Learning Generative Population Models From Multiple Clinical Datasets Via Probabilistic Programming

Anonymous Authors[1]

Abstract

Accurate, efficient generative models of clinical populations could accelerate clinical research and improve patient outcomes. For example, such models could infer probable treatment outcomes for different subpopulations, generate high-fidelity synthetic data that can be shared across organizational boundaries, and discover new relationships among clinical variables. Using Bayesian structure learning, we show that it is possible to learn probabilistic program models of clinical populations by combining data from multiple, sparsely overlapping clinical datasets. Through experiments with multiple clinical trials and real-world evidence from census health surveys, we show that our model generates higher quality synthetic data than neural network baselines, supports more accurate inferences across datasets than traditional statistical methods, and can be queried more efficiently than both, opening up new avenues for accessible and efficient AI assistance in clinical research.

1. Introduction

Clinical research and practice both depend on patterns and predictions gleaned from data. There has been growing interest in using machine learning to analyze such data at scale (Bourne et al., 2015), potentially accelerating clinical research and improving treatment outcomes—for instance, by analyzing heterogeneous treatment effects (Chernozhukov et al., 2018; Goldstein & Rigdon, 2019), generating synthetic control arms for clinical trials (Popat et al., 2022; Yoshino et al., 2023), or discovering new relationships between clinical variables (Su et al., 2013).

A promising avenue to scale these analyses is to learn generative models that draw on many data sources, such as multiple clinical trials and real-world evidence stemming from surveys or electronic health records (Sherman et al., 2016). The traditional statistical methods used to analyze clinical trials (Food and Drug Administration, 2023) are fast and interpretable, but typically place restrictive assumptions on the data distribution—such as linearity, homoskedasticity, and normality—that are violated when integrating data from multiple sources. Conversely, deep generative models provide a flexible framework for modeling the data-generating process, achieving impressive results in unstructured domains such as text and images (Kingma & Welling, 2013; Vaswani et al., 2017; Goodfellow et al., 2020; Ho et al., 2020; Papamakarios et al., 2021). But they struggle with mixed types and sparse data, and lack efficient efficient ways to perform operations such as computing probabilities, conditioning, or marginalizing.

We build upon **Generative Population Models (GPMs)** (Saad & Mansinghka, 2016), a method that marries the approaches' strengths: like traditional statistics, our models can be queried efficiently; like deep generative models, they can pool heterogeneous data sources and accurately model rich multivariate probability distributions. But unlike (Saad & Mansinghka, 2016), our approach is implemented via **probabilistic programming**, particularly the Sum-Product Programming Language (SPPL) (Saad et al., 2021), allowing for compact, editable representations of model and queries, and exact and efficient inference.

This extended abstract makes three contributions. First, it shows that it is possible to learn generative population models from multiple breast cancer trials combined with real-world evidence from census health surveys, despite the sparse overlaps between these datasets – and that the resulting models are more accurate than popular neural network baselines for modeling tabular data. Second, it shows that several important kinds of clinically relevant queries can be encoded naturally in existing probabilistic programming languages, equipped with generative population models. Third, it shows that generative population models support cross-dataset inferences, in some cases more accurately than transitional statistical methods, and with significantly lower compute costs for querying.

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the ICML 2024 Workshop on Accessible and Efficient Foundation Models for Biological Discovery. Do not distribute.

