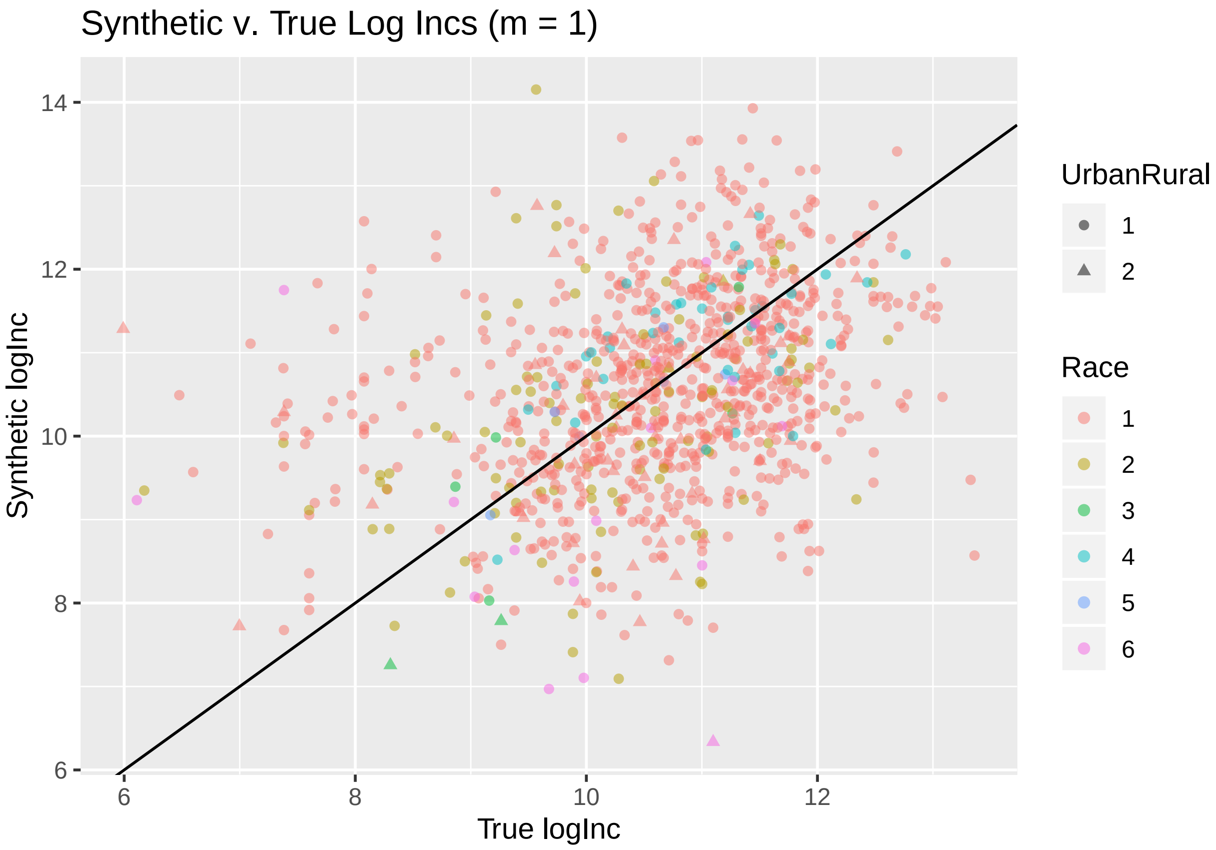
```
posterior_list = lapply(colnames(posterior_df), function(x) posterior_df%>%pull(x))%>%
  `names<-`(colnames(posterior_df))
```

I also define my synthetic sampler functions, which let the user choose their desired m. Importantly, I randomly sample the row to which each $i = 1, \dots, m$ correspondents, in the spirit of randomizing wherever possible.

