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

AnigeD

Finanziert von der Europäischen Union
NextGenerationEU

Bundesministerium für Bildung und Forschung

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

PSD2024: Privacy in Statistical Databases 2024,
26. September, 2024

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

SECTION 1: OVERVIEW

- Common perception that making synthetic data is easy
- We want to show that its complicated
  - You need to know your data
  - You need to know your synthetic data generator (SDG)
  - The problem: Individuals who know the data may not be the same as those who know the SDG
- Conclusion
  - On the one hand, generating synthetic data is easy
  - On the other hand, users need to be aware of challenges
  - The point: One should be skeptical about automated, one-size-fits-all processes for generating synthetic data

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 (Ping et al., 2017): 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

- We must ask:
  - Why are we creating synthetic data? (generating code, education, the public, etc.)
    - How different should it be?
    - How do we measure the difference?
  - How does the synthesizer work?
    - Variables types (categorical, continuous)
    - Variable values (missing, extreme, and ‘spikey’ values, etc.)
    - How long it takes to generate synthetic data from original data (i.e. ‘computational efficiency’)
- Answering these questions will help us make decisions about which synthesizer is the right choice

OUR GOAL IS TO ILLUSTRATE THE CHALLENGES

- Know your data (1 dataset)
  - Social Diagnosis 2011 (SD2011)
- Know your generator
  - Examine 3 synthetic data generators (SDGs): DataSynthesizer, CTGAN, Synthpop
- 3 utility measures
  - Compare univariate frequency – difference/similarity in one variable
  - Propensity score mean – difference/similarity in the dataset (lower values better)
  - Computationally efficiency – duration in time required to generate synthetic data (lower values better)

SECTION 2: KNOW YOUR DATA (SD2011)

- Social Diagnosis 2011 (SD2011)
  - Loads with Synthpop
- Like real data, has messy variables with unusual values
  - Missing values
  - ‘Errors’
  - Generated variables (Can be problematic for SDGs)
  - ‘Spikey’ or discontinuous values

DATA (SD2011)

Table 1

Number	Variable	Description	Type	Observations	Unique.Values	Missings	Negative.values	Generated	Messy
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
4	placesize	Category of the place of residence	factor	5000	6	0	0		
5	region	Region (voivodeship)	factor	5000	16	0	0		
6	edu	Highest educational qualification, 2011	factor	5000	5	7	0		
7	eduspec	Discipline of completed qualification	factor	5000	28	20	0		
				…					
10	income	Personal monthly net income	numeric	5000	407	683	603		Yes
11	marital	Marital status	factor	5000	7	9	0		
12	mmarr	Month of marriage	numeric	5000	13	1350	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		Yes
23	smoke	Smoking cigarettes	factor	5000	3	10	0		
24	nociga	Number of cigarettes smoked per day	numeric	5000	30	0	3737		Yes
				…					
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 - kg/(height - cm^2)*10000)	numeric	5000	1396	59	0	Yes	Yes

SECTION 3: KNOW YOUR GENERATOR

Use 3 SDGs to illustrate the challenges with generating synthetic data from real data

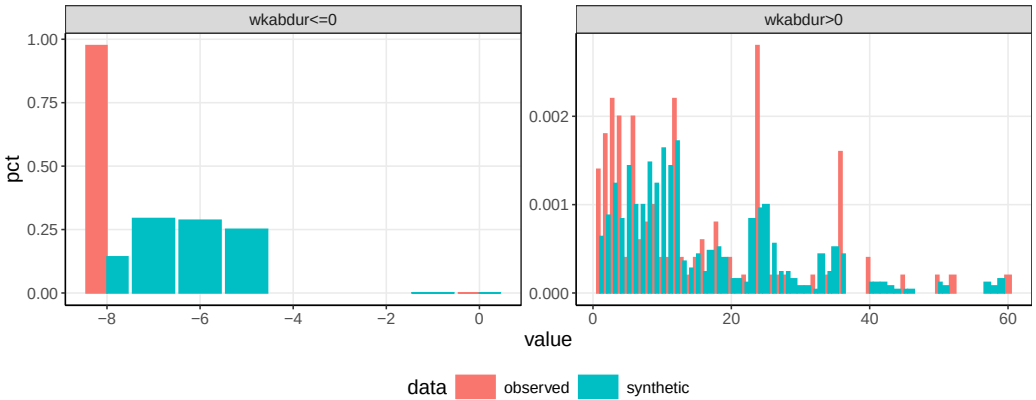
- DataSynthesizer - Bayesian network
- CTGAN - GAN
- Synthpop - CART

