Kung Faux Pandas
Simplifying privacy protection

James King, Seth Russell, Tellen D. Bennett, Debashis Ghosh
Data Science to Patient Value (D2V)
School of Medicine
University of Colorado Anschutz Medical Campus
Aurora, CO
{james.king, seth.russell, tell.bennett, debashis.ghosh}@ucdenver.edu

Abstract

There are many barriers to data access and data sharing, especially in the domain of machine learning on health care data. Legal constraints, such as HIPAA, protect patient privacy but slow access to data and limit reproducibility. We provide a description of an end-to-end system called *Kung Faux Pandas* for easily generating de-identified or synthetic data which is statistically similar to real data but lacks sensitive information. This system focuses on data synthesis and de-identification narrowed to a specific research question to allow for self-service data access without the complexities required to generate an entire population of data that is not needed for a given research project. Kung Faux Pandas is an open source publicly available[1] system that lowers barriers to HIPAA- and GDPR-compliant data sharing for enabling reproducibility and other purposes.

1 Introduction

Independent reproduction and replication of results are critical components of scientific inquiry. Barriers to data access and sharing are the most important impediments to reproducibility in machine learning (ML) research. Health care research has unique challenges because of necessary personal data privacy protections.

Several methods exist to de-identify (remove key identifiers, group individuals, and related anonymization techniques [1]) and synthesize data (create data through algorithmic means, population level statistics, randomization, ideally with preservation of multivariable relationships between records [2, 3, 4]). However, no pipeline currently exists with which ML investigators can routinely *anonymize* or *synthesize* data to facilitate access and sharing. This work addresses that gap.

We have created an open source software library called *Kung Faux Pandas (KFP)* which allows for the modular combination of established privacy protection methods with the popular Python Pandas data science library[5]. KFP also provides a Structured Query Language (SQL) interface which enables users to query a database and receive a data set without personal information through either anonymization or synthetic data generation.

2 Background

Recent work has clarified the definition and importance of reproducibility and a closely related concept, replicability. Leek and Peng have defined reproducibility as the "ability to recompute data analytic results given an observed dataset and knowledge of the data analysis pipeline," and

[1] https://github.com/CUD2V/kungfauxpandas

Preprint. Work in progress.

replicability as "the chance that an independent experiment targeting the same scientific question will produce a consistent result."[6, 7] Others have used the terms in a reverse fashion[8]. Despite the semantic differences, there is broad agreement that an independent experiment with confirmatory results is the strongest support of any experiment. One early influential paper on the topic of reproducibility in scientific computing lists the key factors in reproducibility as: available data, input parameters, documentation, software code, and an environment capable of running the provided software code. These items are difficult to completely convey in a traditional research paper [9].

In the context of ML, sufficient detail is required such that an independent investigator could create the same hardware and software configuration, data used for all experiments, source code, documentation on data and how to run/configure software, and tests that verify the software runs correctly. Once a result can be reproduced, new researchers can then build upon the methods, gather new data for testing/validation, or discover alternative methods to replicate a result.

Many domains in which ML investigators work have legal barriers that make it difficult to enable independent verification of ML models which frequently require enormous amounts of training data. As an example, in the health care domain in the United States, a key law governing health data is the Health Insurance Portability and Accountability Act (HIPAA). HIPAA is backed by significant civil penalties as well as specifics about data security and what can and cannot be done with protected health information [10]. While there are other domain specific data privacy laws, there are also some broad rules that affect entire countries such as the European Union's General Data Protection Regulation (GDPR). The GDPR puts in place new rules about data ownership and control in an attempt to give individuals more control over their information.

The current legal environment creates a difficult situation for scientists in the health care domain. Early examples of re-identification attacks [11] have have created apprehension on the part of businesses and legal experts in data sharing [12]. Getting access to source health care data is slow and isn't always granted [13, 14]. More recently, research in the data sharing/privacy protection space has focused more on synthesis techniques such as in the case of the Synthea system, a tool to generate an entire outpatient Electronic Health Record (EHR) based on population statistics [2]. Current solutions are not without their drawbacks however as they frequently require significant configuration of metadata before they can generate data or they attempt to solve the entire solution space and consequently are very slow to run.