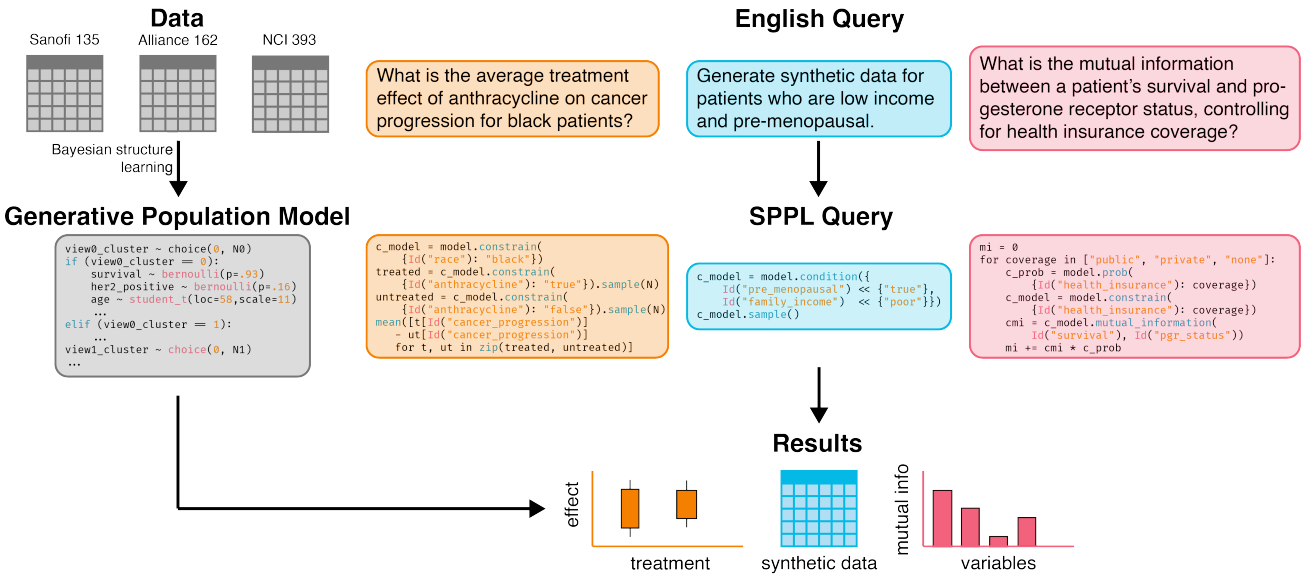

Figure 1. **Generative Population Models for clinical data.** We build accurate generative models from multiple sparsely-overlapping clinical datasets, using Bayesian structure learning (Saad, 2022; Mansinghka et al., 2016; Saad et al., 2019). Model and queries are represented as probabilistic programs (Saad et al., 2021), making them modularly editable (both by humans and by structure learning systems), unlike the entangled weights of a neural network model, and also allowing users to perform exact queries efficiently.

2. Generative Population Models (GPMs)

2.1. Tractable Probabilistic Circuits

In order to build generative models from heterogeneous tabular data sources, we want a class of models that: (i) supports missing data and mixed type distributions (with continuous variables such as age and tumor size, and discrete variables such as cancer stage and ethnicity), (ii) can be conditioned both on discrete and continuous variables, and (iii) permits fast inference for a broad class of queries.

Probabilistic circuits are a class of models that satisfy these desiderata while still being universal approximators of probability densities (see Choi et al. (2020) for a comprehensive review). A probabilistic circuit is represented as a tree that serves both as a sampler and a density evaluator for a fixed-dimensional probability distribution over $\mathbb{R}^n$. The tree contains three types of nodes: sum, product, and leaf. Sum nodes are mixtures with density $P_+(\mathbf{x}) = \sum_{i=1}^n w_i P_i(\mathbf{x})$; when sampling, they are traversed by visiting one child i at random with probability proportional to w_i. Product nodes are factorizations $P_*(\mathbf{x}) = \prod_{i=1}^n P_i(\mathbf{x}_{\text{scope}(i)})$; when sampling, they are traversed by visiting all children. Leaf nodes $P_{\text{L}}(x)$ are "base distributions", which admit both tractable density computation and sampling.

We implement probabilistic circuits using the Sum-Product Probabilistic Language (SPPL, Saad et al. (2021)). Beyond the advantages of probabilistic circuits discussed above, SPPL provides automated compositional queries, which we use to instantiate complex queries as seen in Figure 1. One can also use SPPL as a backend through different querying frontends, such as GenSQL (Huot et al., 2024), which uses a SQL-like DSL allowing more complex query workflows.

2.2. Bayesian Structure Learning

Although non-Bayesian structure learning algorithms exist for probabilistic circuits (Gens & Pedro, 2013; Peharz et al., 2020; Nock & Guillame-Bert, 2022; Watson et al., 2023; Nock & Guillame-Bert, 2023), recent experiments have shown that Bayesian structure learning with non-parametric priors can—perhaps surprisingly—outperform these non-Bayesian methods (Saad & Mansinghka, 2021). Moreover, clinical datasets are often messy and noisy, and only sparsely overlapping after harmonization (i.e., the process of bringing many disparate datasets to a shared space); in this regime, it is natural to take a Bayesian structure learning approach (Saad et al., 2019; Saad, 2022). We therefore infer the structure and parameters of a nested product-of-sums probabilistic circuit by doing Bayesian inference using a hierarchical nonparametric prior (Mansinghka et al., 2016).

