Marcel Neunhoeffer

Generative Adversarial Nets for Social Scientists

Mannheim, 2023

Marcel Neunhoeffer

Generative Adversarial Nets for Social Scientists

Inaugural dissertation submitted in partial fulfillment of the requirements for the degree Doctor of Social Sciences
in the Graduate School of Economic and Social Sciences
at the University of Mannheim

written by
Marcel Neunhoeffer

Mannheim, 2023

Dean: Prof. Dr. Michael Diehl

First Supervisor: Prof. Thomas Gschwend, Ph.D.
Second Supervisor: Prof. Dr. Frauke Kreuter

First Examiner: Prof. Thomas Gschwend, Ph.D.
Second Examiner: Prof. Dr. Frauke Kreuter
Third Examiner: Prof. Dr. Richard Traunmüller

Date of oral examination: 31st March, 2023

Contents

1

Introduction

On July 28, 2016, Yann LeCun—Deep Learning pioneer and Chief Artificial Intelligence Scientist at Facebook—was asked, "What are some recent and potentially upcoming breakthroughs in deep learning?" in a Public Quora Q&A session.[1] He answered: "The most important one, in my opinion, is adversarial training (also called GAN for Generative Adversarial Networks). This is an idea that was originally proposed by Ian Goodfellow when he was a student with Yoshua Bengio at the University of Montreal […].

This, and the variations that are now being proposed is the most interesting idea in the last 10 years in ML [machine learning, the author], in my opinion."

So what is a Generative Adversarial Net? As indicated in the quote above,

[1] As of February 26, 2023, you can find the session here: https://quorasessionwithyannlecun.quora.com.

GANs were first introduced by Goodfellow et al. (2014). In short, a GAN is a likelihood-free method that tries to learn an arbitrary joint distribution by comparison. It samples data from a proposal distribution and compares them to samples from the true distribution. Comparing the difference between the two samples—samples from the proposal distribution and samples from the true distribution—it improves the model for the proposal distribution such that it generates more and more realistic samples. The basic idea of a GAN is surprisingly intuitive. At its core, a GAN is a minimax game with two competing actors—a discriminator trying to tell real from synthetic samples and a generator to produce realistic synthetic samples from random noise.

Goodfellow et al. (2014) illustrate the idea with the following example: "The generative model can be thought of as analogous to a team of counterfeiters, trying to produce fake currency and use it without detection, while the discriminative model is analogous to the police, trying to detect the counterfeit currency. Competition in this game drives both teams to improve their methods until the counterfeits are indistinguishable from the genuine articles" (p. 1).

In GANs, the team of counterfeiters, the generator, is a neural network trained to produce realistic synthetic data examples from random noise. And the police, the discriminator, is a neural network to classify fake and real data. The generator network is trained to fool the discriminator network and uses the feedback of the discriminator to generate increasingly realistic simulated data that should eventually be indistinguishable from the real data. At the same time, the discriminator is constantly adapting to the improving generator. Thus, the threshold where the discriminator is fooled increases along with the faking capabilities of the generator. This goes on until (in theory) an equilibrium is reached. At the equilibrium, the discriminator can no longer distinguish between real and

fake samples. The generator can achieve this goal if it samples from the underlying distribution of the real data.

The promise of drawing additional and new samples from an underlying data distribution without explicitly modeling it makes GANs so appealing. Computer scientists achieved impressive results generating images with GANs, but the idea is also very interesting for data beyond images. In this dissertation, I focus on the introduction of Generative Adversarial Nets (GANs) for Social Scientists and explore the potential of GANs in social science applications. The main goal of this work is to make GANs more accessible for researchers in the social sciences by providing them with user-friendly tools and showing applications of GANs to social science problems like data sharing with privacy-preserving synthetic data and multiple imputation of missing values.

In chapter 2, I present an R-package `RGAN` that facilitates the implementation of GANs and allows researchers to quickly generate synthetic data for their studies. In chapter 3, I address the issue of privacy and confidentiality by exploring how GANs can be used to create synthetic data that preserves the privacy of study participants with formal privacy guarantees. Finally, in chapter 4, I evaluate the use of GANs for multiple imputation, a method commonly used in social science research to handle missing data. While GANs might be a promising approach to impute missing data, current implementations do not meet the current state of the art in multiple imputation. I also show that the translation from computer science to social science applications requires care and that only some things that are hyped in one field are indeed an improvement over existing methods in another. However, a good understanding of GANs is necessary to assess these shortcomings.

1.1 GANs in computer science, media, and politics

GANs became a hot topic at the leading computer science conferences since they first were proposed by Goodfellow et al. (2014).[2] Panel (A) in Figure 1.1 shows this impressive volume of GAN-related research in computer science. In total 659 papers with the acronym "GAN" or the term "Generative Adversarial" in the title have been published at these conferences.[3] Of these papers, the vast majority use GANs to generate image or video data. This can also be seen from the sheer volume of GAN-related papers (299 out of 659) at the computer vision focused conferences CVPR and ICCV.

Unsurprisingly, this also means that the citations of the original GAN paper (Goodfellow et al., 2014) skyrocketed. As of February 26, 2023, the article was cited 55,641 times according to Google Scholar.[4] Panel (B) in Figure 1.1 shows the yearly citations of Goodfellow et al. (2014). As a comparison, the most cited paper in a political science journal also published in 2014 (as an online first article) is "Comparative Politics and the Synthetic Control Method" by Abadie et al. (2015), which was cited 2,208 times.

The current main application of GANs in academia is the generation of realistic-looking images. I show examples of a generated face and a generated cat from a StyleGAN2 implementation (Karras et al., 2020) in Figure 1.2.

GANs have also received significant media attention mainly due to their use

[2]According to the h5 index by Google Scholar accessed on February 26, 2023, on https://scholar.google.com/citations?view_op=top_venues&hl=en&vq=eng, the top five peer-reviewed conferences are the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), the International Conference on Learning Representations (ICLR), Neural Information Processing Systems (NeurIPS), the IEEE/CVF International Conference on Computer Vision (ICCV, bi-annual), and the International Conference on Machine Learning (ICML).

[3]In total 24,095 papers have been published at these conferences since 2014. This means that about 2.7% of all papers were directly GAN related.

[4]I obtained all citation data using the R package `scholar` (Yu et al., 2022).

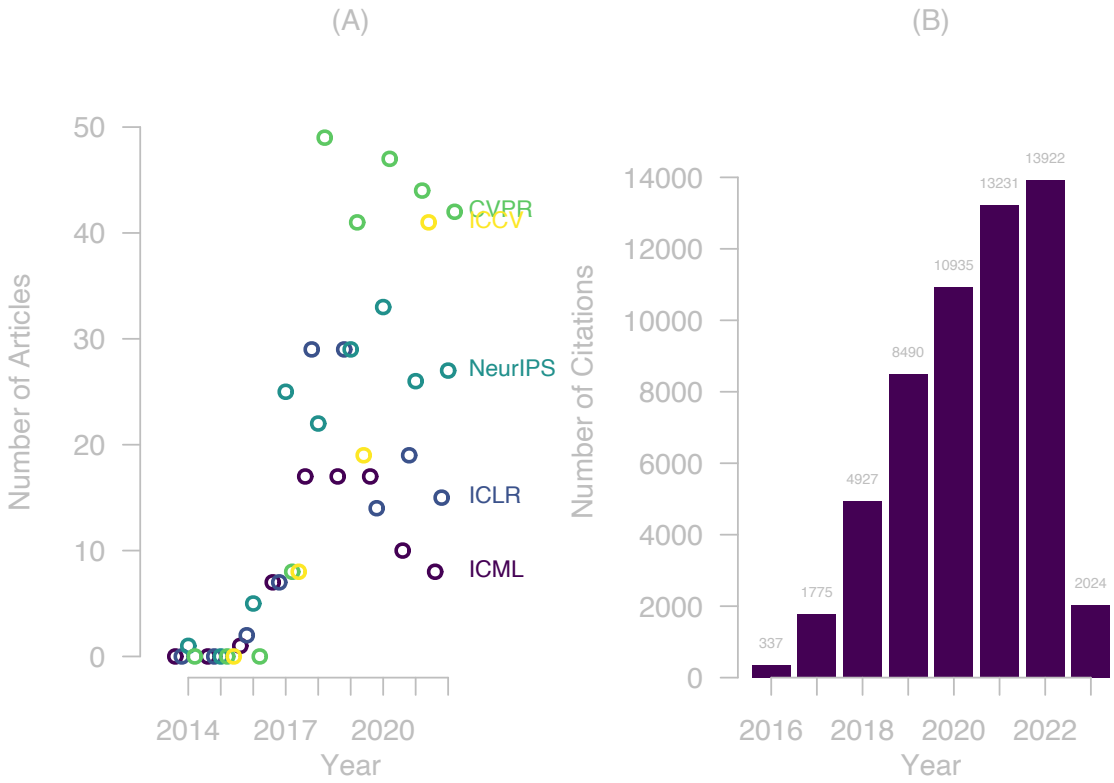

Figure 1.1: Panel (A): Number of Papers with the term "GAN" or "Generative Adversarial" in the title at leading computer science conferences. Panel (B): Number of Citations per year of "Generative Adversarial Networks" by Goodfellow et al. (2014).

Figure 1.2: Images produced by a StyleGAN2 (Karras et al., 2020).

in creating realistic-looking fake images. These images are sometimes called deepfakes. Deepfakes are realistic videos or images that are manipulated using GANs to swap the face of one person onto the body of another.[5] While these applications have raised concerns about the potential misuse of GANs for disinformation and propaganda, there have also been positive uses of GANs that received widespread media coverage. For example, in 2018, the French artist collective obvious[6] created a GAN-generated portrait of a fictional character called Edmond de Belamy. The print sold for $432,500 at Christie's auction house, making it the first AI-generated artwork to be sold at a major auction house.[7] This demonstrates the growing interest and recognition of GANs in various fields and their potential for creative applications beyond computer science. However, it also highlights the need for ethical considerations and responsible use of GANs to prevent misuse and potential harm.

Figure 1.3: GAN-generated portrait of Edmond de Belamy. Image from https://commons.wikimedia.org/wiki/File:Edmond_de_Belamy.png.

[5] See, for example, https://www.vox.com/2018/4/18/17252410/jordan-peele-obama-deepfake-buzzfeed (last accessed February 26, 2023) or https://www.tagesschau.de/investigativ/wdr/deep-fakes-103.html (last accessed February 26, 2023).

[6] https://obvious-art.com/portfolio/edmond-de-belamy/

[7] See, for example, https://www.sueddeutsche.de/kultur/kunst-kunst-per-algorithmus-christie-s-versteigert-ki-gemaelde-dpa.urn-newsml-dpa-com-20090101-181025-99-518993 (last accessed February 26, 2023).

The so-called deepfakes and the responsible use of GANs are also of interest to policymakers. For example, the 116th U.S. Congress passed the "IOGAN Act"[8] in 2020. This act aims to support research to identify the output of GANs. In particular concerning fake images. Similarly, the final report of the Enquete commission on artificial intelligence of the 19th German Bundestag[9], also published in 2020, explicitly mentions Generative Adversarial Nets in the context of deepfakes.

1.2 GANs for Social Scientists

Despite their popularity in computer science, media, and attention from politics, there is still much to be explored regarding applying GANs in social science research. In fact, as of February 26, 2023, a search for publications in leading political science journals[10] returned no results.

Therefore, it is a natural question: *Why should Social Scientists learn about GANs?* First, I am convinced that it is valuable in itself to understand new technology and make it accessible to a broader audience. Second, while GANs have gained popularity for their ability to generate realistic images and videos, they also have great potential for generating synthetic data beyond visual content. For example, GANs can be used to create synthetic data sets that mimic real-world data with the promise to preserve privacy and confidentiality (Beaulieu-Jones et al., 2019; Yoon et al., 2019; Xie et al., 2018) or to generate

[8] Available at https://www.congress.gov/bill/116th-congress/senate-bill/2904/text (last accessed February 26, 2023).

[9] The report is available in German at https://dserver.bundestag.de/btd/19/237/1923700.pdf (last accessed February 26, 2023).

[10] According to the h5 index by Google Scholar accessed on February 26, 2023, on https://scholar.google.com/citations?view_op=top_venues&hl=en&vq=soc_politicalscience, the top five peer-reviewed political science journals are the American Journal of Political Science (AJPS), the American Political Science Review (APSR), the Journal of European Public Policy, the Journal of Politics (JoP) and the British Journal of Political Science (BJPS).

more realistic synthetic data for Monte Carlo studies (Athey et al., 2021). Additionally, GANs can be used for multiple imputation (Yoon et al., 2018; Yoon and Sull, 2020), a statistical technique commonly used in social science research to handle missing data. Overall, GANs have significant potential for generating synthetic data in various domains beyond visual content, which can contribute to advancing social science research and innovation.

1.3 Contributions of this Dissertation

This dissertation contains three papers. In the first paper, "An Introduction to Generative Adversarial Nets in `R`—The `RGAN` package," I introduce GANs to Social Scientists and make the technology accessible to a broader audience by providing an accompanying R-package `RGAN`. I hope that this facilitates the development of new applications. At the same time, it helps social scientists to form an informed opinion on these new technologies. The `RGAN` package runs on top of the `torch` library in R, offering an easy entry point to many quantitive social scientists who are comfortable programming in R. The `RGAN` package is available on the Comprehensive R Archive Network (CRAN). As of February 26, 2023, the `RGAN` package has already been downloaded 2,415[11]. This shows an interest in applying GAN by scientists who work mainly in R. The package facilitates experimentation with different design choices for GANs—such as value functions, other network architectures, different noise distributions, the choice of optimization algorithms and update schemes, dropout during data generation, and the post-processing of generated GAN samples. Furthermore, advanced users can provide customized functions for each design choice.

In the second paper, "Private Post-GAN Boosting" (co-authored with Zhiwei

[11] An up-to-date number of downloads is available at https://cranlogs.r-pkg.org/badges/grand-total/RGAN.

Steven Wu and Cynthia Dwork) (Neunhoeffer et al., 2021), we show how to enhance the synthetic data generated from GANs while protecting the privacy of individuals in the original data with formal privacy guarantees[12]. The paper was published at the International Conference on Learning Representations (ICLR) 2021. Differentially private GANs have proven to be a promising approach for generating realistic synthetic data without compromising the privacy of individuals. Due to the privacy-protective noise introduced in training, the convergence of GANs becomes even more elusive, often leading to poor utility in the output generator at the end of training. We propose Private post-GAN boosting (Private PGB), a differentially private method that combines samples produced by the sequence of generators obtained during GAN training to create a high-quality synthetic data set. We evaluate Private PGB on two-dimensional toy data, MNIST images, US Census data, and a standard machine learning prediction task. Our experiments show that Private PGB improves upon a common private GAN approach across quality measures. We also provide a non-private variant of PGB that improves the data quality of traditional GAN training.

In the third paper, "A Common Benchmark to Evaluate Multiple Imputation Algorithms," I provide a novel benchmark for evaluating multiple imputation algorithms and explore the current potential and limitations of a GAN-based approach to multiple imputation. Multiple imputation (MI) of missing data is an essential tool for quantitative research in the social sciences. Many different MI models are available, and new MI models—e.g., GAN-based models—are introduced to the literature. However, how to evaluate new MI models or compare existing ones needs to be clarified. The R-package `MIBench` accompanying this

[12]Recent research by Carlini et al. (2023) on synthetic data highlights the importance of formal privacy guarantees. Synthetic data without formal privacy guarantees can directly leak training data.

paper makes applying the benchmark and comparing existing imputation models easy. I also provide benchmark results and show that no one-fits-all solution to handle missing data exists. It is striking that seemingly more flexible imputation methods, particularly the GAN-based method GAIN (Yoon et al., 2018), could perform better. The results highlight a need to carefully evaluate MI models before using them in applied research. This also shows that the translation of computer science models to social science applications is more complex than it sometimes might seem. Collaboration between the fields has the potential for new innovative solutions.

All three papers included in this dissertation present novel methods for using GANs in social science research but also come with open-source software that allows for easy implementation of the proposed methods and replication of the presented results. By providing open-source software, the papers ensure that the methods are accessible to a broader audience and that the findings are transparent and replicable. The software packages I provide include the `RGAN` R-package for easy access to GANs[13], replication code for privacy-preserving synthetic data methods[14], and GAN-based multiple imputation methods[15]. This allows researchers to quickly adopt and apply the proposed GAN-based methods in their research, contributing to the advancement of the field.

1.4 Future Research

I agree with Yann LeCun that GANs are an exciting idea. This dissertation is just the beginning of many exciting applications of GANs in social science re-

[13] `RGAN` available at https://cran.r-project.org/web/packages/RGAN/index.html (last accessed on February 26, 2023).

[14] The replication code for Private Post-GAN Boosting (Neunhoeffer et al., 2021) is available at https://github.com/mneunhoe/post-gan-boosting (last accessed on February 26, 2023).

[15] The multiple imputation benchmark including code for GAN-based multiple imputation is available at https://github.com/mneunhoe/MIBench (last accessed on February 26, 2023).

search. In the future, there are several directions that research on GANs in social science could take. One area of interest is to evaluate the statistical properties of synthetic data generated by GANs. Similar to the Multiple Imputation Benchmark, there is a need for methods to evaluate the utility of fully synthetic data. Additionally, improving privacy-preserving training of GANs is an important area of research, as it would enhance the confidentiality and utility of privacy-protected synthetic data. Moreover, exploring the potential of GANs for generating synthetic counterfactuals for statistical inference is a promising direction that can contribute to causal inference. Finally, there is a need for identifying and fingerprinting the output of GANs to ensure that synthetic data is used ethically and responsibly. Overall, the potential applications of GANs in social science research are vast, and further research in this area can lead to significant advancements in the field.

References

Abadie, Alberto, Alexis Diamond, and Jens Hainmueller. 2015. "Comparative Politics and the Synthetic Control Method." *American Journal of Political Science* 59 (2): 495–510. https://doi.org/10.1111/ajps.12116.

Athey, Susan, Guido W. Imbens, Jonas Metzger, and Evan Munro. 2021. "Using Wasserstein Generative Adversarial Networks for the Design of Monte Carlo Simulations." *Journal of Econometrics*. https://doi.org/10.1016/j.jeconom.2020.09.013.

Beaulieu-Jones, Brett K., Zhiwei Steven Wu, Chris Williams, Ran Lee, Sanjeev P. Bhavnani, James Brian Byrd, and Casey S. Greene. 2019. "Privacy-Preserving Generative Deep Neural Networks Support Clinical Data Sharing." *Circulation: Cardiovascular Quality and Outcomes* 12 (7): e005122. https://doi.org/10.1161/CIRCOUTCOMES.118.005122.

Carlini, Nicholas, Jamie Hayes, Milad Nasr, Matthew Jagielski, Vikash Sehwag, Florian Tramèr, Borja Balle, Daphne Ippolito, and Eric Wallace. 2023. "Extracting Training Data from Diffusion Models." arXiv. https://doi.org/10.48550/ARXIV.2301.13188.

Goodfellow, Ian, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. "Generative Adversarial Nets." *Advances in Neural Information Processing Systems 27*, 2672–80. https://doi.org/10.1017/CBO9781139058452.

Karras, Tero, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. 2020. "Analyzing and Improving the Image Quality of StyleGAN." In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Neunhoeffer, Marcel, Steven Wu, and Cynthia Dwork. 2021. "Private Post-

GAN Boosting." In *International Conference on Learning Representations*. https://openreview.net/forum?id=6isfR3JCbi.

Xie, Liyang, Kaixiang Lin, Shu Wang, Fei Wang, and Jiayu Zhou. 2018. "Differentially Private Generative Adversarial Network." *CoRR* abs/1802.06739. http://arxiv.org/abs/1802.06739.

Yoon, Jinsung, James Jordon, and Mihaela van der Schaar. 2018. "GAIN: Missing Data Imputation Using Generative Adversarial Nets." In *Proceedings of the 35th International Conference on Machine Learning*, edited by Jennifer Dy and Andreas Krause, 80:5689–98. Proceedings of Machine Learning Research. PMLR. https://proceedings.mlr.press/v80/yoon18a.html.

———. 2019. "PATE-GAN: Generating Synthetic Data with Differential Privacy Guarantees." In *International Conference on Learning Representations*. https://openreview.net/forum?id=S1zk9iRqF7.

Yoon, Seongwook, and Sanghoon Sull. 2020. "GAMIN: Generative Adversarial Multiple Imputation Network for Highly Missing Data." In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Yu, Guangchuang, James Keirstead, and Gregory Jefferis. 2022. *Scholar: Analyse Citation Data from Google Scholar*. https://github.com/YuLab-SMU/scholar.

2

An Introduction to Generative Adversarial Nets in `R` - The `RGAN` package

An Introduction to Generative Adversarial Nets in R—The RGAN package

Marcel Neunhoeffer
LMU München *and* Boston University

Abstract

This article introduces Generative Adversarial Nets, a powerful neural net architecture, and presents the **RGAN** package. **RGAN** makes it easy to implement Generative Adversarial Nets in R: It facilitates experimentation with different design choices for GANs—such as value functions, different network architectures, the choice of different noise distributions, the choice of optimization algorithms and update schemes, dropout during data generation and the post-processing of generated GAN samples. Furthermore, advanced users can provide customized functions for each design choice. **RGAN** is a lightweight package and runs on top of the **torch** library in R.

Keywords: Generative Adversarial Network, neural nets, R.

1. Introduction

On July 28 2016 Yann LeCun—Deep Learning pioneer and Chief Artificial Intelligence Scientist at Facebook—was asked "What are some recent and potentially upcoming breakthroughs in deep learning?" in a Public Quora Q&A session.[1] He answered: "The most important one, in my opinion, is adversarial training (also called GAN for Generative Adversarial Networks). This is an idea that was originally proposed by Ian Goodfellow when he was a student with Yoshua Bengio at the University of Montreal [...]).

This, and the variations that are now being proposed is the most interesting idea in the last 10 years in ML [machine learning, the author], in my opinion."

So what is a Generative Adversarial Network (GAN)? As indicated in the quote above, GANs were first introduced by Goodfellow, Pouget-Abadie, Mirza, Xu, Warde-Farley, Ozair, Courville, and Bengio (2014). They illustrate the idea with the following example: "The generative model can be thought of as analogous to a team of counterfeiters, trying to produce fake currency and use it without detection, while the discriminative model is analogous to the police, trying to detect the counterfeit currency. Competition in this game drives both teams to improve their methods until the counterfeits are indistinguishable from the genuine articles" (Goodfellow *et al.* 2014, p. 1).

In short, a GAN is a model that tries to learn an arbitrary joint distribution by comparison. It samples data from a proposal distribution and compares them to samples from the true distribution. Comparing the two samples—samples from the proposal distribution and

[1] As of March 30 2022 you can find the session here: https://www.quora.com/q/quorasessionwithyannlecun.

samples from the true distribution— improves the model for the proposal distribution such that it generates more and more realistic samples. The basic idea of a GAN is surprisingly intuitive. At its core, a GAN is a minimax game with two competing actors—a generator to produce realistic synthetic samples from random noise and a discriminator (or critic)[2] trying to tell real from synthetic samples.

In GANs, the team of counterfeiters, the generator, is a neural network that is trained to produce realistic synthetic data examples from random noise. And the police, the discriminator, is a neural network to classify fake and real data. The generator network is trained to fool the discriminator network and uses the feedback of the discriminator to generate increasingly realistic fake data that should eventually be indistinguishable from the real data. At the same time, the discriminator is constantly adapting to the improving generator. Thus, the threshold where the discriminator is fooled increases along with the faking capabilities of the generator. This goes on until (in theory) an equilibrium is reached. At the equilibrium, the discriminator cannot tell anymore what the real and the fake samples are. The generator can achieve this goal by sampling from the underlying distribution of the real data.[3]

With the **RGAN** package, I bring these neural networks to R. The goal of the package is to facilitate experimentation with GANs for an audience mostly working with R. This way, GANs can be included in existing workflows, for example for data synthesising.

RGAN is a lightweight package that only relies on **torch** (Falbel and Luraschi 2022). This makes **RGAN** fast since there is no reliance on calling python through e.g., **reticulate** (Ushey, Allaire, and Tang 2022). For applications, the focus of **RGAN** is on tabular data, yet it is easy to implement GANs for image data. Both use cases are demonstrated in section 4.

In python, software libraries in like **CTGAN** (Xu, Skoularidou, Cuesta-Infante, and Veeramachaneni 2019), **TF-GAN** that is part of **tensorflow** (Abadi, Agarwal, Barham, Brevdo, Chen, Citro, Corrado, Davis, Dean, Devin, Ghemawat, Goodfellow, Harp, Irving, Isard, Jia, Jozefowicz, Kaiser, Kudlur, Levenberg, Mané, Monga, Moore, Murray, Olah, Schuster, Shlens, Steiner, Sutskever, Talwar, Tucker, Vanhoucke, Vasudevan, Viégas, Vinyals, Warden, Wattenberg, Wicke, Yu, and Zheng 2015), **torchgan** (Pal and Das 2021) or **mimicry** (Lee and Town 2020) can serve a similar purpose as **RGAN**. Most of these libraries focus on image data.

In R, **ganGenerativeData** (Müller 2022) makes it possible to synthesize tabular data with a GAN through **reticulate** and **tensorflow**. The focus of **ganGenerativeData** is on producing synthetic data with one specific GAN architecture. This contrasts to **RGAN** that makes it easy to customize GANs for experimentation.

The rest of the paper is organized as follows: In section 2, I give a more detailed introduction to GANs. In section 3, I introduce the different design decisions a researcher faces when developing a GAN and show how **RGAN** facilitates experimentation with these design choices. Section 4 offers two working examples with code: The example in section 4.1 shows how to quickly get started with the synthesization of tabular data. The example in section 4.2 generates fake images with a customized neural network architecture in **RGAN**. Section 5 provides an outlook on potential extensions of **RGAN**.

[2]More on the difference between a discriminator and a critic can be found in section 3.

[3]A GAN is, therefore, a dynamic system where the optimization process is seeking not a minimum (or maximum) but an equilibrium.

2. A Brief Introduction to GANs

A GAN is a model that tries to learn an arbitrary joint distribution by comparison. It samples data from a proposal distribution and compares them to samples from the true distribution. Comparing the difference between the two samples it improves the model for the proposal distribution such that it generates more realistic samples.

In GANs the proposal distribution (fake data) is typically generated by a deep neural network—the generator (G). The comparison of fake and real data is done by a second deep neural network—the discriminator (D). The generator is trained to produce realistic synthetic data examples from random noise. At the same time, the discriminator has the goal to correctly distinguish fake from real data[4].

