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

AnigeD

GENERATING SYNTHETIC DATA IS COMPLICATED: KNOW YOUR DATA AND KNOW YOUR GENERATOR

Wiesbaden,
21. März, 2024

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

Section 1: Introduction

WHAT IS OUR GOAL TODAY?

- Common perception that making synthetic data is easy
- We want to show that its complicated
  - You need to know your data
    - Missing values, messy data, etc.
  - You need to know your generator
    - How does it deal with missing values?
    - How efficient is it (data dimensionality)?
    - How does it meet privacy standards?
- Conclusion
  - There is no one, single solution to creating synthetic data
  - The right generator depends on the goal
  - Eventually ... make some recommendations (but not today)

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)
- How do we evaluate utility (and fidelity)? There is no one measure of either.
  - 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).
- Efficiency (i.e. 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.
- How do we evaluate privacy?
  - Is privacy a function of the generator?
  - Is privacy a function of the data?

OUR GOAL IS TO ILLUSTRATE THE CHALLENGES

- Evaluate 3 synthetic data generators (SDG)
  - DataSynthesizer
  - CTGAN
  - Synthpop
- Know your data
  - Cleaning/pre-processing
  - Utility: Propensity score mean-squared error (pMSE) - the mean-squared difference between the estimated probabilities and the true proportion of the synthetic data in the combined records (0.5). Lower values indicate better performance.
- Know your generator
  - How does it actually create data?
  - How efficient is it with respect to data dimensionality?
  - Thinking about what this means for privacy

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
    - `bmi` = 451, but `height` = 149 and `weight` = NA (999)
  - 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)

DataSynthesizer, a Python package, implements a version of the PrivBayes (Zhang et al., 2017) algorithm. DataSynthesizer learns a differentially private Bayesian Network which captures the correlation structure between attributes and then draws samples (Little et al., 2021).

Variable type: The Bayesian network only works with discrete variables. One way to discretize continuous variables is by binning them.

DATASYNTHESIZER

- Hyperparameters
  - ϵ DP: 0 (default 0.1)
  - k-degree Bayesian network (parents): 1 (independent), 2, 3, or 4 (default is ‘greedy’)
  - In Fig. 1, $k = 2$, but not known in reality

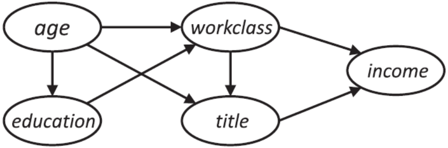
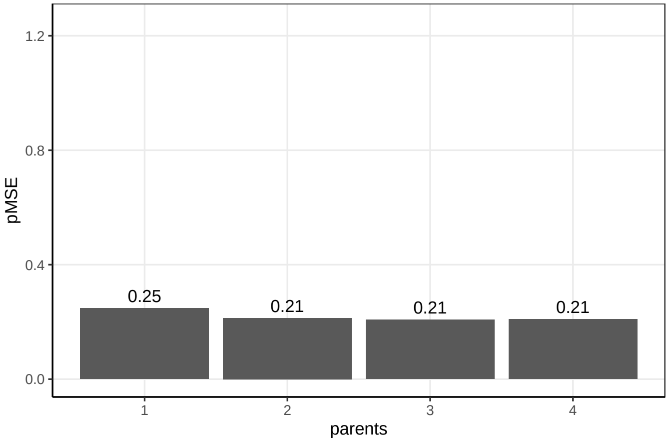


Fig. 1. A Bayesian network $\mathcal{N}_1$ over five attributes.