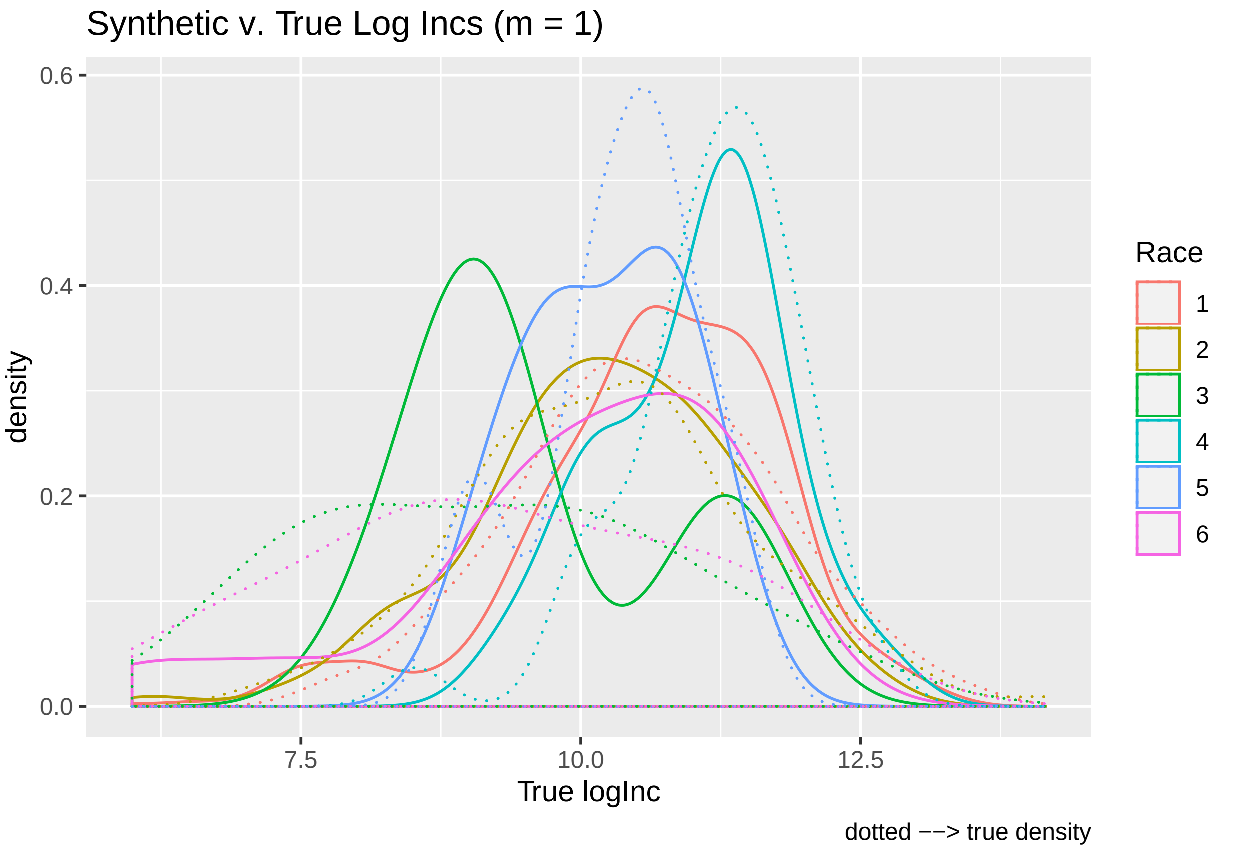
```
synth_helper <- function(posterior_list, df){

  #randomly select posterior draw to use

  idx_grab = sample(1:nrow(df), 1)

  beta_suffixes = paste0(".", df$Race, ".", df$UrbanRural, ".")
  sigma_suffixes = paste0(".", df$Race, ".")

  b0 = sapply(beta_suffixes, function(x) posterior_list[[paste0("B_0", x)]][idx_grab])%>%as.vector()
  b1 = sapply(beta_suffixes, function(x) posterior_list[[paste0("B_1", x)]][idx_grab])%>%as.vector()


  mu_post = b0+b1*df%>%pull(logExp)
  sig_post = sapply(sigma_suffixes, function(x) posterior_list[[paste0("sigma", x)]][idx_grab])%>%as.vec

  result = rnorm(nrow(df), mu_post, sig_post)

  return(result)
}


synth_logInc <- function(posterior_list, df, m=1, ...){
  set.seed(m%%3)

  lapply(1:m,
         function(y)
           synth_helper(posterior_list, df, ...)
         )%>%
    return()
}
```