The generator is trained to be able to fool the discriminator network and uses the feedback of the discriminator to generate increasingly realistic fake data that should eventually be indistinguishable from the real data. At the same time, the discriminator is constantly adapting to the improving generating abilities of the generator. Thus, the threshold where the discriminator is fooled increases along with the faking capabilities of the generator. This goes on until (in theory) an equilibrium is reached. In the classical GAN, described by the value function V in equation 1, an optimal discriminator at the equilibrium would be assigning 0.5 probability to both real and fake samples. This is the point where the discriminator cannot distinguish between real and fake samples anymore.

GANs turn a typical unsupervised learning task—learning a joint density—into a supervised learning problem—learning to distinguish between fake and real data. Using the observation that it is easier to sample from p than to explicitly learn the distribution.

Formally, this two-player minimax game can be written as:

$$\min_G \max_D V(D,G) = E_{x\sim p_{\text{data}}(x)}\Big[\log D(x)\Big] + E_{z\sim p_z(z)}\Big[\log(1-D(G(z)))\Big] \tag{1}$$

where $p_{data}(x)$ is the joint distribution of the real data, x is a sample from $p_{data}(x)$. The generator network $G(z)$ takes as an input z from $p_z(z)$, where z is a random sample from a probability distribution $p_z(z)$. This sample is also called a noise sample. Usually, GANs are set up to either sample z from uniform or Gaussian distributions.[5] Passing the noise sample z through G then generates a sample of synthetic data which is then fed into the discriminator D. The discriminator takes as input a set of labeled data, either real examples from $p_{data}(x)$ or generated examples from $G(z)$, and is trained to distinguish between real data and fake data. In the original GAN setup (Goodfellow *et al.* 2014) this is a standard binary classification problem.[6]

D is trained to maximize the probability of assigning the correct label to training examples and samples from $G(z)$. G is trained to minimize $\log(1-D(G(z)))$. Thus, the goal of the discriminator is to maximize function V, whereas the goal of the generator is to minimize it.

[4] It wants to discriminate between fake and real data, hence the name. However, novel architectures don't necessarily use classifiers to compare fake and real data. In these GANs the second network is usually named a critic network.

[5] Recent research suggests that depending on the application other prior distributions might be preferable. Huster, Cohen, Lin, Chan, Kamhoua, Leslie, Chiang, and Sekar (2021) show that for skewed and heavy-tailed real data, a Pareto distribution performs better than a normal distribution.

[6] Thus, the output layer of the discriminator uses a sigmoid function as the activation function and the standard binary cross-entropy loss can be used.

In practice, this is achieved by iteratively updating the two networks, holding the weights of the other network constant.

The equilibrium point for a GAN is achieved when G achieves to produce samples that come from the true underlying data distribution $p_{data}(x)$ and D is uncertain about the origin of the samples.

3. Designing a GAN

So far many different GAN architectures have been proposed. Many of these GAN models are named—e.g. WGAN (Arjovsky, Chintala, and Bottou 2017), WGAN-GP (Gulrajani, Ahmed, Arjovsky, Dumoulin, and Courville 2017), KL-WGAN (Song and Ermon 2020), DCGAN (Radford, Metz, and Chintala 2016), CGAN (Mirza and Osindero 2014), CTGAN (Xu *et al.* 2019), StyleGAN (Karras, Laine, and Aila 2019), CycleGAN (Zhu, Park, Isola, and Efros 2017) or DiscoGAN (Kim, Cha, Kim, Lee, and Kim 2017) to name a few. Typically, these new GAN models address some weaknesses in prior models, where most of the development focuses on generating more and more realistic images.

In this section, I give a brief overview of some of these design decisions researchers face when developing a GAN for their own application. Some of these design decisions are very similar to those when designing neural network models (or other machine learning models).[7]

Other decisions are unique to the set up of GAN models.[8] In general, the interaction of all these design choices is not yet well understood and is an area of active research.

In the following, I provide a brief description of the design choices to be made, how **RGAN** implements default options, and how **RGAN** can be used to customize all these design choices to facilitate experimentation.

3.1. Choosing a value function

Description of the design choices. An important design decision is the choice of the value function V. In the original GAN by Goodfellow *et al.* (2014) and described by the value function in equation 1, the value function was motivated by turning the unsupervised learning task into a binary classification problem. After all the original GAN value function is just binary-cross entropy loss[9].

While this is an intuitive formulation it turned out that if the discriminator is very good at distinguishing between real and fake data it is very hard to learn (i.e. update the weights in the networks) due to vanishing gradients. This means that the gradient values are close to 0 and therefore cannot provide meaningful information to update the weights during backpropagation.

To address this weakness Arjovsky *et al.* (2017) proposed an alternative value function based on the Wasserstein distance. They use the observation that the original GAN value function essentially minimizes the Jensen-Shannon divergence between encoded fake and real data and

[7] For example, data pre-processing, the choice of the network architecture(s), or the choice of an optimizer with its related hyperparameters.

[8] For example, the choice of the noise distribution $p(z)$ or the GAN value function.

[9] Note that this is the same as the negative log-likelihood of the Bernoulli distribution.

note that using the Wasserstein distance instead has some desirable properties. This leads to the following value function:

$$\min_G \max_D V(D,G)_{||D||_L \leq 1} = E_{x \sim p_{\text{data}}(x)}\Big[D(x)\Big] + E_{z \sim p_z(z)}\Big[(1 - D(G(z)))\Big] \tag{2}$$

This is the value function for the so-called Wasserstein GAN or short WGAN as defined in (Arjovsky *et al.* 2017). Note that for the Wasserstein value function, the discriminator needs to be a Lipschitz continuous function with a Lipschitz constant ≤ 1. In practice, this is achieved by clipping the weights of the discriminator to a pre-specified range e.g. $[-0.01, 0.01]$.

More recent work by Song and Ermon (2020) shows that f-divergences and Wasserstein GANs can be bridged. They propose a novel GAN value function that leads to better empirical results on many tasks.

A more general formulation for the value function is therefore given by

$$\min_G \max_D V(D,G) = E_{x \sim p_{\text{data}}(x)}\Big[f_1(D(x))\Big] + E_{z \sim p_z(z)}\Big[f_2(1 - D(G(z)))\Big] \tag{3}$$

Where in the original GAN $f_1(a) = f_2(a) = \log(a)$, in the Wasserstein GAN $f_1(a) = f_2(a) = a$ and for the GAN value function proposed by Song and Ermon (2020) $f_1(a) = a$ and $f_2(a) = wa$, where w is a weight for the discriminator scores on fake examples, calculated as $w = e^a / E\Big[e^a\Big]$.

Implementation of default design choices RGAN. In **RGAN** these three value functions are readily implemented (`original`, `wasserstein` and `f-wgan`) and can be chosen as an option e.g. `gan_trainer(..., value_function = "wasserstein")`, the default value function is `"original"`.

Further customization with RGAN. Users can also provide custom value functions to work with the **RGAN** `gan_trainer()` function. The only requirements are that the value function takes as inputs the discriminator scores of real and fake data, works on and with **torch** data types and outputs a named list with the entries `d_loss` and `g_loss`. For example, this is what the original GAN value function looks like in code:

```
GAN_value_fct <- function(real_scores, fake_scores) {
  d_loss <-
    torch::torch_log(real_scores) + torch::torch_log(1 - fake_scores)
  d_loss <- -d_loss$mean()


  g_loss <- torch::torch_log(1 - fake_scores)

  g_loss <- g_loss$mean()

  return(list(d_loss = d_loss,
              g_loss = g_loss))

}
```

3.2. Choosing and customizing network architectures

Description of the design choices. Parts of the network architecture, specifically the architecture of the discriminator (or critic) network, depend on the chosen value function. For the original GAN value function the output layer of the neural network should return values between 0 and 1, thus, the activation function for the output layer is typically the sigmoid function[10]. For WGAN the output does not need to be restricted, therefore, a linear output layer can be used.[11]

In general, it is important that the discriminator has enough capacity. Where the meaning of enough capacity is dependent on the application. For image synthesization convolutional neural network architectures achieve the best results, for time series recurrent neural network architectures are sensible choices, for tabular data, simpler fully connected network architectures are usually sufficient.

Implementation of default design choices RGAN. In **RGAN** some commonly used architectures are already implemented e.g. DCGAN[12] (Radford *et al.* 2016) for image data and a GAN with fully connected neural networks for tabular data. If no networks are specified in a call to `gan_trainer` **RGAN** defaults to fully connected networks for both the generator and discriminator, with two hidden layers, 128 neurons in each layer, and a dropout rate of 0.5.

Further customization with RGAN. The included fully connected network architecture can easily be customized. For example, if users want to customize the number of layers and neurons per layer they can create custom neural networks with a call to `generator <- Generator(noise_dim = 2, data_dim = 2, hidden_units = list(256, 128, 64))` and `discriminator <- Discriminator(data_dim = 2, hidden_units = list(256, 128, 64), sigmoid = TRUE)` respectively. These networks can then be fed to `gan_trainer(..., generator = generator, discriminator = discriminator)`. Where `data_dim` needs to match the number of columns in the input dataset and `noise_dim` needs to match the dimensions of the noise distribution (more in 3.3. The list passed to `hidden_units` specifies both the depth of the neural network and the width of each layer. `list(256, 128, 64)` means that the network will have three hidden layers (determined by the length of the list), where the first hidden layer will contain 256 neurons, the second hidden layer 128, and the third hidden layer 64. For the discriminator network, users can also specify whether the output layer should have a sigmoid activation function or be a linear output layer. This choice depends on the value function chosen for a specific GAN (see section 3.1).

If users want to synthesize images and use the DCGAN architecture included in **RGAN** they can achieve this by setting up the DCGAN networks by calling `generator <- DCGAN_Generator()` and `DCGAN_Discriminator()`[13]. Again, these networks can be included in the call to `gan_trainer(...,`

[10] Which is one of the reasons for the vanishing gradients.

[11] For WGAN to work, however, restrictions on the weights of the discriminator need to be imposed. Either by weight clipping or a gradient penalty (e.g. in the WGAN-GP implementation (Gulrajani *et al.* 2017)).

[12] Short for Deep Convolutional GAN.

[13] If no options are specified this will initialize the default DCGAN architecture with a `noise_dim = 100` and a linear output layer for the discriminator.

`generator = generator, discriminator = discriminator)`. I describe a full working example with the DCGAN architecture and loading images from a user-specified folder in section 4.

Furthermore, users can provide their custom network architectures for both the discriminator and generator networks by passing a suitable `torch::nn_module` object to the `gan_trainer` function. For the generator that means the input to the `torch::nn_module` needs to match the dimensions of the chosen noise distribution and the output needs to line up with the dimensions of the real data. For the discriminator, the input needs to match the dimensions of the real data and the output needs to be just a 1-dimensional vector with the discriminator scores.

The number of hidden layers, the depth and width of these layers, and their activation functions are hyperparameters that should be optimized for a particular application. In **RGAN** it is easy to experiment with these hyperparameters and tailor them for a specific application.

3.3. Choosing the noise distribution

Description of the design choices. In most GAN applications the noise distribution for the generator $p(z)$ is typically a standard normal distribution or a uniform distribution on the interval $(-1, 1)$.

However, Huster *et al.* (2021) show that the choice of the noise distribution influences the behavior of GAN training. For example, it is hard to learn skewed distributions[14] with the standard noise distributions. And other noise distributions such as sampling from a Pareto distribution could have preferable properties for different applications.

Implementation of default design choices RGAN. This is an active area of research and **RGAN** makes it easy to implement custom prior distributions. Users can easily choose between a `normal` distribution and a `uniform` distribution when specifying the options of `gan_trainer`.

Further customization with RGAN. On top of these default options, users can also provide their own functions. For example, this is how sampling from the normal distribution is implemented:

```
normal_noise <- torch::torch_randn
```

and this function would be used as follows `gan_trainer(..., noise_distribution = normal_noise)` for training a GAN.

3.4. Choosing optimization algorithms

Description of the design choices. The optimization of GANs is notoriously hard. Recent research shows (see e.g. Heusel, Ramsauer, Unterthiner, Nessler, and Hochreiter 2017;

[14] A typical social science example for such skewed distributions could be the distribution of income.

Schaefer and Anandkumar 2019) that gradient ascent descent—i.e. iteratively training the generator and discriminator with the same optimizer—might not be the best option.

In particular, Heusel *et al.* (2017) have shown that using two different learning rates for the optimization of the discriminator and generator—they call this scheme "two time-scale update rule"—improves GAN training. Additional to the learning rate, the concrete choice of an optimizer can also influence GAN training. The batch size, how many examples are sampled from the training data per iteration, and potentially further hyperparameters of common optimizers (typically momentum hyperparameters) can also influence how quickly a GAN can learn.

Implementation of default design choices RGAN. In **RGAN** it is easy to put the optimizers for the generator and discriminator on two separate time scales by setting the `ttur_factor` in `gan_trainer()`. The `ttur_factor` is a multiplier for the provided `base_lr`, such that the generator will be updated with the `base_lr` and the discriminator with `ttur_factor*base_lr`. By default **RGAN** uses the Adam optimizer (Kingma and Ba 2014) as implemented in `torch::optim_adam` and the default minibatch size is set to 50 examples per iteration.

Further customization with RGAN. **RGAN** facilitates experimentation with different optimizers and hyperparameter settings. Users can directly pass any **torch** optimizer with any self-chosen hyperparameter settings to `gan_trainer` e.g. `gan_trainer(..., generator_optimizer = torch::optim_sgd(g_net$parameters, lr = 0.1))`.

3.5. Choosing dropout during training and data generation

Description of the design choices. In neural network architectures so-called dropout layers (Srivastava, Hinton, Krizhevsky, Sutskever, and Salakhutdinov 2014) can be used to regularize network training and prevent overfitting. In each training iteration, a dropout layer randomly sets a pre-specified percentage of all neurons to 0. A helpful side effect of such dropout layers in classical neural networks is that dropout can also be used to estimate the model uncertainty of neural networks (Gal and Ghahramani 2016).

Anecdotal evidence shows that dropout layers might help training GANs as well[15]. Furthermore, Isola, Zhu, Zhou, and Efros (2016) argue that dropout in the generator during data generation provides stochasticity for the synthetic data that promotes diversity of the generated data.

Yet, a systematic evaluation of the effect of dropout in GANs on the produced synthetic data—especially on the statistical utility of tabular synthetic data—is still missing and warrants more research.

Implementation of default design choices RGAN. By making it easy to experiment with dropout layers in both the discriminator and generator networks, **RGAN** can facilitate such research. When working with the default networks in `gan_trainer` the dropout rate is

[15]For example this `https://github.com/soumith/ganhacks` list of tips for GAN training by some of the authors of influential GAN papers (Arjovsky *et al.* 2017; Radford *et al.* 2016; Denton, Gross, and Fergus 2016; Zhao, Mathieu, and LeCun 2016) mentions dropout in the generator as a means to stabilize GAN training.

set to 0.5 in both the discriminator and generator networks. By default, dropout is disabled during the generation of synthetic data but can be enabled by setting `gan_trainer(..., eval_dropout = TRUE)`.

Further customization with RGAN. By providing custom networks, the dropout rate during training can be adjusted. In section 4 I provide a simple example of an experiment of synthetic data generated with and without dropout during data generation.

3.6. Post-Processing GAN samples

Description of the design choices. Since convergence to equilibrium cannot be guaranteed the samples produced after the last update need not be the best. However, several methods have been proposed to boost the fidelity of the samples by sampling from multiple generators.[16] These post-processing methods leverage two empirical observations. First, that the learned discriminator can be used to assess the quality of generated examples (Azadi, Olsson, Darrell, Goodfellow, and Odena 2019). And second, while an arbitrary generator (for example the last generator) can produce a bad representation of the underlying data distribution, a mixture distribution of samples from multiple generators can improve the quality of generated examples (see e.g. Beaulieu-Jones, Wu, Williams, Lee, Bhavnani, Byrd, and Greene 2019; Neunhoeffer, Wu, and Dwork 2021). Methods include Discriminator Rejection Sampling (DRS) (Azadi *et al.* 2019), Metropolis-Hastings GAN (MH) (Turner, Hung, Frank, Saatchi, and Yosinski 2019) or Post-GAN Boosting (PGB) (Neunhoeffer *et al.* 2021).

Implementation of default design choices RGAN. Currently, DRS and PGB are implemented as part of **RGAN**.

4. Illustrations

In this section, I provide two illustrations of how **RGAN** can be used. The first example shows how to quickly train a first GAN on tabular data with the default settings of **RGAN**, including a simple experiment of what dropout during synthetic data generation does. The second example shows that **RGAN** can also be used to synthesize image data and along the way serves as a tutorial on how to provide custom neural network architectures and optimizers to `gan_trainer`.

4.1. Synthesizing tabular data with default settings

As a simple illustration to get started with **RGAN** I show how to generate synthetic copies of a simple tabular data set. First, I generate some toy data[17] and transform it—for numeric

[16] There are many to sample from multiple generators. E.g. each update step during training produces a slightly different generator, so the sequence of generators can be thought of as a distribution of generators. Or when using dropout in the generator during the generation of synthetic examples and repeating this step multiple times, this will also produce a distribution of generators. Furthermore, methods that train multiple generators in parallel exist as well (e.g. Arora, Ge, Liang, Ma, and Zhang 2017; Hoang, Nguyen, Le, and Phung 2018).

[17] The default in `sample_toydata` is $N = 1000$.

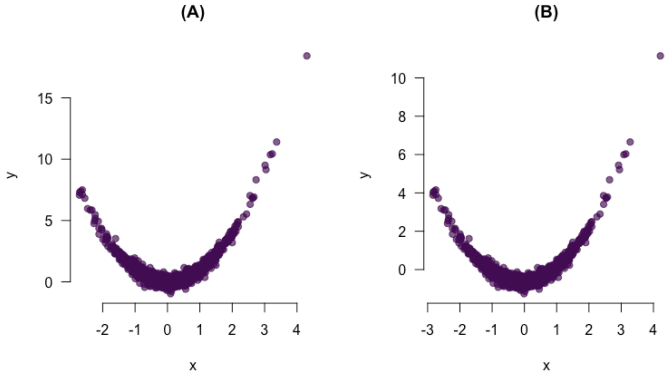

Figure 1: The real data before training the GAN. Panel (A) shows the data on the original scale and panel (B) shows the data after applying the `data_transformer`.

data which means standardizing the data to facilitate training.

```
R> data <- sample_toydata()
R> transformer <- data_transformer$new()
R> transformer$fit(data)
R> transformed_data <-  transformer$transform(data)
```

The real data for this example is displayed in panel (A) of figure 1. x on the x-axis is drawn from a standard normal distribution and y on the y-axis is $x^2 + \mathcal{N}(0, 0.3)$. Panel (B) of figure 1 shows the data after transformation.

Now the `gan_trainer()` can be used to train a GAN to generate synthetic data from `transformed_data`.[18] By default, `gan_trainer` shows a simple progress bar to monitor training and get an estimate for how long training will take.[19]

```
R> trained_gan <- gan_trainer(transformed_data)

  Training the GAN                              26% | ETA: 42s
```

This will set up the necessary generator and discriminator as well as the respective optimizers. The default value function is set to the original GAN value function described in equation 1. Then, the `gan_trainer` trains the GAN for 150 epochs using mini-batches of 50 real examples ($m = 50$) at each update step.[20]

[18] Similarily, data could be used directly. A first experiment could be to assess the convergence of the GAN when using the original data instead of the transformed data.

[19] The `gan_trainer` also has the option to observe the intermediary output of the generator during training. If users set the option `plot_progress = TRUE` a plot of real and synthetic data will be produced after each epoch. For cases where an epoch might take too long or it is sufficient to look at the intermediary output in a longer interval, users can input the number of steps after which a new plot should be produced to the option `plot_interval`.

[20] This means that the GAN is trained for a total of $\frac{N}{m} \cdot epochs = \frac{1000}{50} \cdot 150 = 3000$ update steps.

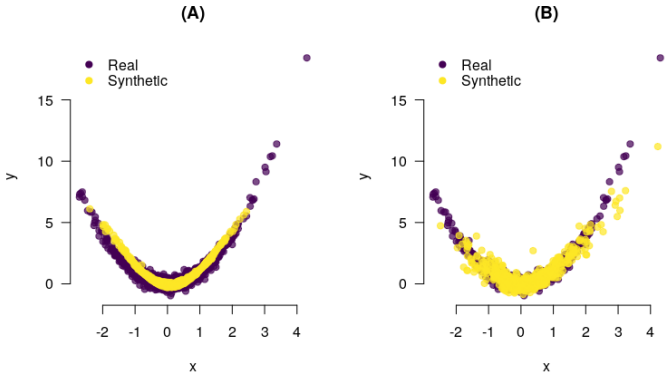

Figure 2: The synthetic data after training the GAN. Panel (A) shows the generated data without dropout during generation on the original scale. Panel (B) shows the generated data with dropout during data generation.

With the trained networks—which will be collected as part of the output of class `trained_RGAN` in the object `trained_gan`—it is then easy to sample synthetic data from the last generator using the function `sample_synthetic_data`. Note that to transform the synthetic data back to the original scale of the data I pass the `transformer` to the `sample_synthetic_data` function. To see the data alongside the synthetic data, both can be passed to the `GAN_update_plot` function. This produces the plot in panel (A) of figure 2.

```
R> synthetic_data <- sample_synthetic_data(trained_gan, transformer)
```

```
R> GAN_update_plot(
+  data = data,
+  synth_data = synthetic_data,
+  main = "(A)"
+ )
```

Looking at the generated examples in panel (A) of figure 2 shows that while the GAN seems to approximate the functional form of the underlying data well, the produced examples have a smaller variance than the original data.

As a simple experiment, I also produce synthetic data from the same generator but with dropout during data generation. I do this by setting the `eval_dropout` setting of `trained_gan` to `TRUE`. This produces the plot in panel (B) of figure 2.

```
R> trained_gan$settings$eval_dropout <- TRUE
```

```
R> synthetic_data <- sample_synthetic_data(trained_gan, transformer)
```

```
R> GAN_update_plot(
+  data = data,
+  synth_data = synthetic_data,
+  main = "(B)"
+ )
```

Now the generated synthetic data in yellow seems to look more like the underlying real data in purple. How good the data is could now be assessed by using several utility metrics for synthetic data (see e.g. Snoke, Raab, Nowok, Dibben, and Slavkovic 2016).

4.2. Synthesizing images with a DCGAN architecture

To showcase the **RGAN** customization options this illustration generates synthetic images using a DCGAN (Radford *et al.* 2016) architecture. The training data for this example is the CelebA dataset (Liu, Luo, Wang, and Tang 2015)[21] that contains 202,599 images[22] of celebrities' faces.[23]

First, I use **torchvision** to make images in a folder on my hard drive available for training the networks with **torch**. Furthermore, the DCGAN architecture expects input images to be of size 64×64 pixels. To ensure that the loaded images have the expected format, I apply some standard transformations to the raw images.

```
R> dataset <- torchvision::image_folder_dataset(root = "path/to/celeba",
+ transform = function(x) {
+ x = torchvision::transform_to_tensor(x)
+ x = torchvision::transform_resize(x, size = c(64, 64))
+ x = torchvision::transform_center_crop(x, c(64, 64))
+ x = torchvision::transform_normalize(x, c(0.5, 0.5, 0.5), c(0.5, 0.5, 0.5))
+ return(x)
+ })
```

Next, I load the `DCGAN_Generator` and `DCGAN_Discriminator` and make them available on the GPU by passing them to the `"cuda"` device.[24] The `noise_dim`—the number of input features for the generator network—is set to 100.

```
R> device <- "cuda"
R> g_net <- DCGAN_Generator(dropout_rate = 0, noise_dim = 100)$to(device = device)
R> d_net <- DCGAN_Discriminator(dropout_rate = 0, sigmoid = FALSE)$to(device = device)
```

Next, the optimizers for both networks can be set up. I use the same optimizers and hyperparameters as Radford *et al.* (2016) in the original DCGAN paper.

[21] The images can be obtained from the authors' website: https://mmlab.ie.cuhk.edu.hk/projects/CelebA.html. Last accessed: March 31, 2022.

[22] With a batch size of 128, as in the DCGAN paper, this means the GAN is updated for 1,582 steps per epoch.

[23] Training this model on a machine with a GPU (here a GeForce RTX 2070) takes about 17 minutes per epoch. On a notebook without a GPU one epoch takes about 7 hours.)

[24] If no GPU is available, the device should be set to `"cpu"`.

```
R> g_optim <- torch::optim_adam(g_net$parameters, lr = 0.0002, betas = c(0.5, 0.999))

R> d_optim <- torch::optim_adam(d_net$parameters, lr = 0.0002, betas = c(0.5, 0.999))
```

Now, these objects along with a couple of additional options can be passed to `gan_trainer`. Note that the DCGAN architecture expects the noise samples to be in a three-dimensional tensor (`noise_dim = c(100, 1, 1)`, even though the second and third dimensions are only of length 1. Another important option is to set `data_type = "image"`, to make sure the `gan_trainer` produces the output in the right format.

```
R> trained_gan <- gan_trainer(
+  data = dataset,
+  noise_dim = c(100, 1, 1),
+  noise_distribution = "uniform",
+  data_type = "image",
+  value_function = "wasserstein",
+  generator = g_net,
+  generator_optimizer = g_optim,
+  discriminator = d_net,
+  discriminator_optimizer = d_optim,
+  plot_progress = TRUE,
+  plot_interval = 10,
+  batch_size = 128,
+  synthetic_examples = 16,
+  device = device,
+  eval_dropout = FALSE,
+  epochs = 1
+  )
```

To observe the progress of training I set `plot_progress = TRUE` and the `plot_interval = 10` this means that 16 images[25] will be shown after every tenth update step.

In figure 3, I show 16 examples of generated faces after just one epoch of training the DCGAN with the settings as in the code above. Note, that I do not expect to generate state-of-the-art fake images after training just for one epoch with a relatively small and simple DCGAN architecture. However, even after a few update steps with the settings described above it should be possible to make out the outlines of faces in the generated images.

With **RGAN** more elaborate network architectures could be provided and training could be optimized.[26]

[25] 16 is an arbitrary number but it is easy to show these images in a 4×4 grid.

[26] Yet, achieving state-of-the-art performance can be prohibitively expensive. For example, the developers of StyleGAN2 (Karras, Laine, Aittala, Hellsten, Lehtinen, and Aila 2020)—a GAN that produces high definition real looking images—transparently state the computing resources that went into the project: "The entire project, including all exploration, consumed 132 MWh of electricity, of which 0.68 MWh went into training the final FFHQ model. In total, we used about 51 single-GPU years of computation (Volta class GPU)" (8). A large amount of computing time, especially from the perspective of a statistician or social scientist. At the time of writing, the price for such a GPU e.g. an Nvidia Tesla V100 GPU is around $9000, i.e. to get all the experiments done in one year you would need at least $459000 just to buy the GPUs. Using cloud computing e.g. through Amazon Web Services (AWS) you can access a computer with eight such GPUs for $12.24 per

Figure 3: Generated faces after one epoch of training the DCGAN with the parameters specified in the main text.