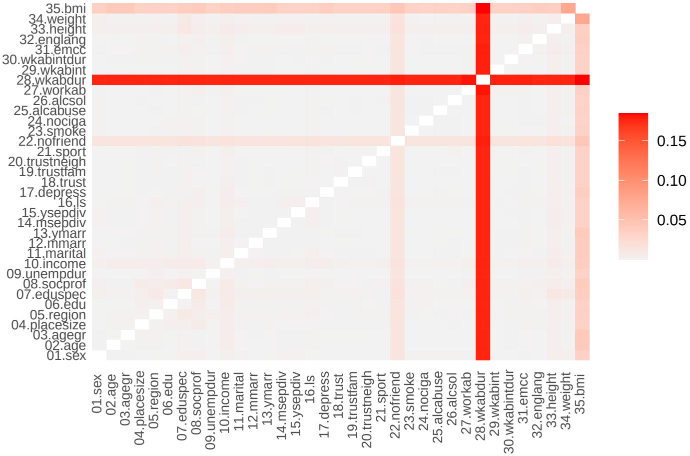
SD2011 - PMSE BY NUMBER OF PARENTS

Figure 1: Model fit does not improve after $k = 2$



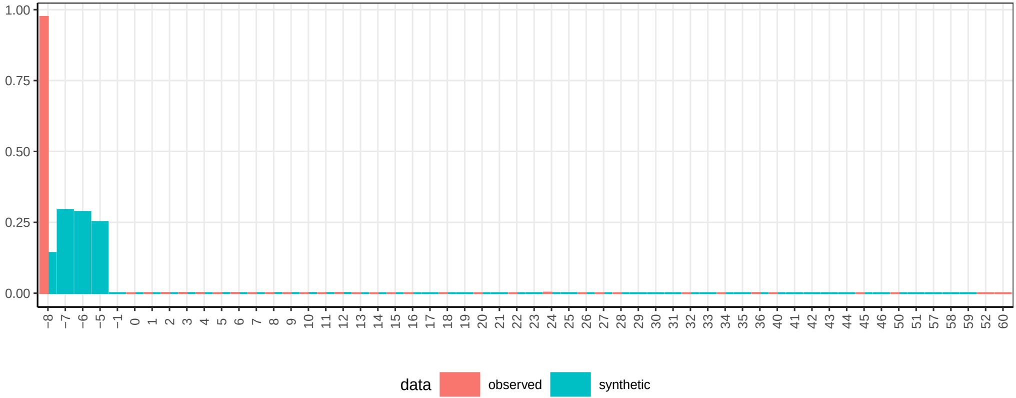
TWO-WAY UTILITY: PMSE FOR PAIRS OF VARIABLES

Figure 2: SD2011(a) – Raw data



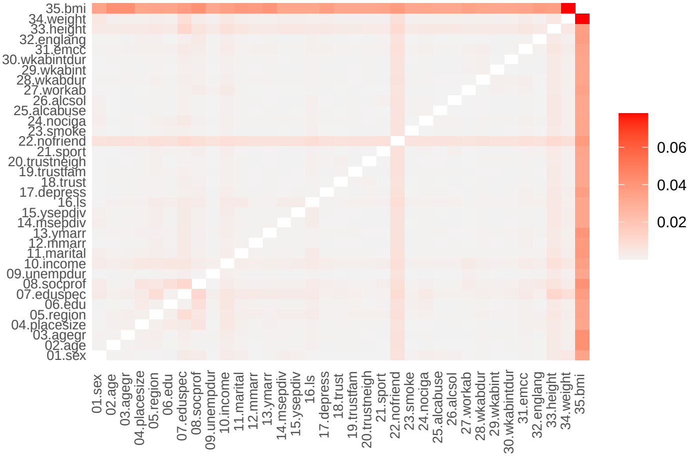
VARIABLE: WKABDUR (WORK ABROAD DURATION)

Figure 3: Captures values < 0 as continuous, not missing/categorical



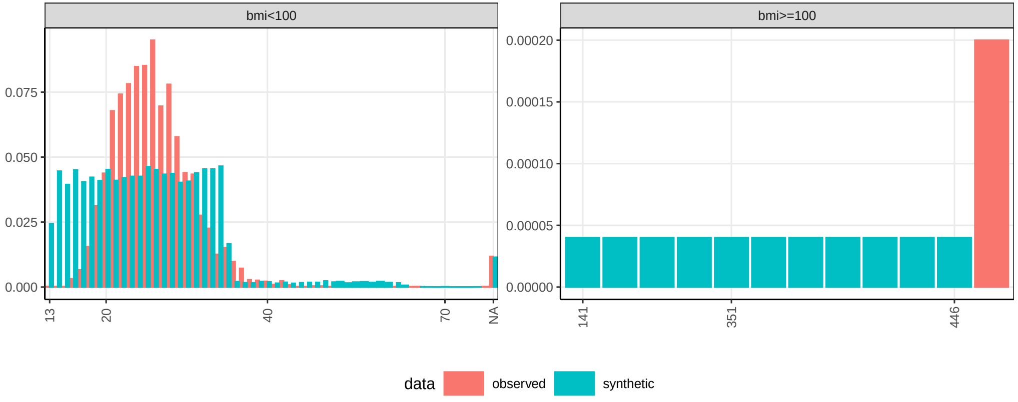
TWO-WAY UTILITY: PMSE FOR PAIRS OF VARIABLES