Finally, I perform an $m = 1$ synthetic sample. Results are discussed below.

```
synth_single = synth_logInc(posterior_list, CEsample, m = 1)

ggplot(data.frame(logInc_true = CEsample$logInc,
                  logInc_synth = unlist(synth_single),
                  Race = as.factor(CEsample$Race),
                  UrbanRural = as.factor(CEsample$UrbanRural)),
       aes(x = logInc_true, y = logInc_synth, color = Race, shape = UrbanRural))+
  geom_point(alpha = .5) +
  geom_abline(slope = 1) +
  labs(x = "True logInc", y = "Synthetic logInc", title = "Synthetic v. True Log Incs (m = 1)")
```



```r
ggplot(data.frame(logInc_true = CEsample$logInc,
                  logInc_synth = unlist(synth_single),
                  Race = as.factor(CEsample$Race),
                  UrbanRural = as.factor(CEsample$UrbanRural)),
       aes(x = logInc_true,color = Race))+
  geom_density() +
  geom_density(aes(x = logInc_synth,color = Race), linetype = "dotted")+
  labs(x = "True logInc",title = "Synthetic v. True Log Incs (m = 1)",
       caption = "dotted --> true density")
```



i.)

At first glance, it doesn't appear as though my synthetic dataset did a particularly good job. For one, the synthetic densities (filled lines) of `logInc` are positioned and stretched differently than the true densities (dotted lines) – in fact, the hierarchcial model seemed to introduce too much pooling/shrinkage into the synthetic distribution. Further, in examining the scatterplot, we do not see a "centering" about the $y = x$ line that represents true 1:1 correspondence. This also suggests that this synthetic generator may not do the best job of recreating the underlying distribution.

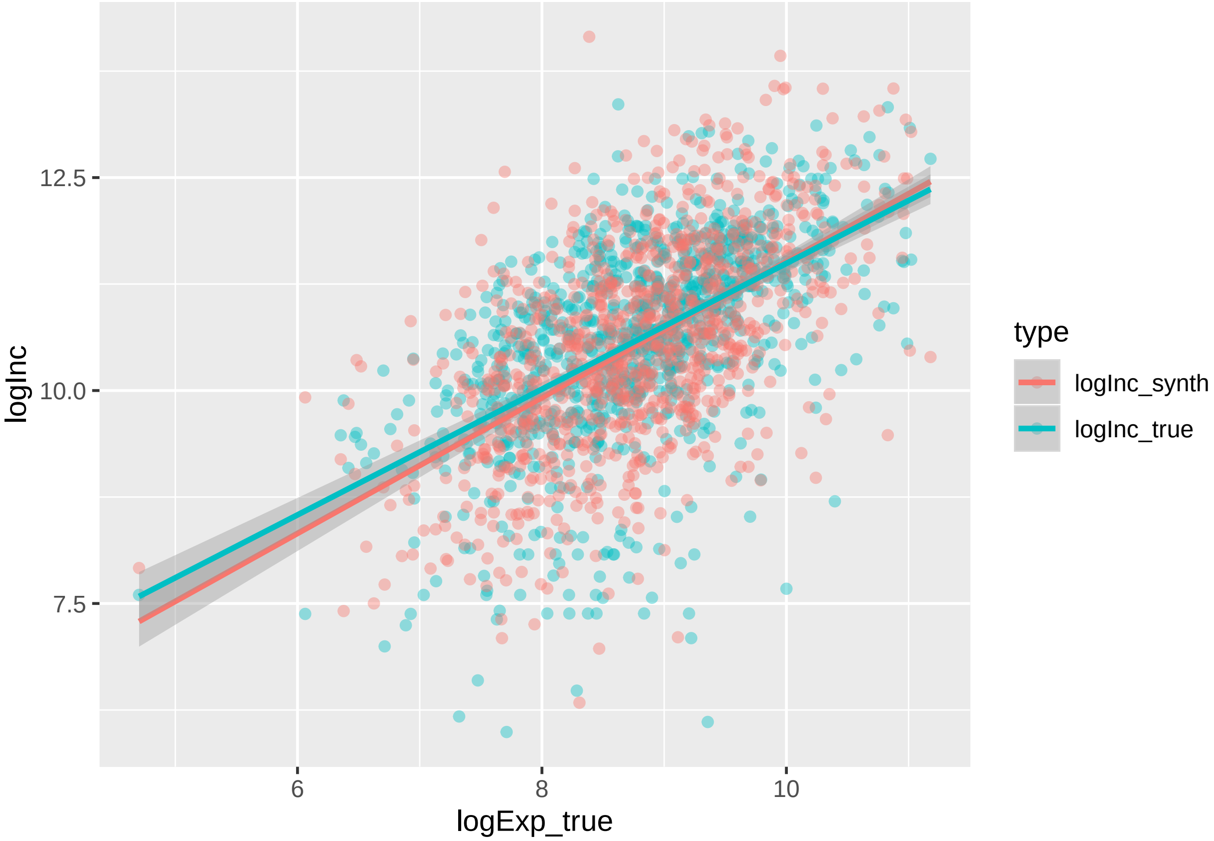
The poor performance here likely stems from either a coding error, or a poorly fit model – perhaps, despite the shared prior/hyperprior, parameter sampling is still occuring wildly. However, it's also worth nothing that this was run for $m = 1$. As detailed in the slides, traditional practice is to use $m > 1$, so as to minimize reliance on a single, perhaps outlandish posterior draw.