3 System Details

One way around the difficulties of source data access is to separate the dependency between data and analysis in such a way that analysis can be performed without the analyst having access to the source data. Such a process can be possible if a sufficiently detailed description of the sensitive data is provided. Appropriate models can be developed and initially validated before either being handed off to a data custodian who then runs the processes on protected data and returns the results to the researcher or access to the protected data has been granted directly to the researcher. Solutions of this form have been possible for some time, but are awkward in practice because activities such as data cleaning, exploration, and model building are interactive and iterative processes that require more than just descriptions of data.

KFP addresses the issue of data access by generating data that is "statistically similar" to the real data but does not itself contain any sensitive data. We collectively refer to de-identified data and synthetic data as *faux-data*. We use the term faux as it is used in the fashion sense of the word: the use of man made materials that look and feel like real leather, fur, etc., but without the problematic derivation from live animals and reliance upon limited resources. This faux-data can be analyzed, cleaned, feature-engineered, etc. as if it were the actual source data. KFP provides a standard mechanism for wrapping a privacy protection method into a plug-in which integrates the method with the Pandas DataFrame model. Two of these plug-ins are provided with full documentation for creating other plug-ins.

As the data generated through KFP has no protected component, it can be shared along with code or methods to improve reproducibility as well as aid in replicability. To further the aim of reproducibility we have provided our code and a sample of a randomly generated dataset for testing purposes through our GitHub repository in a range of formats: Python environment files (conda and pip) to allow others to start with the same software libraries and specific versions that we used, a Docker container with

code and sample data installed and already configured, and all source code along with version history so others can follow our development process.

KFP focuses on generating faux-data narrowed to a specific subset that a researcher is interested in rather than attempting to generate an entire population of data such as an entire EHR or longitudinal patient record. By focusing on just the data needed, performance is improved, the need for extensive modeling or metadata configuration is reduced, and the amount of faux-data generated and stored is minimized. Although the plugable methods included for faux-data generation have been shown to effectively replace source data for analysis, KFP facilitates a more through process that includes an initial exploration with faux-data followed by a repetition of methods on original source data through query logging and thereby aiding in replication.

3.1 System Architecture

Ideally a ML investigator should develop, test, and validate models using all available data. This is most easily accomplished when they have unfettered access to all data in the problem domain they are working in. Figure 1 shows the novel architecture of KFP: data access through an intermediary that can synthesize data on-demand.

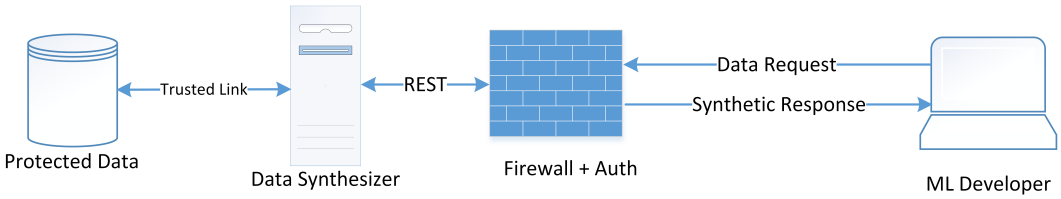

Figure 1: System Architecture.

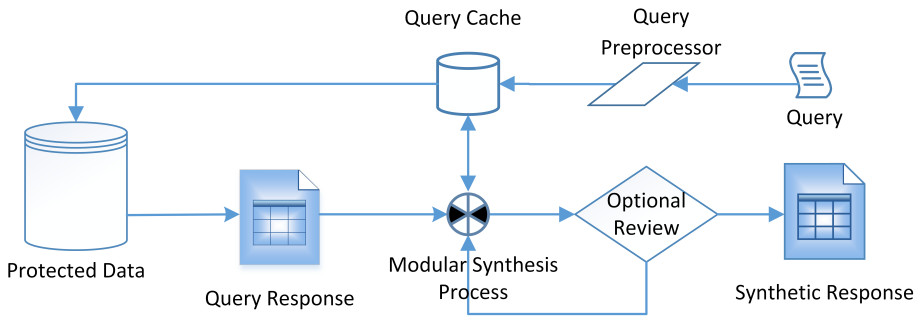

Figure 2: Data synthesis process.

