PFunk-H: Approximate Query Processing using Perceptual Models

ABSTRACT

With the vast amount of data available for analyses these days, in order to speeden ad-hoc analysis and experimental evaluation, analysts frequently resort to using samples of the original dataset to approximate query answers. Numerous approximate query processing frameworks exist to solve this problem. But none of these frameworks make use of human perceptual accuracy in calculating the approximate answers. As a result, many approximation frameworks use more computing resources than needed because the approximation process meant for visualization is oblivious to human perception. For any query answers meant for display on a front-end visualization system, we can bound the error of our approximation by utilizing perceptual functions that model human perceptual error. Perceptual functions can be used to obtain high-confidence lower bounds for human perceptual error on data visualizations.

In this paper, we explore a preliminary model of sampling-based approximate query processing that uses perceptual functions to construct approximate answers intended for visualization. We present initial results that show that (with a very high probability) the approximate answers differ from the true answers by a *perceptually indiscernible* amount, determined by the perceptual function.

1. INTRODUCTION

Sampling-based approximate query processing (AQP) systems have been designed to make ad-hoc querying of large datasets more fluid. With the vast amounts of data available these days, analysts can use subsets of the dataset to quickly get approximate query answers. Offline sampling-based AQP systems like BlinkDB [2] have been deployed at large companies (such as Facebook Inc. and Conviva Inc.) to help data analysts tradeoff result accuracy for a more immediate response time and space.

Sometimes, the result of queries fed to AQP systems are meant to be visualized. Using approximation in these scenarios make the resulting visualizations more interactive. In these cases, the approximation could be appropriate because human beings are limited by perceptual accuracy anyways. The field of graphical perception explores how humans decode information on graphs. Often times, humans cannot very accurately perceive the value of visualizations [10, 9]. Therefore, if we had knowledge of human perceptual error, we could this information to bound the error of our approximation. In this paper, we present a preliminary method that can use knowledge of human perceptual error (encoded in perceptual functions) to provide approximate answers that differ from the true answers by a perceptually indiscernible answer.

Every form of approximation can be defined in terms of an error or confidence bound to show how good the approximation is. That is, for any given approximate answer, there is a corresponding interval/range for which we can be confident that the true answer lies in. The smaller the confidence interval for our approximate answer, the better our approximation (or the closer we are to our true answer). The bigger our confidence interval, the worse our approximation is. Often times, the size of the confidence interval is determined by the resource availability. For example, BlinkDB [2] can use a user-specified time limit to determine the number of samples that should be chosen, which correspondingly also determines the width of the confidence interval. Sometimes, our approximation is determined by certain visual guarantees. For example, IFocus [6], an online sampling algorithm, selects samples until – with a very high probability – the visual property of correct ordering is satisfied, whereby groups or bars in a visualization are ordered correctly.

In this paper, we present PFunk-H, an online sampling algorithm that can use perceptual functions (see section 1.1) to automatically determine the confidence interval for approximate answers. Since the width of the confidence interval determines both the efficiency and the correctness of query approximation, we would like to be able to automatically determine the width of our confidence interval such that the interval is lax enough so that we do not consume too many samples but strict enough to ensure correctness. The approximation process of PFunk-H directly depends on the perceptual function used. More conservative functions will produce stricter confidence intervals than less conservative ones.

The gains of using perceptual models during query approximation is accentuated in highly interactive visualizations. Such visualizations generate bursts of highly correlated queries whose results are ultimately perceived by humans [33].

HILDA '16 June 26, 2016, San Francisco, CA, USA
© 2016 ACM. ISBN 978-1-4503-2138-9. . . $15.00
DOI: 10.1145/1235

We acknowledge that human perceptual error varies by person, visualization, and features. Different people perceive visualizations differently. For a particular person, a certain visualization or graphic might be more discernable than every other rendering of the same dataset. For example, figures 1a and 1b are two visual renderings of the same dataset. To at least one of the authors, figure 1b is more easily and quickly discernible than figure 1a. This suggests that human perceptual error is a mixture of distributions that varies by person, visual rendering, and accompanying features of the rendered graphic. Modeling visual perceptual accuracy truly is a complex art [33]. In section 1.1, we present a simple model that is easy for PFunk-H to use.