ii.)

The mean of my synthetic sample was 10.5429789, while the median was 10.5404816. In truth, the mean of log income was 10.5950712, while the median was 10.7057357. So in this respect, the model was at least a bit closer.

iii.)

```
compare_lm = data.frame(logInc_true = CEsample$logInc,
                        logExp_true = CEsample$logExp,
                         logInc_synth = unlist(synth_single))

ggplot(compare_lm%>%gather(type, logInc, -logExp_true),
       aes(x = logExp_true, y = logInc, color = type))+
  geom_point(alpha = .4)+
  stat_smooth(method = "lm")
```



```
summary(lm(logInc_true~logExp_true, compare_lm)); summary(lm(logInc_synth~logExp_true, compare_lm))
```

```
##
## Call:
## lm(formula = logInc_true ~ logExp_true, data = compare_lm)
##
## 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 ***
## logExp_true  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
```

```
##
## Call:
## lm(formula = logInc_synth ~ logExp_true, data = compare_lm)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.8268 -0.6315 -0.0043  0.6621  3.9267
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3.53761    0.31830   11.11   <2e-16 ***
## logExp_true  0.79751    0.03606   22.12   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9901 on 992 degrees of freedom
## Multiple R-squared:  0.3302, Adjusted R-squared:  0.3296
## F-statistic: 489.1 on 1 and 992 DF,  p-value: < 2.2e-16
```

In comparing linear regressions of both the synthetic and true incomes against true expense, we see that the parameters for B_0 and B_1 were reasonably close, with the former in the 3.7-4.1 range and the latter in the .73-74 range.

3.)

i.)

According to the paper, synthesized variable fields included `Payroll`, `Employment`, `Multiunit`, `Firstyear`, `Lastyear`, while `SIC` was left as is. Further, `County` (and by extension, `State`) was simply not released, while `Year` and `ID` were created and released. The paper also notes the existence of additional, confidential variables that were not used to generate this synthetic dataset, but may be used in future sets.

ii/iii.)

In order to simulate the multiple variables in play, Kinney et al. built a nested series of models – i.e. they imputed "outwards" from the categorical targets to the continuous targets. Importantly, the output of one synthesis could be an input for the next, thus giving the "nested" structure described above. More specifically, this structure involved

1.) Model category $Firstyear|County, SIC \sim Multinom$ with (flat) dirichlet priors, i.e. the multinomial-dirichlet conjugacy.

2.) Model category $Lastyear|Firstyear, County, SIC \sim Multinom$ with (flat) dirichlet priors, i.e. the multinomial-dirichlet conjugacy. Note that the output of 1 is an input here.

3.) Model category $Multiunit|Lastyear, Firstyear, County, SIC \sim Multinom$ with (flat) dirichlet priors, i.e. the multinomial-dirichlet conjugacy. Here, the outputs of 1 and 2 are inputs.

4.) Model continuous $Employment|Multiunit, Lastyear, Firstyear, County, SIC$ via a hybrid AR1/MLR model, with an added KDE smoothing transformation at the end. Note that the outputs of 1-3 were, in addition to the AR1's lag component, covariates in this linear regression.

