Suppelementary Material for Noise-Aware Statistical Inference with Differentially Private Synthetic Data

A Privacy theorems for NAPSU-MQ

The privacy bounds of the Gaussian mechanism depend on the *sensitivity* of the function f, which is an upper bound on the change in the value of f for neighbouring datasets.

Definition A.1. *The L_2-sensitivity of a function f is $\Delta_2 f = \sup_{X \sim X'} ||f(X) - f(X')||_2$. $X \sim X'$ denotes that X and X' are neighbouring.*

Theorem A.2. *Let a be the concatenation of n_s full sets of marginal queries. Then $\Delta_2 a \leq \sqrt{2n_s}$.*

Proof. Let $a_1, \ldots, a_{n_s}$ be the full sets of marginal queries that form a. Because all of the queries of a_i have the same set of variables, the vector $a_i(x)$ has a single component of value 1, and the other components are 0 for any $x \in \mathcal{X}$. Then, for any neighbouring $X, X' \in \mathcal{X}^n$, $||a_i(X) - a_i(X')||_2^2 \leq 2$. Then

$$\Delta_2 a = \sup_{X \sim X'} ||a(X) - a(X')|| \tag{5}$$

$$= \sup_{X \sim X'} \sqrt{\sum_{i=1}^{n_s} ||a_i(X) - a_i(X')||_2^2} \tag{6}$$

$$\leq \sup_{X \sim X'} \sqrt{\sum_{i=1}^{n_s} 2} \tag{7}$$

$$= \sqrt{2n_s} \tag{8}$$

□

Theorem A.3 (Balle and Wang [3])**.** *The Gaussian mechanism for function f with L_2-sensitivity Δ_2 and noise variance σ^2 is (ϵ, δ)-DP with*

$$\delta \geq \Phi\left(\frac{\Delta_2}{2\sigma} - \frac{\epsilon\sigma}{\Delta_2}\right) - e^{\epsilon}\Phi\left(-\frac{\Delta_2}{2\sigma} - \frac{\epsilon\sigma}{\Delta_2}\right) \tag{9}$$

where Φ is the cumulative distribution function of the standard Gaussian distribution.

B Multiple Imputation

In order to compute uncertainty estimates for downstream analyses from the noise-aware posterior with NA+MI, we use Rubin's rules for synthetic data [30, 31].

After the synthetic datasets X_i^{Syn} for $1 \leq i \leq m$ are released by the data holder, the data analyst runs their downstream analysis on each X_i^{Syn}. For each synthetic dataset, the analysis produces a point estimate q_i and a variance estimate v_i for q_i.

The estimates $q_1, \ldots, q_m$ and $v_1, \ldots, v_m$ are combined as follows [30]:

$$\bar{q} = \frac{1}{m}\sum_{i=1}^{m} q_i, \quad \bar{v} = \frac{1}{m}\sum_{i=1}^{m} v_i, \quad b = \frac{1}{m-1}\sum_{i=1}^{m}(q_i - \bar{q})^2. \tag{10}$$

We use $\bar{q}$ as the combined point estimate, and set

$$T = \left(1 + \frac{1}{m}\right) b - \bar{v}, \qquad T^* = \begin{cases} T & \text{if } T \geq 0 \\ \frac{n_{Syn}}{n}\bar{v} & \text{otherwise.} \end{cases} \tag{11}$$

T an estimate of the combined variance. T can be negative, which is corrected using T^* instead [31].

We compute confidence intervals and hypothesis tests using the t-distribution with mean $\bar{q}$, variance T^*, and degrees of freedom

$$\nu = (m-1)(1-r^{-1})^2, \tag{12}$$

where $r = (1 + \frac{1}{m})\frac{b}{\bar{v}}$ [31].

These combining rules apply when q is a univariate estimate. Reiter [32] derives appropriate combining rules for multivatiate estimates, which can be applied with NA+MI.

Rubin's rules make many assumptions on the different distributions that are involved [30, 34], such as the normality of the distribution of q_i when sampling data from the population. These assumptions may not hold for some types of estimates, such as probabilities [23] or quantile estimates [40]. Further work [12, 34] tries to reduce these assumptions, especially in the context of missing data. Their results for synthetic data generation can be applied with our method.

Si and Reiter [34] propose to remove some of these assumptions by approximating the integral that Rubin's rules are derived from by sampling instead of using the analytical approximations in (10) and (11). They find that their sampling-based approximation can be effective, especially with a small number of datasets, but is computationally more expensive.

In the missing data context, when the downstream task uses Bayesian inference, Gelman et al. [12] propose to mix the samples of each downstream posterior, and use the mixed posterior for inferences, which doesn't require the normality assumptions that Rubin's rules require. However, this is restricted to Bayesian downstream tasks, and was originally proposed for the missing data context, but it may be applicable to synthetic data and our method.