2.3. Additional Challenges

Aside from learning the model, there are two key challenges to building generative models of clinical populations in practice that warrant addressing.

Challenge 1: Integrating Heterogeneous Datasets. In order to use data from multiple sources as input to our model,

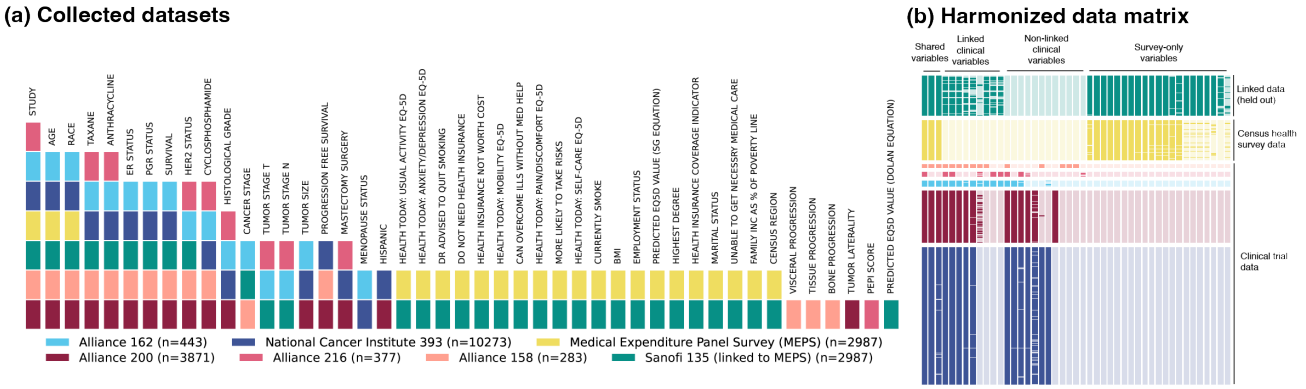

Figure 2. **Data harmonization: (a) Collected datasets**: shared variables for the 7 datasets used. Columns are represented along the x-axis, studies are color-coded. **(b) Harmonized data matrix**: block sparsity pattern for the full harmonized data matrix (n=21,221): rows represent patients, columns represent variables. Rows are colored by study, transparent cells are missing data.

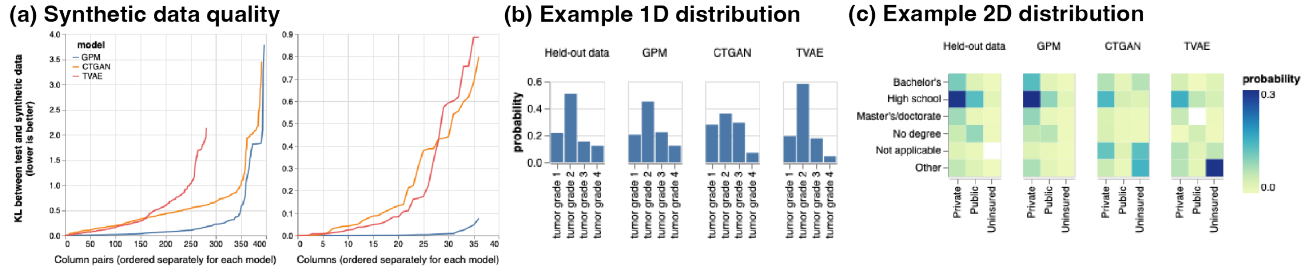

Figure 3. **(a) Synthetic data quality**: KL divergence between test data and synthetic data for the different models, for column pairs and columns. **(b) Example 1D distribution** for the variable "Histological grade", **(c) Example 2D distribution** for the pair of variables "Highest degree" and "Health insurance coverage indicator".

we need to *harmonize* the datasets such that they can be coalesced into a single, block-sparse table (see Figure 2 (a) for an illustration of the different variables across the data sources we used and (b) for the resulting data matrix). Harmonization is an extremely time-consuming process, especially for clinical trials, which vary greatly in their collected information, variable coding, and even file structures. We tackle harmonization by designing a domain-specific language (DSL) that handles a wide range of the most common variable transformations across datasets and crafting a novel Large Language Model "in-the-loop" workflow to assist in writing harmonization programs in our DSL (see Appendix). All resulting harmonization programs are manually assessed by two humans from our author team.

Challenge 2: Efficient Querying. A second key challenge is querying a learned model: after learning from available datasets, we must be able to express a wide range of questions we might ask the model, and efficiently compute answers to those. Here, our model relies on SPPL both for representing queries (see Figure 1) and for answering them exactly with small runtimes (see Figure 4 (c)).