5. Summary and outlook

With **RGAN** I introduce a package to facilitate the training of and experimentation with GANs in R. The goal is to open up this powerful neural net framework to users that primarily work in R—especially to facilitate experimentation with synthetic data from GANs from a statistical perspective.

I provide two illustrations of how **RGAN** can be used. The first illustration shows the basic functionality of **RGAN** and how to quickly synthesize tabular data. The second illustration shows that training GANs with **RGAN** is easy to customize and synthesize images with a DCGAN architecture.

Future iterations of **RGAN** will include additional network architectures, for example, a CTGAN (Xu *et al.* 2019) inspired architecture to make the generation of mixed tabular data (i.e. data with real-valued and categorical columns) easier. Another area of development will focus on making the **opacus** (Yousefpour, Shilov, Sablayrolles, Testuggine, Prasad, Malek,

hour. To run all the experiments that (Karras *et al.* 2020) did, you would need 55845 hours of such a server, totaling ≈ $685000.

Nguyen, Ghosh, Bharadwaj, Zhao, Cormode, and Mironov 2021) python library available in **RGAN** to train GANs with formal privacy guarantees.

The latest stable version of **RGAN** is available from CRAN and development versions can be found on https://github.com/mneunhoe/RGAN.

Computational details

The results in this paper were obtained using R 4.1.3 with the **RGAN** 0.1.1 package, the **torch** 0.8.0 package, and the **torchvision** 0.4.1 (Falbel 2022) package to load images into **torch**. For the color palettes in the visualizations, I used the **viridis** 0.6.2 (Garnier, Ross, Rudis, Camargo, Pedro, Sciaini, and Scherer 2021) package. R itself and all packages used are available from the Comprehensive R Archive Network (CRAN) at https://CRAN.R-project.org/.

Acknowledgments

I am grateful to Thomas Gschwend, Christian Arnold, Pia Kürzdörfer, Guido Ropers, Oliver Rittmann and Oke Bahnsen for feedback on earlier versions of this manuscript.

References

Abadi M, Agarwal A, Barham P, Brevdo E, Chen Z, Citro C, Corrado GS, Davis A, Dean J, Devin M, Ghemawat S, Goodfellow I, Harp A, Irving G, Isard M, Jia Y, Jozefowicz R, Kaiser L, Kudlur M, Levenberg J, Mané D, Monga R, Moore S, Murray D, Olah C, Schuster M, Shlens J, Steiner B, Sutskever I, Talwar K, Tucker P, Vanhoucke V, Vasudevan V, Viégas F, Vinyals O, Warden P, Wattenberg M, Wicke M, Yu Y, Zheng X (2015). "TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems." Software available from tensorflow.org, URL https://www.tensorflow.org/.

Arjovsky M, Chintala S, Bottou L (2017). "Wasserstein Generative Adversarial Networks." In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, p. 214223. JMLR.org.

Arora S, Ge R, Liang Y, Ma T, Zhang Y (2017). "Generalization and Equilibrium in Generative Adversarial Nets (GANs)." doi:10.48550/ARXIV.1703.00573. URL https://arxiv.org/abs/1703.00573.

Azadi S, Olsson C, Darrell T, Goodfellow I, Odena A (2019). "Discriminator Rejection Sampling." In *International Conference on Learning Representations*. URL https://openreview.net/forum?id=S1GkToR5tm.

Beaulieu-Jones BK, Wu ZS, Williams C, Lee R, Bhavnani SP, Byrd JB, Greene CS (2019). "Privacy-Preserving Generative Deep Neural Networks Support Clinical Data Sharing." *Circulation: Cardiovascular Quality and Outcomes*, **12**(7), e005122. doi:10.

1161/CIRCOUTCOMES.118.005122. https://www.ahajournals.org/doi/pdf/10.1161/CIRCOUTCOMES.118.005122.

Denton E, Gross S, Fergus R (2016). "Semi-Supervised Learning with Context-Conditional Generative Adversarial Networks." doi:10.48550/ARXIV.1611.06430. URL https://arxiv.org/abs/1611.06430.

Falbel D (2022). *torchvision: Models, Datasets and Transformations for Images*. R package version 0.4.1, URL https://CRAN.R-project.org/package=torchvision.

Falbel D, Luraschi J (2022). *torch: Tensors and Neural Networks with 'GPU' Acceleration*. R package version 0.7.2, URL https://CRAN.R-project.org/package=torch.

Gal Y, Ghahramani Z (2016). "Dropout as a bayesian approximation: Representing model uncertainty in deep learning." In *international conference on machine learning*, pp. 1050–1059. PMLR.

Garnier S, Ross N, Rudis R, Camargo AP, Pedro A, Sciaini M, Scherer C (2021). *viridis - Colorblind-Friendly Color Maps for R*. doi:10.5281/zenodo.4679424. R package version 0.6.2, URL https://sjmgarnier.github.io/viridis/.

Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A, Bengio Y (2014). "Generative Adversarial Nets." *Advances in Neural Information Processing Systems 27*, pp. 2672–2680. ISSN 10495258. doi:10.1017/CBO9781139058452. arXiv:1406.2661v1, URL http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf.

Gulrajani I, Ahmed F, Arjovsky M, Dumoulin V, Courville AC (2017). "Improved Training of Wasserstein GANs." In I Guyon, UV Luxburg, S Bengio, H Wallach, R Fergus, S Vishwanathan, R Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc. URL https://proceedings.neurips.cc/paper/2017/file/892c3b1c6dccd52936e27cbd0ff683d6-Paper.pdf.

Heusel M, Ramsauer H, Unterthiner T, Nessler B, Hochreiter S (2017). "GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium." In I Guyon, UV Luxburg, S Bengio, H Wallach, R Fergus, S Vishwanathan, R Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc. URL https://proceedings.neurips.cc/paper/2017/file/8a1d694707eb0fefe65871369074926d-Paper.pdf.

Hoang Q, Nguyen TD, Le T, Phung D (2018). "MGAN: Training Generative Adversarial Nets with Multiple Generators." In *International Conference on Learning Representations*. URL https://openreview.net/forum?id=rkmu5b0a-.

Huster T, Cohen J, Lin Z, Chan K, Kamhoua C, Leslie NO, Chiang CYJ, Sekar V (2021). "Pareto GAN: Extending the Representational Power of GANs to Heavy-Tailed Distributions." In M Meila, T Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 4523–4532. PMLR. URL https://proceedings.mlr.press/v139/huster21a.html.

Isola P, Zhu JY, Zhou T, Efros AA (2016). "Image-to-Image Translation with Conditional Adversarial Networks." doi:10.48550/ARXIV.1611.07004. URL https://arxiv.org/abs/1611.07004.

Karras T, Laine S, Aila T (2019). "A Style-Based Generator Architecture for Generative Adversarial Networks." In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Karras T, Laine S, Aittala M, Hellsten J, Lehtinen J, Aila T (2020). "Analyzing and Improving the Image Quality of StyleGAN." In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Kim T, Cha M, Kim H, Lee JK, Kim J (2017). "Learning to Discover Cross-Domain Relations with Generative Adversarial Networks." In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, p. 18571865. JMLR.org.

Kingma DP, Ba J (2014). "Adam: A Method for Stochastic Optimization." doi:10.48550/ARXIV.1412.6980. URL https://arxiv.org/abs/1412.6980.

Lee KS, Town C (2020). "Mimicry: Towards the Reproducibility of GAN Research." doi:10.48550/ARXIV.2005.02494. URL https://arxiv.org/abs/2005.02494.

Liu Z, Luo P, Wang X, Tang X (2015). "Deep Learning Face Attributes in the Wild." In *Proceedings of International Conference on Computer Vision (ICCV)*.

Mirza M, Osindero S (2014). "Conditional Generative Adversarial Nets." 1411.1784.

Müller W (2022). *ganGenerativeData*. R package version 1.3.3, URL https://cran.r-project.org/package=ganGenerativeData.

Neunhoeffer M, Wu S, Dwork C (2021). "Private Post-{GAN} Boosting." In *International Conference on Learning Representations*. URL https://openreview.net/forum?id=6isfR3JCbi.

Pal A, Das A (2021). "TorchGAN: A Flexible Framework for GAN Training and Evaluation." *Journal of Open Source Software*, **6**(66), 2606. doi:10.21105/joss.02606. URL https://doi.org/10.21105/joss.02606.

Radford A, Metz L, Chintala S (2016). "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks." In Y Bengio, Y LeCun (eds.), *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*. URL http://arxiv.org/abs/1511.06434.

Schaefer F, Anandkumar A (2019). "Competitive Gradient Descent." In H Wallach, H Larochelle, A Beygelzimer, F d'Alché-Buc, E Fox, R Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc. URL https://proceedings.neurips.cc/paper/2019/file/56c51a39a7c77d8084838cc920585bd0-Paper.pdf.

Snoke J, Raab G, Nowok B, Dibben C, Slavkovic A (2016). "General and specific utility measures for synthetic data." doi:10.48550/ARXIV.1604.06651. URL https://arxiv.org/abs/1604.06651.

Song J, Ermon S (2020). "Bridging the gap between f-gans and wasserstein gans." In *International Conference on Machine Learning*, pp. 9078–9087. PMLR.

Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R (2014). "Dropout: a simple way to prevent neural networks from overfitting." *The journal of machine learning research*, **15**(1), 1929–1958.

Turner R, Hung J, Frank E, Saatchi Y, Yosinski J (2019). "Metropolis-Hastings Generative Adversarial Networks." In K Chaudhuri, R Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 6345–6353. PMLR. URL https://proceedings.mlr.press/v97/turner19a.html.

Ushey K, Allaire J, Tang Y (2022). *reticulate: Interface to 'Python'*. R package version 1.24, URL https://CRAN.R-project.org/package=reticulate.

Xu L, Skoularidou M, Cuesta-Infante A, Veeramachaneni K (2019). "Modeling Tabular data using Conditional GAN." In H Wallach, H Larochelle, A Beygelzimer, F d'Alché-Buc, E Fox, R Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc. URL https://proceedings.neurips.cc/paper/2019/file/254ed7d2de3b23ab10936522dd547b78-Paper.pdf.

Yousefpour A, Shilov I, Sablayrolles A, Testuggine D, Prasad K, Malek M, Nguyen J, Ghosh S, Bharadwaj A, Zhao J, Cormode G, Mironov I (2021). "Opacus: User-Friendly Differential Privacy Library in PyTorch." *arXiv preprint arXiv:2109.12298*.

Zhao J, Mathieu M, LeCun Y (2016). "Energy-based Generative Adversarial Network." doi:10.48550/ARXIV.1609.03126. URL https://arxiv.org/abs/1609.03126.

Zhu JY, Park T, Isola P, Efros AA (2017). "Unpaired Image-To-Image Translation Using Cycle-Consistent Adversarial Networks." In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.

3

Private Post-GAN Boosting

PRIVATE POST-GAN BOOSTING

Marcel Neunhoeffer
University of Mannheim
mneunhoe@mail.uni-mannheim.de

Zhiwei Steven Wu
Carnegie Mellon University
zstevenwu@cmu.edu

Cynthia Dwork
Harvard University
dwork@seas.harvard.edu

ABSTRACT

Differentially private GANs have proven to be a promising approach for generating realistic synthetic data without compromising the privacy of individuals. Due to the privacy-protective noise introduced in the training, the convergence of GANs becomes even more elusive, which often leads to poor utility in the output generator at the end of training. We propose *Private post-GAN boosting (Private PGB)*, a differentially private method that combines samples produced by the sequence of generators obtained during GAN training to create a high-quality synthetic dataset. To that end, our method leverages the Private Multiplicative Weights method (Hardt and Rothblum, 2010) to reweight generated samples. We evaluate Private PGB on two dimensional toy data, MNIST images, US Census data and a standard machine learning prediction task. Our experiments show that Private PGB improves upon a standard private GAN approach across a collection of quality measures. We also provide a non-private variant of PGB that improves the data quality of standard GAN training.

1 INTRODUCTION

The vast collection of detailed personal data, including everything from medical history to voting records, to GPS traces, to online behavior, promises to enable researchers from many disciplines to conduct insightful data analyses. However, many of these datasets contain sensitive personal information, and there is a growing tension between data analyses and data privacy. To protect the privacy of individual citizens, many organizations, including Google (Erlingsson et al., 2014), Microsoft (Ding et al., 2017), Apple (Differential Privacy Team, Apple, 2017), and more recently the 2020 US Census (Abowd, 2018), have adopted *differential privacy* (Dwork et al., 2006) as a mathematically rigorous privacy measure. However, working with noisy statistics released under differential privacy requires training.

A natural and promising approach to tackle this challenge is to release *differentially private synthetic data*—a privatized version of the dataset that consists of fake data records and that approximates the real dataset on important statistical properties of interest. Since they already satisfy differential privacy, synthetic data enable researchers to interact with the data freely and to perform the same analyses even without expertise in differential privacy. A recent line of work (Beaulieu-Jones et al., 2019; Xie et al., 2018; Yoon et al., 2019) studies how one can generate synthetic data by incorporating differential privacy into *generative adversarial networks* (GANs) (Goodfellow et al., 2014). Although GANs provide a powerful framework for synthetic data, they are also notoriously hard to train and privacy constraint imposes even more difficulty. Due to the added noise in the private gradient updates, it is often difficult to reach convergence with private training.

In this paper, we study how to improve the quality of the synthetic data produced by private GANs. Unlike much of the prior work that focuses on fine-tuning of network architectures and training techniques, we propose *Private post-GAN boosting* (Private PGB)—a differentially private method that boosts the quality of the generated samples after the training of a GAN. Our method can be viewed as a simple and practical amplification scheme that improves the distribution from any ex-

isting black-box GAN training method – private or not. We take inspiration from an empirical observation in Beaulieu-Jones et al. (2019) that even though the generator distribution at the end of the private training may be a poor approximation to the data distribution (due to e.g. mode collapse), there may exist a high-quality mixture distribution that is given by several generators over different training epochs. PGB is a principled method for finding such a mixture at a moderate privacy cost and without any modification of the GAN training procedure.

To derive PGB, we first formulate a two-player zero-sum game, called *post-GAN* zero-sum game, between a *synthetic data* player, who chooses a distribution over generated samples over training epochs to emulate the real dataset, and a *distinguisher* player, who tries to distinguish generated samples from real samples with the set of discriminators over training epochs. We show that under a "support coverage" assumption the synthetic data player's mixed strategy (given by a distribution over the generated samples) at an equilibrium can successfully "fool" the distinguisher–that is, no mixture of discriminators can distinguish the real versus fake examples better than random guessing. While the strict assumption does not always hold in practice, we demonstrate empirically that the synthetic data player's equilibrium mixture consistently improves the GAN distribution.

The Private PGB method then privately computes an approximate equilibrium in the game. The algorithm can be viewed as a computationally efficient variant of MWEM (Hardt & Rothblum, 2010; Hardt et al., 2012), which is an inefficient query release algorithm with near-optimal sample complexity. Since MWEM maintains a distribution over exponentially many "experts" (the set of all possible records in the data domain), it runs in time exponential in the dimension of the data. In contrast, we rely on private GAN to reduce the support to only contain the set of privately generated samples, which makes PGB tractable even for high-dimensional data.

We also provide an extension of the PGB method by incorporating the technique of *discriminator rejection sampling* (Azadi et al., 2019; Turner et al., 2019). We leverage the fact that the distinguisher's equilibrium strategy, which is a mixture of discriminators, can often accurately predict which samples are unlikely and thus can be used as a rejection sampler. This allows us to further improve the PGB distribution with rejection sampling without any additional privacy cost since differential privacy is preserved under post-processing. Our Private PGB method also has a natural non-private variant, which we show improves the GAN training without privacy constraints.

We empirically evaluate both the Private and Non-Private PGB methods on several tasks. To visualize the effects of our methods, we first evaluate our methods on a two-dimensional toy dataset with samples drawn from a mixture of 25 Gaussian distributions. We define a relevant quality score function and show that the both Private and Non-Private PGB methods improve the score of the samples generated from GAN. We then show that the Non-Private PGB method can also be used to improve the quality of images generated by GANs using the MNIST dataset. Finally, we focus on applications with high relevance for privacy-protection. First we synthesize US Census datasets and demonstrate that the PGB method can improve the generator distribution on several statistical measures, including 3-way marginal distributions and pMSE. Secondly, we evaluate the PGB methods on a dataset with a natural classification task. We train predictive models on samples from Private PGB and samples from a private GAN (without PGB), and show that PGB consistently improves the model accuracy on real out-of-sample test data.

Related work. Our PGB method can be viewed as a modular boosting method that can improve on a growing line of work on differentially private GANs (Beaulieu-Jones et al., 2019; Xie et al., 2018; Frigerio et al., 2019; Torkzadehmahani et al., 2020). To obtain formal privacy guarantees, these algorithms optimize the discriminators in GAN under differential privacy, by using private SGD, RMSprop, or Adam methods, and track the privacy cost using moments accounting Abadi et al. (2016); Mironov (2017). Yoon et al. (2019) give a private GAN training method by adapting ideas from the PATE framework (Papernot et al., 2018).

Our PGB method is inspired by the Private Multiplicative Weigths method (Hardt & Rothblum, 2010) and its more practical variant MWEM (Hardt et al., 2012), which answer a large collection of statistical queries by releasing a synthetic dataset. Our work also draws upon two recent techniques (Turner et al. (2019) and Azadi et al. (2019)) that use the discriminator as a rejection sampler to improve the generator distribution. We apply their technique by using the mixture discriminator computed in PGB as the rejection sampler. There has also been work that applies the idea of boosting to (non-private) GANs. For example, Arora et al. (2017) and Hoang et al. (2018) propose methods

that directly train a mixture of generators and discriminators, and Tolstikhin et al. (2017) proposes AdaGAN that reweighes the real examples during training similarly to what is done in AdaBoost (Freund & Schapire, 1997). Both of these methods may be hard to make differentially private: they either require substantially more privacy budget to train a collection of discriminators or increase the weights on a subset of examples, which requires more adding more noise when computing private gradients. In contrast, our PGB method boosts the generated samples *post* training and does not make modifications to the GAN training procedure.

2 PRELIMINARIES

Let $\mathcal{X}$ denote the data domain of all possible observations in a given context. Let p_d be a distribution over $\mathcal{X}$. We say that two datasets $X, X' \in \mathcal{X}^n$ are adjacent, denoted by $X \sim X'$, if they differ by at most one observation. We will write p_X to denote the empirical distribution over X.

Definition 1 (Differential Privacy (DP) (Dwork et al., 2006)). *A randomized algorithm $\mathcal{A} : \mathcal{X}^n \to \mathcal{R}$ with output domain $\mathcal{R}$ (e.g. all generative models) is (ε, δ)-differentially private (DP) if for all adjacent datasets $X, X' \in \mathcal{X}^n$ and for all $S \subseteq \mathcal{R}$: $P(\mathcal{A}(X) \in S) \leq e^{\varepsilon} P(\mathcal{A}(X') \in S) + \delta$.*

A very nice property of differential privacy is that it is preserved under post-processing.

Lemma 1 (Post-processing). *Let $\mathcal{M}$ be an (ε, δ)-differentially private algorithm with output range R and $f : R \to R'$ be any mapping, the composition $f \circ \mathcal{M}$ is (ε, δ)-differentially private.*

As a result, any subsequent analyses conducted on DP synthetic data also satisfy DP.

The *exponential mechanism* (McSherry & Talwar, 2007) is a private mechanism for selecting among the best of a discrete set of alternatives $\mathcal{R}$, where "best" is defined by a quality function $q \colon \mathcal{X}^n \times \mathcal{R} \to \mathcal{R}$ that measures the quality of the result r for the dataset X. The sensitivity of the quality score q is defined as $\Delta(q) = \max_{r \in \mathcal{R}} \max_{X \sim X'} |q(X, r) - q(X', r)|$. Then given a quality score q and privacy parameter ε, the exponential mechanism $\mathcal{M}_E(q, \varepsilon, X)$ simply samples a random alternative from the range $\mathcal{R}$ such that the probability of selecting each r is proportional to $\exp(\varepsilon q(X, r)/(2\Delta(q)))$.

2.1 DIFFERENTIALLY PRIVATE GAN

The framework of *generative adversarial networks (GANs)* (Goodfellow et al., 2014) consists of two types of neural networks: *generators* and *discriminators*. A generator G is a function that maps random vectors $z \in Z$ drawn from a prior distribution p_z to a sample $G(z) \in \mathcal{X}$. A discriminator D takes an observation $x \in \mathcal{X}$ as input and computes a probability $D(x)$ that the observation is real. Each observation is either drawn from the underlying distribution p_d or the induced distribution p_g from a generator. The training of GAN involves solving the following joint optimization over the discriminator and generator:

$$\min_G \max_D \; \mathbb{E}_{x \sim p_X}[f(D(x))] + \mathbb{E}_{z \sim p_z}[f(1 - D(G(z)))]$$

where $f \colon [0, 1] \to \mathbb{R}$ is a monotone function. For example, in standard GAN, $f(a) = \log a$, and in Wasserstein GAN (Arjovsky et al., 2017), $f(a) = a$. The standard (non-private) algorithm iterates between optimizing the parameters of the discriminator and the generator based on the loss functions:

$$L_D = -\mathbb{E}_{x \sim p_X}[f(D(x))] - \mathbb{E}_{z \sim p_z}[f(1 - D(G(z)))], \quad L_G = \mathbb{E}_{z \sim p_z}[f(1 - D(G(z)))]$$

The private algorithm for training GAN also performs the same alternating optimization, but it optimizes the discriminator under differential privacy while keeping the generator optimization the same. In general, the training proceeds over epochs $\tau = 1, \ldots, N$, and at the end of each epoch τ the algorithm obtains a discriminator D_τ and a generator G_τ by optimizing the loss functions respectively. In Beaulieu-Jones et al. (2019); Xie et al. (2018), the private optimization on the discriminators is done by running the private SGD method Abadi et al. (2016) or its variants. Yoon et al. (2019) performs the private optimization by incorporating the PATE framework Papernot et al. (2018). For all of these private GAN methods, the entire sequence of discriminators $\{D_1, \ldots, D_N\}$ satisfies privacy, and thus the sequence of generators $\{G_1, \ldots, G_N\}$ is also private since they can be viewed as post-processing of the discriminators. Our PGB method is agnostic to the exact private GAN training methods.

3 PRIVATE POST-GAN BOOSTING

The noisy gradient updates impede convergence of the differentially private GAN training algorithm, and the generator obtained in the final epoch of the training procedure may not yield a good approximation to the data distribution. Nonetheless, empirical evidence has shown that a mixture over the set of generators can be a realistic distribution (Beaulieu-Jones et al., 2019). We now provide a principled and practical scheme for computing such a mixture subject to a moderate privacy budget. Recall that during private GAN training method produces a sequence of generators $\mathcal{G} = \{G_1, \ldots, G_N\}$ and discriminators $\mathcal{D} = \{D_1, \ldots, D_N\}$. Our boosting method computes a weighted mixture of the G_j's and a weighted mixture of the D_j's that improve upon any individual generator and discriminator. We do that by computing an equilibrium of the following *post-GAN (training)* zero-sum game.

3.1 POST-GAN ZERO-SUM GAME.

We will first draw r independent samples from each generator G_j, and let B be the collection of the rN examples drawn from the set of generators. Consider the following *post-GAN* zero-sum game between a *synthetic data player*, who maintains a distribution ϕ over the data in B to imitate the true data distribution p_X, and a *distinguisher player*, who uses a mixture of discriminators to tell the two distributions ϕ and p_X apart. This zero-sum game is aligned with the minimax game in the original GAN formulation, but is much more tractable since each player has a finite set of strategies. To define the payoff in the game, we will adapt from the Wasserstein GAN objective since it is less sensitive than the standard GAN objective to the change of any single observation (changing any single real example changes the payoff by at most $1/n$), rendering it more compatible with privacy tools. Formally, for any $x \in B$ and any discriminator D_j, define the payoff as

$$U(x, D_j) = \mathbb{E}_{x' \sim p_X}\left[D_j(x')\right] + (1 - D_j(x))$$

For any distribution ϕ over B, let $U(\phi, \cdot) = \mathbb{E}_{x \sim \phi}[U(x, \cdot)]$, and similarly for any distribution ψ over $\{D_1, \ldots, D_N\}$, we will write $U(\cdot, \psi) = \mathbb{E}_{D \sim \psi}[U(\cdot, D)]$. Intuitively, the payoff function U measures the predictive accuracy of the distinguisher in classifying whether the examples are drawn from the synthetic data player's distribution ϕ or the private dataset X. Thus, the synthetic data player aims to minimize U while the distinguisher player aims to maximize U.

Definition 2. *The pair $(\overline{D}, \overline{\phi})$ is an α-approximate equilibrium of the post-GAN game if*

$$\max_{D_j \in \mathcal{D}} U(\overline{\phi}, D_j) \leq U(\overline{\phi}, \overline{D}) + \alpha, \quad \textit{and} \quad \min_{\phi \in \Delta(B)} U(\phi, \overline{D}) \geq U(\overline{\phi}, \overline{D}) - \alpha \tag{1}$$

By von Neumann's minimax theorem, there exists a value V – called the *game value* – such that

$$V = \min_{\phi \in \Delta(B)} \max_{j \in [N]} U(\phi, D_j) = \max_{\psi \in \Delta(\mathcal{D})} \min_{x \in B} U(x, \psi)$$

The game value corresponds to the payoff value at an exact equilibrium of the game (that is $\alpha = 0$). When the set of discriminators cannot predict the real versus fake examples better than random guessing, the game value $V = 1$. We now show that under the assumption that the generated samples in B approximately cover the support of the dataset X, the distinguisher player cannot distinguish the real and fake distributions much better than by random guessing.

Theorem 1. *Fix a private dataset $X \in (\mathbb{R}^d)^n$. Suppose that for every $x \in X$, there exists $x_b \in B$ such that $\|x - x_b\|_2 \leq \gamma$. Suppose $\mathcal{D}$ includes a discriminator network $D^{1/2}$ that outputs $1/2$ for all inputs, and assume that all networks in $\mathcal{D}$ are L-Lipschitz. Then there exists a distribution $\phi \in \Delta(B)$ such that $(\phi, D^{1/2})$ is a $L\gamma$-approximate equilibrium, and so $1 \leq V \leq 1 + L\,\gamma$.*

We defer the proof to the appendix. While the support coverage assumption is strong, we show empirically the synthetic data player's mixture distribution in an approximate equilibrium improves on the distribution given by the last generator G_N even when the assumption does not hold. We now provide a method for computing an approximate equilibrium of the game.