Figure 2 shows the automated faux-data request/response process. All queries submitted to the system are first pre-processed to remove clauses such as "ORDER BY" which are made ineffective by the synthesis process. All queries are logged for later analysis and results cached to optionally improve synthesis performance for repeated queries. Next the processed query is executed against the protected data set and results are passed into the modular synthesis process. While the current implementation only utilizes a couple of synthesis methods, ML developers or data warehouse owners can insert their own custom method or modify existing methods. Previously removed "ORDER BY" clauses are re-applied to the faux-data before results are returned to the user. Finally, although not implemented in this version of KFP, synthetic results could be held back until reviewed and approved for release by the data owner.

KFP provides multiple methods by which a ML developer can access on-demand faux-data. As shown in Figure 3, a single page web application (SPA) allows for users to directly enter SQL queries, submit them to the connected database, and receive faux-data according to the selected generation method. Data can either be viewed directly in browser or downloaded in csv format for use in a ML process. Alternatively, a Hypertext Transfer Protocol (HTTP) Representational State Transfer (REST) service [15], utilized by the SPA, allows for cross language compatible requesting and receiving of faux-data, again via SQL query. In order to facilitate querying and understanding of the data model,

the database metadata is presented to the user in a collapsible section. Lastly, for Python based software or languages with an interface to Python, the kungfauxpandas.py file and associated classes can be imported and called natively.

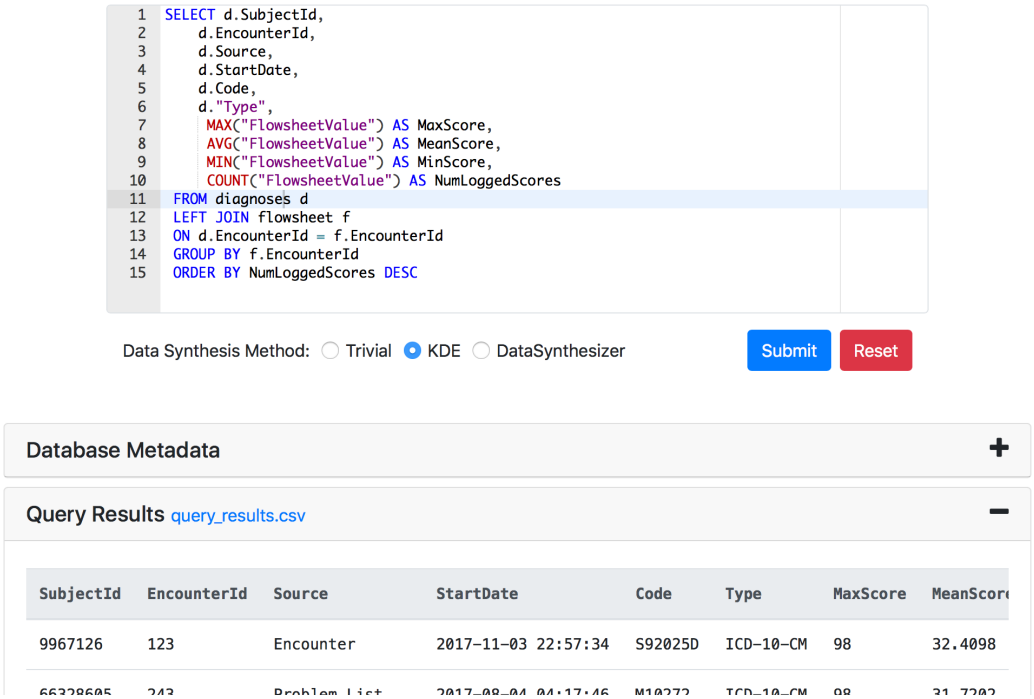

Figure 3: User Interface.