3. Experiments

Data

We harmonize five clinical trials in the Project DataSphere platform (Green et al., 2015) as well as data from the Medical Expenditure Panel Survey (Cohen et al., 2009) collected by the American Census Bureau, and a *linked dataset* which connects patient information from a held-out clinical trial with administrative data Cohen & Unangst (2018). Our harmonized data comprises n=21221 rows, each representing a patient, and 45 columns, each representing a patient attribute (e.g., histological grade, tumor size, age, highest level of education attained; see Figure 2(a) and (b)). Additional details on our harmonization process are included in the Appendix. We hold out the linked dataset for experiment 3.3, and for the rest of the data use a 70/30 train/test split, randomized over all patients (across trials).

3.1. Synthetic Data Quality

We compare our model to two tabular neural networks generative models – a Generative Adversarial Network (CTGAN) and a Variational Autoencoder (TVAE) (Xu et al.,

(a) Cross-dataset prediction

Shared variables								Predicted linked clinical variables		
age	race	taxane	anthracycline	ER status	PGR status	survival	HER2 status	cancer stage	T stage	N stage
70	white	no	no	positive	positive	alive	negative	stage 4	t1	n0
43	white	no	yes	negative	positive	alive	negative	stage 0	t1m	n0(i+)
79	white	no	yes	negative	negative	alive	negative	stage 2A	t3	n1mic
56	asian	no	no	positive	negative	alive	negative	stage 3C	t2	n0

(b) Cross-dataset prediction accuracy

Model	logp(clinical|shared)
GLM	-14.49 ± 9.33
GPM	-9.00 ± 2.38

(c) Cross-dataset query efficiency

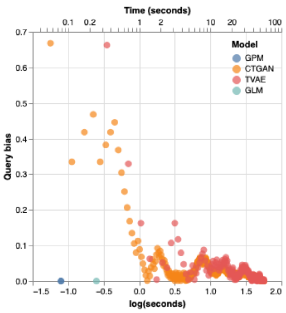

Figure 4. **(a) Cross-dataset prediction:** Four randomly picked rows containing shared variables in the manually linked data from (Cohen & Unangst, 2018) along with samples of GLM predictions of linked clinical trial variables conditioned on shared variables—the prediction task in Experiment 3.3. **(b) Cross-dataset prediction accuracy:** Model-wise probability of clinical trial data variables conditioned on survey data variables for held-out linked data. Higher log probability is better. Error bounds represent standard deviation across rows (i.e., patients). **(c) Cross-dataset query efficiency**: Model-wise bias – the absolute difference between a model's true query probability and its estimated probability – over time for the query "What is the probability of a black patient in the northeast with primary tumor stage T4 being progesterone receptor positive?".

2019). Figure 2(a) shows GPMs consistently outperform neural baselines. Figure 3(b) and (c) show examples of marginal distributions on the test data: we find that the neural baselines often exhibit mode-seeking behavior or greatly underfit the data, whereas GPMs capture qualitative patterns throughout. In particular, GPMs most improve fit over neural baselines for the variables: cancer stage, health insurance coverage, EQ-5D (a quality of life score), and the survey questions of whether a patient thinks they can overcome their illness without medical help and whether health insurance was not worth the cost.

3.2. Query Efficiency

We assess the efficiency of querying each model by computing the probability of an event under a rare condition. Figure 4 (c) shows that the neural network baselines admit significant bias even after tens of seconds of inference as they approximate the query by sampling, in contrast to GPMs which can compute exact results after only a tenth of a second. While GLMs also produce exact results, they take rougly twice as much time as GPMs as they need to be trained from scratch for every query.

3.3. Cross-Dataset Inferences

We assess GPM's ability to perform cross-dataset inferences, by evaluating against *linked data* which connects patient information from a held-out clinical trial with administrative data Cohen & Unangst (2018) (see Appendix A.3 and B.3). We compare against Generalized Linear Models (GLMs), which are commonly used in clinical trial analyses (Food and Drug Administration, 2023). Figure 4 (b) shows that the linked data is approximately three orders of magnitude more likely under GPMs than under GLMs. GPMs also achieve much tighter variance than GLMs.