Figure 4: SD2011(b) – missing are numerical values < 0 and " " categorical values



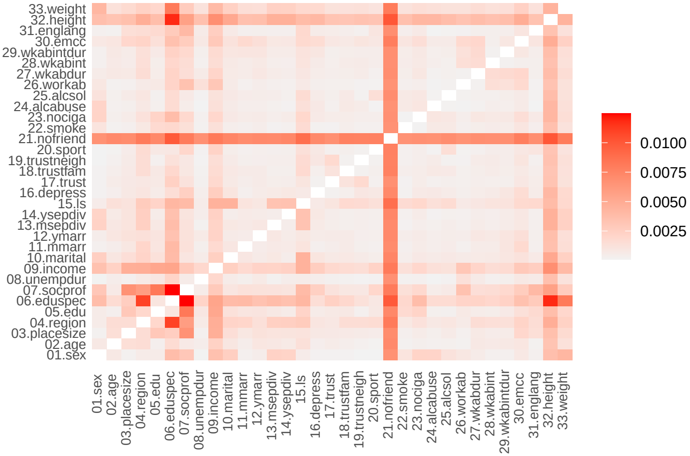
VARIABLE: BMI

Figure 5: BMI $<$ 20 is underweight/malnourished



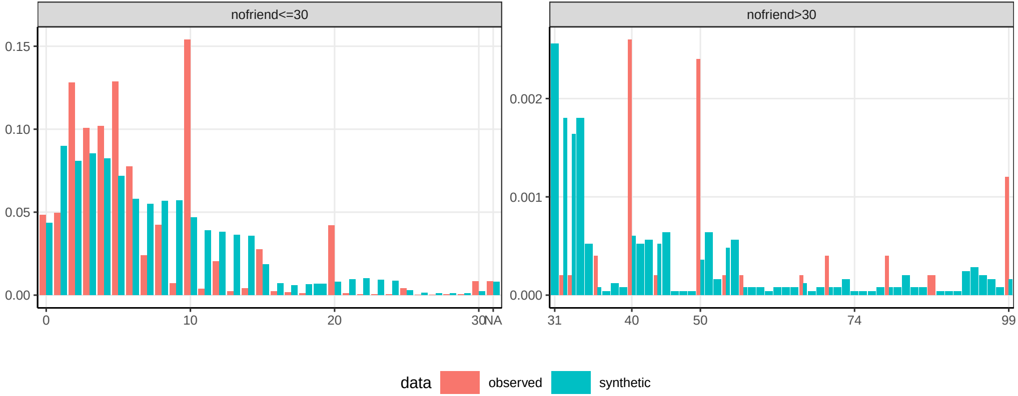
TWO-WAY UTILITY: PMSE FOR PAIRS OF VARIABLES

Figure 6: SD2011(c) – drop generated variables (bmi and agegr)