5.) Model continuous $Payroll|Employment, Multiunit, Lastyear, Firstyear, County, SIC$ via a hybrid AR1/MLR model, with an added KDE smoothing transformation at the end. Note that the outputs of 1-4 were, in addition to the AR1's lag component, covariates in this linear regression.

As shown above, this model was built "outwards" from the categorical fits above. In this way, they were able to synthesize multiple variables at once. Notably, this puts principal importance on models 1-2, as the remainder of the models rely/are conditional on their accuracy.

iv.)

As noted on page 365, they generated a single synthetic entry for over 21 million records. So it appears that $m = 1$.

v.)

Annual comparisons of each synthetic marginal values/summary stats with the true marginal values/summary stats was one way the authors measured utility: for instance, they remarked that between 1976 and 2000, synthetic `Employment` and true `Employment` had an average 1.3% discrepancy (think MAE, converted to percentage) over those years, while synthetic `Payroll` and true `Payroll` had an average 8% discrepancy over those years. Further, the authors note that within non-synthesized categories of location and industry, employments, establishments, and payrolls generally aligned with the true summary statistics in these categories. This finding appears to have been made via visual inspection/comparison of group-wise scatterplots for these variables (similar to that in question 2).

Further, the authors also corroborated the accuracy of their synthesis by comparing transformations of the synthetic data with transformations of the true data. For instance, job creation/destruction metrics can be backed out of year-to-year employment figures, so the authors calculated these measures for the true data and then calculated the same measures for their own data. To their credit, these transformations aligned closely, although the synthetic data consistently showed a lower job creation rate than the true data. In a similar manner, the authors also plugged both the synthesic data and the true data into a pre-fit linear economic growth regression, before comparing the performances of the two. Again, the synthetic data exhibited similar trends to that of the true data, when plugged into the model, further illustrating the robustness of the synthetic data.

Though their analysis of utility/accuracy was fairly exhaustive, other utility measures could have included:

- correlations between the various transformations of the two datasets (i.e. job flow metrics, the growth regression), as well as correlations for the plots that were subject to visual inspection.
- MAE estimates not just for marginal rates (i.e. one variable in a vacuum), but perhaps MAEs/percentage discrepancies when faceted over one or two additional covariates. This reporting would, of course, be subject to it not disclosing individual entries.
- A brief case study in how small perturbations in the early components of the nested model (such as `Firstyear`, `Lastyear`), that is those outputs most foundational to the overall model, would have affected accuracy. This could have spoken to the structural "stability" of the model.

vi.)

Due to a paucity of matching algorithms/appropriate software, the authors could not have a computer comb through the synthetic data and pick out individuals. As such, the authors analyzed disclosure risks in the following ways. First, they analyzed the probability that true birth/death years (`Firstyear`, `Lastyear`, respectively) matched those of the synthetic – in most cases, this probability was low, as the synthetic value was close but not exactly equal. This alone, they argue, "confounded" reidentification substantially.

Second, the authors also point to the KDE transform of the continuous variables as an additional buffer against reidentification – that is, a single vector of covariates cannot be "tracked" through the data, as the smoother obfuscates this vector's potential uniqueness.

Third, the authors also note that while the overall shape of the synthetic data matches that of the true data, correlations between the value of each entry are minimal. This also speaks to limited disclosure risks.

Fourth, as discussed on the authors found that outliers in the true data rarely align with the outliers in the synthetic data: they found that in fewer than 5% of cases, the maximum synthetic employment value corresponded to the maximum true employment value.

Fifth, the authors applied more complex algorithms, which compared year-to-year transition probabilities, to see if year-to-year behavior in the synthetic dataset aligned with year-to-year behavior in the true dataset. It did not – in fact, even if the intruder knew pretty much everything about the true and synthetic data, they might still have trouble.

This disclosure risk analysis seemed pretty robust to me, and with little knowledge of the field, I don't have too many additional disclosure risk tests to add. However, I think there is one non-mathematical check that would be worthwhile: take the synthetic data to the accounting departments/C-Suites of the 10-20 most "vulnerable" companies (provided they can be contacted), and see if they can identify themselves in the synthetic data. If most cannot, then identities are likely secure.