3.2 BOOSTING VIA EQUILIBRIUM COMPUTATION.

Our post-GAN boosting (PGB) method computes an approximate equilibrium of the post-GAN zero-sum game by simulating the so-called *no-regret dynamics*. Over T rounds the synthetic data

player maintains a sequence of distributions $\phi^1, \ldots, \phi^T$ over the set B, and the distinguisher plays a sequence of discriminators $D^1, \ldots, D^T$. At each round t, the distinguisher first selects a discriminator D using the exponential mechanism $\mathcal{M}_E$ with the payoff $U(\phi^t, \cdot)$ as the score function. This will find an accurate discriminator D^t against the current synthetic distribution ϕ^t, so that the synthetic data player can improve the distribution. Then the synthetic data player updates its distribution to ϕ^t based on an online no-regret learning algorithm–the multiplicative weights (MW) method Kivinen & Warmuth (1997). We can view the set of generated examples in B as a set of "experts", and the algorithm maintains a distribution over these experts and, over time, places more weight on the examples that can better "fool" the distinguisher player. To do so, MW updates the weight for each $x \in B$ with

$$\phi^{t+1}(x) \propto \phi^t \exp\left(-\eta U(x, D^t)\right) \propto \exp\left(\eta D^t(x)\right) \tag{2}$$

where η is the learning rate. At the end, the algorithm outputs the average plays $(\overline{D}, \overline{\phi})$ for both players. We will show these form an approximate equilibrium of the post-GAN zero-sum game (Freund & Schapire, 1997).

Algorithm 1 Differentially Private Post-GAN Boosting

Require: a private dataset $X \in \mathcal{X}^n$, a synthetic dataset B generated by the set of generators $\mathcal{G}$, a collection of discriminators $\{D_1, \ldots, D_N\}$, number of iterations T, per-round privacy budget ϵ_0, learning rate parameter η.

Initialize ϕ^1 to be the uniform distribution over B

for $t = 1, \ldots, T$ **do**

Distinguisher player: Run exponential mechanism $\mathcal{M}_E$ to select a discriminator D^t using quality score $q(X, D_j) = U(\phi^t, D_j)$ and privacy parameter ϵ_0.

Synthetic data player: Multiplicative weights update on the distribution over B: for each example $b \in B$:

$$\phi^{t+1}(b) \propto \phi^t(b)\ \exp(\eta D^t(b))$$

Let $\overline{D}$ be the discriminator defined by the uniform average over the set $\{D^1, \ldots, D^T\}$, and $\overline{\phi}$ be the distribution defined by the average over the set $\{\phi^1, \ldots, \phi^T\}$

Note that the synthetic data player's MW update rule does not involve the private dataset, and hence is just a post-processing step of the selected discriminator D^t. Thus, the privacy guarantee follows from applying the advacned composition of T runs of the exponential mechanism.[1]

Theorem 2 (Privacy Guarantee)**.** *For any $\delta \in (0, 1)$, the private MW post-amplification algorithm satisfies (ϵ, δ)-DP with $\epsilon = \sqrt{2\log(1/\delta)T}\epsilon_0 + T\epsilon_0(\exp(\epsilon_0) - 1)$.*

Note that if the private GAN training algorithm satisfies (ϵ_1, δ_1)-DP and the Private PGB method satisfies (ϵ_2, δ_2)-DP, then the entire procedure is $(\epsilon_1 + \epsilon_2, \delta_1 + \delta_2)$-DP.

We now show that the pair of average plays form an approximate equilibrium of the game.

Theorem 3 (Approximate Equilibrium)**.** *With probability $1-\beta$, the pair $(\overline{D}, \overline{\phi})$ is an α-approximate equilibrium of the post-GAN zero-sum game with $\alpha = 4\eta + \frac{\log|B|}{\eta T} + \frac{2\log(NT/\beta)}{n\epsilon_0}$. If $T \geq n^2$ and $\eta = \frac{1}{2}\sqrt{\log(|B|)/T}$, then*

$$\alpha = O\left(\frac{\log(nN|B|/\beta)}{n\epsilon_0}\right)$$

We provide a proof sketch here and defer the full proof to the appendix. By the result of Freund & Schapire (1997), if the two players have low regret in the dynamics, then their average plays form an approximate equilibrium, where the regret of the two players is defined as $R_{\text{syn}} = \sum_{t=1}^T U(\phi^t, D^t) - \min_{b \in B} \sum_{t=1}^T U(b, D^t)$ and $R_{\text{dis}} = \max_{D_j} \sum_{t=1}^T U(\phi^t, D_j) - \sum_{t=1}^T U(\phi^t, D^t)$. Then the approximate equilibrium guarantee directly follows from bounding R_{syn} with the regret bound of MW and R_{dis} with the approximate optimality of the exponential mechanism.

[1] Note that since the quality scores from the GAN Discriminators are assumed to be probabilities and the score function takes an average over n probabilities (one for each private example), the sensitivity is $\Delta(q) = \frac{1}{n}$.

Non-Private PGB. The Private PGB method has a natural non-private variant: in each round, instead of drawing from the exponential mechanism, the distinguisher player will simply compute the exact best response: $D^t = \arg\max_{D_j} U(\phi^t, D_j)$. Then if we set learning rate $\eta = \frac{1}{2}\sqrt{\log(|B|)/T}$ and run for $T = \log(|B|)/\alpha^2$ rounds, the pair $(\overline{D}, \overline{\phi})$ returned is an α-approximate equilibrium.

Extension with Discriminator Rejection Sampling. The mixture discriminator $\overline{D}$ at the equilibrium provides an accurate predictor on which samples are unlikely. As a result, we can use $\overline{D}$ to further improve the data distribution $\overline{\phi}$ by the *discriminator rejection sampling* (DRS) technique of Azadi et al. (2019). The DRS scheme in our setting generates a single example as follows: first draw an example x from $\overline{\phi}$ (the proposal distribution), and then accept x with probability proportional to $\overline{D}(x)/(1-\overline{D}(x))$. Note that the optimal discriminator D^* that distinguishes the distribution $\overline{\phi}$ from true data distribution p_d will accept x with probability proportional to $p_d(x)/p_{\overline{\phi}}(x) = D^*(x)/(1-D^*(x))$. Our scheme aims to approximate this ideal rejection sampling by approxinating D^* with the equilibrium strategy $\overline{D}$, whereas prior work uses the last discriminator D_N as an approximation.

4 Empirical Evaluation

We empirically evaluate how both the Private and Non-Private PGB methods affect the utility of the generated synthetic data from GANs. We show two appealing advantages of our approach: 1) non-private PGB outperforms the last Generator of GANs, and 2) our approach can significantly improve the synthetic examples generated by a GAN under differential privacy.

Datasets. We assess our method with a toy dataset drawn from a mixture of 25 Gaussians, which is commonly used to evaluate the quality of GAN (Srivastava et al., 2017; Azadi et al., 2019; Turner et al., 2019) and synthesize MNIST images. We then turn to real datasets from the American Census, and a standard machine learning dataset (Titanic).

Privacy budget. For the tasks with privacy, we set the privacy budget to be the same across all algorithms. Since Private PGB requires additional privacy budget this means that the differentially private GAN training has to be stopped earlier as compared to running only a GAN to achieve the same privacy guarantee. Our principle is to allocate the majority of the privacy budget to the GAN training, and a much smaller budget for our Private PGB method. Throughout we used 80% to 90% of the final privacy budget on DP GAN training.[2]

Utility measures. Utility of synthetic data can be assessed along two dimensions; general utility and specific utility (Snoke et al., 2018; Arnold & Neunhoeffer, 2020). General utility describes the overall distributional similarity between the real data and synthetic datasets, but does not capture specific use cases of synthetic data. To assess general utility, we use the propensity score mean squared error (pMSE) measure (Snoke et al., 2018) (detailed in the Appendix). Specific utility of a synthetic dataset depends on the specific use an analyst has in mind. In general, specific utility can be defined as the similarity of results for analyses using synthetic data instead of real data. For each of the experiments we define specific utility measures that are sensible for the respective example. For the toy dataset of 25 gaussians we look at the number of high quality samples. For the American Census data we compare marginal distributions of the synthetic data to marginal distributions of the true data and look at the similarity of regression results.

4.1 Evaluation of Non-Private PGB

Mixture of 25 Gaussians. We first examine the performance of our approach on a two dimensional dataset with a mixture of 25 multivariate Gaussian distributions, each with a covariance matrix of $0.0025I$. The left column in Figure 1 displays the training data. Each of the 25 clusters consists

[2]Our observation is that the DP GAN training is doing the "heavy lifting". Providing a good "basis" for PGB requires a substantial privacy expenditure in training DP GAN. The privacy budget allocation is a hyperparameter for PGB that could be tuned. In general, the problem of differentially private hyperparameter selection is extremely important and the literature is thin (Liu & Talwar, 2019; Chaudhuri & Vinterbo, 2013).

of $1,000$ observations. The architecture of the GAN is the same across all results.[3] To compare the utility of the synthetic datasets with the real data, we inspect the visual quality of the resultsand calculate the proportion of high quality synthetic examples similar to Azadi et al. (2019),Turner et al. (2019) and Srivastava et al. (2017).[4]

Visual inspection of the results without privacy (in the top row of Figure 1) shows that our proposed method outperforms the synthetic examples generated by the last Generator of the GAN, as well as the last Generator enhanced with DRS. PGB over the last 100 stored Generators and Discriminators trained for $T = 1,000$ update steps, and the combination of PGB and DRS, visibly improves the results. The visual impression is confirmed by the proportion of high quality samples. The data from the last GAN generator have a proportion of 0.904 high quality samples. The synthetic data after PGB achieves a higher score of 0.918. The DRS samples have a proportion of 0.826 high quality samples, and the combination of PGB and DRS a higher proportion of 0.874 high quality samples.[5]

MNIST Data. We further evaluate the performance of our method on an image generation task with the MNIST dataset. Our results are based on the DCGAN GAN architecture (Radford et al., 2015) with the KL-WGAN loss (Song & Ermon, 2020). To evaluate the quality of the generated images we use a metric that is based on the Inception score (IS) (Salimans et al., 2016), where instead of the Inception Net we use a MNIST Classifier that achieves 99.65% test accuracy. The theoretical best score of the MNIST IS is 10, and the real test images achieve a score of 9.93. Without privacy the last GAN Generator achieves a score of 8.41, using DRS on the last Generator slightly decreases the score to 8.21, samples with PGB achieve a score of 8.76, samples with the combination of PGB and DRS achieve a similar score of 8.77 (all inception scores are calculated on 5,000 samples). Uncurated samples for all methods are included in the Appendix.

4.2 Evaluation of Private PGB

Mixture of 25 Gaussians. To show how the differentially private version of PGB improves the samples generated from GANs that were trained under differential privacy, we first re-run the experiment with the two-dimensional toy data.[6] Our final value of ϵ is 1 and δ is $\frac{1}{2N}$. For the results with PGB, the GAN training contributes $\epsilon_1 = 0.9$ to the overall ϵ and the Private PGB algorithm $\epsilon_2 = 0.1$. Again a first visual inspection of the results in Figure 1 (in the bottom row) shows that post-processing the results of the last GAN Generator with Private PGB is worthwhile. Private PGB over the last 100 stored Generators and Discriminators trained for $T = 1,000$ update steps, again, visibly improves the results. Again, our visual impression is confirmed by the proportion of high quality samples. The last Generator of the differentially private GAN achieves a proportion of 0.031 high quality samples. With DRS on top of the last Generator, the samples achieve a quality score of 0.035. The GAN enhanced with Private PGB achieves a proportion of 0.044 high quality samples, the combination of Private PGB and DRS achieves a quality score of 0.053.

MNIST Data. On the MNIST data, with differential privacy ($\epsilon = 10$, $\delta = \frac{1}{2N}$) the last DP GAN Generator achieves an inception score of 8.07, using DRS on the last Generator the IS improves to 8.18. With Private PGB the samples achieve an IS of 8.58, samples with the combination of Private PGB and DRS achieve the highest IS of 8.66.[7] Uncurated samples for all methods are included in the Appendix.

Private Synthetic 1940 American Census Samples. While the results on the toy dataset are encouraging, the ultimate goal of private synthetic data is to protect the privacy of actual persons in

[3] A description of the architecture is in the Appendix. The code for the GANs and the PGB algorithm will be made available on GitHub.

[4] Note that the scores in Azadi et al. (2019) and Turner et al. (2019) do not account for the synthetic data distribution across the 25 modes. We detail our evaluation of high quality examples in the Appendix.

[5] The lower scores for the DRS samples are due to the capping penalty in the quality metric. Without the capping penalty the scores are 0.906 for the last Generator, 0.951 for PGB , 0.946 for DRS and 0.972 for the combination of PGB and DRS.

[6] To achieve DP, we trained the Discriminator with a DP optimizer as implemented in `tensorflow_privacy` or the `opacus` library. We keep track of the values of ϵ and δ by using the moments accountant (Abadi et al., 2016; Mironov, 2017).

[7] All inception scores are calculated on $5,000$ samples.

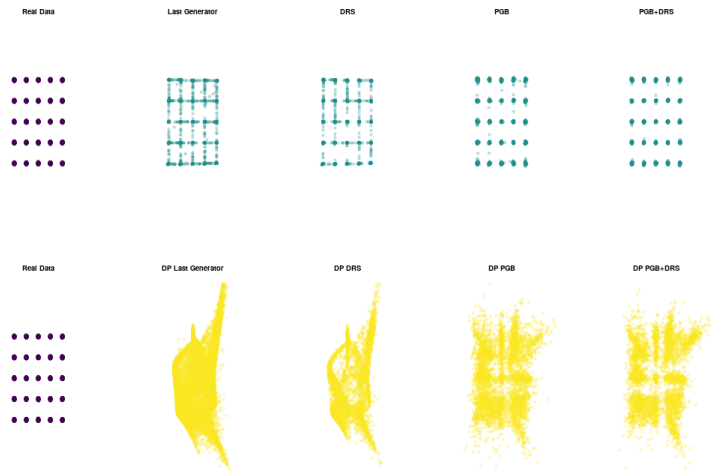

Figure 1: Real samples from 25 multivariate normal distributions, synthetic examples without privacy from a GAN, DRS, Non-Private PGB and the combination of PGB and DRS (top row). Synthetic examples from a GAN with differential privacy, DP DRS, Private PGB and the combination of Private PGB and DRS (bottom row).

data collections, and to provide useful data to interested analysts. In this section we report the results of synthesizing data from the 1940 American Census. We rely on the public use micro data samples (PUMS) as provided in Ruggles et al. (2019).[8] For 1940 we synthesize an excerpt of the 1% sample of all Californians that were at least 18 years old.[9] Our training sample consists of 39,660 observations and 8 attributes (sex, age, educational attainment, income, race, Hispanic origin, marital status and county). The test set contains another 9,915 observations. Our final value of ϵ is 1 and δ is $\frac{1}{2N} \approx 6.3 \times 10^{-6}$ (after DP GAN training with $\epsilon_1 = 0.8$ and PGB with $\epsilon_2 = 0.2, \delta_1 = \frac{1}{2N}, \delta_2 = 0$). The general utility scores as measured by the pMSE ratio score are 2.357 (DP GAN), 2.313 (DP DRS), 2.253 (DP PGB), and 2.445 (DP PGB+DRS). This indicates that PGB achieves the best general utility. To assess the specific utility of our synthetic census samples we compare one-way marginal distributions to the same marginal distributions in the original data. In panel (A) of Figure 2 we show the distribution of race membership. Comparing the synthetic data distributions to the true distribution (in gray), we conclude that PGB, improves upon the last Generator. To underpin the visual impression we calculate the total variation distance between each of the synthetic distributions and the real distribution, the data from the last GAN Generator has a total variation distance of 0.58, DP DRS of 0.44, DP PGB of 0.22 and DP PGB+DRS of 0.13. Furthermore, we evaluate whether more complex analysis models, such as regression models, trained on synthetic samples could be used to make sensible out-of-sample predictions. Panel (B) of Figure 2 shows a parallel coordinate plot to compare the out-of-sample root mean squared error of regression models trained on real data and trained on synthetic data. The lines show the RMSE for predicted income for all linear regression models trained with three independent variables from the set of on the synthetic data generated with Private PGB as compared to the last GAN generator and other post processing methods like DRS.

[8] Further experiments using data from the 2010 American Census can be found in the appendix.

[9] A 1% sample means that the micro data contains 1% of the total American (here Californian) population.

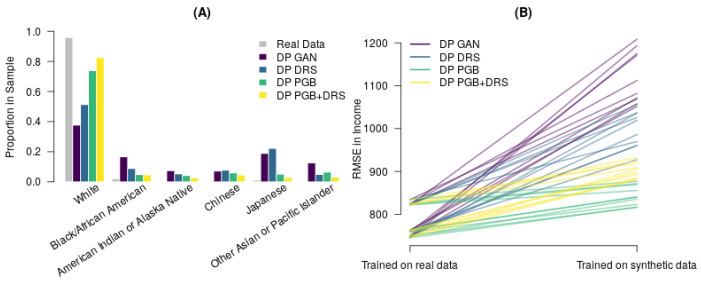

Figure 2: Specific Utility of Synthetic 1940 American Census Data. Panel (A): Distribution of Race Membership in Synthetic Samples. Panel (B): Regression RMSE with Synthetic Samples.

Machine Learning Prediction with Synthetic Data. In a final set of experiments we evaluate the performance of machine learning models trained on synthetic data (with and without privacy) and tested on real out-of-sample data. We synthesize the Kaggle Titanic[10] training set (891 observations of Titanic passengers on 8 attributes) and train three machine learning models (Logistic Regression, Random Forests (RF) (Breiman, 2001) and XGBoost (Chen & Guestrin, 2016)) on the synthetic datasets to predict whether someone survived the Titanic catastrophe. We then evaluate the performance on the test set with 418 observations. To address missing values in both the training set and the test set we independently impute values using the MissForest (Stekhoven & Bühlmann, 2012) algorithm. For the private synthetic data our final value of ϵ is 2 and δ is $\frac{1}{2N}$ (for PGB this implies DP GAN training with $\epsilon_1 = 1.6$ and PGB $\epsilon_2 = 0.4$). The models trained on synthetic data generated with our approaches (PGB and PGB+DRS) consistently perform better than models trained on synthetic data from the last generator or DRS – with or without privacy.[11]

ACKNOWLEDGMENTS

This work began when the authors were at the Simons Institute participating in the "Data Privacy: Foundations and Applications" program. We thank Thomas Steinke, Adam Smith, Salil Vadhan, and the participants of the DP Tools meeting at Harvard for helpful comments. Marcel Neunhoeffer is supported by the University of Mannheim's Graduate School of Economic and Social Sciences funded by the German Research Foundation. Zhiwei Steven Wu is supported in part by an NSF S&CC grant 1952085, a Google Faculty Research Award, and a Mozilla research grant. Cynthia Dwork is supported by the Alfred P. Sloan Foundation, "Towards Practicing Privacy" and NSF CCF-1763665.

[10] https://www.kaggle.com/c/titanic/data

[11] Table 1 in the appendix summarizes the results in more detail. We present the accuracy, ROC AUC and PR AUC to evaluate the performance.

REFERENCES

Martin Abadi, Andy Chu, Ian Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, pp. 308–318, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-4139-4. doi: 10.1145/2976749.2978318. URL http://doi.acm.org/10.1145/2976749.2978318.

John M. Abowd. The U.S. census bureau adopts differential privacy. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19-23, 2018*, pp. 2867, 2018. doi: 10.1145/3219819.3226070. URL https://doi.org/10.1145/3219819.3226070.

Martín Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein GAN. *CoRR*, abs/1701.07875, 2017. URL http://arxiv.org/abs/1701.07875.

Christian Arnold and Marcel Neunhoeffer. Really useful synthetic data–a framework to evaluate the quality of differentially private synthetic data. *arXiv preprint arXiv:2004.07740*, 2020.

Sanjeev Arora, Elad Hazan, and Satyen Kale. The multiplicative weights update method: a meta-algorithm and applications. *Theory of Computing*, 8(6):121–164, 2012. doi: 10.4086/toc.2012.v008a006. URL http://www.theoryofcomputing.org/articles/v008a006.

Sanjeev Arora, Rong Ge, Yingyu Liang, Tengyu Ma, and Yi Zhang. Generalization and equilibrium in generative adversarial nets (GANs). In Doina Precup and Yee Whye Teh (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 224–232, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR. URL http://proceedings.mlr.press/v70/arora17a.html.

Samaneh Azadi, Catherine Olsson, Trevor Darrell, Ian J. Goodfellow, and Augustus Odena. Discriminator rejection sampling. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, 2019. URL https://openreview.net/forum?id=S1GkToR5tm.

Brett K. Beaulieu-Jones, Zhiwei Steven Wu, Chris Williams, Ran Lee, Sanjeev P. Bhavnani, James Brian Byrd, and Casey S. Greene. Privacy-preserving generative deep neural networks support clinical data sharing. *Circulation: Cardiovascular Quality and Outcomes*, 12(7):e005122, 2019. doi: 10.1161/CIRCOUTCOMES.118.005122.

Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

Kamalika Chaudhuri and Staal A Vinterbo. A stability-based validation procedure for differentially private machine learning. In *Advances in Neural Information Processing Systems*, pp. 2652–2660, 2013.

Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pp. 785–794, 2016.

Differential Privacy Team, Apple. Learning with privacy at scale. https://machinelearning.apple.com/docs/learning-with-privacy-at-scale/appledifferentialprivacysystem.pdf, December 2017.

Bolin Ding, Janardhan Kulkarni, and Sergey Yekhanin. Collecting telemetry data privately. In *Advances in Neural Information Processing Systems 30*, NIPS '17, pp. 3571–3580. Curran Associates, Inc., 2017.

Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Proceedings of the 3rd Theory of Cryptography Conference*, volume 3876, pp. 265–284, 2006.

Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. RAPPOR: Randomized aggregatable privacy-preserving ordinal response. In *Proceedings of the 2014 ACM Conference on Computer and Communications Security*, CCS '14, pp. 1054–1067, New York, NY, USA, 2014. ACM.

Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119 – 139, 1997. ISSN 0022-0000. doi: https://doi.org/10.1006/jcss.1997.1504. URL http://www.sciencedirect.com/science/article/pii/S002200009791504X.

Lorenzo Frigerio, Anderson Santana de Oliveira, Laurent Gomez, and Patrick Duverger. Differentially private generative adversarial networks for time series, continuous, and discrete open data. In *ICT Systems Security and Privacy Protection - 34th IFIP TC 11 International Conference, SEC 2019, Lisbon, Portugal, June 25-27, 2019, Proceedings*, pp. 151–164, 2019. doi: 10.1007/978-3-030-22312-0_11. URL https://doi.org/10.1007/978-3-030-22312-0_11.

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, pp. 2672–2680, Cambridge, MA, USA, 2014. MIT Press. URL http://dl.acm.org/citation.cfm?id=2969033.2969125.

Moritz Hardt and Guy N. Rothblum. A multiplicative weights mechanism for privacy-preserving data analysis. In *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, October 23-26, 2010, Las Vegas, Nevada, USA*, pp. 61–70, 2010. doi: 10.1109/FOCS.2010.85. URL https://doi.org/10.1109/FOCS.2010.85.

Moritz Hardt, Katrina Ligett, and Frank McSherry. A simple and practical algorithm for differentially private data release. In *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States.*, pp. 2348–2356, 2012.

Quan Hoang, Tu Dinh Nguyen, Trung Le, and Dinh Phung. MGAN: Training generative adversarial nets with multiple generators. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=rkmu5b0a-.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

Jyrki Kivinen and Manfred K. Warmuth. Exponentiated gradient versus gradient descent for linear predictors. *Information and Computation*, 132(1):1 – 63, 1997. ISSN 0890-5401. doi: https://doi.org/10.1006/inco.1996.2612. URL http://www.sciencedirect.com/science/article/pii/S0890540196926127.

Jingcheng Liu and Kunal Talwar. Private selection from private candidates. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pp. 298–309, 2019.

Chris J Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*, 2016.

Frank McSherry and Kunal Talwar. Mechanism design via differential privacy. In *Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science*, FOCS '07, pp. 94–103, Washington, DC, USA, 2007. IEEE Computer Society. ISBN 0-7695-3010-9. doi: 10.1109/FOCS.2007.41. URL http://dx.doi.org/10.1109/FOCS.2007.41.

Ilya Mironov. Rényi differential privacy. In *30th IEEE Computer Security Foundations Symposium, CSF 2017, Santa Barbara, CA, USA, August 21-25, 2017*, pp. 263–275, 2017. doi: 10.1109/CSF.2017.11. URL https://doi.org/10.1109/CSF.2017.11.

Nicolas Papernot, Shuang Song, Ilya Mironov, Ananth Raghunathan, Kunal Talwar, and Úlfar Erlingsson. Scalable private learning with PATE. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, 2018. URL https://openreview.net/forum?id=rkZB1XbRZ.

Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

Steven Ruggles, Sarah Flood, Ronald Goeken, Josiah Grover, Erin Meyer, Jose Pacas, and Matthew Sobek. Ipums usa: Version 9.0 [dataset]. *Minneapolis, MN: IPUMS*, 10:D010, 2019. doi: 10.18128/D010.V9.0.

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *Advances in neural information processing systems*, pp. 2234–2242, 2016.

Joshua Snoke, Gillian M. Raab, Beata Nowok, Chris Dibben, and Aleksandra Slavkovic. General and specific utility measures for synthetic data. *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, 181(3):663–688, 2018. doi: 10.1111/rssa.12358. URL https://rss.onlinelibrary.wiley.com/doi/abs/10.1111/rssa.12358.

Jiaming Song and Stefano Ermon. Bridging the gap between f-gans and wasserstein gans, 2020.

Akash Srivastava, Lazar Valkov, Chris Russell, Michael U Gutmann, and Charles Sutton. Veegan: Reducing mode collapse in gans using implicit variational learning. In *Advances in Neural Information Processing Systems*, pp. 3308–3318, 2017.

Daniel J Stekhoven and Peter Bühlmann. Missforest—non-parametric missing value imputation for mixed-type data. *Bioinformatics*, 28(1):112–118, 2012.

Ilya Tolstikhin, Sylvain Gelly, Olivier Bousquet, Carl-Johann Simon-Gabriel, and Bernhard Schölkopf. Adagan: Boosting generative models. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pp. 5430–5439, USA, 2017. Curran Associates Inc. ISBN 978-1-5108-6096-4. URL http://dl.acm.org/citation.cfm?id=3295222.3295294.

Reihaneh Torkzadehmahani, Peter Kairouz, and Benedict Paten. DP-CGAN: differentially private synthetic data and label generation. *CoRR*, abs/2001.09700, 2020. URL https://arxiv.org/abs/2001.09700.

Ryan Turner, Jane Hung, Eric Frank, Yunus Saatchi, and Jason Yosinski. Metropolis-Hastings generative adversarial networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 6345–6353, Long Beach, California, USA, 09–15 Jun 2019. PMLR. URL http://proceedings.mlr.press/v97/turner19a.html.