3.2 Included Synthesis Methods

KFP provides a framework by which any number of methods for data privacy can be carried out. For demonstration purposes we have implemented two plugins that use different techniques for data synthesis. The kernel density estimator from scipy.stats.gaussian_kde uses the method developed by Silverman [16] which handles inter-related ordinal and ratio data yet runs quickly on standard consumer level hardware. The DataSynthesizer method, created by Ping et al. [17] is built on differential privacy mechanisms and has multiple internal methods for data generation ranging from random generation based on data type to Bayesian modeling of inter-relationships among columns of data. As a result of their methods, the DataSynthezier offers a unique contribution to the space: it can work with no configuration other than input data, or can be configured to provide more custom results. In DataSynthesizer, the authors show that the synthetic data generated preserves privacy while still being useful enough for actual data analysis (see also [18]).

4 Experiments

In this section we evaluate the performance of the KFP plugins to see how closely the methods meet the stated goals of ease of use, similarity to input data, and total time to run. Hardware for these performance and evaluation tests was a 2017 MacBook Pro, 3.1 GHz Intel Core i7, 16GB 2133 MHz LPDDR3 with macOS 10.12.6. KFP software is written in Python 3.6; all packages and dependencies can be installed with identical versions using the provided requirements.txt, environment.yml file, or Docker container. Database queries were run against the relational database PostgreSQL version 9.6.9. Data was generated using the notebook test_data_generator.ipynb available in our GitHub repository. The sample data population is composed of 100,000 patient encounters with various dimensions spread across several tables, the largest of which contains 10,539,549 rows of data. The sample query generating the "real" input data is composed of 8 total columns composed of IDs, categorical variables, and floating point numbers.

Table 1: Kung Faux Pandas data synthesis plugin timings (mean seconds)

Method	10 rows	100 rows	1000 rows	10000 rows	100000 rows
Trivial	2.5e-06	2.3e-06	2.3e-06	2.2e-06	2.2e-06
KDE	0.086	0.079	0.074	0.074	0.122
DS Independent	0.044	0.050	0.103	0.268	2.608
DS Independent w/Config	0.045	0.040	0.064	0.120	0.799
DS Correlated	0.188	1.419	3.460	16.66	159.9
DS Correlated w/Config	0.172	0.627	1.393	342.7	2124

4.1 Timing Method

Table 1 documents results obtained from running the various plugins. Source code for running these performance tests are available in the Jupyter notebook performance_tests.ipynb. The "Trivial" method provides a baseline comparison for evaluating the overhead of Pandas and correct input/output of the plugin class and function overloading mechanism; it returns the input data frame unmodified. The KDE method is the custom method that acts as an example of how to create plugins. A couple of DataSynthesizer (abbreviated DS in the table) modes are used to demonstrate performance differences as well as generated synthetic data differences. "Independent" mode generates synthetic data with each attribute being modeled independently from the others. "Correlated" attribute mode finds inter-column relationships to more accurately generate synthetic data. Both DataSynthesizer modes are run with no configuration and again "w/Config" by passing in some column-wise dataset specific configuration.

	SubjectId	EncounterId	Source	Code	Type	maxscore	minscore	numloggedscores
0	54382804	64760	Encounter	A327	ICD-10-CM	100.0	0.0	423
1	43596351	57124	Encounter	A403	ICD-10-CM	99.0	0.0	378
2	30688269	71764	Encounter	R364	ICD-10-CM	99.0	0.0	354

Figure 4: Sample of source data

	SubjectId	EncounterId	Source	Code	Type	maxscore	minscore	numloggedscores
0	54219998.0	51970.0	Encounter	gopudh	ICD-10-CM	99.891767	0.0	127.0
1	73784750.0	93130.0	Encounter	yfqp	ICD-10-CM	97.672444	0.0	137.0
2	57100318.0	53866.0	Problem List	diihze	ICD-10-CM	99.823126	0.0	125.0

Figure 5: Sample of synthetic data

4.2 Insights

Figure 4 provides a few rows of the source data for the synthetic plugins. Figure 5 provides a few rows of the synthetic data generated by DataSynthesizer in "Independent" mode.