B.1 Unbiasedness of Rubin's Rules

Rubin's rules make several assumptions on the downstream analysis method, and several normal approximations when deriving the rules. Raghunathan, Reiter, and Rubin [30] derive conditions under which Rubin's rules give an unbiased estimate.

Rubin's rules aim to estimate a quantity Q of the entire population P, of which X is a sample. Conceptually, the sampling of the synthetic datasets is done in two stages: first, synthetic populations P_i^{Syn} for $1 \leq i \leq m$ are sampled. Second, a synthetic dataset X_i^{Syn} is sampled from P_i^{Syn}. This is equivalent to the sampling process for X_i^{Syn} described in Section 2, and makes stating the assumptions of Rubin's rules easier.

Let Q_i denote the quantity of interest Q computed from the synthetic population P_i^{Syn} instead of P. Let V_i denote the sampling variance of q_i from the synthetic population P_i^{Syn}. Let $\hat{Q}_D$ and $\hat{U}_D$ be the point and variance estimates of Q when sampling from the population P.

Assumption B.1. *For all $1 \leq i \leq m$, q_i is unbiased for Q_i and asymptotically normal with respect to sampling from the synthetic population P_i^{Syn}, with sampling variance V_i.*

Assumption B.2. *For all $1 \leq i \leq m$, v_i is unbiased for V_i, and the sampling variability in v_i is negligible. That is $v_i \mid P_i^{Syn} \approx V_i$. Additionally, the variation in V_i across the synthetic populations is negligible.*

Assumption B.3. $\hat{Q}_D \mid P \sim \mathcal{N}(Q, \hat{U}_D)$

Assumptions B.1-B.3 ensure that the downstream analysis method used to estimate Q is accurate, for both point and variance estimates, when applied to real data, regardless of the population.

Assumption B.4. $Q_i \mid X \sim \mathcal{N}(\hat{Q}_D, \hat{U}_D)$

Assumption B.4 requires that the generation of synthetic datasets does not bias the downstream analysis. For query-based methods like NAPSU-MQ, it may not hold when the queries do not contain the relevant information for the downstream task.

With Assumptions B.1-B.4, Raghunathan, Reiter, and Rubin [30] show that $\bar{q}$ is an unbiased estimate of Q, and T in an asymptotically unbiased variance estimate.

Theorem B.5 (Raghunathan, Reiter, and Rubin [30])**.** *Assumptions B.1-B.4 imply that*

1. $E(\bar{q} \mid P) = Q$,
2. $E(T \mid P) = \mathrm{Var}(\bar{q} \mid P)$,
3. *Asymptotically* $\frac{\bar{q}-Q}{\sqrt{T}} \sim \mathcal{N}(0,1)$,
4. *For moderate* m, $\frac{\bar{q}-Q}{\sqrt{T}} \sim t_\nu(0,1)$ *[31]*.

C Finding an Identifiable Parameterisation

In this section, we describe the process we use to ensure the parameterisation of the posterior in NAPSU-MQ is identifiable. We ensure identifiability by dropping some of the selected queries, chosen using the the canonical parameterisation of MED_θ to ensure no information is lost. First, we give some background on Markov networks, which is necessary to understand the canonical parameterisation.

Markov Networks A Markov network is a representation of a probability distribution that is factored according to an undirected graph. Specifically, a Markov network distribution P is a product of *factors*. A factor is a function from a subset of the variables to non-negative real numbers. The subset of variables is called the *scope* of the factor. The joint distribution is given by

$$P(x) = \frac{1}{Z} \prod_{I \subset S} \phi_I(x_I) \tag{13}$$

where S is the set of scopes for the factors. The undirected graph is formed by representing each variable as a node, and adding edges such that the scope of each factor is a clique in the graph.

Canonical Parametrisation The canonical parametrisation is given in terms of *canonical factors* [1]. The canonical factors depend on an arbitrary assignment of variables x^*. We simply choose $x^* = (0, \ldots, 0)$. In the following, x_U denotes the selection of components in the set U from the vector x, and x_{-U} denotes the selection of all components except those in U.

Definition C.1. *A canonical factor ϕ^*_D with scope D is defined as*

$$\phi^*_D(x) = \exp\left(\sum_{U \subseteq D} (-1)^{|D-U|} \ln P(x_U, x^*_{-U}) \right)$$

The sum is over all subsets of D, including D itself and the empty set. $|D-U|$ is the size of the set difference of D and U.