Liyang Xie, Kaixiang Lin, Shu Wang, Fei Wang, and Jiayu Zhou. Differentially private generative adversarial network. *CoRR*, abs/1802.06739, 2018. URL http://arxiv.org/abs/1802.06739.

Jinsung Yoon, James Jordon, and Mihaela van der Schaar. PATE-GAN: Generating synthetic data with differential privacy guarantees. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=S1zk9iRqF7.

A PROOFS

A.1 PROOF OF THEOREM 1

Proof of Theorem 1. Note that if the synthetic data player plays the distribution over X, then $U(p_X, D) = \mathbb{E}_{x\sim p_X}[D(x')] + \mathbb{E}_{x\sim\phi}[1 - D(x)] = 1$ for any discriminator $D \in \mathcal{D}$. Now let us replace each element in X with its γ-approximation in B and obtain a new dataset X_B, and let p_{X_B} denote the empirical distribution over X_B. By the Lipschitz conditions, we then have $|U(p_X, D) - U(p_{X_B}, D)| \leq L\,\gamma$. This means $U(p_{X_B}, D) \in [1 - L\gamma, 1 + L\gamma]$ for all D. Also, for all $\phi \in \Delta(B)$, we have $U(\phi, D^{1/2}) = 1$. Thus, $(p_{X_b}, D^{1/2})$ satisfies equation 1 with $\alpha = L\gamma$. □

A.2 PROOF OF THE APPROXIMATE EQUILIBRIUM

Proof. We will use the seminal result of Freund & Schapire (1997), which shows that if the two players have low regret in the dynamics, then their average plays form an approximate equilibrium. First, we will bound the regret from the data player. The regret guarantee of the multiplicative weights algorithm (see e.g. Theorem 2.3 of Arora et al. (2012)) gives

$$\sum_{t=1}^{T} U(\phi^t, D^t) - \min_{b\in B} \sum_{t=1}^{T} U(b, D^t) \leq 4\eta T + \frac{\log|B|}{\eta} \tag{3}$$

Next, we bound the regret of the distinguisher using the accuracy guarantee of the exponential mechanism (McSherry & Talwar, 2007). For each t, we know with probability $(1 - \beta/T)$,

$$\max_{D_j} U(\phi^t, D_j) - U(\phi^t, D^t) \leq \frac{2\log(NT/\beta)}{n\epsilon_0}$$

Taking a union bound, we have this accuracy guarantee holds for all t, and so

$$\max_{D_j} \sum_{t=1}^{T} U(\phi^t, D_j) - \sum_{t=1}^{T} U(\phi^t, D^t) \leq \frac{2T\log(NT/\beta)}{n\epsilon_0} \tag{4}$$

Then following the result of Freund & Schapire (1997), their average plays $(\overline{D}, \overline{\phi})$ is an α-approximate equilibrium with

$$\alpha = 4\eta + \frac{\log|B|}{\eta T} + \frac{2\log(NT/\beta)}{n\epsilon_0}$$

Plugging in the choices of T and η gives the stated bound. □

B ADDITIONAL DETAILS ON THE QUALITY EVALUATION

B.1 ON THE CALCULATION OF THE PMSE.

To calculate the pMSE one trains a discriminator to distinguish between real and synthetic examples. The predicted probability of being classified as real or synthetic is the propensity score. Taking all propensity scores into account the mean squared error between the propensity scores and the proportion of real data examples is calculated. A synthetic dataset has high general utility, if the model can at best predict probabilities of 0.5 for both real and synthetic examples, then the pMSE would be 0.

B.2 SPECIFIC UTILITY MEASURE FOR THE 25 GAUSSIANS.

In the real data, given the data generating process outlined in section 4.1, at each of the 25 modes 99% of the observations lie within a circle with radius $r = \sqrt{0.0025 \cdot 9.21034}$ around the mode centroids, where 9.21034 is the critical value at $p = 0.99$ of a χ^2 distribution with 2 degrees of freedom, and 0.0025 is the variance of the spherical gaussian.

To calculate the quality score we count the number of observations within each of these 25 circles. If one of the modes contains more points than we would expect given the true distribution the count is capped accordingly. Our quality score for the toy dataset of 25 gaussians can be expressed as $Q = \sum_i^{25}(\min(p^i_{real} \cdot N_{syn}, N^i_{syn})/N_{syn})$, where i indexes the clusters, p_{real} is the true distribution of points per cluster, N^i_{syn} the number of observations at a cluster within radius r, and N_{syn} the total number of synthetic examples.

C GAN ARCHITECTURES

C.1 DETAILS ON THE EXPERIMENTS WITH THE 25 GAUSSIANS.

The generator and discriminator are neural nets with two fully connected hidden layers (Discriminator: 128, 256; Generator: 512, 256) with Leaky ReLu activations. The latent noise vector Z is of dimension 2 and independently sampled from a gaussian distribution with mean 0 and standard deviation of 1. For GAN training we use the KL-WGAN loss (Song & Ermon, 2020). Before passing the Discriminator scores to PGB we transform them to probabilities using a sigmoid activation.

C.2 GAN ARCHITECTURE FOR THE 1940 AMERICAN CENSUS DATA.

The GAN networks consist of two fully connected hidden layers (256, 128) with Leaky ReLu activation functions. To sample from categorical attributes we apply the Gumbel-Softmax trick (Maddison et al., 2016; Jang et al., 2016) to the output layer of the Generator. We run our PGB algorithm over the last 150 stored Generators and Discriminators and train it for $T = 400$ update steps.

D PRIVATE SYNTHETIC 2010 AMERICAN DECENNIAL CENSUS SAMPLES.

We conducted further experiments on more recent Census files. The 2010 data is similar to the data that the American Census is collecting for the 2020 decennial Census. For this experiment, we synthesize a 10% sample for California with 3,723,669 observations of 5 attributes (gender, age, Hispanic origin, race and puma district membership). Our final value of ϵ is 0.795 and δ is $\frac{1}{2N} \approx 1.34 \times 10^{-7}$ (for PGB the GAN training contributes $\epsilon = 0.786$ and PGB $\epsilon = 0.09$). The pMSE ratio scores are 1.934 (DP GAN), 1.889 (DP DRS), 1.609 (DP PGB) and 1.485 (DP PGB+DRS), here PGB achieves the best general utility. For specific utility, we compare the accuracy of three-way marginals on the synthetic data to the proportions in the true data.[12] We tabulate race (11 answer categories in the 2010 Census) by Hispanic origin (25 answer categories in the 2010 Census) by gender (2 answer categories in the 2010 Census) giving us a total of 550 cells. To assess the specific utility for these three-way marginals we calculate the average accuracy across all 550 cells. Compared to the true data DP GAN achieves 99.82%, DP DRS 99.89%, DP PGB 99.89% and the combination of DP PGB and DRS 99.93%. Besides the average accuracy across all 550 cells another interesting metric of specific utility is the number of cells in which each synthesizer achieves the highest accuracy compared to the other methods, this is the case 43 times for DP GAN, 30 times for DP DRS, 90 times for DP PGB and 387 times for DP PGB+DRS. Again, this shows that our proposed approach can improve the utility of private synthetic data.

E DETAILED RESULTS OF MACHINE LEARNING PREDICTION WITH SYNTHETIC DATA

Table 1 summarizes the results for the machine learning prediction experiment with the Titanic data. We present the accuracy, ROC AUC and PR AUC to evaluate the performance. It can be seen that the models trained on synthetic data generated with our approach consistently perform better than models trained on synthetic data from the last generator or DRS – with or without privacy. To put these values into perspective, the models trained on the real training data and tested on the same out-of-sample data achieve the scores in table 2.

[12] A task that is similar to the tables released by the Census.

Table 1: Predicting Titanic Survivors with Machine Learning Models trained on synthetic data and tested on real out-of-sample data. Median scores of 20 repetitions with independently generated synthetic data. With differential privacy ϵ is 2 and δ is $\frac{1}{2N} \approx 5.6 \times 10^{-4}$.

	GAN	DRS	PGB	PGB + DRS
Logit Accuracy	0.626	0.746	0.701	**0.765**
Logit ROC AUC	0.591	0.760	0.726	**0.792**
Logit PR AUC	0.483	0.686	0.655	**0.748**
RF Accuracy	0.594	0.724	0.719	**0.742**
RF ROC AUC	0.531	0.744	0.741	**0.771**
RF PR AUC	0.425	0.701	0.706	**0.743**
XGBoost Accuracy	0.547	0.724	0.683	**0.740**
XGBoost ROC AUC	0.503	0.732	0.681	**0.772**
XGBoost PR AUC	0.400	0.689	0.611	**0.732**
	DP GAN	DP DRS	DP PGB	DP PGB +DRS
Logit Accuracy	0.566	0.577	0.640	**0.649**
Logit ROC AUC	0.477	0.568	0.621	**0.624**
Logit PR AUC	0.407	0.482	0.532	**0.547**
RF Accuracy	0.487	0.459	0.481	**0.628**
RF ROC AUC ROC AUC	0.512	0.553	0.558	**0.652**
RF PR AUC PR AUC	0.407	0.442	0.425	**0.535**
XGBoost Accuracy	0.577	0.589	0.609	**0.641**
XGBoost ROC AUC	0.530	0.586	**0.619**	0.596
XGBoost PR AUC	0.398	0.479	0.488	**0.526**

Table 2: Predicting Titanic Survivors with Machine Learning Models trained on real data and tested on real out-of-sample data.

Model	Score
Logit Accuracy	0.764
Logit ROC AUC	0.813
Logit PR AUC	0.785
RF Accuracy	0.768
RF ROC AUC	0.809
RF PR AUC	0.767
XGBoost Accuracy	0.768
XGBoost ROC AUC	0.773
XGBoost PR AUC	0.718

F SYNTHETIC MNIST SAMPLES

Figure 3 shows uncurated samples from the last Generator after 30 epochs of training without differential privacy in Panel 3a and with differential privacy ($\epsilon =$, $\delta =$) in Panel 3b. Figure 4 shows uncurated samples with DRS on the last Generator. Figure 5 shows uncurated samples after PGB and Figure 6 shows uncurated samples after the combination of PGB and DRS. In Figure 7 we show the 100 samples with the highest PGB probabilities.

(a) Without Differential Privacy.

(b) With Differential Privacy ($\epsilon = 10$, $\delta = \frac{1}{2N}$)

Figure 3: Uncurated MNIST samples from last GAN Generator after 30 epochs.

(a) Without Differential Privacy.

(b) With Differential Privacy ($\epsilon = 10$, $\delta = \frac{1}{2N}$)

Figure 4: Uncurated MNIST samples with DRS on last Generator after 30 epochs.

(a) Without Differential Privacy.

(b) With Differential Privacy ($\epsilon = 10$, $\delta = \frac{1}{2N}$)

Figure 5: Uncurated MNIST samples with PGB after 30 epochs (without DP) and 25 epochs (with DP).

(a) Without Differential Privacy.

(b) With Differential Privacy ($\epsilon = 10$, $\delta = \frac{1}{2N}$)

Figure 6: Uncurated MNIST samples with PGB and DRS after 30 epochs (without DP) and 25 epochs (with DP).

(a) Without Differential Privacy.

(b) With Differential Privacy ($\epsilon = 10$, $\delta = \frac{1}{2N}$)

Figure 7: Top 100 MNIST samples after PGB after 30 epochs (without DP) and 25 epochs (with DP).

4

A Common Benchmark to Evaluate Multiple Imputation Algorithms

A Common Benchmark to Evaluate Multiple Imputation Algorithms

Marcel Neunhoeffer
LMU Munich & Boston University

Abstract

Multiple imputation (MI) of missing data is an essential tool for quantitative research in political science. Many different MI models are available, and new MI models are introduced to the political science literature. However, it needs to be clarified how to evaluate new MI models or compare existing ones. This, in turn, means for applied researchers that, it is hard to choose between imputation models. This letter presents a new benchmark for evaluating multiple imputation (MI) models. The R-package `MIBench` accompanying this letter makes applying the benchmark and comparing existing imputation models easy. I also provide benchmark results and show that no one-fits-all solution to handle missing data exists. It is striking that seemingly more flexible imputation methods, e.g., based on neural nets, do not perform well. Furthermore, I show that comparing results from MI to other methods, like complete case analysis on the same data set, can only be conclusive with further assumptions. The results highlight a need for careful evaluation of MI models before using them in applied research. I conclude with practical advice for applied researchers and developers of new imputation algorithms.

Abstract: 187 words
Manuscript: 2959 words

Multiple imputation (MI) of missing data is an essential tool for quantitative research in political science. Many different MI models are available and new MI models are introduced. However, it needs to be clarified how to evaluate new MI models or compare existing ones. This, in turn, means for applied researchers that, it is hard to choose between imputation models. In this letter, I propose a new benchmark for MI models. I also show the results of the benchmark with established imputation models. These results highlight that developing new MI models and choosing imputation models for applied research needs to be done more carefully. In particular, seemingly more flexible imputation methods (e.g., based on neural nets) fail to achieve the goal of statistically valid inferences in almost all benchmark tasks. Furthermore, I show that comparing results from MI to other methods, like complete case analysis on the same data set, can only be conclusive with further assumptions. The benchmark is accompanied by the R-package `MIBench` which makes it easy to apply.

1 Background on Multiple Imputation

MI of missing data is a well-established (Rubin, 1978, 1996, 2004) and popular general-purpose framework to handle missing values. The appeal of MI comes from the fact that the analysis model (e.g., a regression model) does not need any modification. It just needs to be run repeatedly on multiple completed data sets. Then the results can be combined according to Rubin's rules (Rubin, 1987). MI is easy to implement, as different software packages are available.However, it is important to remember the central goal of MI; uncertainty propagation for proper inference (Rubin, 1996). To achieve statistically valid inference with MI two assumptions need to hold. First, missing data needs to be at least missing at random (MAR)[1] meaning that the missingness is random conditional on observed values of the data[2]. Otherwise, if missing

[1]Definitions of the different missingness mechanisms can be found in the supporting information SI.2.

[2]Note that it is also possible that even with MAR missingness of some variables, the statistic of interest can be recovered without multiple imputation (Arel-Bundock and Pelc, 2018; Mohan and Pearl, 2021; Pepinsky, 2018).

values depend on unobserved values, no statistical or machine learning model based on observed values can generate reasonable imputations. Second, the relationship between the imputation and analysis models is important (Xie and Meng, 2017). For valid inference, the analysis model needs to be congenial to the imputation model, i.e., the imputation model should contain the analysis model as a sub-model and potentially be more general than the analysis model to allow for more analyses (Bartlett et al., 2015; Meng, 1994; van Buuren, 2018). Suppose the analysis model assumes more than the imputation model. For example, if it includes an interaction term that was not included in the imputation model, the results can be seriously biased (Bartlett et al., 2015).

1.1 Multiple Imputation in Political Science

The importance of MI in quantitative political science research can be seen both in applications as well as in methodological contributions. I reviewed 57 papers with the term "multiple imputation" anywhere in the text published in some of the top journals of the discipline[3] between 2017 and 2022[4]. Of these 57 papers, 38 are applications that used MI as part of their analyses. Of these 38 applied papers, 18 used the `amelia` (Honaker et al., 2011) R-package, nine used the `Stata mi` (see, e.g., Royston, 2004) functions, five used the `mice` (van Buuren and Groothuis-Oudshoorn, 2011) R-package, three the `mi` (Su et al., 2011) R-package, and the authors of one paper wrote their own custom MI model.[5] Four papers are methodological contributions (Arel-Bundock and Pelc, 2018; Lall and Robinson, 2022; Marbach, 2022; Pepinsky, 2018)[6].

In addition to the methodological contributions above, several new MI models have been introduced in and to the political science literature over the past two decades. Starting with

[3]For this study, I include the American Political Science Review, the American Journal of Political Science, the Journal of Politics and Political Analysis.

[4]The search query and table SI.1 with all papers included in this analysis can be found in the Supporting Information SI.1.

[5]For two out of the 36 applied papers, I could not determine the MI model the authors used.

[6]The remaining 15 papers showed up in the search because they have the term "multiple imputation" somewhere in the full text but do not use MI in their analyses.

`Amelia` in the seminal paper by King et al. (2001), several multiple imputation models like `hotdeck` (Cranmer and Gill, 2013), and `mi` (Kropko et al., 2014) have been introduced in political science journals. Furthermore, other MI models like in Hollenbach et al. (2021) have been co-developed by political scientists.

Recent advances in imputation models in and outside of political science focus on non-parametric and modern machine learning model based imputation models (see, e.g., Hollenbach et al., 2021; Lall and Robinson, 2022; Stekhoven and Bühlmann, 2012; Yoon et al., 2018). The goals and hopes of these novel models are to offer more flexible ways to learn the joint distribution and, in turn, reduce the burden on the imputer of correctly specifying the imputation model.

With so many different MI models available, it is hard for developers of new imputation models and for applied researchers to compare them directly.Without good comparisons of different imputation models, there is also no clear guidance for applied researchers who want to choose an MI model for their analyses.

2 The Multiple Imputation Benchmark

My main contribution is a MI benchmark to standardize the Monte Carlo experiments to evaluate competing MI models on common tasks. As "there is essentially only one scientific way to evaluate and compare statistical procedures — show how they work when applied repeatedly, in reality, via simulation or in thought experiments" (Xie and Meng, 2017, 1491). By using a common benchmark, the results of new methods can be directly compared with existing methods without the need to rerun all Monte Carlo experiments on all combinations of data-generating processes and missingness mechanisms. For the initial version of `MIBench`, I focus on combinations of data-generating processes and missingness mechanisms used in previous MI papers, with the addition of one new data-generating process.

2.1 Data-Generating Processes and Missingness Mechanisms

I include four different data-generating processes that highlight different aspects of challenges that applied political scientists might encounter when they want to use MI models to handle missing data.

The Amelia experiments. The first set of experiments reruns the experiments from the first Amelia paper King et al. (2001). The complete data is drawn from a multivariate normal distribution, and missingness is induced according to five missingness mechanisms (two MAR mechanisms, two MCAR mechanisms, and one NI mechanism). The goal is to recover the coefficients of a linear regression model.

The Hot Deck experiments. The second set of experiments was first introduced by (Cranmer and Gill, 2013). Here the focus is on binary variables with three different MAR missingness mechanisms, which essentially differ by the amount of missing data. The goal is to recover the coefficients of a logistic regression model.

The Marbach experiments. The third data-generating process was first introduced by Marbach (2022). It combines continuous and binary variables with the added difficulty of an interaction effect in the data-generating process and one MAR missingness mechanism. The goal is to recover the regression coefficients of a linear model.

The Mixed experiments. The setup of the fourth and novel data-generating process is similar to the one introduced by Marbach (2022). It also combines continuous and binary variables with the added difficulty of an interaction effect in the data-generating process. However, the interaction effect is more influential, and MCAR and MAR missingness mechanisms induce missingness. The goal is to recover the regression coefficients of a linear model.

These four data-generating processes, with their respective missingness mechanisms, make up a set of eleven different experiments. I describe the data-generating processes, missingness mechanisms, and the analysis models in further detail in the Supporting Information SI.3.

2.2 Measures to Evaluate Inferential Quality

Two quality measures of inferential quality that also facilitate a direct comparison of different MI models are *empirical coverage* and the *width* of confidence intervals of regression

coefficients (see also van Buuren, 2018). If CI_l^s and CI_u^s are the lower and upper limit of the confidence interval for a parameter of interest θ, for a particular run of the procedure s, with S total Monte Carlo simulation repetitions, then the empirical coverage can be defined as the proportion of confidence intervals that cover the true value $\frac{1}{S}\sum_{s=1}^{S}\mathbb{I}(CI_l^s \leq \theta \leq CI_u^s)$, where $\mathbb{I}$ is the indicator function. The width of a confidence interval is the difference between the upper and the lower limit of the confidence interval, $CI_u^s - CI_l^s$ of a given repetition of the Monte Carlo simulation s.

The best MI model for a particular task achieves empirical coverage of at least the nominal significance level with the shortest confidence interval among the competing models.

3 Results

Table 1 summarizes the coverage rate and width of confidence intervals of 1000 repeated runs of nine different MI models on each of the eleven combinations of data-generating processes and missingness mechanisms. For each run, I sample a new data set from the data-generating process, induce missingness according to the respective missingness mechanism and then run each imputation model on the same data sets with missing data to generate $m = 10$ completed data sets with each method. I then analyze each completed data set with the respective analysis model and combine the results according to Rubin's rules (Rubin, 1987) (assuming that the imputation model and analysis model are congenial, indicated with a ^{C} in table 1) and according to the combining rules by Xie and Meng (2017) (assuming that the imputation model and analysis model are uncongenial, indicated with a ^{U} in table 1).[7]

In each cell of table 1, I report the proportion of the shortest confidence intervals with valid inference (i.e., coverage of at least the nominal level) of a MI model compared against all other MI models for that experiment. Empty cells mean that a given MI model did not provide valid inference.

For example, in the Amelia experiment with MAR-1 missingness, the MI model Amelia II with Rubin's rules to combine the results produced the shortest intervals in 41% of the

[7]Both combining rules are included in SI.2.

Table 1: Proportion of shortest confidence intervals with valid inference. Results of running `MIBench` for eleven different combinations of data-generating processes and missingness mechanisms with nine popular MI models. The results from each MI model are combined according to Rubin's rules for congenial models (^{C}) (Rubin, 1987) and the combining rules for uncongenial models (^{U}) suggested by Xie and Meng (2017). Each column presents the results for one combination of a data-generating process and a missingness mechanism. If a cell for a MI model in a column is empty, the MI model did not produce valid inferences (the confidence intervals did not cover the true parameters in at least 0.938—accounting for Monte Carlo error with 1000 repetitions—of all confidence intervals for a nominal 0.95 confidence level). A 0 in a cell means that the MI model produced valid confidence intervals, but compared to other MI models or complete case analyses (on the same data), they were never the shortest. The cell highlighted in **bold** had the highest proportion of valid shortest intervals for that combination of data-generating process and missingness mechanism.

	Amelia					Hot Deck			Marbach	Mixed	
	MAR-1	MAR-2	MCAR-1	MCAR-2	NI	MAR-1	MAR-2	MAR-3	MAR	MAR	MCAR
Joint MVN											
Amelia II^C	**0.41**		0.18	0.15							
Amelia II^U	0	0	0	0		0					
Conditional											
hotdeck^C			0.14	0.2		0.03	**0.99**				
hotdeck^U	0	0	0	0		0	0				
mi^C	0.1	**0.54**	0.15	0.12		0.03		**0.92**			
mi^U	0	0	0	0		0	0	0			
mice hd^C	0.12	0.45	**0.21**	0.18		0			**0.51**		
mice hd^U	0	0	0	0		0	0		0		0
mice pmm^C	0.36		0.18	0.14		0	0.01	0.08			
mice pmm^U	0	0	0	0		0	0	0			
mice rf^C			0.15	**0.2**		0.04					
mice rf^U	0	0	0	0		0	0				
Neural Net											
rMIDAS 1^C											
rMIDAS 1^U			0			0					
rMIDAS 2^C						**0.89**					
rMIDAS 2^U			0			0	0				
GAIN^C											
GAIN^U			0			0					
Complete case			0			0			0.49		**1**

simulations.For the Amelia MAR-1 experiment, none of the neural net based imputation models provided valid inference. Furthermore, unsurprisingly, none of the methods provide valid inference in the Amelia setting with non-ignorable (NI) missingness.

I want to highlight two main results. First, applying the benchmark to established MI models shows that currently, no single best solution to handle missing values exists. The **bold** cells are scattered throughout the table. This means that for most experiments, a different MI model produces the shortest valid confidence intervals.

Second, novel neural net based MI models failed to produce valid inferences under almost all combinations of missingness mechanisms and data-generating processes. With the default values of `rMIDAS` (rMIDAS 1 in table 1) and Rubin's congenial combining rules as suggested by Lall and Robinson (2022), the method did not produce valid inference for any of the experiments. The same is true for GAIN with the default values suggested by Yoon et al. (2018) and Rubin's rules to combine the results.

For further analysis of the results, I include figures for each of the eleven runs combinations of data-generating processes and missingness mechanisms in the Supporting Information SI.6.

3.1 What can I conclude when the results from MI and complete case analysis are (not) different?

The answer depends on the data-generating process, the missingness process, and the interplay of the imputation and analysis model. The results from the Monte Carlo simulations illustrate all five cases.

Case 1: The results of both MI and complete case analysis are similar, and both provide valid inferences. This can be the case when the data is MCAR, and both the imputation and analysis models are correctly specified. An example is the Amelia MCAR-1 experiment. In this case, the question is, which method is using your available data more efficiently? As the results in table 1 show, several imputation models, provide shorter confidence intervals than complete case analysis. Almost all methods provide valid inferences.

Case 2: The results of both MI and complete case analysis are similar, and both provide invalid inferences. An example is an MCAR missingness mechanism, where the imputation and

analysis models are congenial but wrongly specified (i.e., both the imputer and the analyst forget important interaction terms). This happens, for example, in the Mixed MCAR experiment if the analyst also forgets the interaction term in the analysis model.

Case 3: The results of MI and complete case analysis are different, and both provide invalid inferences. This happens, for example, in the Amelia NI experiment when the missingness mechanism is non-ignorable.

Case 4: The results of MI and complete case analysis are different, and MI provides valid inferences. A good example is the Amelia MAR-1 experiment, with MAR missingness and correctly specified imputation and analysis models. This is one of the original example cases for MI. When the data is MAR, results from complete case analysis provide invalid inferences, as they are biased. A correctly specified imputation model combined with a congenial analysis model recovers unbiased valid inferences.

Case 5: The results of MI and complete case analysis are different, and complete case analysis provides valid inferences. The Mixed MCAR experiment is an excellent example for this case. If the data are MCAR but the imputation model is wrongly specified, complete case analysis recovers unbiased valid inferences, whereas, most MI models fail.

In applied settings, understanding similarities and differences between results from complete case analysis and MI only makes sense with further reasoning about the data-generating process and the missingness mechanism. Observing similarities or differences between estimates of MI and complete case analysis is not enough to decide which results are trustworthy.

4 Conclusion

In this letter, I present a novel benchmark to evaluate MI models. Applying the benchmark to established MI models shows that currently, no single best solution to handle missing values exists. Novel neural net based MI models—like MIDAS (Lall and Robinson, 2022) and GAIN (Yoon et al., 2018)—failed to produce valid inferences under almost all combinations of missingness mechanisms and data-generating processes. The results also highlight that (no) differences between results from MI models and complete case analysis should be interpreted with caution.

This leads me to offer concrete advice for applied political scientists, where answers to the following questions could go a long way in improving empirical practice:

- Do you need MI to recover your statistic of interest?
- If so, what assumptions about the data-generating process and missingness mechanism do you need to make for a particular MI model?
- Are the MI model and the analysis model compatible/congenial?

