INSTITUTE FOR EMPLOYMENT RESEARCH
The Research Institute of the Federal Employment Agency

AnigeD

ITS COMPLICATED: KNOW YOUR DATA AND KNOW YOUR GENERATOR

Wiesbaden,
21. März, 2024

Jonathan Latner, PhD
Dr. Marcel Neuenhoeffer
Prof. Dr. Jörg Drechsler

SECTIONS

1. Introduction
2. Know your data (SD2011)
3. Know your generator
   - DataSynthesizer
   - CTGAN
   - Synthpop
4. Conclusion

Section 1: Introduction

SYNTHETIC DATA: WHATS THE GOAL?

- According to Rubin (1993), there are multiple goals for generating synthetic data
  - For the survey participants, they could be confident that their data would never be released, which could increase the likelihood not only of more survey respondents, but also more truthful answers.
  - For the 'snooper' or illegitimate user, the value of the data would be eliminated because they could not discover actual, confidential information.
  - For the legitimate user, the value of the data would increase not only because the data are easier to access, but also more accurate than the actual data.
  - For the researcher, synthetic data can be used to develop code before they have access to the original data. As a result, high quality synthetic data can increase knowledge creation by allowing more people to use better data.
- Synthetic data can accelerate development. Good quality synthetic data can significantly accelerate data science projects and reduce the cost of development...it contributes to data democratisation (Jordan et al., 2022).

SYNTHETIC DATA: WHATS THE PROBLEM?

- For the survey participants, it is not known whether their participation is affected by the public, protected release of their data
- For the snooper, without denying the importance of so-called ‘intruder’ studies that evaluate risk, reports of malicious re-identifications are rare to non-existent (Francis and Wagner, XXXX)
- For the legitimate user, more accurate data can only be true in the context that Rubin originally considered: if you had design variables available for the entire frame and you condition on them when generating new data. This almost never applies in practice.
- For the researcher, the theoretical and realistic goal are the same: synthetic data can make time with original data more efficient and effective.
- As a result, even if synthetic data cannot be a replacement for real data, high quality synthetic data can be a valuable tool to accelerate knowledge creation.

THE GOOD NEWS - MAKING SYNTHETIC DATA IS EASY

- `Gretel.ai`: The synthetic data platform for developers. Generate artificial datasets with the same characteristics as real data, so you can develop and test AI models without compromising privacy.
- `Mostly.ai`: Synthetic Data. Better than real. Still struggling with real data? Use existing data for synthetic data generation. Synthetic data is more accessible, more flexible, and simply...smarter.
- `Statice.ai`: Generating synthetic data comes down to learning the joint probability distribution in an original, real dataset to generate a new dataset with the same distribution. The more complex the real dataset, the more difficult it is to map dependencies correctly. Deep learning models such as generative adversarial networks (GAN) and variational autoencoders (VAE) are well suited for synthetic data generation.
- `hazy.com`: Synthetic data does not contain any real data points so can be shared freely. Say goodbye to lengthy governance processes associated with real data. Specifically, Hazy data is designed to preserve all the patterns, statistical properties and correlations in the source data, so that it can be used as a drop-in replacement for it.
- DataSynthesizer: The distinguishing feature of DataSynthesizer is its usability — the data owner does not have to specify any parameters to start generating and sharing data safely and effectively.

THE BAD NEWS - MAKING SYNTHETIC DATA IS HARD

- According to the Alan Turing Institute (Jordan et al., 2022)
  - Synthetic data is not automatically private.
  - Evaluating the privacy can be problematic.
    - Privacy is a function of the mechanism that generated a synthetic dataset.
    - Can privacy be evaluated by comparing synthetic and real data?
- There is no one measure for utility, fidelity, or privacy
  - Utility and fidelity are sometimes called general/broad and specific/narrow measures within the single concept of utility (Snoke et al., 2018; Drechsler and Reiter, 2009)
  - One example of utility is propensity score mean-squared error (pMSE)
  - One example of fidelity is confidence interval overlap (CIO)
- Efficiency (duration in time) is important and often ignored. The algorithm should scale well with the dimension of the data space in a relational way, not exponential way

COMPARE 3 SYNTHETIC DATA GENERATOR (SDG)

- DataSynthesizer (Bayes)
- CTGAN (GAN)
- Synthpop (CART)

Section 2: Know your data (SD2011)