Theorem C.2 (Abbeel, Koller, and Ng [1](Theorem 3))**.** *Let P be a Markov network with factor scopes S. Let $S^* = \cup_{D \in S} \mathcal{P}(D) - \emptyset$. Then*

$$P(x) = P(x^*) \prod_{D^* \in S^*} \phi^*_{D^*}(x_{D^*})$$

There are more canonical factors than original factors, so it might seem that there are more parameters in the canonical parametrisation than in the original parametrisation. However, many values in the canonical factors turn out to be ones. We can select the queries corresponding to non-one canonical factor values to obtain a set of queries with the same information as the original queries, but without linear dependencies [19]. We call this set of queries the *canonical queries*.

Many of the canonical factor scopes are subsets of the original factor scopes, so using the canonical queries as is would introduce new marginal query sets and potentially increase the sensitivity of the queries. As all of the new queries are sums of existing queries, we can replace each new query with the old queries that sum to the new query, and use the same θ value for all of the added queries to preserve identifiability. If one of the added queries was already included, it does not need to be added again, because two instances of a single query can be collapsed into a single instance with it's own parameter value. Because of this, we did not need to fix the θ values of any queries to the same value in the settings we studied.

D NAPSU-MQ vs. PGM

The PGM algorithm [27] generates synthetic data based on the same marginal queries a and noise addition as NAPSU-MQ. PGM also models the original data using the MED_θ distribution. Unlike NAPSU-MQ, PGM finds the parameters θ by minimising the l_2-distance $||\tilde{s} - n\mu(\theta)||_2$ between the observed noisy query values $\tilde{s}$ and the expected query values $n\mu(\theta) = nE_{x\sim\mathrm{MED}_\theta}(a(x))$ In the following, we'll replace the query values s and $\tilde{s}$ that are summed over datapoints with $u = \frac{s}{n}$ and $\tilde{u} = \frac{\tilde{s}}{n}$ that represent mean query values over datapoints. Then the PGM objective is equivalent to $||\tilde{u} - \mu(\theta)||_2$.

We can view the PGM minimisation problem as a maximum likelihood estimation in the NAPSU-MQ probabilistic model

$$X \sim \mathrm{MED}_\theta^n, \quad s = a(X), \quad \tilde{s} \sim \mathcal{N}(s, \sigma_{DP}^2 I), \tag{14}$$

where we replace normal approximation that NAPSU-MQ uses with a law of large numbers approximation. Specifically, first replace s with u in (14):

$$X \sim \mathrm{MED}_\theta^n, \quad u = \frac{a(X)}{n}, \quad \tilde{u} \sim \mathcal{N}(u, \sigma_{DP}^2 I/n^2). \tag{15}$$

Because u is a mean of sufficient statistics for individual datapoints, by the law of large numbers, asymptotically $u \sim \delta_{\mu(\theta)}$. With this approximation, the probabilistic model is

$$u \sim \delta_{\mu(\theta)}, \quad \tilde{u} \sim \mathcal{N}(u, \sigma_{DP}^2 I/n^2). \tag{16}$$

u can be marginalised from the likelihood of this model:

$$\begin{aligned} p(\tilde{u}|\theta) &= \int p(\tilde{u}, u|\theta)\mathrm{d}u && (17)\\ &= \int p(\tilde{u}|u)p(u|\theta)\mathrm{d}u && (18)\\ &= \int \mathcal{N}(\tilde{u}|u, \sigma_{DP}^2 I/n^2)\delta_{\mu(\theta)}(u)\mathrm{d}u && (19)\\ &= \mathcal{N}(\tilde{u}|\mu(\theta), \sigma_{DP}^2 I/n^2) && (20) \end{aligned}$$

The marginalised log-likelihood is then

$$\ln p(\tilde{u}|\theta) = -\frac{n^2}{\sigma_{DP}^2}||\tilde{u} - \mu(\theta)||_2^2 + \text{constant}, \tag{21}$$

so maximising the log-likelihood is equivalent to minimising the PGM objective.

If we made a normal approximation instead of the law of large numbers approximation in (15), we would get

$$\tilde{u} \sim \mathcal{N}(\mu(\theta), \Sigma(\theta)/n + \sigma_{DP}^2 I/n^2), \tag{22}$$

so maximising the likelihood is still possible. Unlike PGM, this maximum likelihood objective includes the covariance $\Sigma(\theta)$. We leave any comparisons between maximising this objective and PGM to future work.