SECTION 3a): KNOW YOUR GENERATOR (DATASYNTHESIZER)

- Hyperparameters
  - ϵ Differential Privacy (DP): we turn it off (default 0.1)
  - k-degree Bayesian network (parents): 2 (default is ‘greedy’)
- Utility measure - compare univariate frequency between original and synthetic data

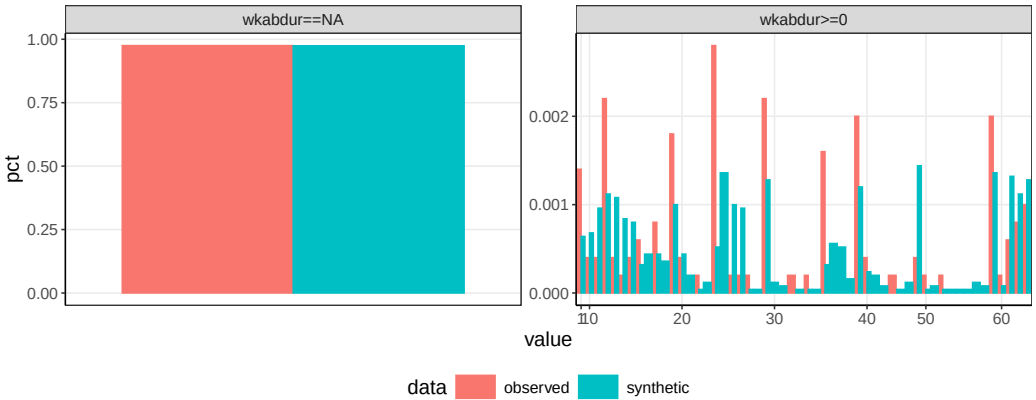
NEGATIVE VALUES (I.E. MISSING) AND ‘SPIKEY’ VALUES

Figure 1: Work abroad duration - in weeks (wkabdur)



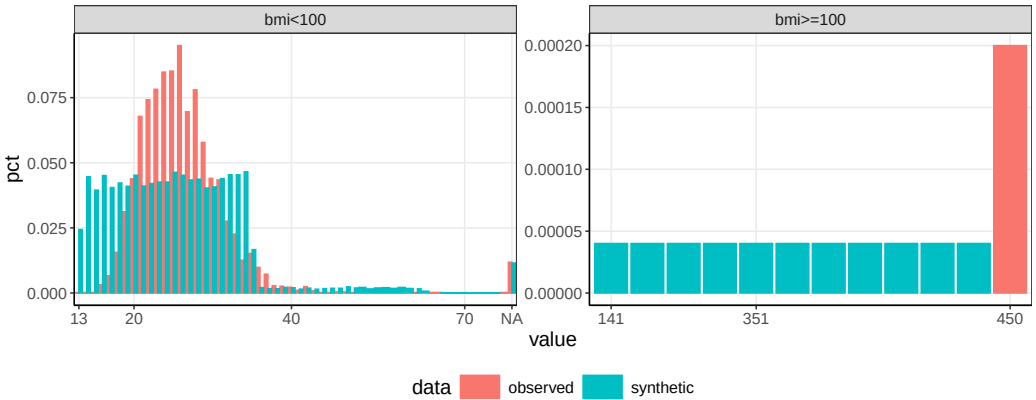
SOLUTION: PREPROCESS NEGATIVE VALUES AS MISSING

Figure 2: Work abroad duration - in weeks (wkabdur)