1.1 Modeling Visual Perceptual Accuracy

We model human perceptual error as a mixture of gaussian distributions. Specifically, for a given visual rendering type and a specific value to be rendered, we model human visual perceptual error as gaussian noise with an unknown mean and variance. Similar models have been used to represent object perception [24]. In our bayesian inference model, we assume a gaussian prior for every possible rendered value. Notationally, for a rendered object or value x on a particular visualization graphic, perceptual error is a random variable

$$E_i \sim \mathcal{N}(m(x), \sigma(x)) \tag{1}$$

where $m(x), \sigma(x)$ are the unknown mean and variance of the perceptual error on x.

This representation becomes unwieldy quickly since the perceptual error on every possible value is a distribution not a single point. *Perceptual functions* essentially collapse this distribution into a function, mapping x to high-confidence error lower bounds for $\mathcal{N}(m(x), \sigma(x))$. As a simple example, if $x = 100$ and $E_i \sim \mathcal{N}(3, 2)$ for a specific data visualization V, then the mean of the perceptual error on values of intensity 100 is 3 and the std is 2. Suppose we used 50 error values on $x = 100$ to calculate this sample mean and std, then we could construct a perceptual function as follows. If we wanted to make a function P_v using the 99% lower bounds for the E_i, then $P_v(100) = 2.34$, the 99-th percentile lower bound for $\mathcal{N}(3, 2)$. Figure 2 shows three perceptual functions fit to cubic functions. Cubic models were used (as opposed to linear or quadratic models, for example) because in this particular case, cubic models gave the minimal sum of squared residuals after fitting. Several functions can be tested and the model with the best fit can be utilizaed.

We have described a univariate perceptual function $P : \mathbb{R} \to \mathbb{R}$ that maps a visually encoded value to the perceptual error. The second formulation is a bivarite perceptual function of the form $P : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ that maps a pair of encoded values to the error in the perceived proportional differences. A simple linear univariate perceptual function is $P(v) = 5 \cdot 10^{-5} v$. $P(105) = 5$ could be interpreted as: a user can perceive a 105 encoded value within a margin of error of ± 5.

In this paper, we introduce a sampling algorithm, PFunk-H that can use perceptual functions to provide approximate answers with formal visual guarantees. We also present preliminary performance experiments in section 4 that show how the sampling and runtime performance of different perceptual functions used by PFunk-H. As expected, very conservative perceptual functions sample more and have a higher runtime than less conservative ones.

2. THE ALGORITHM AND ITS ANALYSIS

In this section, we present PFunk-H. This algorithm supports approximate answers for `SUM` and `COUNT` based aggregation functions and we present the version for the `AVG` aggregate function. For ease of description, the text assumes that each record is a single numerical value, however our implementation supports `SELECT-PROJECT-GROUPBY` queries over full records.

2.1 The PFunk-H Algorithm

PFunk-H uses a single perceptual function to return an approximate average of the dataset $\mathbf{X}(= \{x_1, \ldots, x_N\})$. Suppose μ is the population mean. At the end of the algorithm, we want to estimate with high probability a sample mean v such that

$$v \in [\mu - P^{(u)}(\mu), \mu + P^{(u)}(\mu)] \tag{2}$$

Equation 7 is a guarantee that the margin of perceptual error is $\leq P^{(u)}(\mu)$ with probability $\geq (1 - \delta)$. In other words, with a high probability, the final approximation error is not perceptually discernible by humans, as defined by the perceptual function $P^{(u)}$. Table 2 describes the symbols used in Algorithm 2.

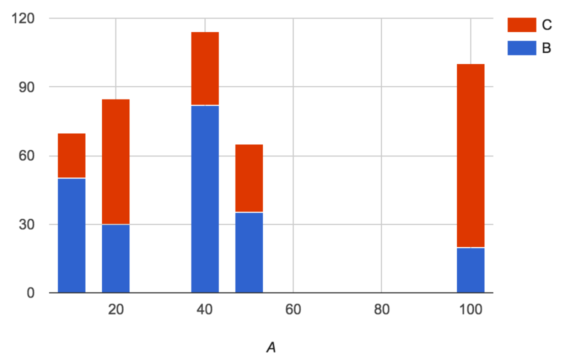
Algorithm 1: Basic Univariate PFunk-H Algorithm

```
  Data: $P^{(u)}, \mathbf{X}, \delta$
1 Initialize $s = 0, v = R(\mathbf{X}), t = P^{(u)}(v)$;
2 while $t > P^{(u)}(v - t)$ and $s < N$ do
3     $s = s + 1$;
4     $t = \sqrt{\frac{\log(2/\delta)}{2s}}$;
5     $v = \frac{s-1}{s} \cdot v + \frac{1}{s} \cdot R(\mathbf{X})$;
6 return $v$;
```