4. Discussion

Our work demonstrates that it is possible to learn probabilistic programs that (i) generatively model sparsely-overlapping clinical datasets, (ii) predict clinically-meaningful attributes as or more accurately than neural and statistical baselines, and (iii) allow efficient probabilistic querying about clinical sub-populations. More research is needed to scale our approach into a global model of not just cancer but clinical populations more broadly. Challenges include automating data harmonization; handling high degrees of sparsity and low-overlap in Bayesian structure learning; simultaneously cleaning and modeling data (perhaps leveraging probabilistic programming approaches such as PClean (Lew et al., 2021)); and learning richer probabilistic programs that simultaneously reduce compute cost and improve predictive accuracy (Saad & Mansinghka, 2021). GPMs are poised to unlock new possibilities for AI chatbots that can help empower doctors and patients to better navigate diagnosis, prognosis, and personalized treatment. For instance, GPMs could be used as "world models" of cancer to ground conversational AI expert systems (Wong et al., 2023) and support controlled generation and parsing between natural language and the probabilistic programs illustrated in this paper (Lew et al., 2023).

References

Bourne, P. E., Bonazzi, V., Dunn, M., Green, E. D., Guyer, M., Komatsoulis, G., Larkin, J., and Russell, B. The NIH Big Data to Knowledge (BD2K) initiative. *Journal of the American Medical Informatics Association*, 22(6): 1114–1114, 11 2015. ISSN 1067-5027. doi: 10.1093/jamia/ocv136. URL https://doi.org/10.1093/jamia/ocv136.

Chernozhukov, V., Demirer, M., Duflo, E., and Fernandez-Val, I. Generic machine learning inference on heterogeneous treatment effects in randomized experiments, with an application to immunization in india. Technical report, National Bureau of Economic Research, 2018.

Choi, Y., Vergari, A., and Van den Broeck, G. Probabilistic circuits: A unifying framework for tractable probabilistic models. pp. 6, 2020. URL http://starai.cs.ucla.edu/papers/ProbCirc20.pdf.

Cohen, J. W., Cohen, S. B., and Banthin, J. S. The medical expenditure panel survey: a national information resource to support healthcare cost research and inform policy and practice. *Medical care*, 47(7_Supplement_1):S44–S50, 2009.

Cohen, S. B. and Unangst, J. Data integration innovations to enhance analytic utility of clinical trial content to inform health disparities research. *Frontiers in Oncology*, 8:365, 2018.

Food and Drug Administration. Adjusting for covariates in randomized clinical trials for drugs and biological products. Technical Report FDA-2019-D-0934, 10001 New Hampshire Ave., Hillandale Bldg., 4th Floor, Silver Spring, MD, 2023.

Gens, R. and Pedro, D. Learning the structure of sum-product networks. In Dasgupta, S. and McAllester, D. (eds.), *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pp. 873–880, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR.

Goldstein, B. A. and Rigdon, J. Using machine learning to identify heterogeneous effects in randomized clinical trials—moving beyond the forest plot and into the forest. *JAMA network open*, 2(3):e190004–e190004, 2019.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

Green, A. K., Reeder-Hayes, K. E., Corty, R. W., Basch, E., Milowsky, M. I., Dusetzina, S. B., Bennett, A. V., and Wood, W. A. The project data sphere initiative: accelerating cancer research by sharing data. *The oncologist*, 20 (5):464–e20, 2015.

Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

Huot, M., Ghavamizadeh, M., Lew, A. K., Schaechtle, U., Freer, C. E., Shelby, Z., Rinard, M. C., Saad, F. A., and Mansinghka, V. K. Gensql: A probabilistic programming system for querying generative models of database tables. In *PLDI 2024: Proceedings of the 45th ACM SIGPLAN Conference on Programming Language Design and Implementation*, volume 8. ACM, 2024.

Kingma, D. P. and Welling, M. Auto-encoding variational Bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Lew, A., Agrawal, M., Sontag, D., and Mansinghka, V. Pclean: Bayesian data cleaning at scale with domain-specific probabilistic programming. In *International Conference on Artificial Intelligence and Statistics*, pp. 1927–1935. PMLR, 2021.

Lew, A. K., Zhi-Xuan, T., Grand, G., and Mansinghka, V. K. Sequential monte carlo steering of large language models using probabilistic programs. *arXiv preprint arXiv:2306.03081*, 2023.

Mansinghka, V., Shafto, P., Jonas, E., Petschulat, C., Gasner, M., and Tenenbaum, J. B. Crosscat: A fully bayesian nonparametric method for analyzing heterogeneous, high dimensional data. *Journal of Machine Learning Research*, 17(138):1–49, 2016.