REAL DATA

- Social Diagnosis 2011 (SD2011)
- Loads with Synthpop
  - `http://www.diagnoza.com/index-en.html`
  - Not entirely clear how original data is created or cleaned to create data in Synthpop
- Like real data, has ‘quirks’ or unusual values/variables
  - Includes missings
    - Informative (i.e. month married, but single)
    - Non-informative
  - Includes ‘errors’
    - `smoke` - Does not smoke is NO, but `nociga` - 20/22 cigarettes per day
  - Includes generated variables
    - `bmi`, `agegr`
    - Can be problematic for SDG

DATA (SD2011)

Number	Variable	Description	Type	Observations	Unique.Values	Missings	Negative.values	Generated	Quirks
1	sex	Sex	factor	5000	2	0	0		
2	age	Age of person, 2011	numeric	5000	79	0	0		
3	agegr	Age group, 2011	factor	5000	7	4	0	Yes	Yes
				...					
7	eduspec	Discipline of completed qualification	factor	5000	28	20	0		Yes
				...					
10	income	Personal monthly net income	numeric	5000	407	683	603		
11	marital	Marital status	factor	5000	7	9	0		
12	mmarr	Month of marriage	numeric	5000	13	1350	0		
13	ymarr	Year of marriage	numeric	5000	75	1320	0		
14	msepdiv	Month of separation/divorce	numeric	5000	13	4300	0		
15	ysepdiv	Year of separation/divorce	numeric	5000	51	4275	0		
				...					
22	nofriend	Number of friends	numeric	5000	44	0	41		
23	smoke	Smoking cigarettes	factor	5000	3	10	0		
24	nociga	Number of cigarettes smoked per day	numeric	5000	30	0	3737		Yes
				...					
27	workab	Working abroad in 2007-2011	factor	5000	3	438	0		
28	wkabdur	Total time spent on working abroad	numeric	5000	33	0	4875		Yes
				...					
33	height	Height of person	numeric	5000	65	35	0		
34	weight	Weight of person	numeric	5000	91	53	0		
35	bmi	Body mass index (weight/(height^2)*10000	numeric	5000	1396	59	0	Yes	Yes

Section 3a): Know your generator (DataSynthesizer)

TASKS

- Run default model - correlated attribute mode
- Hyperparameters
  - ϵ DP: 0 (default 0.1)
  - Bayesian network ($\mathcal{N}$) parents: 1, 2, or 3 (default is 'greedy')

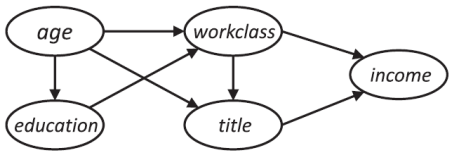
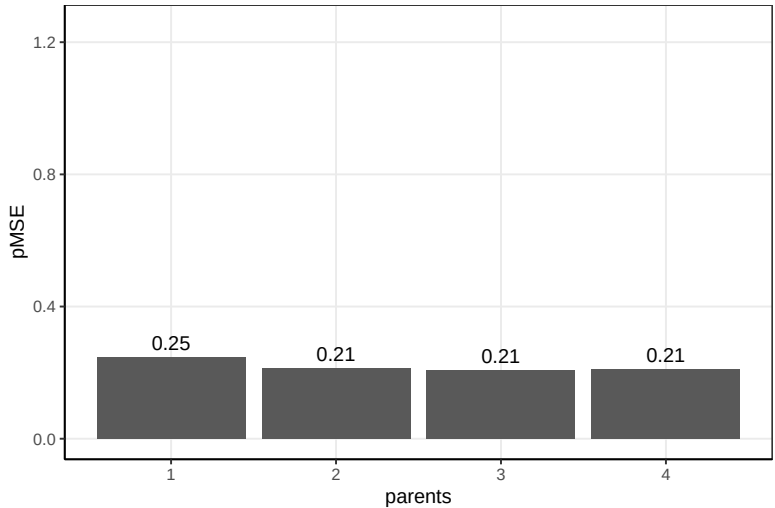