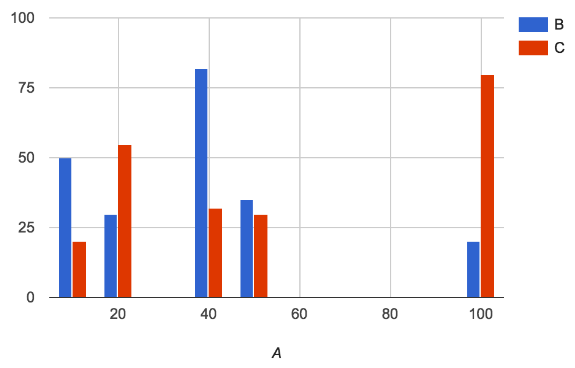
Algorithm 2 depicts the iterative algorithm in detail. s represents the total number of samples required for a margin of error of t. On each iteration, we sample without replacement one more element from the dataset using $R(\mathbf{X})$. Using Hoeffding's inequality [**?**], we can compute the decreased margin of error t. We continue this procedure until $t \leq P^{(u)}(v - t)$, when the number of samples ensures that equation 7 is satisfied. The challenge is that the perceptual function $P^{(u)}$ is conditiioned on the true value μ, and we outline the proof in Subsection 3.1.1.

EXAMPLE 1. *We now provide an example execution sequence of algorithm 2. Suppose the dataset is generated from a normal distribution $x_1, \ldots, x_N \sim \mathcal{N}(0.2, 1)$ where $N = 1$ million points, and the perceptual function is linear $P^{(u)}(x) = \frac{x}{10}$. The maximum allowable perceptual error when computing the* `AVG` *is defined as $P^{(u)}(\mu = 0.2) = 0.02$, however we do not know the value of μ and must learn it as part of the algorithm such that, by the time the algorithm terminates, the confidence interval for the sample mean v should be a sub-interval of $[\mu \pm P^{(u)}(\mu)] = [0.2 \pm 0.02]$.*

The conditional in Line 2 compares the empirical margin of error t with the output of the perceptual function over the sample mean minus the margin of error $v - t$. After taking 5867 samples we find that $v = 0.21$, and $t = 0.018$ is less than the true perceptual error 0.02 and ensures that the approximated result is not perceptually discernible.



(a) Rendering of a dataset in stacked bars chart



(b) Rendering of a dataset in multi-series column chart

Figure 1

Var	Description
$\mathbf{X}$	Dataset $0 \leq X_0, \ldots, X_n \leq 1$
$R(\mathbf{X})$	Proc to return a random without-replacement sample from $\mathbf{X}$
$P^{(u)}$	Univariate perceptual function
δ	Chance of failure of algorithm (default: 0.05)
Δ	Step size for sample set size

Table 1: Table of Notation for the PFunk-H Algorithms

Note that the goal is to only use the minimum number of samples such that the empirical margine of error is, with high probability, less than 0.02.

To extend this algorithm to records with multiple attributes, we simple need to keep track of the margins of error t_{attr} for each attribute *attr*. Furthermore, the current algorithm assumes that all values ar within $[0, 1]$ to simplify the proof, however it is simple to extend to handle values within an fixed numerical range $[a, b]$ where $a < b$.

Algorithm 2 uses Hoeffding's classical inequality [19]. Serfling [28] refined the inequality to give tighter error bounds as the number of samples chosen approaches the size of the dataset. In particular, using the Hoeffding-Serfling corollary[5], we can make the margin of error tighter by replacing line 4 of algorithm 2 with

$$t = \sqrt{\frac{\rho_s \log(2/\delta)}{2s}} \tag{3}$$

where

$$\rho_s = \left\{ \begin{array}{ll} \left(1 - \frac{s-1}{N}\right), & \text{for } s \leq N/2 \\ \left(1 - \frac{s}{N}\right)\left(1 + \frac{1}{s}\right), & \text{for } s > N/2 \end{array} \right\} \tag{4}$$

Sometimes, using the empirical variance (which can be updated in a streaming fashion) might produce even tighter bounds depending on the distribution of the underlying dataset.