Nock, R. and Guillame-Bert, M. Generative trees: Adversarial and copycat. *arXiv preprint arXiv:2201.11205*, 2022.

Nock, R. and Guillame-Bert, M. Generative forests. *arXiv preprint arXiv:2308.03648*, 2023.

OpenAI. ChatGPT, 2024. URL https://openai.com/blog/chatgpt.

Papamakarios, G., Nalisnick, E., Rezende, D. J., Mohamed, S., and Lakshminarayanan, B. Normalizing flows for probabilistic modeling and inference. *Journal of Machine Learning Research*, 22(57):1–64, 2021.

Peharz, R., Vergari, A., Stelzner, K., Molina, A., Shao, X., Trapp, M., Kersting, K., and Ghahramani, Z. Random sum-product networks: A simple and effective approach to probabilistic deep learning. In *Uncertainty in Artificial Intelligence*, pp. 334–344. PMLR, 2020.

Popat, S., Liu, S. V., Scheuer, N., Hsu, G. G., Lockhart, A., Ramagopalan, S. V., Griesinger, F., and Subbiah, V. Addressing challenges with real-world synthetic control arms to demonstrate the comparative effectiveness of pralsetinib in non-small cell lung cancer. *Nature communications*, 13(1):3500, 2022.

Saad, F. and Mansinghka, V. K. A probabilistic programming approach to probabilistic data analysis. In Lee, D., Sugiyama, M., Luxburg, U., Guyon, I., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. URL https://proceedings.neurips.cc/paper_files/paper/2016/file/46072631582fc240dd2674a7d063b040-Paper.pdf.

Saad, F. A. and Mansinghka, V. K. Hierarchical infinite relational model. In *Uncertainty in Artificial Intelligence*, pp. 1067–1077. PMLR, 2021.

Saad, F. A., Cusumano-Towner, M. F., Schaechtle, U., Rinard, M. C., and Mansinghka, V. K. Bayesian synthesis of probabilistic programs for automatic data modeling. *Proceedings of the ACM on Programming Languages*, 3 (POPL):1–32, 2019.

Saad, F. A., Rinard, M. C., and Mansinghka, V. K. SPPLs: probabilistic programming with fast exact symbolic inference. In *Proceedings of the 42nd acm sigplan international conference on programming language design and implementation*, pp. 804–819, 2021.

Saad, F. A. K. *Scalable Structure Learning, Inference, and Analysis with Probabilistic Programs*. PhD thesis, Massachusetts Institute of Technology, 2022.

Sherman, R. E., Anderson, S. A., Dal Pan, G. J., Gray, G. W., Gross, T., Hunter, N. L., LaVange, L., Marinac-Dabic, D., Marks, P. W., Robb, M. A., et al. Real-world evidence—what is it and what can it tell us. *N Engl J Med*, 375(23):2293–2297, 2016.

Su, C., Andrew, A., Karagas, M. R., and Borsuk, M. E. Using bayesian networks to discover relations between genes, environment, and disease. *BioData mining*, 6: 1–21, 2013.

Van Buuren, S. and Groothuis-Oudshoorn, K. mice: Multivariate imputation by chained equations in r. *Journal of statistical software*, 45:1–67, 2011.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Watson, D. S., Blesch, K., Kapar, J., and Wright, M. N. Adversarial random forests for density estimation and generative modeling. In Ruiz, F., Dy, J., and van de Meent, J.-W. (eds.), *Proceedings of The 26th International Conference on Artificial Intelligence and Statistics*, volume 206 of *Proceedings of Machine Learning Research*, pp. 5357–5375. PMLR, 25–27 Apr 2023. URL https://proceedings.mlr.press/v206/watson23a.html.

Wong, L., Grand, G., Lew, A. K., Goodman, N. D., Mansinghka, V. K., Andreas, J., and Tenenbaum, J. B. From word models to world models: Translating from natural language to the probabilistic language of thought. *arXiv preprint arXiv:2306.12672*, pp. arXiv–2306, 2023.

Xu, L., Skoularidou, M., Cuesta-Infante, A., and Veeramachaneni, K. Modeling tabular data using conditional gan. *Advances in neural information processing systems*, 32, 2019.

Yoshino, T., Shi, Q., Misumi, T., Bando, H., Wakabayashi, M., Raeisi, M., Andre, T., and de Gramont, A. A synthetic control arm for refractory metastatic colorectal cancer: the no placebo initiative. *Nature Medicine*, 29(10):2389–2390, 2023.