Fast generation of synthetic data is important in a self-service setting where researchers can generate their own data as needed. As shown in the results, some methods are significantly faster than others as dataset size increases. Additionally, the other key constraint is having synthetic data be as close as possible to real data without reducing privacy. While the KDE plugin is a simple method, it quickly generates data that looks similar to the input data and works well for the range of data types used. DataSynthesizer in "Independent" mode is quick for the tested range of dataset sizes, but without additional configuration, it doesn't generate results that closely match the input dataset. As an example the "Code" column should contain valid International Statistical Classification of Diseases and Related Health Problems, Tenth Revision (ICD-10) codes, but instead has random strings that are similar in length to actual ICD-10 codes that do not line up with values in other columns. The DataSynthesizer "Correlated" attribute mode generates very realistic synthetic data, but has a significant computational burden when generating synthetic data from larger dataset sizes.

In all cases, synthetic data generation is a CPU intensive task; no parallelization has been implemented by either KFP nor the plugins used. While memory usage does increase as data size increases, at the 100000 rows seen here, memory usage peaked at around 2GB for the Python process.

5 Discussion

KFP provides a unique contribution to the space of reproducible ML. Through providing ad-hoc faux-data, ML researchers can have lower barriers to accessing private data as well as reduced barriers to sharing faux-data derived from private data. Although one clear area for further research would be to develop plugins for additional faux-data methods, there are several areas that go beyond this simple step that could be researched further: improved code sharing of KFP, addressing institutional specific privacy issues, and evaluation of this system in a ML workflow.

While we have provided software code and a description of our methods, we believe that ML reproducibility should go beyond just making source code available. Peng recommends making code available in any form as an excellent first step towards reproducible research; the next step is to make code available "in a durable non-proprietary format" [19]. This is a level of code sharing rarely achieved in ML research. Currently there is no standard long term format that works for all computational environments. For the Python ecosystem however, the standard for sharing code is via a package on Python Package Index (PyPI). We intend to continue this research to make KFP available as a Python package.

Another area that is open for further research is that of domain and institution specific data privacy needs. In the field of education, statistical methods for calculating sample size based on probability of detecting a specific effect size are commonly used [20] as the key method for privacy protection. However, it is not clear that these methods are appropriate for use in ML in general or domains other than education. Thus, in this work, no attempt has been made to constrain data set sizes nor provide additional proofs of data protection. KFP could be modified to return no results if a minimum threshold is not met or, alternatively, data boosting techniques in combination with data synthesis could be used based on the risk profile of the data owner(s).

The last area recommended for further research is to deploy and observe the use of KFP in an active ML environment where researchers are working with sensitive datasets. A key limitation that may be identified is that of performance characteristics, particularly in a multi-user setting against a large relational database store. Additionally, as no private data was distributed via this system, actual review by security and compliance groups or institutional review boards may uncover additional requirements to reduce risk of data exposure that could require significant rework or further study.

6 Conclusion

KFP removes barriers to data access and data sharing via self-service faux-data generation. Users of KFP do not need to have access to a protected dataset nor does a human have to be involved in translating researchers data needs into a non-protected dataset. This self-service data can be differentially private or meet other privacy needs based on plugin capabilities. With faux-data, researchers can publicly share data upon which their research relies without organizational fear of privacy breaches. KFP also facilitates a process by which evaluation, analysis, or model building is first performed with faux-data followed by repetition upon private data.

Finally, instead of taking the computational and labor intensive step of full EHR de-identification or synthesis, KFP takes the smaller and more easily implemented step of just-in-time data de-identification or synthesis. As shown in our experiments, if the proper method is selected, data can be quickly generated based on real data that the requester need not have direct access to. We surmise that this tool will improve data exploration and reduce dependency on complex processes required for data access. The result of this system will be improved reproducibility in ML.

References

[1] United States Department of Health and Human Services. Hipaa for professionals. https://www.hhs.gov/hipaa/for-professionals/index.html. Accessed: 2018-05-08.