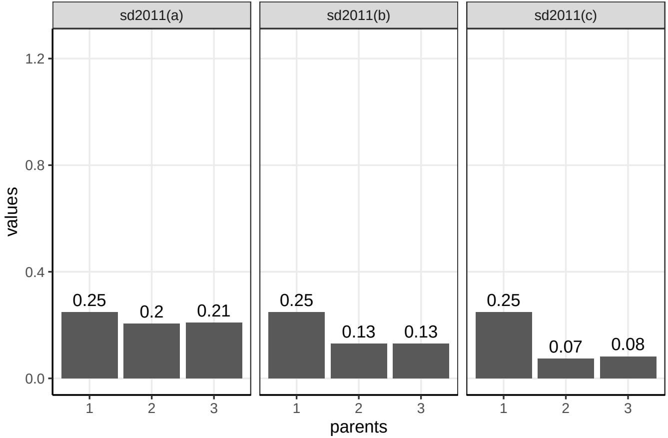
VARIABLE: NOFRIEND

Figure 7: Doesn't capture rounding/discontinuity



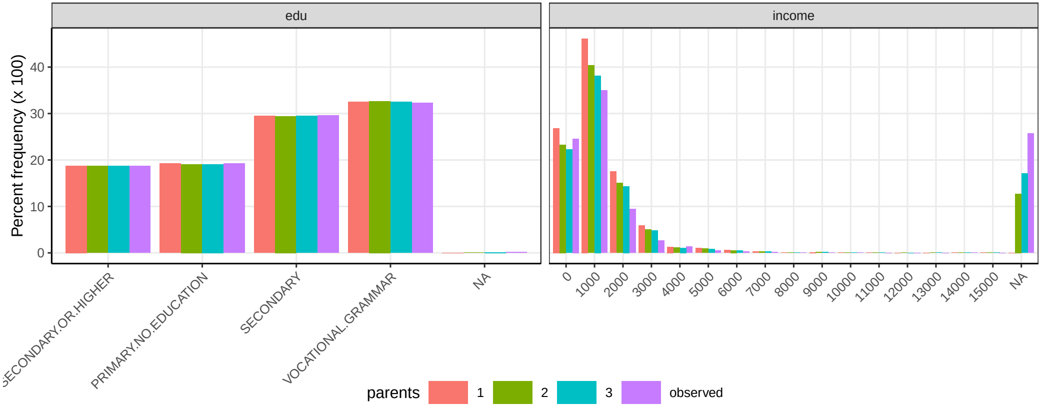
SD2011 - PMSE

Figure 8: We use SD2011(c) - cleaned missing values, dropped generated variables, and $k = 2$



PERCENT FREQUENCY FOR SELECTED VARIABLES BY PARENTS

Figure 9: No missings if parents $<$ 2



SUMMARY

- General lessons
  - You have to ‘know’ your data (missings, negative values, etc.)
  - No need to replicate generated variables
- DataSynthesizer lessons for SD2011
  - Will only capture missing values if parents (k) $>= 2$
  - Better at capturing distribution of categorical variables than continuous variables
- Its the only SDG that incorporates ϵ DP as a setable hyperparameter

Section 3b): Know your generator (CTGAN)

GANs (Goodfellow et al., 2014), simultaneously train two NN models: a generative model which captures the data distribution, and a discriminative model that aims to determine whether a sample is from the model distribution or the data distribution.

The generative model starts off with noise as inputs and relies on feedback from the discriminative model to generate a data sample. This goes back and forth until the discriminator cannot distinguish between the actual data and the generated data.

Unlike DataSynthesizer, GANs were created to deal with continuous variables.

TUNING CTGAN: 'PRIMARY' HYPERPARAMETERS

- epochs = Number of times to train the GAN. Each new epoch can improve the model (default is 300).
- batch size = Number of samples to process in each step (default is 500)

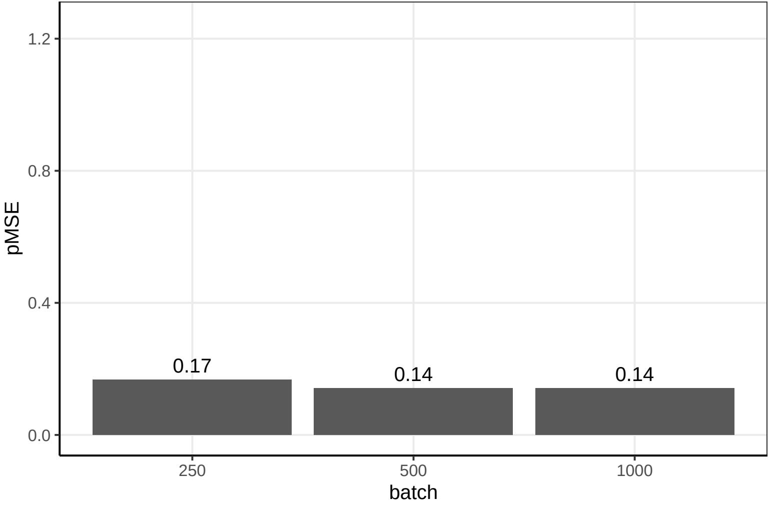
Table 1: Batch size and epochs = actual steps

N	Batch size	Steps per Epoch	Epochs	Actual Steps
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
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