2.1.1 *Proof of Correctness*

CLAIM 1. *Assuming $P^{(u)}$ is a monotonically non-decreasing function, at the end of the algorithm,* $\Pr[P^{(u)}(\mu) > t] > 1 - \delta$ *where t is as defined in the algorithm 2.*

PROOF. We can prove the claim by proving its contrapositive $\Pr[P^{(u)}(\mu) \leq t] \leq \delta$.

We use the one-sided case of Hoeffding's classical inequality. By theorem 1 in [19], $\Pr[v - \mu \geq t] \leq e^{-2st^2}$ where s is the number of samples used in calculating the sample average v. Then, equivalently, $\Pr[\mu \leq v - t] \leq e^{-2st^2}$ and using the monotonicity of $P^{(u)}$,

$$\Pr[P^{(u)}(\mu) \leq P^{(u)}(v - t)] \leq e^{-2st^2} \tag{5}$$

To get a lower bound for s, we solve $e^{-2st^2} \leq \delta$ and obtain that s must be greater than or equal to $\frac{\log(1/\delta)}{2t^2}$. At the end of algorithm 2, $s = \lceil \frac{\log(2/\delta)}{2t^2} \rceil$.

Therefore, $\Pr[P^{(u)}(\mu) \leq P^{(u)}(v-t)] \leq \delta$ which means that $\Pr[P^{(u)}(\mu) > P^{(u)}(v - t)] > 1 - \delta$. The stopping condition for the algorithm ensures that $P^{(u)}(v - t) \geq t$. As a result, $\Pr[P^{(u)}(\mu) > t] > 1 - \delta$.
$\square$

CLAIM 2. *If $s < N$, then $v \in [\mu - P^{(u)}(\mu), \mu + P^{(u)}(\mu)]$*

PROOF. In claim 3, we invoked the one-sided case of Hoeffding's inequality[19]. For the two sided case, we need $\geq \frac{\log(2/\delta)}{2t^2}$ samples so that $v \in [\mu - t, \mu + t]$. By claim 3, with probability $(1 - \delta)$, $P^{(u)}(\mu) > t$. As a result,

$$[\mu - t, \mu + t] \subseteq [\mu - P^{(u)}(\mu), \mu + P^{(u)}(\mu)] \tag{6}$$

Therefore, $v \in [\mu - P^{(u)}(\mu), \mu + P^{(u)}(\mu)]$ with probability $\geq (1 - \alpha)$. $\square$

3. THE ALGORITHM AND ITS ANALYSIS

In this section, we present PFunk-H. It uses a single univariate perceptual function for approximate answers to the `AVG` aggregate function.

We can extend this algorithm to handle other aggregate functions, such as `SUM` and `COUNT`.

3.1 The PFunk-H Algorithm

Algorithm 2 uses a single perceptual function to return an approximate average of the dataset $\mathbf{X}(= X_1, \ldots, X_N)$. Suppose μ is the population mean. At the end of the algorithm, with a very high probability

$$v \in [\mu - P^{(u)}(\mu), \mu + P^{(u)}(\mu)] \tag{7}$$

Equation 7 is a guarantee that the margin of perceptual error is $\leq P^{(u)}(\mu)$ with probability $\geq (1-\delta)$. In other words, with a very high probability, the final approximation error is not perceptually discernible by humans.

Algorithm 2: Basic PFunk-H Algorithm

```
  Data: $P^{(u)}, \mathbf{X}, \delta$
1 Initialize $s = 0, v = R(\mathbf{X}), t = P^{(u)}(v)$;
2 while $t > P^{(u)}(v - t)$ and $s < N$ do
3     $s = s + 1$;
4     $t = \sqrt{\frac{\log(2/\delta)}{2s}}$;
5     $v = \frac{s-1}{s} \cdot v + \frac{1}{s} \cdot R(\mathbf{X})$;
6 return $v$;
```

Table 2 describes the symbols used in algorithms 2. Algorithm 2, with a very high probability, returns $v \in [\mu - P^{(u)}(\mu), \mu + P^{(u)}(\mu)]$. The correctness of this algorithm is detailed in claim 3 and 4 of section 3.1.1.

The algorithm works as follows. s represents the total number of samples required for a margin of error of t. On each iteration, we sample (without replacement) one more element from our dataset using the randomized $R(\cdot)$ procedure. As a result, the margin of error t decreases. We keep sampling until $t \leq P^{(u)}(v - t)$. At this point, the number of samples chosen so far ensures that equation 7 is satisfied.