Furthermore, showing a difference (or no difference) between the results of complete case analysis in comparison to MI is neither a necessary nor sufficient condition to conclude that results with either method are better. An interesting future research direction to guide researchers to reason about missingness in the context of their analyses could be graphical models as proposed by Mohan and Pearl (2021).

Researchers developing new algorithms should have answers to the following questions:

- Under what conditions does your proposed MI model improve on established MI models?
- What are the failure cases of your new model?

The benchmark can be a useful tool to evaluate new MI models against existing methods and facilitate the development of new MI models that are useful for applied researchers.

Software

A software implementation of the proposed method is available as an open-source `R` package, `MIBench`, at https://github.com/mneunhoe/MIBench.

References

Arel-Bundock, V. and K. J. Pelc (2018). When can multiple imputation improve regression estimates? *Political Analysis 26*(2), 240–245.

Bartlett, J. W., S. R. Seaman, I. R. White, and J. R. Carpenter (2015, Aug). Multiple imputation of covariates by fully conditional specification: Accommodating the substantive model. *Statistical Methods in Medical Research 24*(4), 462–487.

Cranmer, S. J. and J. Gill (2013). We have to be discrete about this: A non-parametric imputation technique for missing categorical data. *British Journal of Political Science 43*(2), 425–449.

Hollenbach, F. M., I. Bojinov, S. Minhas, N. W. Metternich, M. D. Ward, and A. Volfovsky (2021). Multiple imputation using gaussian copulas. *Sociological Methods & Research 50*(3), 1259–1283.

Honaker, J., G. King, and M. Blackwell (2011). Amelia II: A program for missing data. *Journal of Statistical Software 45*(7), 1–47.

King, G., J. Honaker, A. Joseph, and K. Scheve (2001). Analyzing Incomplete Political Science Data: An Alternative Algorithm for Multiple Imputation. *American Political Science Review 95*(1), 49–69.

Kropko, J., B. Goodrich, A. Gelman, and J. Hill (2014). Multiple imputation for continuous and categorical data: Comparing joint multivariate normal and conditional approaches. *Political Analysis 22*(4), 497–519.

Lall, R. and T. Robinson (2022). The midas touch: Accurate and scalable missing-data imputation with deep learning. *Political Analysis 30*(2), 179–196.

Marbach, M. (2022). Choosing imputation models. *Political Analysis 30*(4), 597–605.

Meng, X.-L. (1994). Multiple-imputation inferences with uncongenial sources of input. *Statistical Science 9*(4), 538–558.

Mohan, K. and J. Pearl (2021). Graphical models for processing missing data. *Journal of the American Statistical Association 116*(534), 1023–1037.

Pepinsky, T. B. (2018). A note on listwise deletion versus multiple imputation. *Political Analysis 26*(4), 480–488.

Royston, P. (2004). Multiple imputation of missing values. *The Stata Journal 4*(3), 227–241.

Rubin, D. B. (1978). Multiple imputations in sample surveys-a phenomenological bayesian approach to nonresponse. In *Proceedings of the survey research methods section of the American Statistical Association*, Volume 1, pp. 20–34. American Statistical Association Alexandria, VA, USA.

Rubin, D. B. (1987). *Multiple Imputation for Nonresponse in Surveys*. New York: Wiley.

Rubin, D. B. (1996). Multiple imputation after 18+ years. *Journal of the American statistical*

Association 91(434), 473–489.

Rubin, D. B. (2004). *Multiple imputation for nonresponse in surveys*, Volume 81. John Wiley & Sons.

Stekhoven, D. J. and P. Bühlmann (2012). Missforest-Non-parametric missing value imputation for mixed-type data. *Bioinformatics 28*(1), 112–118.

Su, Y.-S., A. Gelman, J. Hill, and M. Yajima (2011). Multiple imputation with diagnostics (mi) in r: Opening windows into the black box. *Journal of Statistical Software 45*(2), 1–31.

van Buuren, S. (2018). *Flexible Imputation of Missing Data. Second Edition.* Boca Raton, FL.: CRC Press.

van Buuren, S. and K. Groothuis-Oudshoorn (2011). mice: Multivariate imputation by chained equations in r. *Journal of Statistical Software 45*(3), 1–67.

Xie, X. and X. L. Meng (2017). Dissecting multiple imput a tion from a multi-phase inference perspective: What happens when god's, imputer's and anal Yst's models are uncongenial? *Statistica Sinica 27*(4), 1485–1545.

Yoon, J., J. Jordon, and M. van der Schaar (2018, 10–15 Jul). Gain: Missing data imputation using generative adversarial nets. In J. Dy and A. Krause (Eds.), *Proceedings of the 35th International Conference on Machine Learning*, Volume 80 of *Proceedings of Machine Learning Research*, pp. 5689–5698. PMLR.

Supporting Information:

A Common Benchmark to Evaluate Multiple Imputation Algorithms

Marcel Neunhoeffer[1,2].

[1]*LMU Munich*.

[2]*Boston University*.

Abstract

This is the Supporting Information for the letter "A Common Benchmark to Evaluate Multiple Imputation Algorithms".

Multiple imputation (MI) of missing data is an essential tool for quantitative research in political science. Many different MI models are available, and new MI models are introduced to the political science literature. However, it needs to be clarified how to evaluate new MI models or compare existing ones. This, in turn, means for applied researchers that, it is hard to choose between imputation models. This letter presents a new benchmark for evaluating multiple imputation (MI) models. The R-package `MIBench` accompanying this letter makes applying the benchmark and comparing existing imputation models easy. I also provide benchmark results and show that no one-fits-all solution to handle missing data exists. It is striking that seemingly more flexible imputation methods, e.g., based on neural nets, do not perform well. Furthermore, I show that comparing results from MI to other methods, like complete case analysis on the same data set, can only be conclusive with further assumptions. The results highlight a need for careful evaluation of MI models before using them in applied research. I conclude with practical advice for applied researchers and developers of new imputation algorithms.

SI.1 Overview of "multiple imputation" in Political Science

To get an overview of the use of MI in political science research, I searched for the term "multiple imputation" in articles published in some of the top political science journals (in particular, the American Journal of Political Science, the American Political Science Review, the Journal of Politics, and Political Analysis) between 2017 and 2022.

The following Google Scholar search query reproduces my search:

https://scholar.google.com/scholar?start=0&q=%22multiple+imputation%22+source:%22Political+Analysis%22+OR+source:%22American+Political+Science+Review%22+OR+source:%22American+Journal+of+Political+Science%22+OR+source:%22Journal+of+Politics%22&hl=en&as_sdt=0,22&as_ylo=2017&as_yhi=2022.

The search returned the 57 articles I present in table SI.1. I then coded each paper on whether the authors use MI for their main analyses, additional analyses, or whether it is an MI methods paper or has nothing to do with MI. To get an overview of what MI methods are popular in the field, I also coded which imputation model/software package the authors used (in some cases, I could not determine the MI model).

38 of the 57 papers are applied studies, of which 24 use MI for their main analysis, and 14 are applications where MI was used for additional analyses (i.e., robustness checks or additional results in an Appendix). Four papers are MI methods papers (all published in Political Analysis), and 15 papers did neither apply MI for their analyses nor are MI methods papers.

As for MI models (as indicated by the use of statistical software packages) used in applied studies, I could determine the used model for 36 of the 38 studies. 18 used the `amelia` (Honaker et al., 2011) package, nine used the `Stata mi` (see, e.g., Royston, 2004) functions, five used the `mice` (van Buuren and Groothuis-Oudshoorn, 2011) R-package, three the `mi` (Su et al., 2011) R-package and one paper wrote their own custom MI model.

Table SI.1: Overview of the 57 "multiple imputation" papers.

Article	MI for main analyses	MI for additional analyses	MI methods paper	Other (no direct use of MI)	MI model
Bonica and Sen (2017)	✓				amelia
Busby et al. (2017)		✓			Stata mi
Coppock et al. (2017)				✓	
Grossman et al. (2017)		✓			Stata mi
Hemker and Rink (2017)		✓			Stata mi
De Kadt (2017)		✓			amelia
Osgood et al. (2017)	✓				amelia
Pietryka and DeBats (2017)	✓				amelia
Rueda (2017)	✓				amelia
Smidt (2017)	✓				amelia
Solt et al. (2017)	✓				mi
Arel-Bundock and Pelc (2018)			✓		
Beazer and Blake (2018)		✓			amelia
Margolis (2018)				✓	
Pan and Xu (2018)	✓				amelia
Pepinsky (2018)			✓		
Rueda (2018)		✓			Stata mi
Slapin et al. (2018)				✓	
Alvarez et al. (2019)				✓	
Carroll and Kenkel (2019)	✓				amelia
Caughey and Warshaw (2019)				✓	
Dorsch and Maarek (2019)	✓				Stata mi
Hangartner et al. (2019)	✓				mice
Harris and Posner (2019)	✓				custom
Pardos-Prado and Sagarzazu (2019)	✓				Stata mi
Schultz and Mankin (2019)				✓	
Wing (2019)				✓	
Beckman and Schleiter (2020)	✓				amelia
Lachapelle (2020)	✓				mice
Lyall et al. (2020)		✓			mi
Rivera (2020)	✓				amelia
Simpser (2020)		✓			
Alizade et al. (2021)	✓				
Clark and Dolan (2021)	✓				mice
Evans et al. (2021)				✓	
Fong and Tyler (2021)				✓	
Harden and Kirkland (2021)	✓				amelia
Ketchley and El-Rayyes (2021)		✓			Stata mi
Todd et al. (2021)		✓			amelia
Bjarnegård et al. (2022)		✓			Stata mi
Clark (2022)	✓				mice
Eggers et al. (2022)	✓				amelia
Erlich et al. (2022)				✓	
Eubank and Fresh (2022)	✓				amelia
Fukumoto (2022)				✓	
Tai et al. (2022)	✓				mi
Lall and Robinson (2022)			✓		rMIDAS
Levy (2022)		✓			amelia
López-Moctezuma et al. (2022)		✓			amelia
Marbach (2022)			✓		
Mullin and Hansen (2022)	✓				mice
Park and Yamauchi (2022)				✓	
Redeker (2022)	✓				amelia
Wang (2022)		✓			Stata mi
Zhirkov (2022)				✓	
Slough (2023)				✓	
Wang and Aronow (2023)				✓	

Note that the papers published in 2023 were already available as online first versions before 2023, and fell under the search query.

SI.2 Definitions

Useful notation: Following King et al. (2001), I use the following notation to describe missing and complete data. The data matrix that contains N observations (both complete and incomplete) in rows and k variables in columns is denoted by D. Now, a missingness matrix M, with the same dimensions as D, can be defined such that $M_{ij} = 1 \forall D_{ij} = \text{NA}$ and $M_{ij} = 0$

elsewhere[8]. This means the entries in matrix M indicate whether a particular value is missing. Now, we can partition the matrix D into two disjoint subsets, D_{obs} and D_{mis}. D_{obs} is the part of D which contains all entries for which the row sum of $M = 0$, and D_{mis} contains all entries for which the row sum of $M \geq 1$.

Complete case analysis: The standard procedure in most software packages only considers cases that are fully observed for all variables in a model. That means that the number of observations can differ between models that use different subsets of variables.

SI.2.1 Missingness Mechanisms

What determines whether a value is observed or missing? It is essential to distinguish between the three following missingness mechanisms:

Missing completely at random (MCAR): Whether or not a particular value is observed is determined by a purely random mechanism. In formal notation, this is when M is independent of D, $P(M|D) = P(M)$. This is typically not problematic, as this implies that the complete cases are just an i.i.d subsample of the data with missingness. In these cases, complete case analysis would generate unbiased results, yet with fewer available cases. In high-dimensional applications, the combination of MCAR and complete case analysis can still be a severe problem when hardly any cases are left for complete case analysis (see, e.g, Wang and Aronow, 2023). Typical multiple imputation algorithms can handle MCAR missingness.

Missing at random (MAR): Whether or not a particular value is observed depends on other observed values only. I.e., conditional on observed values, the missingness mechanism is random. Formally, $P(M|D) = P(M|D_{obs})$. In these cases, an imputation model can be learned based on the observed values. Complete case analysis will typically be biased.

Non-ingorable missingnes (NI) or Not missing at random (NMAR): The missingness depends on values that are not observed. These might, e.g., be the missing values themselves or some variable that is not part of the data set. This is when $P(M|D)$ cannot be simplified, and missingness depends on D_{mis} as well. In these cases, neither complete case analysis nor multiple

[8]Note that King et al. (2001) define M the other way round, but this does not affect any subsequent arguments.

imputation models can give any guarantees. The missing values can not be reasonably filled in based on the observed values only, complete case analysis will also be biased.

SI.2.2 Combining Results from Multiple Completed Data Sets.

For valid inference, the results from the multiple completed data sets need to be combined such that the variance within each of the completed data sets, as well as variance across the completed data sets are accounted for.

Rubin's rules (Rubin, 1987): Suppose you run your complete data analysis on m completed (by the MI model) data sets. Then, the point estimate(s) are the average of the m results on each completed data set. Using the notation from Xie and Meng (2017): $\bar{\theta}_m = \frac{1}{m}\sum_{i=1}^{m}\hat{\theta}^{(i)}$, where $\hat{\theta}^{(i)}$ is a vector, e.g., of regression coefficients, or a single point estimate, of the statistic of interest calculated on the i-th completed data set.

The variance T_m of $\bar{\theta}_m$ is a sum of two terms: $T_m = \bar{U}_m + (1 + \frac{1}{m})B_m$, where $\bar{U}_m$ is the estimated within-imputation variance and B_m the estimated between-imputation variance. $\bar{U}_m = \frac{1}{m}\sum_{i=1}^{m}U^{(i)}$, i.e., $U^{(i)}$ is the variance estimate of the complete data analysis calculated on the i-th completed data set. $B_m = \frac{1}{m-1}\sum_{i=1}^{m}(\hat{\theta}^{(i)} - \bar{\theta}_m)(\hat{\theta}^{(i)} - \bar{\theta}_m)^T$ is the variance across the m complete data estimates $\hat{\theta}^{(i)}$.

Uncongenial rules (Xie and Meng, 2017): As Xie and Meng (2017) point out: "In general, uncongenality should be regarded as the rule rather than the exception, and a simple confidence valid procedure to combat any degree of uncongenality is to double Rubin's MI variance estimate" (1494). This means, the more conservative combining rule for the variance of $\bar{\theta}_m$ is to take $2T_m$ after applying Rubin's rules.

SI.3 Data Generating Processes in the Benchmark

SI.3.1 The Amelia Experiments

These Monte Carlo experiments were first presented in King et al. (2001). The multivariate normal data-generating process makes this the home turf case for multivariate normal imputation models like Amelia II.

The Data-Generating Process. The complete data D is drawn from a multivariate normal distribution with the mean vector $\mu = (0, 0, 0, 0, 0)$, and the variance-covariance matrix $\Sigma = \begin{pmatrix} 1.00 & -0.12 & -0.10 & 0.50 & 0.10 \\ -0.12 & 1.00 & 0.10 & -0.60 & 0.10 \\ -0.10 & 0.10 & 1.00 & -0.50 & 0.10 \\ 0.50 & -0.60 & -0.50 & 1.00 & 0.10 \\ 0.10 & 0.10 & 0.10 & 0.10 & 1.00 \end{pmatrix}$.

$N = 500$ for all experiments presented in this letter, but can be set to different values.

The Missingness Mechanisms. There are five missingness mechanisms for the Amelia experiments. For all missingness mechanisms, first, an auxiliary matrix U, with the same dimensions as D and M, is initialized with i.i.d. draws from a uniform distribution on the interval $(0, 1)$. M is initialized with 0 everywhere (essentially starting with a fully observed data set).

MAR-1: For MAR-1 missingness, the values of the missingness matrix M are updated in the following way:

- The fourth column of D, D_{i4}, is set to always observed, i.e., $M_{i4} = 0$ for all observations.
- The missingness of the first column of D, D_{i1}, is determined by: $M_{i1} = 1 \forall U_{i1} < 0.06$.
- The missingness of the second column of D, D_{i2}, is determined by: $M_{i2} = 1 \forall D_{i4} < -1 \wedge U_{i2} < 0.9$.
- The missingness of the third column of D, D_{i3}, is determined by: $M_{i3} = 1 \forall D_{i4} < -1 \wedge U_{i3} < 0.9$.
- The missingness of the fifth column of D, D_{i5}, is determined by: $M_{i5} = 1 \forall U_{i5} < 0.06$.

This missingness mechanism is MAR, as the missingness of values in the second and third

columns depends on the observed values of the data.

MAR-2: For MAR-2 missingness, the values of the missingness matrix M are updated in the following way:

- The fourth column of D, D_{i4}, is set to always observed, i.e., $M_{i4} = 0$ for all observations.
- The missingness of the first column of D, D_{i1}, is determined by: $M_{i1} = 1 \forall U_{i1} < 0.12$.
- The missingness of the second column of D, D_{i2}, is determined by: $M_{i2} = 1 \forall D_{i4} < -0.4 \wedge U_{i2} < 0.9$.
- The missingness of the third column of D, D_{i3}, is determined by: $M_{i3} = 1 \forall D_{i4} < -0.4 \wedge U_{i3} < 0.9$.
- The missingness of the fifth column of D, D_{i5}, is determined by: $M_{i5} = 1 \forall U_{i5} < 0.12$.

This missingness mechanism is MAR, as the missingness of values in the second and third columns depends on the observed values of the data. In comparison to MAR-1, MAR-2 has a higher number of expected missing values.

MCAR-1: For MCAR-1 missingness, the values of the missingness matrix M are updated in the following way:

- The fourth column of D, D_{i4}, is set to always observed, i.e., $M_{i4} = 0$ for all observations.
- The missingness of the first column of D, D_{i1}, is determined by: $M_{i1} = 1 \forall U_{i1} < 0.06$.
- The missingness of the second column of D, D_{i2}, is determined by: $M_{i2} = 1 \forall U_{i2} < 0.06$.
- The missingness of the third column of D, D_{i3}, is determined by: $M_{i3} = 1 \forall U_{i3} < 0.06$.
- The missingness of the fifth column of D, D_{i5}, is determined by:$M_{i5} = 1 \forall U_{i5} < 0.06$.

This missingness mechanism is MCAR, as the missingness of values is only determined by the values of U independent of the data D.

MCAR-2: For MCAR-2 missingness, the values of the missingness matrix M are updated in the following way:

- The fourth column of D, D_{i4}, is set to always observed, i.e., $M_{i4} = 0$ for all observations.
- The missingness of the first column of D, D_{i1}, is determined by: $M_{i1} = 1 \forall U_{i1} < 0.19$.
- The missingness of the second column of D, D_{i2}, is determined by: $M_{i2} = 1 \forall U_{i2} < 0.19$.
- The missingness of the third column of D, D_{i3}, is determined by: $M_{i3} = 1 \forall U_{i3} < 0.19$.
- The missingness of the fifth column of D, D_{i5}, is determined by:$M_{i5} = 1 \forall U_{i5} < 0.19$.

This missingness mechanism is MCAR, as the missingness of values is only determined by the values of U independent of the data D. Compared to MCAR-1, MCAR-2 has a higher proportion of missing values.

NI: For NI missingness, the values of the missingness matrix M are updated in the following way:

- The fourth column of D, D_{i4}, is set to always observed, i.e., $M_{i4} = 0$ for all observations.
- The fifth column of D, D_{i5}, is set to always observed, i.e., $M_{i5} = 0$ for all observations.
- The missingness of the first column of D, D_{i1}, is determined by: $M_{i1} = 1 \forall D_{i1} < -0.95$.
- The missingness of the second column of D, D_{i2}, is determined by: $M_{i2} = 1 \forall D_{i4} < -0.52$.
- The missingness of the third column of D, D_{i3}, is determined by: $M_{i3} = 1 \forall D_3 > 0.48$.

As the missingness for values in the first and third columns depends on unobserved values of the data, this missingness mechanism is non-ignorable.

The Analysis Model and Statistics of Interest. For all of the Amelia experiments, the analysis model of interest is the following linear regression model: $X_{i1} = \beta_0 + \beta_1 X_{i2} + \beta_2 X_{i3} + \epsilon_i$, with $\epsilon \sim \mathcal{N}(0, \sigma)$. The true coefficients of interest are $\beta_0 = 0$, $\beta_1 = -0.11$, and $\beta_3 = -0.089$.

SI.3.2 The Hot Deck Experiments

These Monte Carlo experiments were first presented in Cranmer and Gill (2013).

The Data-Generating Process. The complete data D is drawn from a multivariate normal distribution with the mean vector $\mu = (0, 0, 0, 0, 0)$, and the variance-covariance matrix $\Sigma = \begin{pmatrix} 1.00 & 0.80 & 0.80 & 0.80 & 0.80 \\ 0.80 & 1.00 & 0.80 & 0.80 & 0.80 \\ 0.80 & 0.80 & 1.00 & 0.80 & 0.80 \\ 0.80 & 0.80 & 0.80 & 1.00 & 0.80 \\ 0.80 & 0.80 & 0.80 & 0.80 & 1.00 \end{pmatrix}$.

All values of $D \leq 0$ are then set to 0, and all values of $D > 0$ are set to 1. The number of observations is $N = 500$.

The Missingness Mechanisms. There are three missingness mechanisms for the Hot Deck experiments. The three mechanisms differ by the amount of missingness.

For all missingness mechanisms, first, an auxiliary matrix U, with the same dimensions as D and M, is initialized with i.i.d. draws from a uniform distribution on the interval $(0, 1)$. Then, U is updated such that $U = 1$ for all rows in which D holds at least one 1 in the third, fourth, or fifth column. Furthermore, $U = 1$ for columns three, four, and five (this will make sure that these values are observed). M is initialized with 0 everywhere (essentially starting with a fully observed data set).

MAR-1: For MAR-1 missingness, the values of the missingness matrix M are updated in the following way:

$M_i j = 1 \forall U_i j < 0.2$

As U is updated before, columns three, four, and five are always fully observed. Missingness, furthermore, depends on values in these three columns, making this a MAR mechanism.

MAR-2 For MAR-2 missingness, the values of the missingness matrix M are updated in the following way:

$M_i j = 1 \forall U_i j < 0.5$

As U is updated before, columns three, four, and five are always fully observed. Missingness, furthermore, depends on values in these three columns, making this a MAR mechanism.

MAR-3: For MAR-3 missingness, the values of the missingness matrix M are updated in the following way:

$M_i j = 1 \forall U_i j < 0.8$

As U is updated before, columns three, four, and five are always fully observed. Missingness, furthermore, depends on values in these three columns, making this a MAR mechanism.

The Analysis Model and Statistics of Interest. For all of the Hot Deck experiments, the analysis model of interest is the logistic regression model $X_{i1} \sim \mathcal{B}(\pi_i)$, with $\pi_i = \frac{1}{1+e^{-(\beta_0+\beta_1 X_{i3}+\beta_2 X_{i4})}}$. The true regression coefficients for the fully observed data-generating process are $\beta_0 = -1.912$, $\beta_1 = 1.912$, and $\beta_3 = 1.912$.

SI.3.3 The Marbach Experiments

These Monte Carlo experiments were first presented in Marbach (2022).

The Data-Generating Process. To generate the data for the fully observed mixed data, I first draw $D_{i2} \sim \mathcal{U}(-5, 5)$, i.e., D_{i2} is drawn from a uniform distribution. Then, $D_{i3} \sim \mathcal{B}(0.5)$. Finally, $D_{i1} = D_{i2}D_{i3} + \epsilon_i$, with $\epsilon_i = \mathcal{N}(\mu = 0, \sigma = 1)$. The number of observations is $N = 1000$.

The Missingness Mechanism. There is one MAR missingness mechanism, *MAR-1*. M is initialized with 0 everywhere (essentially starting with a fully observed data set). D_{i2} and D_{i3} are always fully observed. The proportion of missing values in D_{i1} depends on values of D_{i3}. First, two proportions are drawn from $\mathcal{U}(0.1, 0.5)$. The larger proportion p_l is the amount of missingness for D_{i1}, where $D_{i3} = 1$, and the smaller proportion p_s governs the amount of missingness for D_{i1}, where $D_{i3} = 0$, such that $p_i = p_l \forall D_{i3} = 1$ and $p_i = p_s \forall D_{i3} = 0$. The proportions in p are used to draw an auxiliary variable $U_i \sim \mathcal{B}(p_i)$. Formally, $M_{i1} = 1 \forall U_i = 1$.

This mechanism is MAR, as the missingness depends on observed values of D_{i3}.

The Analysis Model and Statistics of Interest. For the Marbach experiment, the analysis model of interest is the linear regression model $X_{i1} = \beta_0 + \beta_1 X_{i2} + \beta_2 X_{i3} + \beta_4 X_{i2} X_{i3}$. The true coefficients of interest are $\beta_0 = 0$, $\beta_1 = 0$, $\beta_3 = 0$ and $\beta_4 = 1$.

SI.3.4 The Mixed Experiments

The Data-Generating Process. To generate the data for the fully observed mixed data, I first draw $D_{i1} \sim \mathcal{N}(0, 1)$, i.e., D_{i1} is drawn from a standard normal distribution. Then, $D_{i2} \sim \mathcal{B}(0.5)$. Finally, $D_{i3} = D_{i1} - 2D_{i1}D_{i2} + \epsilon_i$, with $\epsilon_i = \mathcal{N}(\mu = 0, \sigma = 0.2)$. The number of observations is $N = 1000$.

The Missingness Mechanisms. There are two missingness mechanisms for the Mixed experiments. For all missingness mechanisms, first, an auxiliary matrix U, with the same dimensions as D and M, is initialized with i.i.d. draws from a uniform distribution on the interval $(0, 1)$. M is initialized with 0 everywhere (essentially starting with a fully observed data set).

MAR-1: For MAR-1 missingness, the values of the missingness matrix M are updated in the following way:

- The third column of D, D_{i3}, is set to always observed, i.e., $M_{i4} = 0$ for all observations.
- The missingness of the first column of D, D_{i1}, is determined by: $M_{i1} = 1 \forall D_{i3} < -1 \wedge U_{i1} < 0.19$.
- The missingness of the second column of D, D_{i2}, is determined by: $M_{i2} = 1 \forall D_{i3} < -1 \wedge U_{i2} < 0.19$.

MCAR-1: For MCAR-1 missingness, the values of the missingness matrix M are updated in the following way:

- The third column of D, D_{i3}, is set to always observed, i.e., $M_{i4} = 0$ for all observations.
- The missingness of the first column of D, D_{i1}, is determined by: $M_{i1} = 1 \forall U_{i1} < 0.19$.
- The missingness of the second column of D, D_{i2}, is determined by: $M_{i2} = 1 \forall U_{i2} < 0.19$.