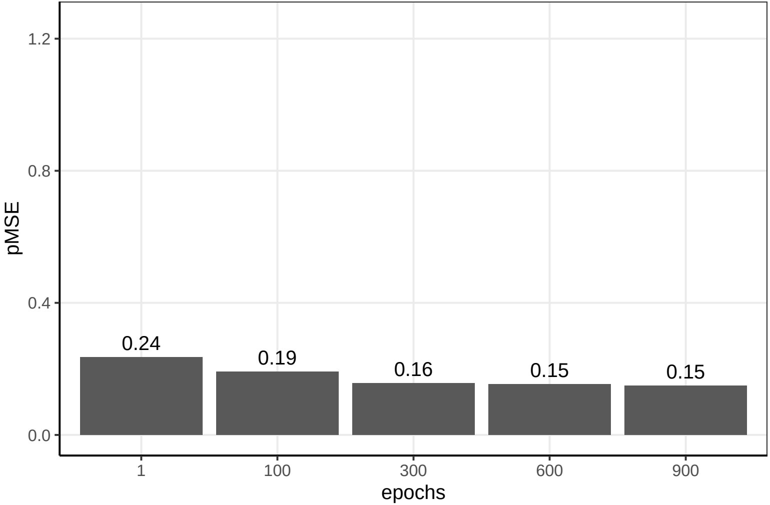
TUNING CTGAN: 'ADVANCED' HYPERPARAMETERS

- dimensionality - The number of layers in the generator/discriminator networks
  - 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 Residual Layer will be created for each one of the values provided. Defaults to (256, 256).
- embedding_dim (int): Size of the random sample passed to the Generator. Defaults to 128.
  - The embedding dimension essentially influences how much the information in the original data set is compressed

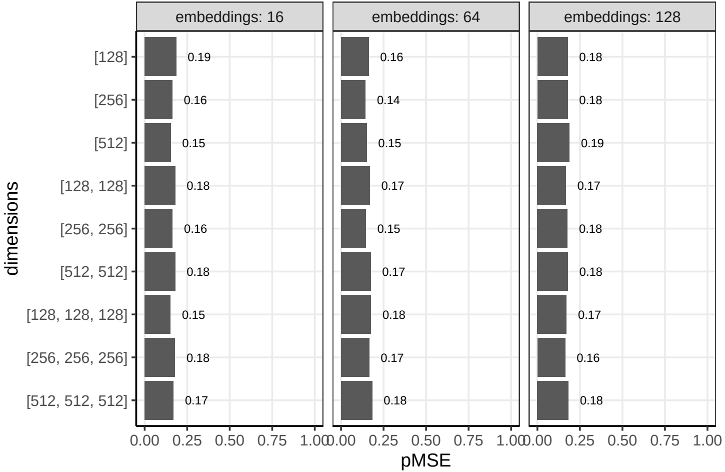
CTGAN: EFFECT OF BATCH SIZE (CONSTANT STEPS)



CTGAN: EFFECT OF EPOCHS (CONSTANT BATCH SIZE)