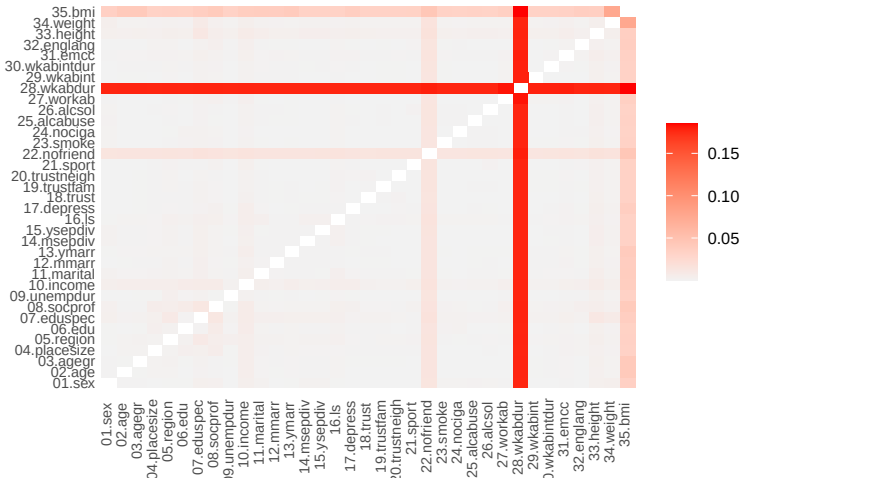
Fig. 1. A Bayesian network $\mathcal{N}_1$ over five attributes.

- In Fig. 1, $\mathcal{N} = 2$, but not known in reality

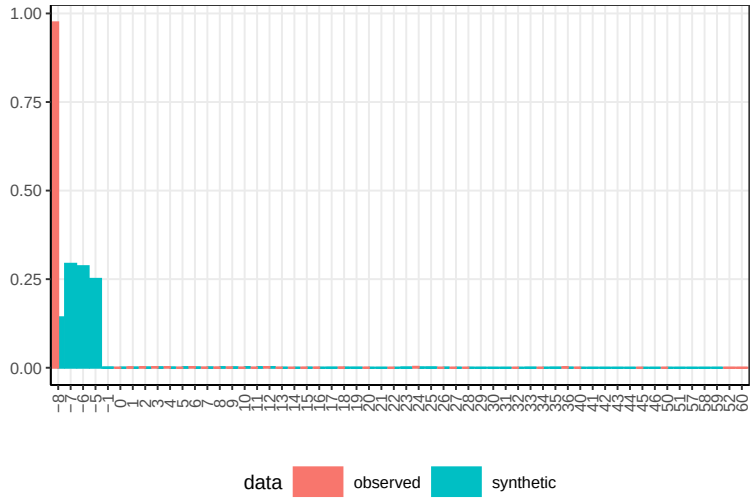
SD2011 - PMSE



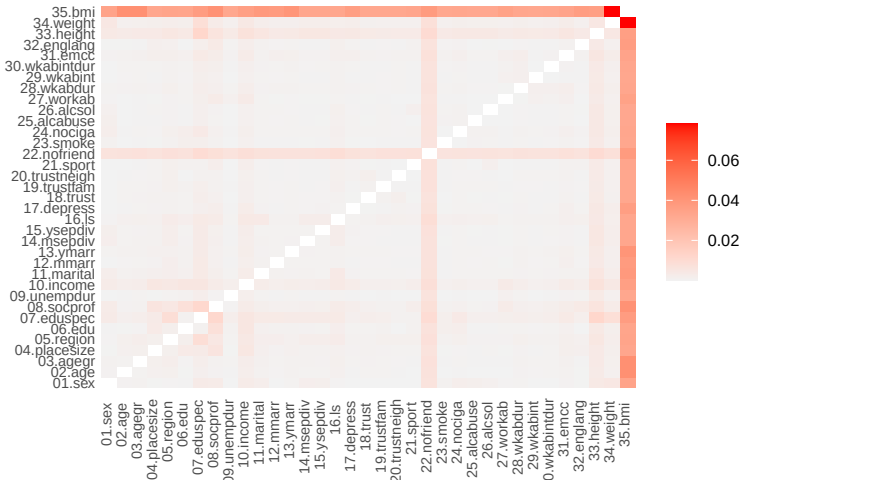
DATASYNTHESIZER - SD2011(A)



VARIABLE: WKABDUR (WORK ABROAD DURATION)