E Hyperparameters

NAPSU-MQ The hyperparameters of NAPSU-MQ are the choice of prior, choice of inference algorithm, and the parameters of that algorithm. For the toy data experiment, we used the Laplace approximation for inference, which approximates the posterior with a Gaussian centered at the maximum aposteriori estimate (MAP). We find the MAP for the Laplace approximation with the LBFGS optimisation algorithm, which we run until the loss improves by less than 10^{-5} in an iteration, up to a maximum of 500 iterations. Sometimes LBFGS failed to converge, which we detect by checking is the loss increased by over 1000 in one iteration, and fix by restarting optimisation from a different starting point. We also restarted if the maximum number of iterations was reached without convergence. For almost all runs, no restarts were needed, and at most 2 were needed.

For the Adult experiment, we used NUTS [15]. We ran 4 chains of 800 warmup samples and 2000 kept samples. We set the maximum tree depth of NUTS to 12. We normalised the posterior using the mean and covariance from the Laplace approximation. For the Laplace approximation, we used the same hyperparameters as with the toy data set, except we set the maximum number of iterations to 6000.

For the prior, we used a Gaussian distribution with mean 0 and standard deviation 10 for all components, without dependencies between components, for both experiments.

PGM and Repeated PGM PGM finds the MED_θ parameters θ that minimise the L_2-error between the expected query values and the noisy query values. The PGM implementation offers several algorithms for this optimisation problem, but we found that the default algorithm and number of iterations works well for both experiments.

RAP RAP minimises the error on the selected queries of a continuous relaxation of the discrete synthetic dataset. After the optimal relaxed synthetic dataset is found, a discrete synthetic dataset is constructed by sampling. This gives two hyperparameters that control the size of the synthetic data: the size of the continuous dataset, and the number of samples for each datapoint in the continuous relaxation. We set the size of the continuous dataset to 1000 for both experiments, as recommended by the paper [2]. For the Adult data experiment, we set the number of samples per datapoint to 46, so that the total size of the synthetic dataset is close to the size of the original dataset. For the toy data experiment, we set the number of samples per datapoint to 50. The RAP paper [2] finds that much smaller values are sufficient, but higher values should only increase accuracy.

In both cases, we weight the synthetic datapoints by $\frac{n}{n_{Syn}}$ before the downstream logistic regression to ensure that the logistic regression does not over- or underestimate variances because of a different sample size from the original data.

RAP also has two other hyperparameters that are relevant in our experiments: the number of iterations and the learning rate for the query error minimisation. After preliminary runs, we set the learning rate at 0.1 for both experiments, and set the number of iterations to 5000 for the toy data experiment, and 10000 for the Adult data experiment.

PEP PEP has two hyperparameters: the number of iterations used to find a distribution with maximum entropy that has approximately correct query values, and the allowed bound on the difference of the query values. The PEP implementation hardcodes the allowed difference to 0. We set the number of iterations to 1000 after preliminary runs for both experiments.

PrivLCM PrivLCM samples the posterior of a Bayesian latent class model, where the number of classes in limited to make inference tractable. The model has hyperparameters for the prior, and the number of latent classes. We leave the prior hyperparameters to their defaults, and set the number of latent classes to 10, which the PrivLCM authors used in a 5-dimensional binary data experiment [29]. The remaining hyperparameter of PrivLCM is the number of posterior samples that are obtained. To keep the runtime of PrivLCM manageable, we set the number of samples to 500 after ensuring that the lower number of samples did not degrade the accuracy of the estimated probabilties for the joint distribution compared to using the default of 5000 samples.

F Toy Data Experiment Details

To demonstrate the necessity of noise-awareness in synthetic data generation, we measure the coverage of confidence intervals computed from DP synthetic data on a generated toy dataset where the data generation process is known. We test the existing algorithms PGM [27], PEP [21], RAP [2], PrivLCM [29] and our pipeline NA+MI, where data generation is implemented with NAPSU-MQ. The authors of PrivLCM also propose using multiple imputation [29], so we use Rubin's rules [30] with the output of PrivLCM.

The original data consists of $n = 2000$ datapoints of 3 binary variables. The first two are sampled by independent coinflips. The third is sampled from logistic regression on the other two variables with coefficients (1, 0).

For all algorithms except PrivLCM, we use the full set of 3-way marginal queries released with the Gaussian mechanism. PrivLCM doesn't implement these, and instead uses all full sets of 2-way marginals, and a different mechanism, which is $(\epsilon, 0)$-DP [29] instead of (ϵ, δ)-DP like the other algorithms. We use the Laplace approximation for NAPSU-MQ inference, as it is much faster than NUTS and works well for this simple setting.

For the privacy bounds, we use $\delta = n^{-2}$, and vary ϵ. We generate $m = 100$ synthetic datasets of size $n_{Syn} = n$ for all algorithms except RAP, where the synthetic dataset size is a function of two hyperparameters. We describe the hyperparameters in detail in Supplemental Section E.