The Analysis Model and Statistics of Interest. For all of the Mixed experiments, the statistics of interest are the coefficients of the following linear model: $X_{i3} = \beta_0 + \beta_1 X_{i1} + \beta_2 X_{i2} + \beta_4 X_{i1} X_{i2}$. The true coefficients of interest are $\beta_0 = 0$, $\beta_1 = 1$, $\beta_3 = 0$ and $\beta_4 = -2$.

SI.4 The `MIBench` R-package

The `MIBench` R-package makes it easy to run the experiments I suggest in this letter.

Running the Monte Carlo experiments for a MI model[9] with all the eleven experiments from this letter is as simple as:

```
MIBench_results <-
  run_MIBench_all_dgps(
    MIalgorithm,
    store_runs = TRUE,
    store_results = TRUE,
    n_repetitions = 1000,
    seed = 221208
  )
```

Furthermore, as re-running all eleven combinations of data-generating processes and missingness mechanisms with all MI models can be resource intensive, I include the results from my runs (that I present in table 1) directly into the package. This means that when running the benchmark with a new model and the same random seeds, the results are directly comparable to the results presented in this letter.

[9] In the Supporting Information I describe how to wrap any MI model into a function that works with `MIBench`.

SI.5 Imputation Models included in the Benchmark

In general, I used the default values provided in the respective packages. In the following, I document the function calls to reproduce the MI models I present in table1.

SI.5.1 Wrapping MI models for `MIBench`

The `MIBench` package can work with any MI model as long as the call to it is wrapped in the following function, which takes the data matrix as, and the number of completed data sets m as input, and returns a named list with a list of the m imputations in the slot "imputations" and the name of the procedure in the slot "MI_name". If a MI model, for example, requires that the data is preprocessed in a particular way this should happen within the function.

```
MIalgorithm <- function(data = NULL, m = 10) {
# In case the data is not in the format needed for the
# MI model, you can pre-process it here.
 if (!is.null(data)) {
    mi_obj <- my_favorite_MI_model(data, m = m)
  } else {
    mi_obj <- NULL
  }
  # In case mi_obj is not the list of m completed data sets,
  # you can post-process them here.
  return(list(imputations = mi_obj,
              MI_name = "favorite"))
}
```

SI.5.2 Amelia II

This uses the `amelia` function provided by the `Amelia` R-package (Honaker et al., 2011). The data is pre-processed, such that nominal and ordered factors are imputed correctly.

```
MIalgorithm <- function(data = NULL, m = 10) {
  if (!is.null(data)) {
    is_binary <- function(x) {
      x0 <- na.omit(x)
      is.numeric(x) && length(unique(x0)) %in% 1:2 && all(x0
         %in% 0:1)
    }

    ok <- sapply(data.frame(data), is_binary)
    data <-
      replace(data.frame(data),
              ok,
              lapply(data.frame(data)[ok], factor, levels =
                 0:1))

    classes <- sapply(data.frame(data), class)


    ords <- which(sapply(classes, function(x)
      "ordered" %in% x))

    if (length(ords) == 0)
      ords <- NULL

    noms <-
      which(sapply(classes, function(x)
        "factor" %in% x & length(x) == 1))

    if (length(noms) == 0)
```

```
      noms <- NULL


    mi_obj <-
      Amelia::amelia(
        data,
        m = m,
        noms = noms,
        ords = ords,
        emburn = c(1, 75)
      )
  } else {
    mi_obj <- NULL
  }
  return(list(imputations = mi_obj$imputations,
              MI_name = "amelia"))
}
```

SI.5.3 hot.deck

This MI model uses the `hot.deck` function provided by the `hot.deck` R-package (Cranmer and Gill, 2013).

```
MIalgorithm <- function(data = NULL, m = 10) {
    if (!is.null(data)) {
    mi_obj <- hot.deck::hot.deck(data.frame(data), m = m)$
       data
    } else {
      mi_obj <- NULL
    }
    return(list(imputations = mi_obj,
```

```
                MI_name = "hotdeck"))
  }
```

SI.5.4 mi

This MI model uses the `mi` functions provided by the `mi` R-package (Su et al., 2011). The data is pre- and post-processed according to the `mi` workflow.

```
MIalgorithm <- function(data = NULL, m = 10) {
    if (!is.null(data)) {
    mi_obj <- mi::missing_data.frame(data)
    mi_obj <- mi::mi(mi_obj)
    mi_obj <- mi::complete(mi_obj, m = m)
  } else {
    mi_obj <- NULL
  }
  return(list(imputations = mi_obj,
              MI_name = "mi"))
  }
```

SI.5.5 mice

The `mice` R-package (van Buuren and Groothuis-Oudshoorn, 2011) comes with various potential MI models. Furthermore, separate models could be specified for each column of a data set.

SI.5.5.1 Predictive mean matching (pmm)

If the user does not specify which MI model to use, `mice` defaults to predictive mean matching (pmm).

```
MIalgorithm <- function(data = NULL, m = 10) {
```

```
  if (!is.null(data)) {
    imp <- mice::mice(data, m = m)
    mi_obj <- mice::complete(imp, action = "all")
  } else {
    mi_obj <- NULL
  }
  return(list(imputations = mi_obj,
              MI_name = "mice_pmm"))
}
```

SI.5.5.2 Hot Deck (hd)

The `mice` R-package also has an implementation of MI with the hot deck technique.

```
MIalgorithm <- function(data = NULL, m = 10) {
  if (!is.null(data)) {
    imp <- mice::mice(data, method = "hotDeck", m = m)
    mi_obj <- mice::complete(imp, action = "all")
  } else {
    mi_obj <- NULL
  }
  return(list(imputations = mi_obj,
              MI_name = "mice_hd"))
}
```

SI.5.5.3 Random Forests (rf)

The `mice` R-package also includes an implementation of MI with the random forests for each column.

```
MIalgorithm <- function(data = NULL, m = 10) {
  if (!is.null(data)) {
```

```
    imp <- mice::mice(data, method = "rf", m = m)
    mi_obj <- mice::complete(imp, action = "all")
  } else {
    mi_obj <- NULL
  }
  return(list(imputations = mi_obj,
              MI_name = "mice_rf"))
}
```

SI.5.6 rMIDAS

MI with denoising autoencoders was introduced to the political science literature by Lall and Robinson (2022). As the default values in the R-package `rMIDAS` differed from the default values suggested in Lall and Robinson (2022), I evaluate two instances of `rMIDAS`. The results between the two settings are pretty different. This leads to a natural question that goes beyond the scope of this letter, how can one choose the optimal hyperparameters for such machine learning based MI models?

The data is pre-processed similarly to the pre-processing necessary for `Amelia` (i.e., defining nominal and ordered factors). To get the results in the same format as from other MI packages, the data is post-processed (`rMIDAS`, for example, re-orders the columns).

SI.5.6.1 rMIDAS package defaults (rMIDAS 1)

The first set of experiments uses the default values provided in the `rMIDAS` package, i.e., no parameters in the function call to `rMIDAS` are adjusted.

```
MIalgorithm <- function(data = NULL, m = 10) {
  if (!is.null(data)) {
    if (is.null(colnames(data))) {
      colnames(data) <- paste0("X", 1:ncol(data))
    }
```

```
is_binary <- function(x) {
  x0 <- na.omit(x)
  is.numeric(x) &&
    length(unique(x0)) %in% 1:2 && all(x0 %in% 0:1)
}
ok <- sapply(data.frame(data), is_binary)
data <-
  replace(data.frame(data),
          ok,
          lapply(data.frame(data)[ok], factor, levels =
             0:1))

classes <- sapply(data.frame(data), class)

ords <- which(sapply(classes, function(x)
  "ordered" %in% x))

if (length(ords) == 0)
  ords <- NULL

noms <-
  which(sapply(classes, function(x)
    "factor" %in% x & length(x) == 1))

if (length(noms) == 0)
  noms <- NULL

imp <-
  rMIDAS::train(rMIDAS::convert(
```

```
        data.frame(data),
        cat_cols = colnames(data)[noms],
        minmax_scale = TRUE
      ))
    mi_obj <- rMIDAS::complete(imp, m = m)

    mi_obj <- lapply(mi_obj, function(x)
      x[, colnames(data)])

    mi_obj <- lapply(mi_obj, function(x)
      sapply(x, as.numeric))
  } else {
    mi_obj <- NULL
  }
  return(list(imputations = mi_obj,
              MI_name = "rmidas"))
}
```

SI.5.6.2 rMIDAS paper defaults (rMIDAS 2)

As Lall and Robinson (2022) describe different default values in their experiments with `rMIDAS` than implemented in the R-package, I rerun the experiments with this second set of hyperparameters. The number of training epochs and the learning rate are adapted accordingly.

```
MIalgorithm <- function(data = NULL, m = 10) {
  if (!is.null(data)) {
    if (is.null(colnames(data))) {
      colnames(data) <- paste0("X", 1:ncol(data))
    }
    is_binary <- function(x) {
```

```
  x0 <- na.omit(x)
  is.numeric(x) &&
    length(unique(x0)) %in% 1:2 && all(x0 %in% 0:1)
}
ok <- sapply(data.frame(data), is_binary)
data <-
  replace(data.frame(data),
          ok,
          lapply(data.frame(data)[ok], factor, levels =
             0:1))

classes <- sapply(data.frame(data), class)

ords <- which(sapply(classes, function(x)
  "ordered" %in% x))

if (length(ords) == 0)
  ords <- NULL

noms <-
  which(sapply(classes, function(x)
    "factor" %in% x & length(x) == 1))

if (length(noms) == 0)
  noms <- NULL

imp <-
  rMIDAS::train(
    rMIDAS::convert(
```

```
          data.frame(data),
          cat_cols = colnames(data)[noms],
          minmax_scale = TRUE
        ),
        training_epochs = 5L,
        learn_rate = 1e-4
      )
    mi_obj <- rMIDAS::complete(imp, m = m)

    mi_obj <- lapply(mi_obj, function(x)
      x[, colnames(data)])

    mi_obj <- lapply(mi_obj, function(x)
      sapply(x, as.numeric))
  } else {
    mi_obj <- NULL
  }
  return(list(imputations = mi_obj,
              MI_name = "rmidas"))
}
```

SI.5.7 GAIN

`GAIN` produces multiple imputations based on Generative Adversarial Nets (GANs) (GAIN is the acronym for Generative Adversarial Imputation Nets) and was first introduced by Yoon et al. (2018).

Changes to the GAIN implementation: When implementing GAIN I made a couple of adjustments to the original code. First, although the authors state that GAIN can be used for MI, they do not provide an implementation of drawing multiple completed data sets from a learned

Generator. The second change is fixing a bug in a pre-processing step within the GAIN model. In the paper Yoon et al. (2018) state that the data should be normalized to the interval $[0, 1]$ as they use a sigmoid activation function in their Generator. Unfortunately, the normalization function the authors provide in their code repository does not normalize data to this interval (the sigmoid activation function is used nevertheless). I implemented normalization to the $[0, 1]$ range. Third, I fixed another minor bug that led the original GAIN implementation to fail in cases where the first row of a data set contained missing values (based on how the python package `numpy` handles missing values).

The updated `gain.py` script is part of the replication materials of this letter.

```
reticulate::source_python("gain.py")
MIalgorithm <- function(data = NULL, m = 10L) {
  if (!is.null(data)) {
    mi_obj <-
      gain(
        data_x = data,
        gain_parameters = list(
          batch_size = 64L,
          hint_rate = 0.9,
          alpha = 100,
          iterations = 10000L
        ),
        m = as.integer(m)
      )
  } else {
    mi_obj <- NULL
  }
  return(list(imputations = mi_obj,
              MI_name = "gain"))
}
```

SI.6 Additional Results

SI.6.1 Additional Results for the Amelia Experiments

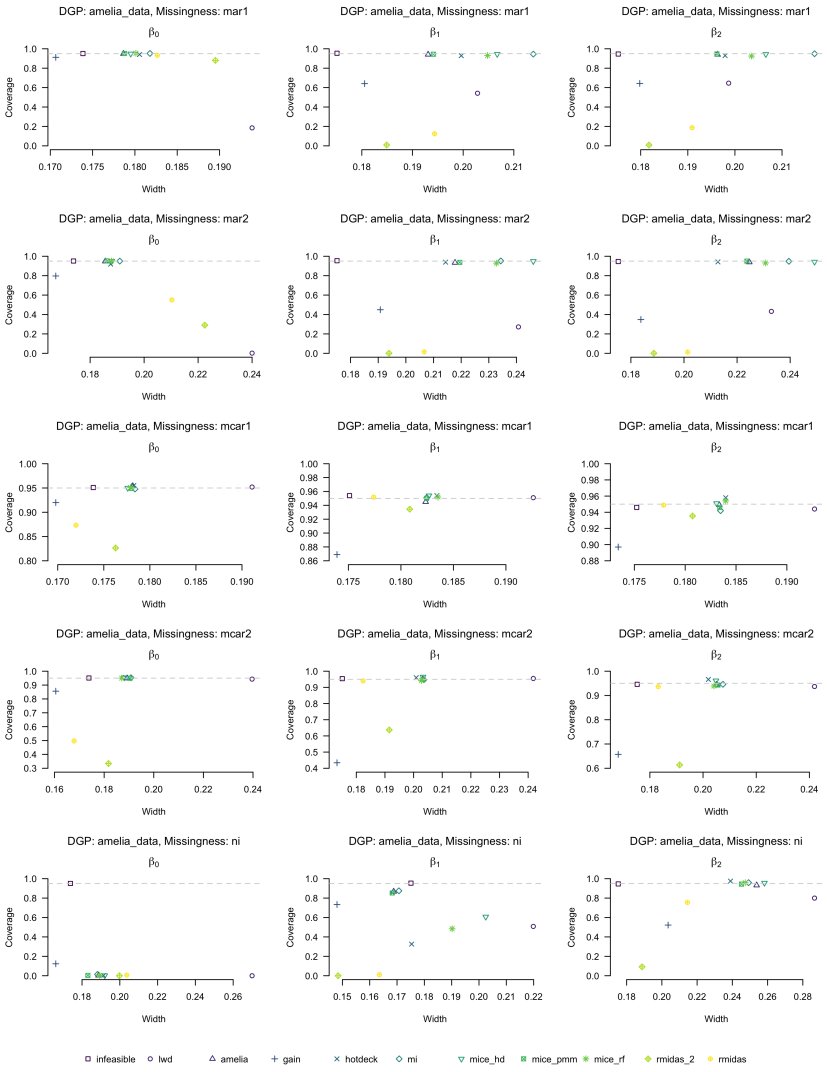

Figure SI.1: Confidence interval width plotted against the empirical coverage of confidence intervals for the regression coefficients in the Amelia experiments.

SI.6.2 Additional Results for the Hot Deck Experiments

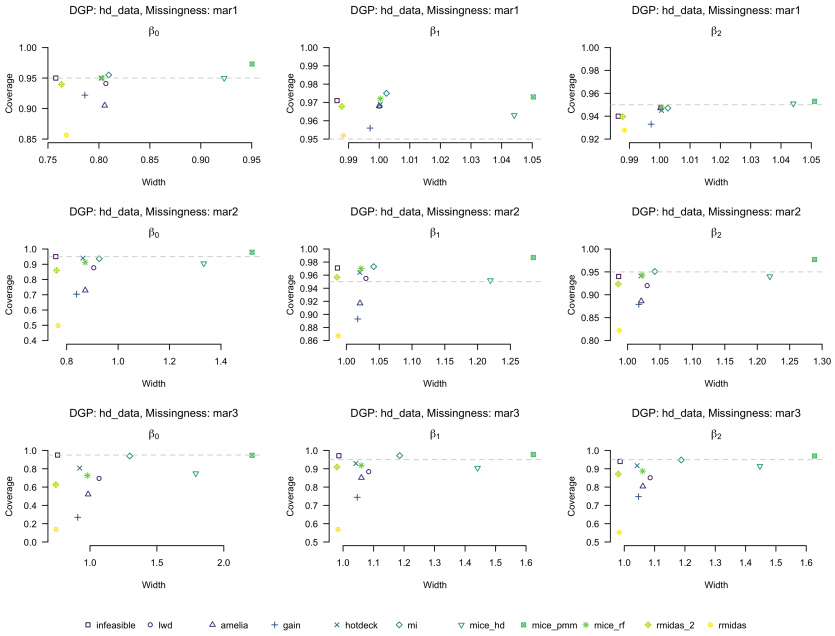

Figure SI.2: Confidence interval width plotted against the empirical coverage of confidence intervals for the regression coefficients in the Hot Deck experiments.

SI.6.3 Additional Results for the Marbach Experiments

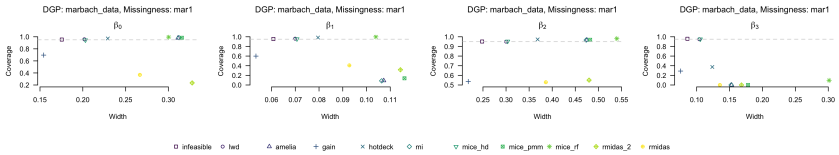

Figure SI.3: Confidence interval width plotted against the empirical coverage of confidence intervals for the regression coefficients in the Marbach experiments.

SI.6.4 Additional Results for the Mixed Experiments

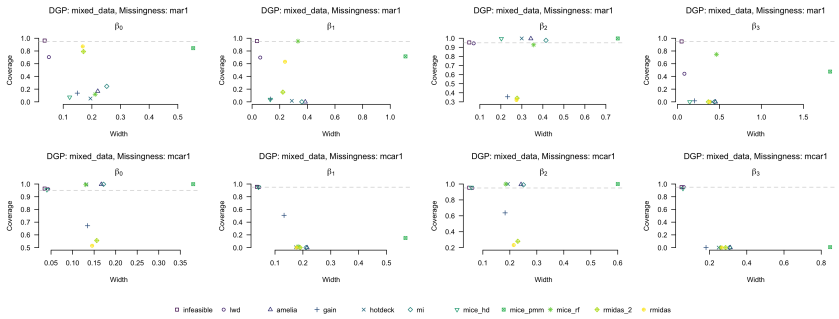

Figure SI.4: Confidence interval width plotted against the empirical coverage of confidence intervals for the regression coefficients in the Mixed experiments.

References

Alizade, J., R. Dancygier, and R. K. Ditlmann (2021). National penalties reversed: The local politics of citizenship and politician responsiveness to immigrants. *The Journal of Politics 83*(3), 867–883.

Alvarez, R. M., L. R. Atkeson, I. Levin, and Y. Li (2019). Paying attention to inattentive survey respondents. *Political Analysis 27*(2), 145–162.

Arel-Bundock, V. and K. J. Pelc (2018). When can multiple imputation improve regression estimates? *Political Analysis 26*(2), 240–245.

Beazer, Q. H. and D. J. Blake (2018). The conditional nature of political risk: How home institutions influence the location of foreign direct investment. *American Journal of Political Science 62*(2), 470–485.

Beckman, T. and P. Schleiter (2020). Opportunistic election timing, a complement or substitute for economic manipulation? *The Journal of Politics 82*(3), 1127–1141.

Bjarnegård, E., A. Engvall, S. Jitpiromsri, and E. Melander (2022). Armed violence and patriarchal values: A survey of young men in thailand and their military experiences. *American Political Science Review*, 1–15.

Bonica, A. and M. Sen (2017). A common-space scaling of the american judiciary and legal profession. *Political Analysis 25*(1), 114–121.

Busby, E. C., J. N. Druckman, and A. Fredendall (2017). The political relevance of irrelevant events. *The Journal of Politics 79*(1), 346–350.

Carroll, R. J. and B. Kenkel (2019). Prediction, proxies, and power. *American Journal of Political Science 63*(3), 577–593.

Caughey, D. and C. Warshaw (2019). Public opinion in subnational politics.

Clark, R. (2022). Bargain down or shop around? outside options and imf conditionality. *The Journal of Politics 84*(3), 000–000.

Clark, R. and L. R. Dolan (2021). Pleasing the principal: Us influence in world bank policymaking. *American Journal of Political Science 65*(1), 36–51.

Coppock, A., A. S. Gerber, D. P. Green, and H. L. Kern (2017). Combining double sampling and bounds to address nonignorable missing outcomes in randomized experiments. *Political*

Analysis 25(2), 188–206.

Cranmer, S. J. and J. Gill (2013). We have to be discrete about this: A non-parametric imputation technique for missing categorical data. *British Journal of Political Science 43*(2), 425–449.

De Kadt, D. (2017). Voting then, voting now: The long-term consequences of participation in south africa's first democratic election. *The Journal of Politics 79*(2), 670–687.

Dorsch, M. T. and P. Maarek (2019). Democratization and the conditional dynamics of income distribution. *American Political Science Review 113*(2), 385–404.

Eggers, A. C., D. Rubenson, and P. J. Loewen (2022). Who votes more strategically? evidence from canada. *The Journal of Politics 84*(3), 000–000.

Erlich, A., S. G. Dantas, B. E. Bagozzi, D. Berliner, and B. Palmer-Rubin (2022). Multi-label prediction for political text-as-data. *Political analysis 30*(4), 463–480.

Eubank, N. and A. Fresh (2022). Enfranchisement and incarceration after the 1965 voting rights act. *American Political Science Review*, 1–16.

Evans, G., G. King, G. Evans, G. King, M. Schwenzfeier, A. Thakurta, J. Katz, G. King, E. Rosenblatt, G. Evans, et al. (2021). Statistically valid inferences from differentially private data releases, ii: Extensions to nonlinear transformations. *Political Analysis 26*(B2), 1161–1165.

Fong, C. and M. Tyler (2021). Machine learning predictions as regression covariates. *Political Analysis 29*(4), 467–484.

Fukumoto, K. (2022). Nonignorable attrition in pairwise randomized experiments. *Political Analysis 30*(1), 132–141.

Grossman, G., J. H. Pierskalla, and E. Boswell Dean (2017). Government fragmentation and public goods provision. *The Journal of Politics 79*(3), 823–840.

Hangartner, D., E. Dinas, M. Marbach, K. Matakos, and D. Xefteris (2019). Does exposure to the refugee crisis make natives more hostile? *American political science review 113*(2), 442–455.

Harden, J. J. and J. H. Kirkland (2021). does transparency inhibit political compromise? *American Journal of Political Science 65*(2), 493–509.

Harris, J. A. and D. N. Posner (2019). (under what conditions) do politicians reward their

supporters? evidence from kenya's constituencies development fund. *American Political Science Review 113*(1), 123–139.

Hemker, J. and A. Rink (2017). Multiple dimensions of bureaucratic discrimination: Evidence from german welfare offices. *American Journal of Political Science 61*(4), 786–803.

Honaker, J., G. King, and M. Blackwell (2011). Amelia II: A program for missing data. *Journal of Statistical Software 45*(7), 1–47.

Ketchley, N. and T. El-Rayyes (2021). Unpopular protest: Mass mobilization and attitudes to democracy in post-mubarak egypt. *The Journal of Politics 83*(1), 291–305.

King, G., J. Honaker, A. Joseph, and K. Scheve (2001). Analyzing Incomplete Political Science Data: An Alternative Algorithm for Multiple Imputation. *American Political Science Review 95*(1), 49–69.

Lachapelle, J. (2020). No easy way out: The effect of military coups on state repression. *The Journal of Politics 82*(4), 1354–1372.

Lall, R. and T. Robinson (2022). The midas touch: Accurate and scalable missing-data imputation with deep learning. *Political Analysis 30*(2), 179–196.

Levy, G. (2022). Evaluations of violence at the polls: Civilian victimization and support for perpetrators after war. *The Journal of Politics 84*(2), 783–797.

López-Moctezuma, G., L. Wantchekon, D. Rubenson, T. Fujiwara, and C. Pe Lero (2022). Policy deliberation and voter persuasion: Experimental evidence from an election in the philippines. *American Journal of Political Science 66*(1), 59–74.

Lyall, J., Y.-Y. Zhou, and K. Imai (2020). Can economic assistance shape combatant support in wartime? experimental evidence from afghanistan. *American Political Science Review 114*(1), 126–143.

Marbach, M. (2022). Choosing imputation models. *Political Analysis 30*(4), 597–605.

Margolis, M. F. (2018). How politics affects religion: Partisanship, socialization, and religiosity in america. *The Journal of Politics 80*(1), 30–43.

Mullin, M. and K. Hansen (2022). Local news and the electoral incentive to invest in infrastructure. *American Political Science Review*, 1–6.

Osgood, I., D. Tingley, T. Bernauer, I. S. Kim, H. V. Milner, and G. Spilker (2017). The

charmed life of superstar exporters: Survey evidence on firms and trade policy. *The Journal of Politics 79*(1), 133–152.

Pan, J. and Y. Xu (2018). China's ideological spectrum. *The Journal of Politics 80*(1), 254–273.

Pardos-Prado, S. and I. Sagarzazu (2019). Economic responsiveness and the political conditioning of the electoral cycle. *The Journal of Politics 81*(2), 441–455.

Park, J. H. and S. Yamauchi (2022). Change-point detection and regularization in time series cross-sectional data analysis. *Political Analysis*, 1–21.

Pepinsky, T. B. (2018). A note on listwise deletion versus multiple imputation. *Political Analysis 26*(4), 480–488.

Pietryka, M. T. and D. A. DeBats (2017). It's not just what you have, but who you know: Networks, social proximity to elites, and voting in state and local elections. *American Political Science Review 111*(2), 360–378.

Redeker, N. (2022). The politics of stashing wealth: The decline of labor power and the global rise in corporate savings. *The Journal of Politics 84*(2), 975–991.

Rivera, C. V. (2020). Loyalty or incentives? how party alignment affects bureaucratic performance. *The Journal of Politics 82*(4), 1287–1304.

Royston, P. (2004). Multiple imputation of missing values. *The Stata Journal 4*(3), 227–241.

Rubin, D. B. (1987). *Multiple Imputation for Nonresponse in Surveys*. New York: Wiley.

Rueda, D. (2018). Food comes first, then morals: Redistribution preferences, parochial altruism, and immigration in western europe. *The Journal of Politics 80*(1), 225–239.

Rueda, M. R. (2017). Small aggregates, big manipulation: Vote buying enforcement and collective monitoring. *American Journal of Political Science 61*(1), 163–177.

Schultz, K. A. and J. S. Mankin (2019). Is temperature exogenous? the impact of civil conflict on the instrumental climate record in sub-saharan africa. *American Journal of Political Science 63*(4), 723–739.

Simpser, A. (2020). The culture of corruption across generations: An empirical study of bribery attitudes and behavior. *The Journal of Politics 82*(4), 1373–1389.

Slapin, J. B., J. H. Kirkland, J. A. Lazzaro, P. A. Leslie, and T. O'grady (2018). Ideology, grandstanding, and strategic party disloyalty in the british parliament. *American Political*

Science Review 112(1), 15–30.