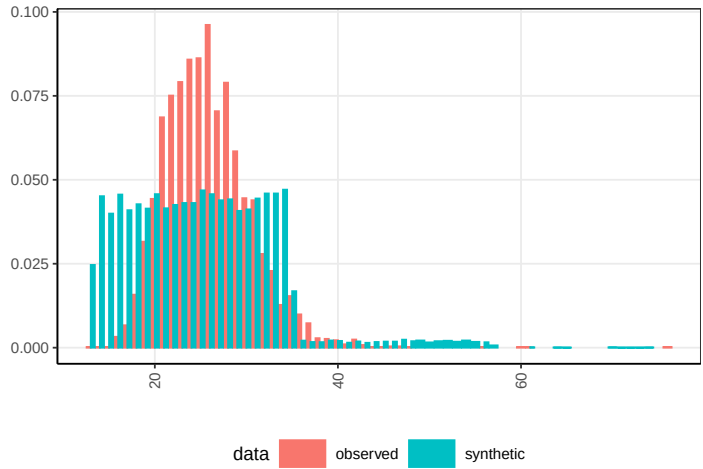


DATASYNTHESIZER - SD2011(B)

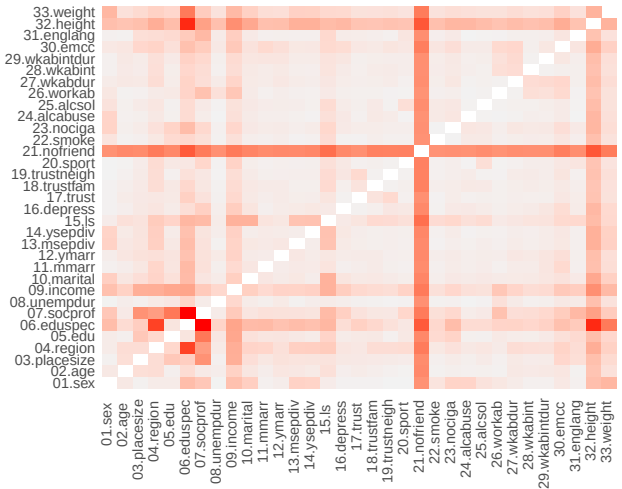


VARIABLE: BMI

Figure 1: BMI $<$ 20 is underweight/malnourished

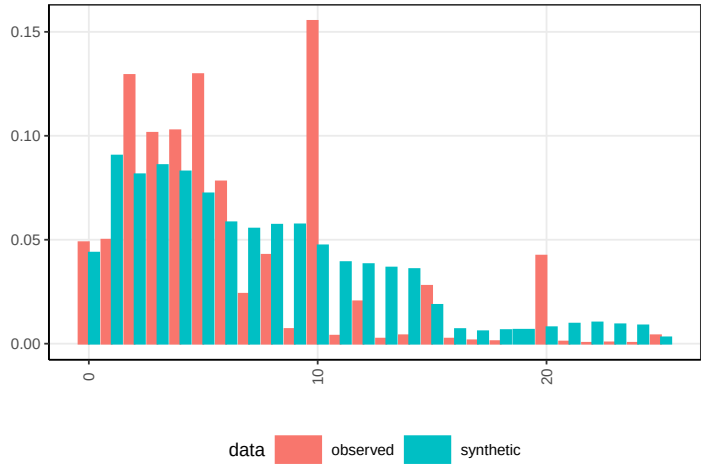


DATASYNTHESIZER - SD2011(C)