The downstream task is inferring the logistic regression coefficients from synthetic data. We repeated all steps 100 times to measure the probability of sampling a dataset giving a confidence interval that includes the true parameter values.

Figure 1 shows the coverages, and Figure 3a shows the widths for the resulting confidence intervals. All of the algorithms apart from ours and PrivLCM are overconfident, even with very loose privacy bounds. Examining the confidence intervals shows the reason: PGM is unbiased, but it produces too narrow confidence intervals, while NAPSU-MQ produces wider confidence intervals. On the other hand, for $\epsilon > 0.25$, PrivLCM produces much wider and too conservative confidence intervals.

G Adult Experiment Details

We include the columns Age, Workclass, Education, Marital Status, Race, Gender, Capital gain, Capital loss, Hours per week and Income of the Adult dataset, and discard the rest to remove redundant columns and keep computation times manageable. We discretise Age and Hours per week to 5 buckets, and discretise Capital gain and Capital loss to binary values indicating whether their value is positive. The Income column is binary from the start, and indicates whether a person has an income $> \$50\,000$.

In the downstream logistic regression, we use income as the dependent variable, and Age, Race and Gender as independent variables. Age is transformed back to a continuous value for the logistic regression by picking the middle value of each discretisation bucket. We did not use all variables for the downstream task, as a smaller set of variables allows including the relevant marginals for synthetic data generation.

For the input queries, we include the 2-way marginals with Hours per week and each of the independent variables Age, Race and Gender and income, as well as the 2-way marginal between Race and Gender. The rest of the queries were selected with the MST algorithm [25]. For MST, we used $\epsilon = 0.5$, but we do not include this in our figures, as we focus on the synthetic data generation, not query selection. The selected queries are shown in Figure S1. The selection is very stable: in 100 repeats of query selection, these queries were selected 99 times.

We chose the number of generated synthetic datasets for NAPSU-MQ and the number of repeats for repeated PGM by comparing the results of the Adult experiment for different choices. The results are shown in Figure S4 for NAPSU-MQ and Figure S2 for repeated PGM. We chose $m = 10$ for NAPSU-MQ because it had slightly better calibration than the other values, and $m = 5$ repeats for repeated PGM because it had the best calibration overall.

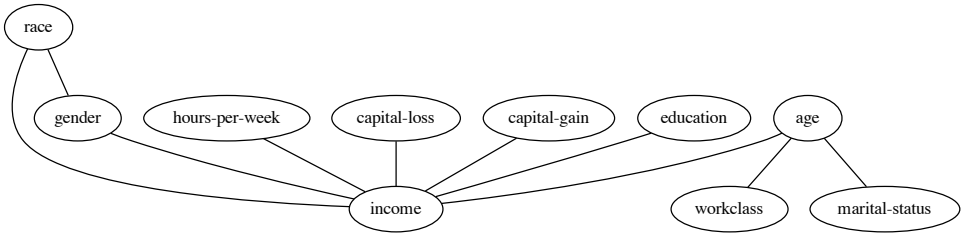


Figure S1: Markov network of selected queries for the Adult experiment. Each edge in the graph represents a selected 2-way marginal.

H Extra Results

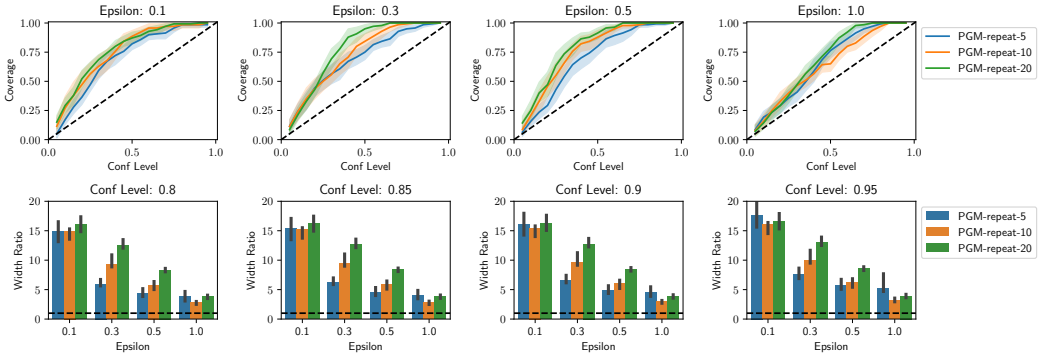


Figure S2: Comparison of different numbers of repetitions for repeated PGM. We chose $m = 5$ repeats to represent repeated PGM in the main experiment, although the differences between the numbers of repeats are small.