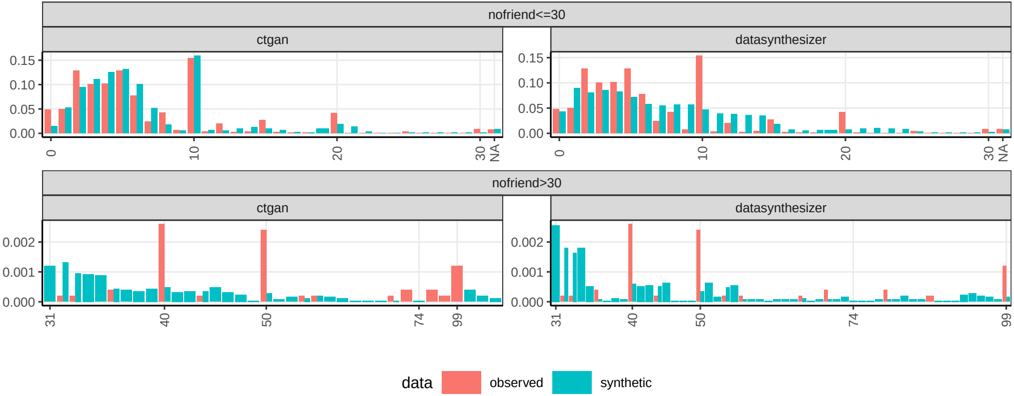


CTGAN: EFFECT OF DIMENSIONS



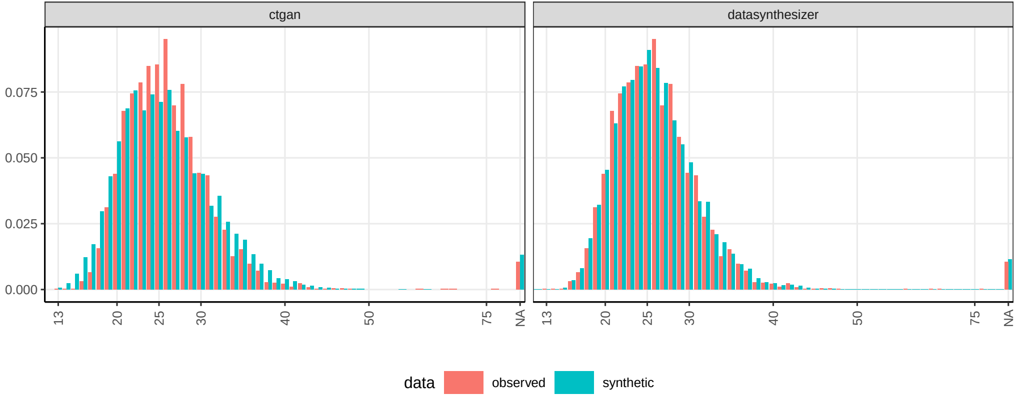
VARIABLE: NOFRIEND

Figure 10: CTGAN is better than DataSynthesizer below 30, but both are bad above 30



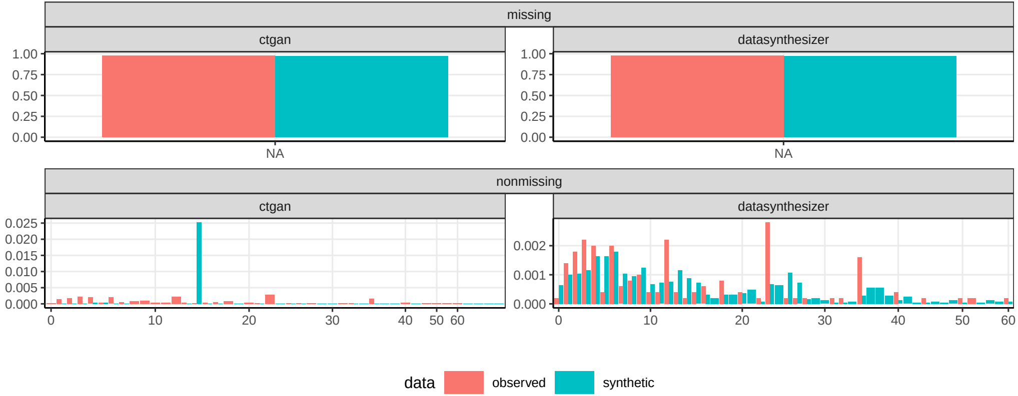
VARIABLE: BMI

Figure 11: CTGAN/DataSynthesizer estimate the median, but CTGAN is skewed a bit more to the right



VARIABLE: WKABDUR (WORK ABROAD DURATION)

Figure 12: CTGAN does not correctly estimate the distribution, DataSynthesizer gets the median (10), but not the rounding



SUMMARY

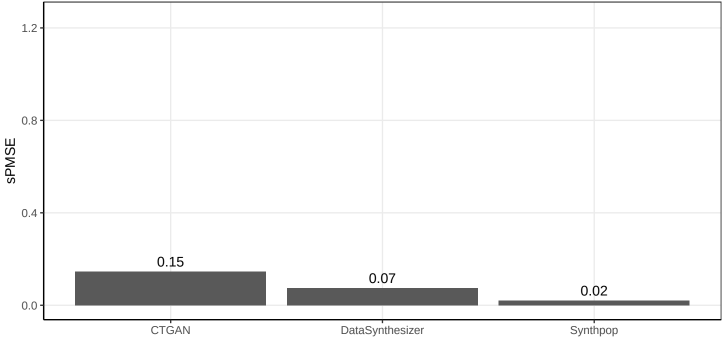
- CTGAN is not a good SDG
- But, CTGAN is not the only GAN
- Distinguish between the package and the synthesizer
- Can we make a better GAN? Yes, we can . . .