Example 2. *Here we provide an example execution sequence of algorithm 2 for illustration purposes. Suppose we have $X_1, \ldots, X_N \sim \mathcal{N}(0.2, 1)$ where $N = 1$ million points and we have a linear univariate perceptual function $P^{(u)}(x) = \frac{x}{10}$. In words, this means that the maximum allowable perceptual error for the approximate mean is 0.02. But before the algorithm executes, we are not aware of that value. At the end of algorithm 2, the confidence interval for the approximate mean should be a sub-interval of $[\mu - P^{(u)}(\mu), \mu + P^{(u)}(\mu)] = [0.2 - 0.02, 0.2 + 0.02]$.*

After some iterations, the algorithm stops, using only 5867 samples to obtain an approximate mean of 0.21 with an empirical margin of error of 0.018. We used just enough samples to ensure that the empirical margin of error is less than 0.02. Note that the number of samples chosen (without replacement) is completely dependent only on $P^{(u)}(\mu)$.

Therefore, using the PFunk-H algorithm, we obtain – with a very high probability – an approximate mean which deviates from the true mean by a perceptually indiscernible amount.

For simplicity, algorithm 2 handles the single attribute case but is easily extensible to handle multiple columns. To do so, we need to independently keep track of the margin of error t on the true mean for each column. Furthermore, algorithm 2 assumes the dataset values belong in range $[0, 1]$ for simplicity of the algorithm and subsequent proof. Section **??** shows how to extend the algorithm to handle values in range $[a, b]$ for any real $a < b$.

Algorithm 2 uses Hoeffding's classical inequality [19]. Serfling [28] refined the inequality to give tighter error bounds as the number of samples chosen approaches the size of the dataset. In particular, using the Hoeffding-Serfling corollary[5], we can make the margin of error tighter by replacing line 4 of algorithm 2 with

$$t = \sqrt{\frac{\rho_s \log(2/\delta)}{2s}} \tag{8}$$

where

$$\rho_s = \left\{ \begin{array}{ll} \left(1 - \frac{s-1}{N}\right), & \text{for } s \leq N/2 \\ \left(1 - \frac{s}{N}\right)\left(1 + \frac{1}{s}\right), & \text{for } s > N/2 \end{array} \right\} \tag{9}$$

$\mathbf{X}$	The dataset $0 \leq X_1, \ldots, X_N \leq 1$ where N is its size
$R(\mathbf{X})$	A procedure that returns a random item in $\mathbf{X}$
$P^{(u)}$	A univariate perceptual function
δ	Chance of failure of algorithm (typically set to 0.05)
$m(v)$	Returns the mean of perceptual errors for v
$\sigma(v)$	Returns the variance of perceptual errors for v
CDF(X, x)	Outputs $\Pr[X \leq x]$
Δ	Step size for sample set size

Table 2: Table of Notation for the PFunk-H Algorithms

Sometimes, using the empirical variance (which can be updated in a streaming fashion) might produce even tighter bounds depending on the distribution of the underlying dataset. In section **??**, we present an algorithm that solves our problem (7) using the Bernstein-Serfling inequality[5].

3.1.1 Proof of Correctness

Claim 3. *Assuming $P^{(u)}$ is a monotonically non-decreasing function, at the end of the algorithm, $\Pr[P^{(u)}(\mu) > t] > 1-\delta$ where t is as defined in the algorithm 2.*

Proof. We can prove the claim by proving its contrapositive $\Pr[P^{(u)}(\mu) \leq t] \leq \delta$.

We use the one-sided case of Hoeffding's classical inequality. By theorem 1 in [19], $\Pr[v - \mu \geq t] \leq e^{-2st^2}$ where s is the number of samples used in calculating the sample average v. Then, equivalently, $\Pr[\mu \leq v - t] \leq e^{-2st^2}$ and using the monotonicity of $P^{(u)}$,

$$\Pr[P^{(u)}(\mu) \leq P^{(u)}(v - t)] \leq e^{-2st^2} \tag{10}$$

To get a lower bound for s, we solve $e^{-2st^2} \leq \delta$ and obtain that s must be greater than or equal to $\frac{\log(1/\delta)}{2t^2}$. At the end of algorithm 2, $s = \lceil \frac{\log(2/\delta)}{2t^2} \rceil$.