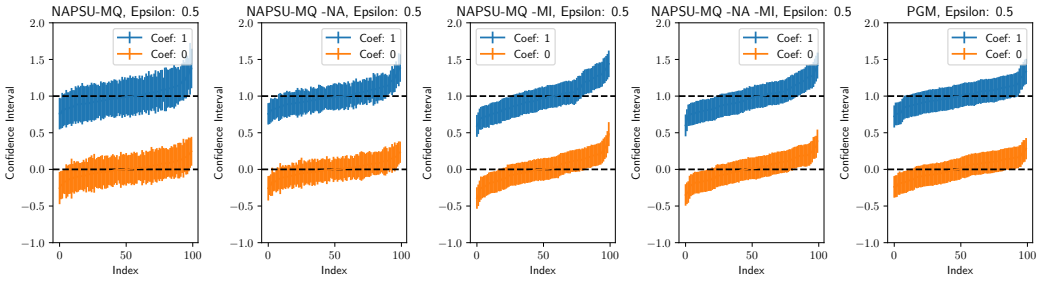


Figure S3: Ablation Study Confidence Intervals

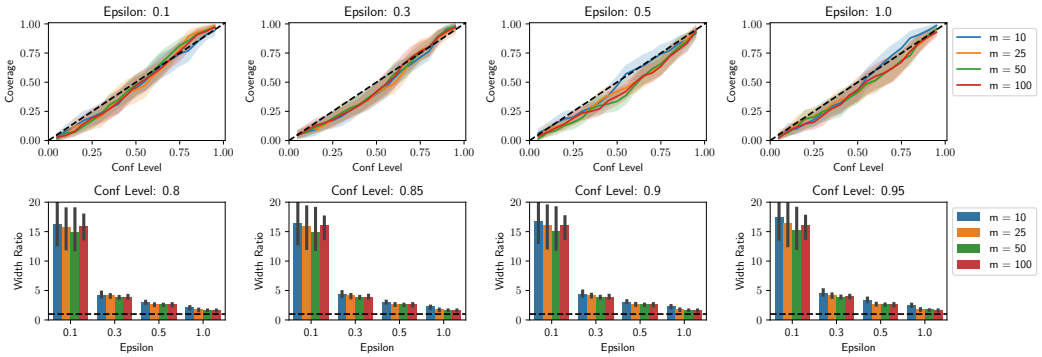
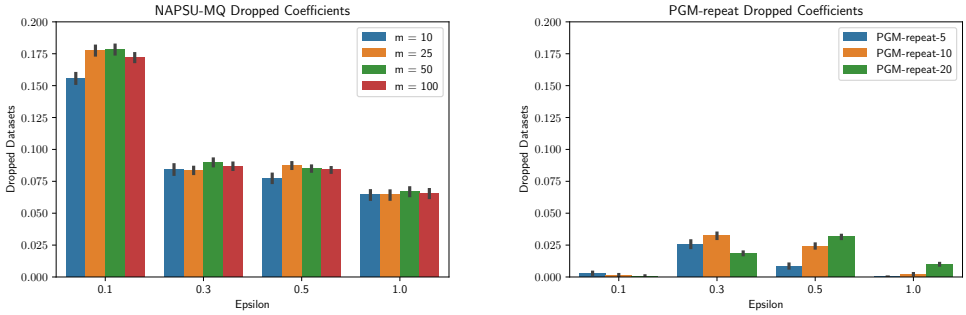


Figure S4: Comparison of different numbers of generated synthetic datasets for NAPSU-MQ. The differences are small, but $m = 10$ synthetic datasets has the best calibration, so we chose it for the main experiment.



(a) (b)

Figure S5: The fraction of coefficients dropped before Rubin's rules because of very high estimated variances from the downstream logistic regression in the Adult data experiment for NAPSU-MQ in (a) and PGM-repeat in (b).

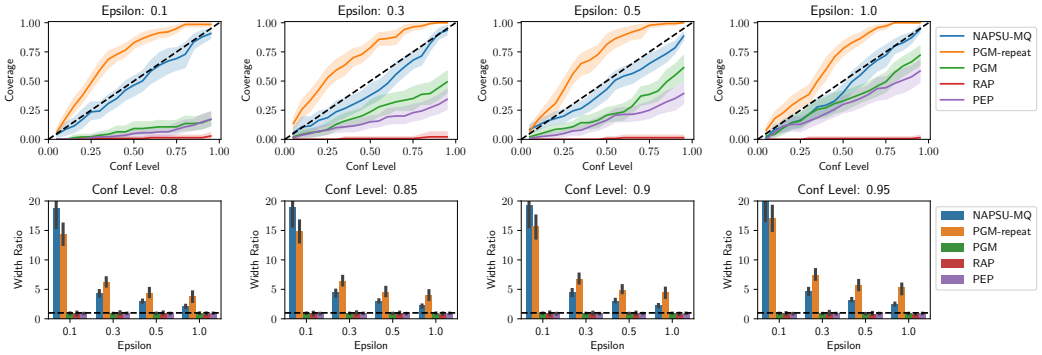


Figure S6: Results from the Adult data experiment with regularised logistic regression instead of removing very high variance estimates. The results are almost identical to Figure 4, except for RAP, which suffers from the regularisation. The regularisation is l_2 with a very small multiplier of 0.00001. Variances for the downstream task are estimated with bootstrapping using 50 bootstrap samples.