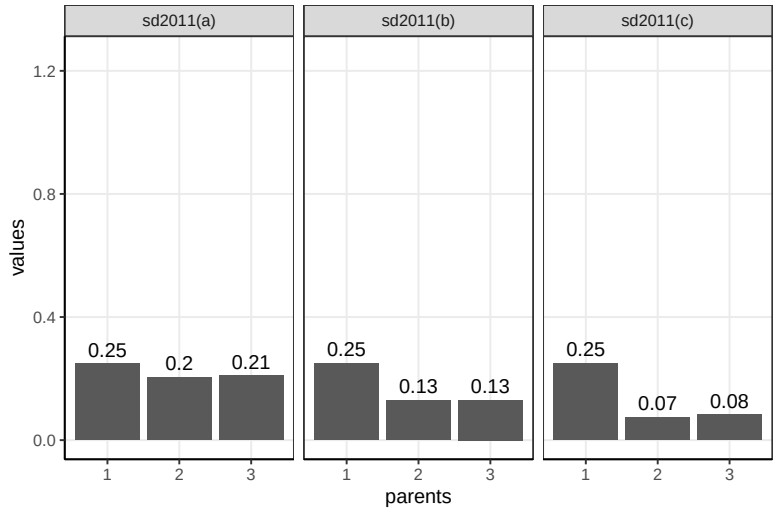


VARIABLE: NOFRIEND

Figure 2: Doesn't capture rounding/discontinuity

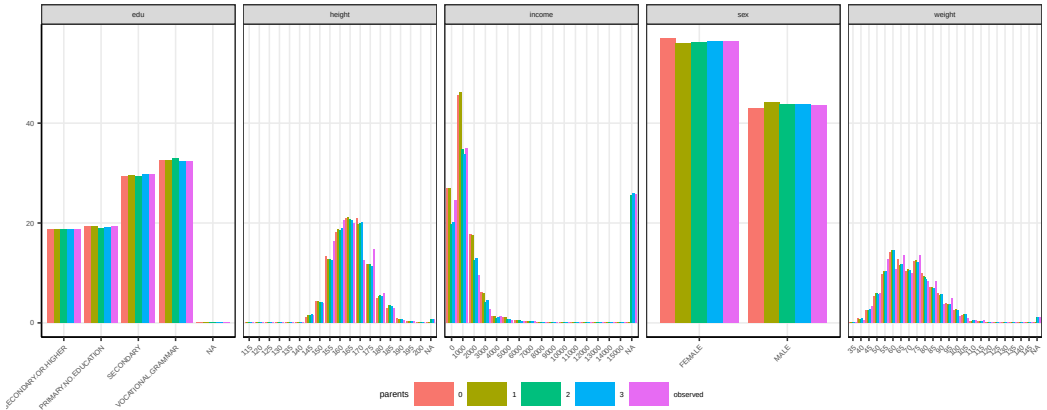


SD2011 - PMSE


sd2011(a)
sd2011(b)
sd2011(c)
values
1.2
0.8
0.4
0.0
0.25
0.2
0.21
0.25
0.13
0.13
0.25
0.07
0.08
1
2
3
parents

DATASYNTHESIZER: SELECTED VARIABLES

Figure 3: No missings if parents < 2



Section 3b): Know your generator (CTGAN)

TUNING CTGAN

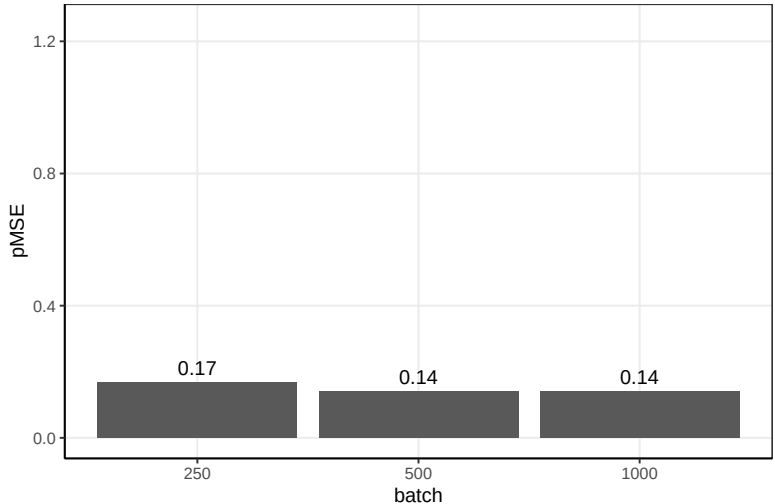
- Batch size (constant steps)
- Epochs (constant batch size)
- Dimensions (2 hyperparameters)
  - embedding_dim (int): Size of the random sample passed to the Generator. Defaults to 128.
  - dimensionality - 2 hyperparameters, but same value for each
    - discriminator_dim (tuple or list of ints): Size of the output samples for each one of the Discriminator Layers. A Linear Layer will be created for each one of the values provided. Defaults to (256, 256).
    - generator_dim (tuple or list of ints): Size of the output samples for each one of the Residuals. A Resiudal Layer will be created for each one of the values provided. Defaults to (256, 256).