Therefore, $\Pr[P^{(u)}(\mu) \leq P^{(u)}(v-t)] \leq \delta$ which means that $\Pr[P^{(u)}(\mu) > P^{(u)}(v - t)] > 1 - \delta$. The stopping condition for the algorithm ensures that $P^{(u)}(v - t) \geq t$. As a result, $\Pr[P^{(u)}(\mu) > t] > 1 - \delta$.
□

Claim 4. *If $s < N$, then $v \in [\mu - P^{(u)}(\mu), \mu + P^{(u)}(\mu)]$*

Proof. In claim 3, we invoked the one-sided case of Hoeffding's inequality[19]. For the two sided case, we need $\geq \frac{\log(2/\delta)}{2t^2}$ samples so that $v \in [\mu - t, \mu + t]$. By claim 3, with probability $(1 - \delta)$, $P^{(u)}(\mu) > t$. As a result,

$$[\mu - t, \mu + t] \subseteq [\mu - P^{(u)}(\mu), \mu + P^{(u)}(\mu)] \tag{11}$$

Therefore, $v \in [\mu - P^{(u)}(\mu), \mu + P^{(u)}(\mu)]$ with probability $\geq (1 - \alpha)$. $\square$

4. EXPERIMENTS

The goals of our preliminary experiments are simple: to compare the sampling complexity of PFunk-H when different perceptual functions are used; and to compare runtime of PFunk-H to the traditional non-approximate table scan.

In the initial performance experiments we ran, we compared the performance of three perceptual functions used in PFunk-H. As control, we also compare the perceptual functions to SCAN, a table scan operation that a traditional non-AQP system uses by default. As expected, our results show that SCAN selects the most number of samples, followed by the strictest perceptual function (P3). The laxest perceptual function (P1) uses the least number of samples.

4.1 Setup

We ran all experiments on a 8-core Intel(R) Xeon(R) CPU E5-2680 2.80GHz server running Ubuntu 12.04.3 LTS. However, all experiments were single-threaded to avoid speedup from parallelization.

PFunk-H and SCAN are evaluated on top of NEEDLETAIL, a database system designed to sample records matching a set of ad-hoc query predicate conditions [25]. NEEDLETAIL uses in-memory bitmap-based indexes to quickly retrieve satisfying tuples.

For our experiments, we generated synthetic datasets with 10 million records, each containing 20 continuous fields and 2 discrete fields. Each continuous field is drawn from a (truncated) normal distribution with mean $\mu \in [0.1, 0.8]$ and std $\sigma \in [0.1, 0.5]$. Our preliminary experiments focused on the continuous fields.

4.2 Learning some Perceptual Functions

Our initial experiments are aimed at verifying that the margins of error produced by laxer perceptual functions is wider than the margins of error produced by more conservative perceptual functions. Figure 2 shows three perceptual functions all fit to cubic functions. We fit the perceptual functions to cubic models (as opposed to linear or quadratic models, for example) because in this particular case, cubic models gave the minimal sum of squared residuals after fitting. Data used for fitting the perceptual functions was obtained from experiments used to replicate prior static perceptual studies done by Cleveland & McGill [10, 9]. In total, 7725 user estimates were obtained via Amazon Mechanical Turk. Five different chart types were used in the experiments: aligned stacked, unaligned stacked, separated, divided, and adjacent bar charts. Of the five, we show the fitted univariate perceptual functions for three of the chart types: aligned stacked, unaligned stacked, and divided bar charts.

For each experiment, we calculate the absolute difference between the true height of bars and the user estimated height. We assume that the absolute errors for each true bar height is normally distributed and use high-confidence error lower bounds (99%) to fit the univariate perceptual functions. All true bar heights were normalized to be in the interval [0, 1].

4.3 Comparing the Perceptual Functions

Now we compare the performance of the three perceptual functions P1, P2, P3 with one another and also with SCAN.