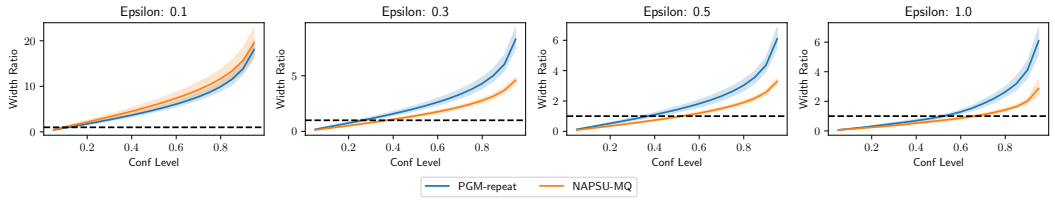


Figure S7: The tradeoff between the confidence level for DP confidence intervals and the width of the intervals. The width ratio on the y-axis is with regards to the original 95% confidence interval, for all confidence levels, so the plot shows how much must the confidence level drop to obtain the same width from a DP confidence interval as a non-DP one. The horizontal line at $y = 1$ shows this point. For $\epsilon = 1$, the confidence level for NAPSU-MQ must be dropped to about 65%, and for PGM-repeat, it must be dropped to about 50%.

Table S1: Runtimes of each inference run for the Adult experiment. Does not include the time taken to generate synthetic data, or run any downstream analysis. The LA rows record the runtime for obtaining the Laplace approximation for NAPSU-MQ that is used in the NUTS inference, so the total runtime for a NAPSU-MQ run with NUTS is the sum of the LA and NUTS rows. Experiments were run on 4 CPU cores of a cluster.

Algorithm	Epsilon	Mean	Standard Deviation
LA	0.1	2 min 53 s	18.5 s
	0.3	3 min 53 s	29.4 s
	0.5	3 min 38 s	35.0 s
	1.0	3 min 25 s	25.5 s
NUTS	0.1	9 h 59 min 6 s	6506 s
	0.3	7 h 33 min 28 s	2701 s
	0.5	4 h 57 min 40 s	3185 s
	1.0	3 h 51 min 34 s	1274 s
PEP	0.1	6 min 50 s	25.4 s
	0.3	7 min 18 s	31.2 s
	0.5	7 min 0 s	33.1 s
	1.0	7 min 7 s	33.7 s
PGM	0.1	15 s	0.5 s
	0.3	17 s	1.5 s
	0.5	15 s	0.4 s
	1.0	15 s	0.6 s
PGM-repeat-10	0.1	2 min 35 s	3.3 s
	0.3	2 min 53 s	13.0 s
	0.5	2 min 37 s	5.0 s
	1.0	2 min 36 s	4.4 s
PGM-repeat-20	0.1	5 min 15 s	10.9 s
	0.3	5 min 58 s	28.4 s
	0.5	5 min 10 s	10.2 s
	1.0	5 min 13 s	12.6 s
PGM-repeat-5	0.1	1 min 17 s	2.7 s
	0.3	1 min 28 s	6.7 s
	0.5	1 min 18 s	2.6 s
	1.0	1 min 18 s	1.9 s
RAP	0.1	32 s	2.4 s
	0.3	34 s	2.2 s
	0.5	32 s	2.1 s
	1.0	31 s	2.1 s

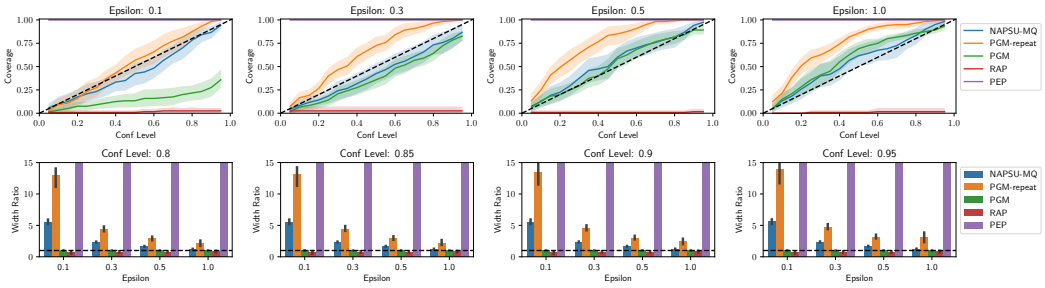


Figure S8: Results for the US Census experiment, showing that only NAPSU-MQ and PGM-repeat are calibrated for both values of ϵ, and NAPSU-MQ produces significantly narrower confidence intervals than PGM-repeat. Like Figure 4, the top row shows the mean coverage over all coefficients and 20 runs for different confidence levels. The bottom row shows median confidence interval widths divided by real data confidence interval widths.