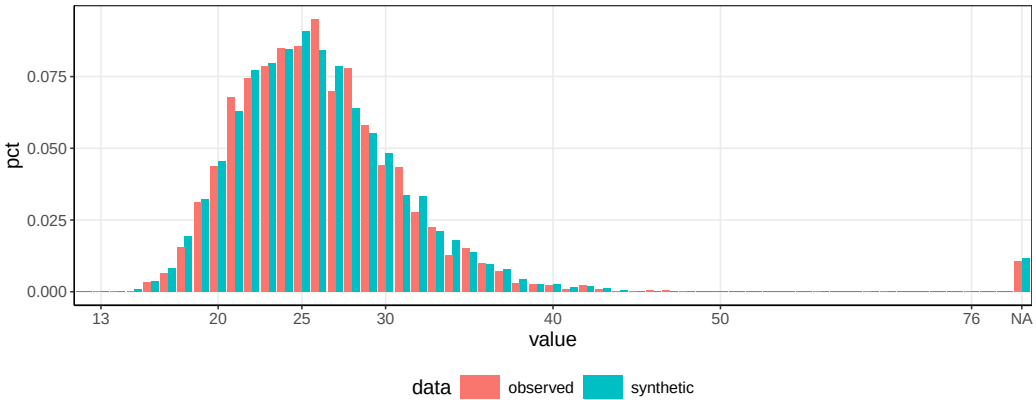
GENERATED VARIABLES AND EXTREME VALUES

Figure 3: Body mass index (bmi)



SOLUTION: DROP GENERATED VARIABLES BEFORE SENDING TO SDG

Figure 4: Regenerate generated values from synthetic values (bmi)



SUMMARY

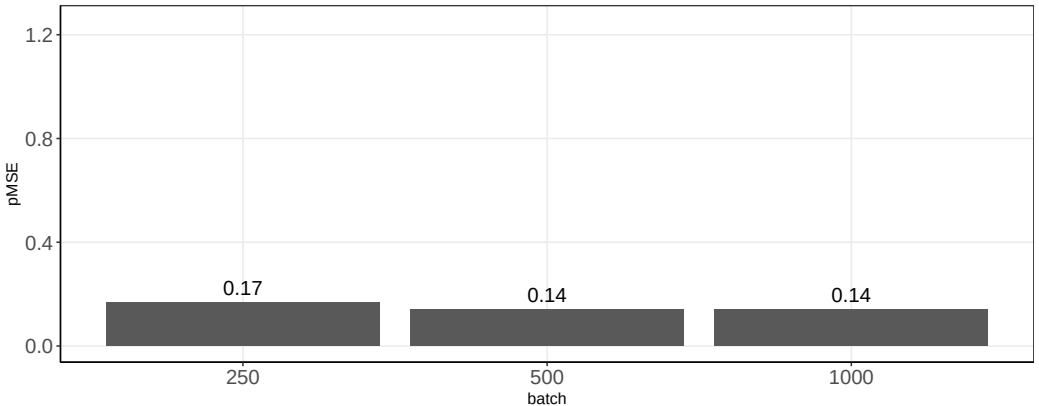
- The problem: there are consequences of generating random values within a discretized bin
- The solution: clean or preprocess the data (code negative values as missing, drop generated variables, etc.)
  - But what about ‘spikey’ or discontinuous values?
  - More generally, cleaning requires knowledge of the data

SECTION 3b): KNOW YOUR GENERATOR (CTGAN)

- Hyperparameters
  - Batch size: is the sample size (or rows of data) the GAN looks at at once. (default is 500)
  - Epochs: Number of times the GAN gets to see the full dataset (default is 300).
  - Discriminator/Generator dimensions: the number of layers and neurons per layer. (Default is 2 layers, each with 256 neurons)
  - Embeddings dimension: the vector size of the compressed representation of the data sent to the generator. (Default 128)
- Utility (pMSE) - similarity or difference between original and synthetic data (lower values are better)

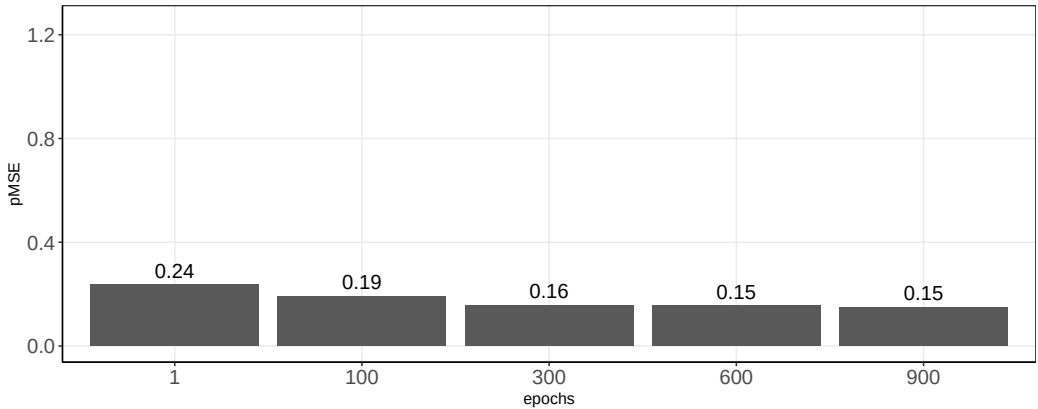
VARY BATCH SIZE

Figure 5: DataSynthesizer (0.07) and Synthpop (0.02)