A. Additional Details on Data and Processing

A.1. DSL for Data Harmonization

As we discuss, a crucial first step to learning a model over multiple heterogenous data sets is to *harmonize* bring them to a shared space. We introduce a new domain specific language (DSL) for reasoning about harmonization. In particular, we support four fundamental harmonization operations `remap`, `discretize`, and `identity`. We sourced these operations by taxonomizing the kinds of harmonization tasks that we found ourselves performing while manually harmonizing patient data.

Our harmonization procedure works as follows. First, the harmonizer (or respective stakeholder) provides a series of target variables that they want to be included in the final data, along with a set of target values that the variable takes on (e.g., all possible cancer stages or the respective range of a continuous variable). Ideally, this would be any possible variable covered in the data; for now, we filter these to variables with reasonable coverage.

Harmonization for each dataset then proceeds with finding the variable(s) in the observed data that correspond to each of the target variables (if present). Once matched, the form of corruption from the target variable to the emitted variable is categorized. Herein, our harmonization DSL comes to the foreground. If the observed variable employs a different categorization than the target (say, referring to a biomarker as being "on" or "off" rather than "positive" or "negative") or coarsens the variable (e.g., storing stage as a patient being "stage IV" or "not stage IV"), the `remap` operation lets the harmonizer specify a dictionary mapping the target variables to how they are observed (see the LLM prompt for an example). Relatedly, if the target variable is a continuous value (like age) but stored in the observed data in discrete bins, the `discretize` type lets the user specify the binning transform. Lastly, if the target and observed variables share the same space, the `identity` operation indicates no change is needed. We plan to extend our DSL to support opertions like `convert` if the target and observed variables are both continuous but stored in different units, such that the user can specify the transformation function (which is a common when reasoning about lab tests), `combine` to support reasoning about cases where two variables are combined in the observed data (e.g., estrogen and progestrone markers) and `aggregate` to support reasoning over aggregated population information.

We then read in the resulting harmonization record (a json) to our harmonization tool, which parses the resulting entries and in the case of an ill-posed mapping (e.g, coarser observed variables), samples from a uniform distribution over the options. Our tool then merges the resulting datasets into a single data file. We emphasize that this tool is preliminary; future work could better automate type checking and handling of uncertainty (e.g., smarter priors). However, we see immense promise in efficiency gains that can ensue from automating parts of the harmonization pipeline.

Additionally, our DSL supports more structured reasoning about the harmonization process. As a nice corrolate, we can use an LLM (e.g., ChatGPT(OpenAI, 2024)) as a first-pass parsing module from relevant dataset documentation[1]. We include our prompt below. We emphasize here, however, that LLMs are *in-the-loop* of a human-driven harmonization. Two authors from our author team manually inspect and intervene on the resulting harmonization parsers, where necessary. We observe several failure modes in the LLM parsing that necessitate human intervention, e.g., making up variables that did not exist, inappropriately matching observed and target columns, or pulling the wrong coding information (as some data dictionaries provide text on multiple coding schemes per variable). Nonetheless, we believe our harmonization pipeline – and importantly, conceptual cast of a DSL – can support more principled, efficient harmonization going forwards.

A.2. LLM Prompting

We prompt ChatGPT with a variant of the following, where "breast_cancer_variables.pdf" is our target set of variables and ideal target coding.

Prompt "Your task is to write a json file that maps all the idealized columns listed in the pdf "breast_cancer_vars.pdf" into the corresponding materialized column names references in the data dictionary "data_dictionary.pdf". The json file should have the following structure:

```
{"null_values": [str1, str2...], "emissions": [ { "idealized_column_name":
   idealized_column1, "materialized_column_name": materialized_column1, "emission": "
   identity" }, { "idealized_column_name": idealized_column2, "materialized_column_name":
    materialized_column2, "emission": "remap", "mapping": {idealized_column_level1:
```

[1]Each PDS clinical trial often comes with a "data dictionary" providing information on the variables collected.

```
    materialized_column_level1, idealized_column_level2: materialized_column_level2} }, {
    "idealized_column_name": idealized_column3, "materialized_column_name":
    materialized_column3, "emission": "discretize", "bins": [10, 20, 30], "names": ["0-10"
    , "10-20", "20-30", ">30"] }, { "idealized_column_name": idealized_column4,
       emission    :    None    , }, ] }
```

null_values should be a list of all strings that are to be read as missing values in the data dictionary.

emissions should contain column mappings for each of these idealized column names:

```
age
race
hispanic
country
cancer_stage
tumor_stage_T
tumor_stage_N
tumor_stage_M
er_status
pgr_status
bmi
mastectomy_surgery
survival
progression_free_survival
treatment
tumor_size
tumor_laterality
her2_status
histological_grade
menopause_status
visceral_progression
tissue_progression
bone_progression
treatment_end_reason
pepi_score
prior_hormonal_therapy
prior_chemotherapy
num_positive_lymph_nodes
```

If there is no matching column, remember to categorize it as None.