Slough, T. (2023). Phantom counterfactuals. *American Journal of Political Science 67*(1), 137–153.

Smidt, C. D. (2017). Polarization and the decline of the american floating voter. *American Journal of Political Science 61*(2), 365–381.

Solt, F., Y. Hu, K. Hudson, J. Song, and D. E. Yu (2017). Economic inequality and class consciousness. *The Journal of Politics 79*(3), 1079–1083.

Su, Y.-S., A. Gelman, J. Hill, and M. Yajima (2011). Multiple imputation with diagnostics (mi) in r: Opening windows into the black box. *Journal of Statistical Software 45*(2), 1–31.

Tai, Y. C., Y. Hu, and F. Solt (2022). Democracy, public support, and measurement uncertainty. *American Political Science Review*, 1–7.

Todd, J. D., E. J. Malesky, A. Tran, and Q. A. Le (2021). Testing legislator responsiveness to citizens and firms in single-party regimes: A field experiment in the vietnamese national assembly. *The Journal of Politics 83*(4), 1573–1588.

van Buuren, S. and K. Groothuis-Oudshoorn (2011). mice: Multivariate imputation by chained equations in r. *Journal of Statistical Software 45*(3), 1–67.

Wang, J. S. and P. M. Aronow (2023). Listwise deletion in high dimensions. *Political Analysis*, 1–7.

Wang, Y. (2022). Blood is thicker than water: Elite kinship networks and state building in imperial china. *American Political Science Review*, 1–15.

Wing, C. (2019). What can instrumental variables tell us about nonresponse in household surveys and political polls? *Political Analysis 27*(3), 320–338.

Xie, X. and X. L. Meng (2017). Dissecting multiple imput a tion from a multi-phase inference perspective: What happens when god's, imputer's and anal Yst's models are uncongenial? *Statistica Sinica 27*(4), 1485–1545.

Yoon, J., J. Jordon, and M. van der Schaar (2018, 10–15 Jul). Gain: Missing data imputation using generative adversarial nets. In J. Dy and A. Krause (Eds.), *Proceedings of the 35th International Conference on Machine Learning*, Volume 80 of *Proceedings of Machine Learning Research*, pp. 5689–5698. PMLR.

Zhirkov, K. (2022). Estimating and using individual marginal component effects from conjoint experiments. *Political Analysis 30*(2), 236–249.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Abadi, M., Chu, A., Goodfellow, I., McMahan, H. B., Mironov, I., Talwar, K., and Zhang, L. (2016). Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, pages 308–318, New York, NY, USA. ACM.

Abadie, A., Diamond, A., and Hainmueller, J. (2015). Comparative politics and the synthetic control method. *American Journal of Political Science*, 59(2):495–510.

Abowd, J. M. (2018). The U.S. census bureau adopts differential privacy. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19-23, 2018*, page 2867.

Alizade, J., Dancygier, R., and Ditlmann, R. K. (2021). National penalties

reversed: The local politics of citizenship and politician responsiveness to immigrants. *The Journal of Politics*, 83(3):867–883.

Alvarez, R. M., Atkeson, L. R., Levin, I., and Li, Y. (2019). Paying attention to inattentive survey respondents. *Political Analysis*, 27(2):145–162.

Arel-Bundock, V. and Pelc, K. J. (2018). When can multiple imputation improve regression estimates? *Political Analysis*, 26(2):240–245.

Arjovsky, M., Chintala, S., and Bottou, L. (2017a). Wasserstein GAN. *CoRR*, abs/1701.07875.

Arjovsky, M., Chintala, S., and Bottou, L. (2017b). Wasserstein generative adversarial networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, pages 214–223. JMLR.org.

Arnold, C. and Neunhoeffer, M. (2020). Really useful synthetic data–a framework to evaluate the quality of differentially private synthetic data. *arXiv preprint arXiv:2004.07740*.

Arora, S., Ge, R., Liang, Y., Ma, T., and Zhang, Y. (2017). Generalization and equilibrium in generative adversarial nets (GANs). In Precup, D. and Teh, Y. W., editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 224–232, International Convention Centre, Sydney, Australia. PMLR.

Arora, S., Hazan, E., and Kale, S. (2012). The multiplicative weights update method: a meta-algorithm and applications. *Theory of Computing*, 8(6):121–164.

Athey, S., Imbens, G. W., Metzger, J., and Munro, E. (2021). Using wasserstein generative adversarial networks for the design of monte carlo simulations.

Journal of Econometrics.

Azadi, S., Olsson, C., Darrell, T., Goodfellow, I., and Odena, A. (2019a). Discriminator rejection sampling. In *International Conference on Learning Representations*.

Azadi, S., Olsson, C., Darrell, T., Goodfellow, I. J., and Odena, A. (2019b). Discriminator rejection sampling. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*.

Bartlett, J. W., Seaman, S. R., White, I. R., and Carpenter, J. R. (2015). Multiple imputation of covariates by fully conditional specification: Accommodating the substantive model. *Statistical Methods in Medical Research*, 24(4):462–487.

Beaulieu-Jones, B. K., Wu, Z. S., Williams, C., Lee, R., Bhavnani, S. P., Byrd, J. B., and Greene, C. S. (2019). Privacy-preserving generative deep neural networks support clinical data sharing. *Circulation: Cardiovascular Quality and Outcomes*, 12(7):e005122.

Beazer, Q. H. and Blake, D. J. (2018). The conditional nature of political risk: How home institutions influence the location of foreign direct investment. *American Journal of Political Science*, 62(2):470–485.

Beckman, T. and Schleiter, P. (2020). Opportunistic election timing, a complement or substitute for economic manipulation? *The Journal of Politics*, 82(3):1127–1141.

Bjarnegård, E., Engvall, A., Jitpiromsri, S., and Melander, E. (2022). Armed violence and patriarchal values: A survey of young men in thailand and their military experiences. *American Political Science Review*, pages 1–15.

Bonica, A. and Sen, M. (2017). A common-space scaling of the american judiciary and legal profession. *Political Analysis*, 25(1):114–121.

Breiman, L. (2001). Random forests. *Machine learning*, 45(1):5–32.

Busby, E. C., Druckman, J. N., and Fredendall, A. (2017). The political relevance of irrelevant events. *The Journal of Politics*, 79(1):346–350.

Carlini, N., Hayes, J., Nasr, M., Jagielski, M., Sehwag, V., Tramèr, F., Balle, B., Ippolito, D., and Wallace, E. (2023). Extracting training data from diffusion models.

Carroll, R. J. and Kenkel, B. (2019). Prediction, proxies, and power. *American Journal of Political Science*, 63(3):577–593.

Caughey, D. and Warshaw, C. (2019). Public opinion in subnational politics.

Chaudhuri, K. and Vinterbo, S. A. (2013). A stability-based validation procedure for differentially private machine learning. In *Advances in Neural Information Processing Systems*, pages 2652–2660.

Chen, T. and Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794.

Clark, R. (2022). Bargain down or shop around? outside options and imf conditionality. *The Journal of Politics*, 84(3):000–000.

Clark, R. and Dolan, L. R. (2021). Pleasing the principal: Us influence in world bank policymaking. *American Journal of Political Science*, 65(1):36–51.

Coppock, A., Gerber, A. S., Green, D. P., and Kern, H. L. (2017). Combining double sampling and bounds to address nonignorable missing outcomes in randomized experiments. *Political Analysis*, 25(2):188–206.

Cranmer, S. J. and Gill, J. (2013). We have to be discrete about this: A non-parametric imputation technique for missing categorical data. *British Journal of Political Science*, 43(2):425–449.

De Kadt, D. (2017). Voting then, voting now: The long-term consequences of participation in south africa's first democratic election. *The Journal of Politics*, 79(2):670–687.

Denton, E., Gross, S., and Fergus, R. (2016). Semi-supervised learning with context-conditional generative adversarial networks.

Differential Privacy Team, Apple (2017). Learning with privacy at scale. https://machinelearning.apple.com/docs/learning-with-privacy-at-scale/appledifferentialprivacysystem.pdf.

Ding, B., Kulkarni, J., and Yekhanin, S. (2017). Collecting telemetry data privately. In *Advances in Neural Information Processing Systems 30*, NIPS '17, pages 3571–3580. Curran Associates, Inc.

Dorsch, M. T. and Maarek, P. (2019). Democratization and the conditional dynamics of income distribution. *American Political Science Review*, 113(2):385–404.

Dwork, C., McSherry, F., Nissim, K., and Smith, A. (2006). Calibrating noise to sensitivity in private data analysis. In *Proceedings of the 3rd Theory of Cryptography Conference*, volume 3876, pages 265–284.

Eggers, A. C., Rubenson, D., and Loewen, P. J. (2022). Who votes more strategically? evidence from canada. *The Journal of Politics*, 84(3):000–000.

Erlich, A., Dantas, S. G., Bagozzi, B. E., Berliner, D., and Palmer-Rubin, B. (2022). Multi-label prediction for political text-as-data. *Political analysis*,

30(4):463–480.

Erlingsson, Ú., Pihur, V., and Korolova, A. (2014). RAPPOR: Randomized aggregatable privacy-preserving ordinal response. In *Proceedings of the 2014 ACM Conference on Computer and Communications Security*, CCS '14, pages 1054–1067, New York, NY, USA. ACM.

Eubank, N. and Fresh, A. (2022). Enfranchisement and incarceration after the 1965 voting rights act. *American Political Science Review*, pages 1–16.

Evans, G., King, G., Evans, G., King, G., Schwenzfeier, M., Thakurta, A., Katz, J., King, G., Rosenblatt, E., Evans, G., et al. (2021). Statistically valid inferences from differentially private data releases, ii: Extensions to nonlinear transformations. *Political Analysis*, 26(B2):1161–1165.

Falbel, D. (2022). *torchvision: Models, Datasets and Transformations for Images*. R package version 0.4.1.

Falbel, D. and Luraschi, J. (2022). *torch: Tensors and Neural Networks with 'GPU' Acceleration*. R package version 0.7.2.

Fong, C. and Tyler, M. (2021). Machine learning predictions as regression covariates. *Political Analysis*, 29(4):467–484.

Freund, Y. and Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119 – 139.

Frigerio, L., de Oliveira, A. S., Gomez, L., and Duverger, P. (2019). Differentially private generative adversarial networks for time series, continuous, and discrete open data. In *ICT Systems Security and Privacy Protection - 34th*

IFIP TC 11 International Conference, SEC 2019, Lisbon, Portugal, June 25-27, 2019, Proceedings, pages 151–164.

Fukumoto, K. (2022). Nonignorable attrition in pairwise randomized experiments. *Political Analysis*, 30(1):132–141.

Gal, Y. and Ghahramani, Z. (2016). Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059. PMLR.

Garnier, S., Ross, N., Rudis, R., Camargo, A. P., Pedro, A., Sciaini, M., and Scherer, C. (2021). *viridis - Colorblind-Friendly Color Maps for R.* R package version 0.6.2.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative Adversarial Nets. *Advances in Neural Information Processing Systems 27*, pages 2672–2680.

Grossman, G., Pierskalla, J. H., and Boswell Dean, E. (2017). Government fragmentation and public goods provision. *The Journal of Politics*, 79(3):823–840.

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. C. (2017). Improved training of wasserstein gans. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Hangartner, D., Dinas, E., Marbach, M., Matakos, K., and Xefteris, D. (2019). Does exposure to the refugee crisis make natives more hostile? *American political science review*, 113(2):442–455.

Harden, J. J. and Kirkland, J. H. (2021). does transparency inhibit political compromise? *American Journal of Political Science*, 65(2):493–509.

Hardt, M., Ligett, K., and McSherry, F. (2012). A simple and practical algorithm for differentially private data release. In *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States.*, pages 2348–2356.

Hardt, M. and Rothblum, G. N. (2010). A multiplicative weights mechanism for privacy-preserving data analysis. In *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, October 23-26, 2010, Las Vegas, Nevada, USA*, pages 61–70.

Harris, J. A. and Posner, D. N. (2019). (under what conditions) do politicians reward their supporters? evidence from kenya's constituencies development fund. *American Political Science Review*, 113(1):123–139.

Hemker, J. and Rink, A. (2017). Multiple dimensions of bureaucratic discrimination: Evidence from german welfare offices. *American Journal of Political Science*, 61(4):786–803.

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., and Hochreiter, S. (2017). Gans trained by a two time-scale update rule converge to a local nash equilibrium. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Hoang, Q., Nguyen, T. D., Le, T., and Phung, D. (2018a). MGAN: Training generative adversarial nets with multiple generators. In *International Conference on Learning Representations*.

Hoang, Q., Nguyen, T. D., Le, T., and Phung, D. (2018b). MGAN: Training generative adversarial nets with multiple generators. In *International Conference on Learning Representations*.

Hollenbach, F. M., Bojinov, I., Minhas, S., Metternich, N. W., Ward, M. D., and Volfovsky, A. (2021). Multiple imputation using gaussian copulas. *Sociological Methods & Research*, 50(3):1259–1283.

Honaker, J., King, G., and Blackwell, M. (2011). Amelia II: A program for missing data. *Journal of Statistical Software*, 45(7):1–47.

Huster, T., Cohen, J., Lin, Z., Chan, K., Kamhoua, C., Leslie, N. O., Chiang, C.-Y. J., and Sekar, V. (2021). Pareto gan: Extending the representational power of gans to heavy-tailed distributions. In Meila, M. and Zhang, T., editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 4523–4532. PMLR.

Isola, P., Zhu, J.-Y., Zhou, T., and Efros, A. A. (2016). Image-to-image translation with conditional adversarial networks.

Jang, E., Gu, S., and Poole, B. (2016). Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*.

Karras, T., Laine, S., and Aila, T. (2019). A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Karras, T., Laine, S., Aittala, M., Hellsten, J., Lehtinen, J., and Aila, T. (2020). Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Ketchley, N. and El-Rayyes, T. (2021). Unpopular protest: Mass mobilization and attitudes to democracy in post-mubarak egypt. *The Journal of Politics*, 83(1):291–305.

Kim, T., Cha, M., Kim, H., Lee, J. K., and Kim, J. (2017). Learning to discover cross-domain relations with generative adversarial networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, pages 1857–1865. JMLR.org.

King, G., Honaker, J., Joseph, A., and Scheve, K. (2001). Analyzing Incomplete Political Science Data: An Alternative Algorithm for Multiple Imputation. *American Political Science Review*, 95(1):49–69.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization.

Kivinen, J. and Warmuth, M. K. (1997). Exponentiated gradient versus gradient descent for linear predictors. *Information and Computation*, 132(1):1 – 63.

Kropko, J., Goodrich, B., Gelman, A., and Hill, J. (2014). Multiple imputation for continuous and categorical data: Comparing joint multivariate normal and conditional approaches. *Political Analysis*, 22(4):497–519.

Lachapelle, J. (2020). No easy way out: The effect of military coups on state repression. *The Journal of Politics*, 82(4):1354–1372.

Lall, R. and Robinson, T. (2022). The midas touch: Accurate and scalable missing-data imputation with deep learning. *Political Analysis*, 30(2):179–196.

Lee, K. S. and Town, C. (2020). Mimicry: Towards the reproducibility of gan research.

Levy, G. (2022). Evaluations of violence at the polls: Civilian victimization and support for perpetrators after war. *The Journal of Politics*, 84(2):783–797.

Liu, J. and Talwar, K. (2019). Private selection from private candidates. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pages 298–309.

Liu, Z., Luo, P., Wang, X., and Tang, X. (2015). Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*.

López-Moctezuma, G., Wantchekon, L., Rubenson, D., Fujiwara, T., and Pe Lero, C. (2022). Policy deliberation and voter persuasion: Experimental evidence from an election in the philippines. *American Journal of Political Science*, 66(1):59–74.

Lyall, J., Zhou, Y.-Y., and Imai, K. (2020). Can economic assistance shape combatant support in wartime? experimental evidence from afghanistan. *American Political Science Review*, 114(1):126–143.

Maddison, C. J., Mnih, A., and Teh, Y. W. (2016). The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*.

Marbach, M. (2022). Choosing imputation models. *Political Analysis*, 30(4):597–605.

Margolis, M. F. (2018). How politics affects religion: Partisanship, socialization, and religiosity in america. *The Journal of Politics*, 80(1):30–43.

McSherry, F. and Talwar, K. (2007). Mechanism design via differential privacy. In *Proceedings of the 48th Annual IEEE Symposium on Foundations*

of Computer Science, FOCS '07, pages 94–103, Washington, DC, USA. IEEE Computer Society.

Meng, X.-L. (1994). Multiple-imputation inferences with uncongenial sources of input. *Statistical Science*, 9(4):538–558.

Mironov, I. (2017). Rényi differential privacy. In *30th IEEE Computer Security Foundations Symposium, CSF 2017, Santa Barbara, CA, USA, August 21-25, 2017*, pages 263–275.

Mirza, M. and Osindero, S. (2014). Conditional generative adversarial nets.

Mohan, K. and Pearl, J. (2021). Graphical models for processing missing data. *Journal of the American Statistical Association*, 116(534):1023–1037.

Müller, W. (2022). *ganGenerativeData*. R package version 1.3.3.

Mullin, M. and Hansen, K. (2022). Local news and the electoral incentive to invest in infrastructure. *American Political Science Review*, pages 1–6.

Neunhoeffer, M., Wu, S., and Dwork, C. (2021). Private post-gan boosting. In *International Conference on Learning Representations*.

Osgood, I., Tingley, D., Bernauer, T., Kim, I. S., Milner, H. V., and Spilker, G. (2017). The charmed life of superstar exporters: Survey evidence on firms and trade policy. *The Journal of Politics*, 79(1):133–152.

Pal, A. and Das, A. (2021). Torchgan: A flexible framework for gan training and evaluation. *Journal of Open Source Software*, 6(66):2606.

Pan, J. and Xu, Y. (2018). China's ideological spectrum. *The Journal of Politics*, 80(1):254–273.

Papernot, N., Song, S., Mironov, I., Raghunathan, A., Talwar, K., and Erlingsson, Ú. (2018). Scalable private learning with PATE. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*.

Pardos-Prado, S. and Sagarzazu, I. (2019). Economic responsiveness and the political conditioning of the electoral cycle. *The Journal of Politics*, 81(2):441–455.

Park, J. H. and Yamauchi, S. (2022). Change-point detection and regularization in time series cross-sectional data analysis. *Political Analysis*, pages 1–21.

Pepinsky, T. B. (2018). A note on listwise deletion versus multiple imputation. *Political Analysis*, 26(4):480–488.

Pietryka, M. T. and DeBats, D. A. (2017). It's not just what you have, but who you know: Networks, social proximity to elites, and voting in state and local elections. *American Political Science Review*, 111(2):360–378.

Radford, A., Metz, L., and Chintala, S. (2015). Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*.

Radford, A., Metz, L., and Chintala, S. (2016). Unsupervised representation learning with deep convolutional generative adversarial networks. In Bengio, Y. and LeCun, Y., editors, *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.

Redeker, N. (2022). The politics of stashing wealth: The decline of labor power and the global rise in corporate savings. *The Journal of Politics*, 84(2):975–991.

Rivera, C. V. (2020). Loyalty or incentives? how party alignment affects bureaucratic performance. *The Journal of Politics*, 82(4):1287–1304.

Royston, P. (2004). Multiple imputation of missing values. *The Stata Journal*, 4(3):227–241.

Rubin, D. B. (1978). Multiple imputations in sample surveys-a phenomenological bayesian approach to nonresponse. In *Proceedings of the survey research methods section of the American Statistical Association*, volume 1, pages 20–34. American Statistical Association Alexandria, VA, USA.

Rubin, D. B. (1987). *Multiple Imputation for Nonresponse in Surveys*. Wiley, New York.

Rubin, D. B. (1996). Multiple imputation after 18+ years. *Journal of the American statistical Association*, 91(434):473–489.

Rubin, D. B. (2004). *Multiple imputation for nonresponse in surveys*, volume 81. John Wiley & Sons.

Rueda, D. (2018). Food comes first, then morals: Redistribution preferences, parochial altruism, and immigration in western europe. *The Journal of Politics*, 80(1):225–239.

Rueda, M. R. (2017). Small aggregates, big manipulation: Vote buying enforcement and collective monitoring. *American Journal of Political Science*, 61(1):163–177.

Ruggles, S., Flood, S., Goeken, R., Grover, J., Meyer, E., Pacas, J., and Sobek, M. (2019). Ipums usa: Version 9.0 [dataset]. *Minneapolis, MN: IPUMS*, 10:D010.

Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., and Chen, X. (2016). Improved techniques for training gans. In *Advances in neural information processing systems*, pages 2234–2242.

Schaefer, F. and Anandkumar, A. (2019). Competitive gradient descent. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Schultz, K. A. and Mankin, J. S. (2019). Is temperature exogenous? the impact of civil conflict on the instrumental climate record in sub-saharan africa. *American Journal of Political Science*, 63(4):723–739.

Simpser, A. (2020). The culture of corruption across generations: An empirical study of bribery attitudes and behavior. *The Journal of Politics*, 82(4):1373–1389.

Slapin, J. B., Kirkland, J. H., Lazzaro, J. A., Leslie, P. A., and O'grady, T. (2018). Ideology, grandstanding, and strategic party disloyalty in the british parliament. *American Political Science Review*, 112(1):15–30.

Slough, T. (2023). Phantom counterfactuals. *American Journal of Political Science*, 67(1):137–153.

Smidt, C. D. (2017). Polarization and the decline of the american floating voter. *American Journal of Political Science*, 61(2):365–381.

Snoke, J., Raab, G., Nowok, B., Dibben, C., and Slavkovic, A. (2016). General and specific utility measures for synthetic data.

Snoke, J., Raab, G. M., Nowok, B., Dibben, C., and Slavkovic, A. (2018). General and specific utility measures for synthetic data. *Journal of the Royal Sta-*

tistical Society: Series A (Statistics in Society), 181(3):663–688.

Solt, F., Hu, Y., Hudson, K., Song, J., and Yu, D. E. (2017). Economic inequality and class consciousness. *The Journal of Politics*, 79(3):1079–1083.

Song, J. and Ermon, S. (2020). Bridging the gap between f-gans and wasserstein gans. In *International Conference on Machine Learning*, pages 9078–9087. PMLR.

Srivastava, A., Valkov, L., Russell, C., Gutmann, M. U., and Sutton, C. (2017). Veegan: Reducing mode collapse in gans using implicit variational learning. In *Advances in Neural Information Processing Systems*, pages 3308–3318.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958.

Stekhoven, D. J. and Bühlmann, P. (2012). Missforest—non-parametric missing value imputation for mixed-type data. *Bioinformatics*, 28(1):112–118.

Su, Y.-S., Gelman, A., Hill, J., and Yajima, M. (2011). Multiple imputation with diagnostics (mi) in r: Opening windows into the black box. *Journal of Statistical Software*, 45(2):1–31.

Tai, Y. C., Hu, Y., and Solt, F. (2022). Democracy, public support, and measurement uncertainty. *American Political Science Review*, pages 1–7.

Todd, J. D., Malesky, E. J., Tran, A., and Le, Q. A. (2021). Testing legislator responsiveness to citizens and firms in single-party regimes: A field experiment in the vietnamese national assembly. *The Journal of Politics*, 83(4):1573–1588.

Tolstikhin, I., Gelly, S., Bousquet, O., Simon-Gabriel, C.-J., and Schölkopf, B. (2017). Adagan: Boosting generative models. In *Proceedings of the 31st In-*

ternational Conference on Neural Information Processing Systems, NIPS'17, pages 5430–5439, USA. Curran Associates Inc.

Torkzadehmahani, R., Kairouz, P., and Paten, B. (2020). DP-CGAN: differentially private synthetic data and label generation. *CoRR*, abs/2001.09700.

Turner, R., Hung, J., Frank, E., Saatchi, Y., and Yosinski, J. (2019a). Metropolis-Hastings generative adversarial networks. In Chaudhuri, K. and Salakhutdinov, R., editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 6345–6353. PMLR.

Turner, R., Hung, J., Frank, E., Saatchi, Y., and Yosinski, J. (2019b). Metropolis-Hastings generative adversarial networks. In Chaudhuri, K. and Salakhutdinov, R., editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 6345–6353, Long Beach, California, USA. PMLR.

Ushey, K., Allaire, J., and Tang, Y. (2022). *reticulate: Interface to 'Python'*. R package version 1.24.

van Buuren, S. (2018). *Flexible Imputation of Missing Data. Second Edition.* CRC Press, Boca Raton, FL.

van Buuren, S. and Groothuis-Oudshoorn, K. (2011). mice: Multivariate imputation by chained equations in r. *Journal of Statistical Software*, 45(3):1–67.

Wang, J. S. and Aronow, P. M. (2023). Listwise deletion in high dimensions. *Political Analysis*, pages 1–7.

Wang, Y. (2022). Blood is thicker than water: Elite kinship networks and state building in imperial china. *American Political Science Review*, pages 1–15.

Wing, C. (2019). What can instrumental variables tell us about nonresponse in household surveys and political polls? *Political Analysis*, 27(3):320–338.

Xie, L., Lin, K., Wang, S., Wang, F., and Zhou, J. (2018). Differentially private generative adversarial network. *CoRR*, abs/1802.06739.

Xie, X. and Meng, X. L. (2017). Dissecting multiple imput a tion from a multiphase inference perspective: What happens when god's, imputer's and anal Yst's models are uncongenial? *Statistica Sinica*, 27(4):1485–1545.

Xu, L., Skoularidou, M., Cuesta-Infante, A., and Veeramachaneni, K. (2019). Modeling tabular data using conditional gan. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Yoon, J., Jordon, J., and van der Schaar, M. (2018). Gain: Missing data imputation using generative adversarial nets. In Dy, J. and Krause, A., editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 5689–5698. PMLR.

Yoon, J., Jordon, J., and van der Schaar, M. (2019). PATE-GAN: Generating synthetic data with differential privacy guarantees. In *International Conference on Learning Representations*.

Yoon, S. and Sull, S. (2020). Gamin: Generative adversarial multiple imputation network for highly missing data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Yousefpour, A., Shilov, I., Sablayrolles, A., Testuggine, D., Prasad, K., Malek, M., Nguyen, J., Ghosh, S., Bharadwaj, A., Zhao, J., Cormode, G., and Mironov, I. (2021). Opacus: User-friendly differential privacy library in PyTorch. *arXiv preprint arXiv:2109.12298*.

Yu, G., Keirstead, J., and Jefferis, G. (2022). *scholar: Analyse Citation Data from Google Scholar*. R package version 0.2.4.

Zhao, J., Mathieu, M., and LeCun, Y. (2016). Energy-based generative adversarial network.

Zhirkov, K. (2022). Estimating and using individual marginal component effects from conjoint experiments. *Political Analysis*, 30(2):236–249.

Zhu, J.-Y., Park, T., Isola, P., and Efros, A. A. (2017). Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.