BATCH SIZE, EPOCHS, AND STEPS

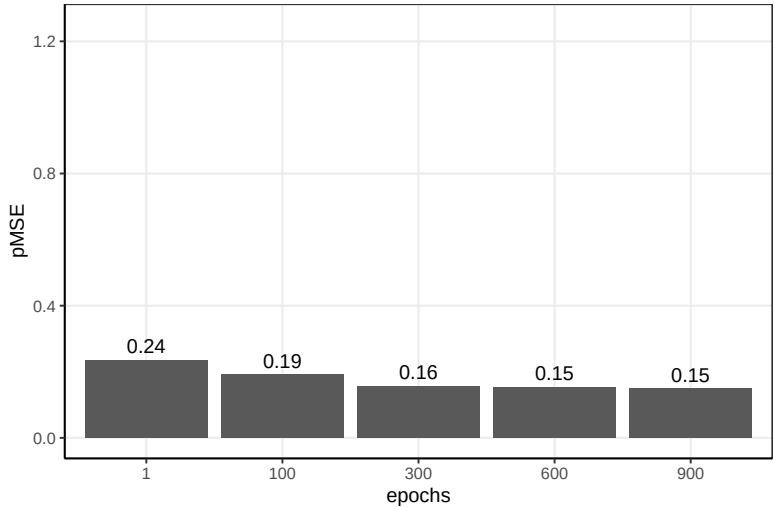
Table 1

N	Batch size	Steps per Epoch	Epochs	Actual Steps
5.000	100	50	60	3,000
5.000	250	20	150	3,000
5.000	500	10	300	3,000
5.000	1.000	5	600	3,000
5.000	500	10	100	1,000
5.000	500	10	300	3,000
5.000	500	10	600	6,000
5.000	500	10	900	9,000

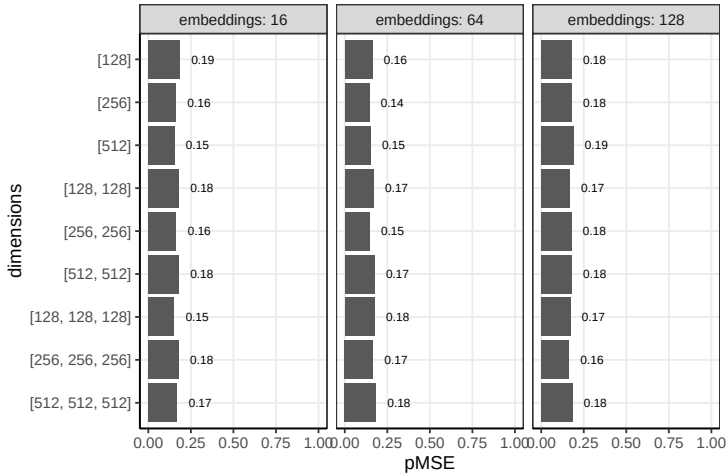
CTGAN: EFFECT OF BATCH SIZE (CONSTANT STEPS)



CTGAN: EFFECT OF EPOCHS (CONSTANT BATCH SIZE)



CTGAN: EFFECT OF DIMENSIONS



Section 3c): Know your generator (Synthpop)

METHODOLOGY

According to the online description, Synthpop follows the two-stage process described by Rubin.

However, if one actually looks at the code, there is an important difference.

Instead of replacing y with y^* generated by the model, Synthpop replaces y with the leaf number from the regression tree (line 523-523).

Then, Synthpop predicts the leaf number (line 527) for y.

Finally, Synthpop replaces the predicted leaf number with sampled values of y from the actual data in that leaf (line 549).

The point: the second-stage replaces values in the synthetic data (D^*) with values in the actual data (D).

SYNTHPOP (SYN.CART)

https://rdrr.io/cran/synthpop/src/R/functions.syn.r

```r
syn.cart <- function(y, x, xp, smoothing = "", proper = FALSE,
                     minbucket = 5, cp = 1e-08, ...)
fit <- rpart(y ~ ., data = as.data.frame(cbind(y, x)), method = "anova",
             minbucket = minbucket, cp = cp, ...)
# get leaf number for observed data
leafnr  <- floor(as.numeric(row.names(fit$frame[fit$where,])))
# replace yval with leaf number in order to predict later node number
# rather than yval (mean y for observations classified to a leaf)
fit$frame$yval <- as.numeric(row.names(fit$frame))
# predict leaf number
nodes       <- predict(object = fit, newdata = xp)

...

uniquenodes <- unique(nodes)
new  <- vector("numeric",nrow(xp))
    for (j in uniquenodes) {
      donors <- y[leafnr == j] # values of y in a leaf
      new[nodes == j] <- resample(donors, size = sum(nodes == j), replace = TRUE)
    }
```

METHOD AND CODE MAY NOT BE THE SAME

The code appears to indicate that the synthetic sample is drawn from the observed data in a predicted leaf, rather than the predicted y value.

Is this different than the methodological description?

Does this violate the definition of synthetic data, described above?

The point (and question): High levels of utility in synthpop may be the result of code that implements the method in a way that is not consistent with the definition of synthetic data

If true, then utility in Synthpop may be artificially high (which lowers the relative advantage to other data synthesizers)

More research is needed

Section 4: Conclusion