[2] Jason Walonoski, Mark Kramer, Joseph Nichols, Andre Quina, Chris Moesel, Dylan Hall, Carlton Duffett, Kudakwashe Dube, Thomas Gallagher, and Scott McLachlan. Synthea: An approach, method, and software mechanism for generating synthetic patients and the synthetic electronic health care record. *Journal of the American Medical Informatics Association*, 25(3):230–238, March 2018.

[3] N. Patki, R. Wedge, and K. Veeramachaneni. The synthetic data vault. In *2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, pages 399–410, 2016.

[4] Edward Choi, Siddharth Biswal, Bradley Malin, Jon Duke, Walter F. Stewart, and Jimeng Sun. Generating multi-label discrete patient records using generative adversarial networks. *arXiv:1703.06490 [cs]*, 2017.

[5] Wes McKinney. Data structures for statistical computing in python. In Stéfan van der Walt and Jarrod Millman, editors, *Proceedings of the 9th Python in Science Conference*, pages 51 – 56, 2010.

[6] Jeffrey T. Leek and Roger D. Peng. Opinion: Reproducible research can still be wrong: Adopting a prevention approach. *Proceedings of the National Academy of Sciences*, 112(6):1645–1646, February 2015.

[7] Roger D. Peng, Francesca Dominici, and Scott L. Zeger. Reproducible Epidemiologic Research. *American Journal of Epidemiology*, 163(9):783–789, May 2006.

[8] Chris Drummond. Replicability is not Reproducibility: Nor is it Good Science. In *Proc. of the Evaluation Methods for Machine Learning Workshop at the 26th ICML*, page 4, Montreal, Canada, 2009.

[9] M. Schwab, M. Karrenbach, and J. Claerbout. Making Scientific Computations Reproducible. *Computing in Science & Engineering*, 2(6):61–67, 2000.

[10] American Medical Association. Hipaa violations & enforcement. https://www.ama-assn.org/practice-management/hipaa-violations-enforcement. Accessed: 2018-05-05.

[11] Latanya Sweeney. k-ANONYMITY: A MODEL FOR PROTECTING PRIVACY. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(05):557–570, October 2002.

[12] Paul Ohm. Broken Promises of Privacy: Responding to the Surprising Failure of Anonymization. SSRN Scholarly Paper ID 1450006, Social Science Research Network, Rochester, NY, August 2009.

[13] Jr James G. Hodge, Lawrence O. Gostin, and Peter D. Jacobson. Legal Issues Concerning Electronic Health Information: Privacy, Quality, and Liability. *JAMA*, 282(15):1466–1471, October 1999.

[14] Sharyl J. Nass, Laura A. Levit, and Lawrence O. Gostin, editors. *Beyond the HIPAA Privacy Rule: Enhancing Privacy, Improving Health Through Research*. The National Academies Collection: Reports funded by National Institutes of Health. National Academies Press (US), Washington (DC), 2009.

[15] W3C Working Group. Web Services Architecture. https://www.w3.org/TR/2004/NOTE-ws-arch-20040211/relwwwrest.

[16] B. W. Silverman. *Density estimation for statistics and data analysis*. Number 26 in Monographs on statistics and applied probability. Chapman and Hall, London ; New York, 1986.

[17] Haoyue Ping, Julia Stoyanovich, and Bill Howe. Datasynthesizer: Privacy-preserving synthetic datasets. In *Proceedings of the 29th International Conference on Scientific and Statistical Database Management*, SSDBM '17, pages 42:1–42:5, New York, NY, USA, 2017. ACM.

[18] Bill Howe, Julia Stoyanovich, Haoyue Ping, Bernease Herman, and Matt Gee. Synthetic Data for Social Good. *arXiv:1710.08874 [cs]*, October 2017. arXiv: 1710.08874.

[19] Roger D. Peng. Reproducible Research in Computational Science. *Science (New York, N.y.)*, 334(6060):1226–1227, December 2011.

[20] National Center for Education Statistics. NAEP Analysis and Scaling - Minimum School and Student Sample Sizes for Reporting Group Results, September 2009.