Columns mappings can have one of 4 types:

```
None
    variable is not present in the other dataset (no field    materialized_column_name    )
Identity
    no change in variable coding.
    signature: {"emission": "identity"}
Remap
    Change a categorical variables' levels. "mapping" is a dict mapping from the idealized
        column's levels to the materialized column's levels
    {"emission": "remap", "mapping": {idealized_column_level1: materialized_column_level1,
        ...}}
Discretize
    Discretize a continuous variable. "bins" is a list of N values along which the
       continuous variable will be split, "names" is a list of N+1 values which will be
       the names of the resulting discrete variables.
    {"emission": "discretize", "bins": [x1, x2...], "names": [name1, name2, name3...]}
```

First, write out a rationale for any tricky conversions. Then write your file below after the line ”—-FILE—“.

In the file, WRITE JSON ONLY. There should be one entry for each of the idealized column names listed above. And at most one self-map for each entry (though there may be none). If you need to think while writing, feel free to add a new key in each entry called ”comments“.”

A.3. Linked Data

In the context of our oncology data, *linked data* are datasets wherein patient information from clinical trials are "matched" in some way with similar people from another dataset such that additional attributes are included for each patient. This takes the form of adding more columns (variables) to the clinical trial, wherein each patient has more information "linked" to them from another data source (e.g., bringing in their likely socioeconomic status, highest level of education attained, etc). Here, we consider the Sanofi clinical trial linkage to MEPS administrative survey data from Cohen & Unangst (2018).

As can be seen in Figure 2, three variables are shared ("bmi", "age", and "race") and nine other clinical variables are linked which we predict ("estrogen receptor status", "progesterone receptor status", "survival", "her-2 status", "cancer stage", "tumor stage (T)", "tumor stage (N)", and two treatments – "taxane" and "anthracycline").

B. Additional Modeling Details

B.1. Neural Baselines

We need to make a few amendments to our setup in order to assess the neural network baselines. As the neural network models cannot model missing data, we need to run an additional step for the baselines; we use Multiple Imputation with Chained Equations (Van Buuren & Groothuis-Oudshoorn, 2011) with a random forest regressor.

B.2. Evaluation of Synthetic Data Quality

Further, while the most natural way to assess the quality of the learned generative models is to compute the probability of the test data under each model, unfortunately, the neural network baselines do not support that operation: the evidence lower-bound in our TVAE baseline is just an approximation to that probability, and the CTGAN has no notion of the probability of a sample. Since all models support sampling, we instead generate synthetic data from each model, and then compute the KL divergence between columns and column pairs in the test data and the synthetic data—approximating the joint distribution by way of marginals. We filter out columns and column pairs which have fewer than 100 patients with recorded data, and discretize continuous columns by quintiles.

B.3. Additional Details on Linked Data Experiment

We consider the same set of clinical trials as in our synthetic data experiment, and add an additional component focused on linked data (as described in A.3) to permit exploration of cross-dataset inferences. Importantly, this clinical trial was not present in our training set. As such, we can use it to test our model's ability to link clinical trial and MEPS data. Specifically, we can compute the probability of the clinical trial variables for the linked patients conditioned on their MEPS variables (which were observed in the training set).

We cannot directly compare to the two neural network generative models from Experiment 3 as they do not naturally support conditional generation. Instead, we compare to Generalized Linear Models (GLMs), which are the norm in clinical trial analyses (Food and Drug Administration, 2023). We fit a separate GLM to the training data to predict each PDS variable using the MEPS variables, and handle missing predictors by performing complete case analysis. The linked Sanofi study has 10 variables that were not present in the MEPS data—to avoid mixing probabilities and densities in the analysis we drop the single continuous column so that we're predicting the joint probability of 9 categorical variables (barring missing entries) for each row, as detailed in A.3 and Figure 1.