Section 3c): Know your generator (Synthpop)

Synthpop, R package, uses methods based on classification and regression trees (CART, developed by Breiman et al. (1984)), which can handle mixed data types and is non-parametric. Synthpop synthesises the data sequentially, one variable at a time; the first is sampled, then the following are predicted using CART (in the default mode) with the previous variables used as predictors. This means that the order of variables is important (and can be set by the user).

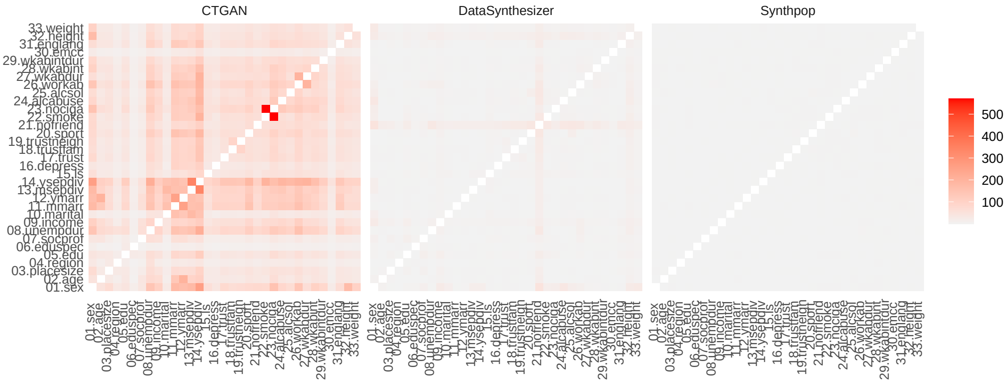
SD2011 - PMSE

Figure 13



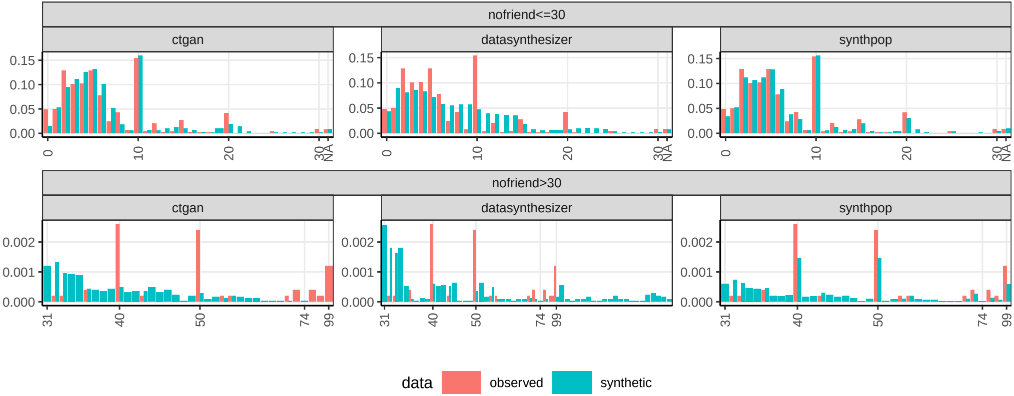
TWO-WAY UTILITY: PMSE FOR PAIRS OF VARIABLES