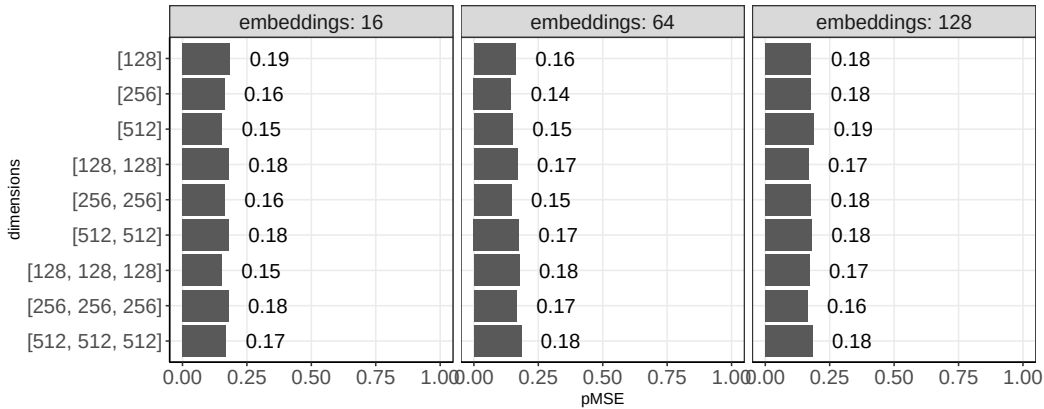
VARY EPOCHS

Figure 6: DataSynthesizer (0.07) and Synthpop (0.02)



OPTIMIZE DIMENSIONS

Figure 7: DataSynthesizer (0.07) and Synthpop (0.02)



SUMMARY

- The problem: CTGAN produces synthetic data with lower utility than DataSynthesizer or Synthpop
- Possible explanations
  - GANs are better for high dimensional data (i.e. images). Our data are low dimensional (33 columns, 5000 rows).
  - Other GANs might be better. CTGAN is not the only GAN. Distinguish between the package and the method.

SECTION 3c): KNOW YOUR GENERATOR (SYNTHPOP)

- Hyperparameters (default)
- Utility measure - Computational efficiency

COMPUTATIONAL EFFICIENCY

Table 2

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
v07_1_20	+ 1 factor variable (20 values)	339.05	264.96	42.23	
v07_1_25	+ 1 factor variable (25 values)	400.28	326.84	137.47	
v07_1_30	+ 1 factor variable (30 values)	339.73	269.72	363.18	
v07_2_20	+ 2 factor variable (20 values)	369.74	339.45	74.96	
v07_2_25	+ 2 factor variable (25 values)	364.56	361.81	631.43	
v07_2_30	+ 2 factor variable (30 values)	373.25	346.15	1222.54	
v07_3_20	+ 3 factor variable (20 values)	393.99	369.58	122.77	
v07_3_25	+ 3 factor variable (25 values)	401.03	383.40	881.53	
v07_3_30	+ 3 factor variable (30 values)	394.44	424.64	3654.59	

SUMMARY

- The problem: categorical variables with many values increases the run-time of CART based synthesizers.
- The solution: move categorical variables with many values to the end
  - But what about Census data with 3-digit occupation, industry, and country codes?

SECTION 4: CONCLUSION

- Contribution: applying SDGs to real data reveal problems not often understood when using clean data
- You need to know your data.
  - Preprocessing or cleaning data is important
  - This requires knowledge of the data
- You need to know your synthesizer
  - DataSynthesizer: Bayesian network model generates random values within discretized bin, but has a hyperparamter for differential privacy
  - CTGAN: GAN architecture may not produce synthetic data with high levels of utility, but can synthesize data that others cannot (i.e. images)
  - Synthpop: CART struggles with computational efficiency on datasets that contain variables with many categories, but produces synthetic data with high utility
- The point:
  - You need to match the right synthesizer with the right data problem (no one-size-fits-all solution)

CONTACT

Jonathan Latner

jonathan.latner@iab.de

Reproducible code:

- Github: https://github.com/jonlatner/KEM_GAN/tree/main/latner/projects/comparison
- Harvard Dataverse: coming soon