I US Census Data Experiment

We conducted an additional experiment on US Census data provided by the UCI repository [28]. We limited the data to individuals who have served in the US Military, and picked 9 columns[2], most relating to military service. Even this subset of the data is large, with $n = 320\,754$. All columns are discrete, and have $10\,800$ possible values, much fewer than the Adult experiment.

As the downstream task, we use logistic regression with iPoverty as the dependent variable and iSex, iKorean, iVietnam and iMilitary as the independent variables. iPoverty has three categories, so we combine the two categories denoting people below the powerty line to make the dependent variable binary for the logistic regression, but not synthetic data generation.

As our queries we use 4 three-way marginals covering the independent and dependent variables, and 3 two-way marginals that include the other variables that are synthesised, but not included in the regression. As the published implementation of RAP [2] does not support a mix of two- and three-way marginals, we replace the two-way marginals with three-way marginals for RAP. As in the Adult experiment, we set $\delta = n^{-2}$, and vary ϵ.

As in the adult experiment, we use $n_{Syn} = n$ for all algorithms except RAP. For PGM-repeat and NAPSU-MQ, we choose m with a preliminary experiment. For NAPSU-MQ, we set $m = 100$, although the differences between the choices are not large. For PGM-repeat, we set $m = 10$. We set the other hyperparameters for all algorithms after testing runs to the same values used in the Adult experiment, except we increased the number of optimisation iterations for PGM to 5000 from the default of 1000, and dropped the number of NUTS chains to 2 and the number of warmup samples to 400 for NAPSU-MQ. We did not use the trick of dropping estimates with very high variances, or using very small regularisation in the logistic regression with the US Census data.

The results are shown in Figure S8. While PGM is calibrated with $\epsilon = 1$, it is severely overconfident with $\epsilon = 0.1$. This is likely caused by the large size of the dataset: at $\epsilon = 1$, there is little noise compared to the large sample size, while at $\epsilon = 0.1$, the noise has a clear effect.

NAPSU-MQ and PGM-repeat are able to produce calibrated results at $\epsilon = 0.1$. Of these, NAPSU-MQ produces clearly narrower confidence intervals for both values of ϵ.

For some reason, PEP fails completely with this dataset. We are not sure what causes this, as the algorithm should work in this setting as well as it did with the Adult dataset, and the size of the dataset should not be an issue.

The runtimes for each algorithm are shown in Table S2. The difference between PGM-repeat and NAPSU-MQ is much smaller than in the Adult data experiment, but is still large. Note that the runtimes are not comparable the Adult experiment runtimes in Table S1, as we could not use the cluster that was used in the Adult experiment due to technical issues.

[2] The columns are dYrsserv, iSex, iVietnam, iKorean, iMilitary, dPoverty, iMobillim, iEnglish and iMarital.

Table S2: Runtimes of each inference run for the US Census experiment. Does not include the time taken to generate synthetic data, or run any downstream analysis. The LA rows record the runtime for obtaining the Laplace approximation for NAPSU-MQ that is used in the NUTS inference, so the total runtime for a NAPSU-MQ run with NUTS is the sum of the LA and NUTS rows. The experiment was run on a M1 Macbook Air, because we could not use the cluster that was used for the Adult experiment due to technical issues. This means that the runtimes are not comparable to the Adult data experiment runtimes.

Algorithm	Epsilon	Mean	Standard Deviation
LA	0.1	42 s	11.3 s
	1.0	52 s	24.6 s
NUTS	0.1	49 min 3 s	667 s
	1.0	21 min 25 s	255 s
PEP	0.1	11 s	2.6 s
	1.0	12 s	2.4 s
PGM	0.1	28 s	5.6 s
	1.0	29 s	4.9 s
PGM-repeat-10	0.1	4 min 53 s	52.6 s
	1.0	4 min 57 s	54.6 s
PGM-repeat-5	0.1	2 min 12 s	17.9 s
	1.0	2 min 27 s	34.8 s
RAP	0.1	21 s	4.7 s
	1.0	19 s	2.9 s