Figure 14



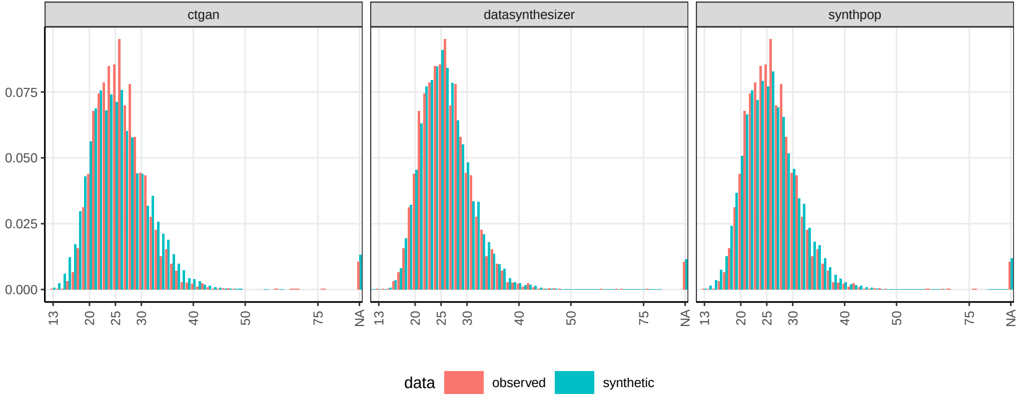
VARIABLE: NOFRIEND

Figure 15: Synthpop captures the distribution



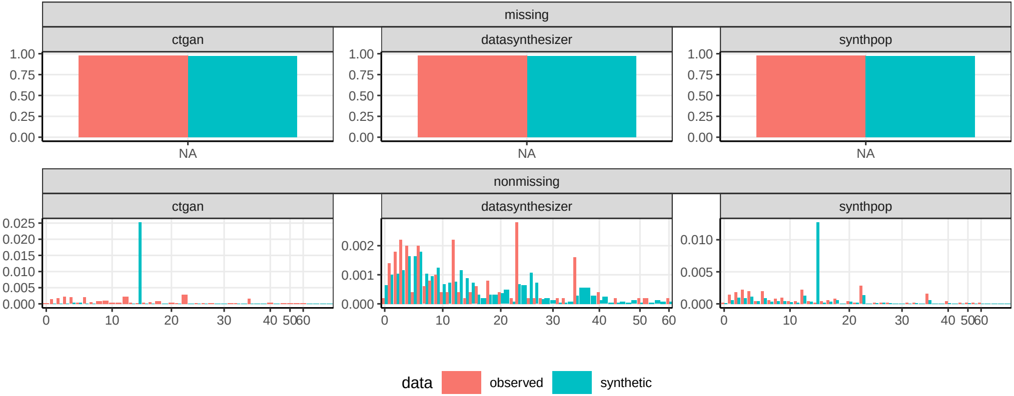
VARIABLE: BMI

Figure 16: DataSynthesizer is similar to Synthpop



VARIABLE: WKABDUR (WORK ABROAD DURATION)

Figure 17: Like CTGAN, Synthpop is higher than median (10), but is better with the distribution than CTGAN



EFFICIENCY - DURATION IN SECONDS

version	description	ctgan	datasynthesizer	synthpop (csv)	synthpop (package)
v00	Raw (SD2011)	331.01	245.37	2132.12	5474.39
v01	Without eduspec or wkabdur	290.30	264.43	10.99	8.45
v02	Without wkabdur	337.07	351.76	13.96	11.02
v03	Without eduspec	306.46	351.24	11.39	8.92
v04	Last variables: eduspec-wkabdur	374.57	344.02	14.23	287.85
v05	Last variables: wkabdur-eduspec	419.60	339.92	14.60	3657.55
v06	as.numeric(wkabdur) and last variable: eduspec	356.02	347.36	14.12	11.05
v08_1_20	+ 1 factor variable (20 values)	339.05	264.96	42.23	
v08_1_25	+ 1 factor variable (25 values)	400.28	326.84	137.47	
v08_1_30	+ 1 factor variable (30 values)	339.73	269.72	363.18	
v08_2_20	+ 2 factor variable (20 values)	369.74	339.45	74.96	
v08_2_25	+ 2 factor variable (25 values)	364.56	361.81	631.43	
v08_2_30	+ 2 factor variable (30 values)	373.25	346.15	1222.54	
v08_3_20	+ 3 factor variable (20 values)	393.99	369.58	122.77	
v08_3_25	+ 3 factor variable (25 values)	401.03	383.40	881.53	
v08_3_30	+ 3 factor variable (30 values)	394.44	424.64	3654.59	

SUMMARY

- Advantages
  - Synthpop is an excellent SDG
  - Much better than CTGAN/DataSynthesizer
- Disadvantages
  - Questions about privacy (not addressed here)
  - Issues with high dimensional data

Section 4: Conclusion

RESULTS

- Know the data
  - Cleaning/preprocessing are important
  -
- Know your synthesizer
  - Synthpop is the best SDG, but struggles with high dimensional data
  - DataSynthesizer is the only one to offer a hyperparameter for privacy
  - CTGAN is bad, but maybe there is a better GAN