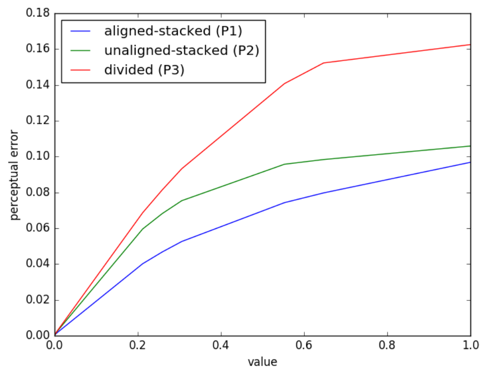


Figure 2: Three perceptual functions fit to cubic functions

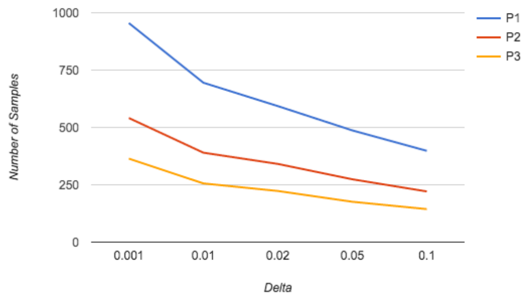


Figure 3: δ vs. Number of Samples

Figure 3 shows how the number of samples chosen by PFunk-H varies with the failure probability of the algorithm. The dataset is drawn from a truncnorm distribution with mean 0.45 and std 0.48. Recall that the δ parameter in algorithm 2 is used to specify the chance of failure of the sampling algorithm. Since the failure probability δ is directly related to the sampling complexity (see algorithm 2), we can expect δ to correlate with the percentage of the dataset sampled. Indeed, it is. Figure 3 shows that (for all three perceptual functions), as the failure probability increases, the number of samples chosen decreases. This makes sense since $\delta \to 0$ would imply that the entire dataset should be used to calculate a non-approximate answer and $\delta \to 1/2$ would mean that our answer would be an incorrect approximation at least half the time.

Figure 3 also shows the relative sampling complexity of the three perceptual functions. As expected, for a constant δ, P1 samples more than P2, P3 and P3 samples less than P1, P2. This result follows the relative ordering of the functions shown in figure 2.

Further, figure 4 shows the runtimes (in seconds) of PFunk-H when used with the three different perceptual functions P1, P2, and P3. Again, PFunk-H runs slowest when using P1 (the most conservative perceptual function) and fastest when used with P3 (the laxest perceptual function).

5. RELATED WORK

The work related to PFunk-H can be placed in a few categories:

Graphical Perception: One of the first studies to for-

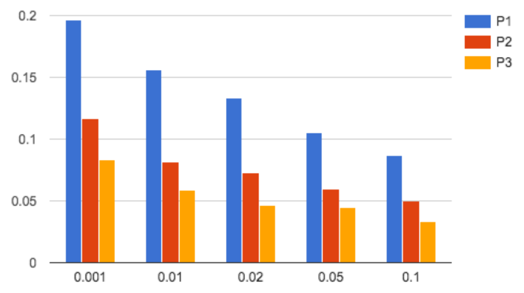


Figure 4: δ vs Total Time (in seconds)

malize visual perceptual accuracy was done by Cleveland & McGill [10]. The field of graphical perception is mainly focused on the visual decoding of information encoded on graphs. Cleveland & McGill ran experiments to quantify the accuracy of pairwise comparison of static objects in data visualizations. The goals of the experiments were two-fold: identify a set of elementary perceptual tasks implicitly performed by humans when extracting quantitative information from graphical models; and – most related to this paper – to order the tasks on the basis of how accurately people perform them [10, 9]. This ordering is based on experiments and on theory in psycophysics. The power law of theoretical psycophysics, which has been shown to be realistic [4], relates the perceived magnitude of a stimuli to the actual magnitude[30]. Therefore, much ground work has been done toward formalizing the theory and experimentation of human visual perceptual accuracy [30, 4, 14, 24].

A range of experiments have been performed to further validate Cleveland & McGill's foundational work on graphical perception. Talbot et al. [29] extend the original experiment to study the effects of distractor bars and separation between non-adjacent bar charts. Heer et al. [17] approximately replicate the original experiment using crowdsourcing and a larger collection of chart types. Zacks et al. [34] evaluates the same perceptual task but on 3D charts.

A concept similar to perceptual functions has been introduced by Demiralp et al.[12]. They formulate perceptual kernels, a distance metric of aggregate pairwise perceptual distances used to measure how color, shape, and size affect graphical perception. But it is not immediately obvious (based on its formal definition) how to use perceptual kernels in sampling-based AQP systems. On the other hand, our initial definition of perceptual functions aim to provide error metrics that can be used in a sampling-based algorithm like PFunk-H. The definition of perceptual functions makes it especially conducive for online algorithms.

Last, weber models (both linear and log-linear) have been formulated to provide concise means to model the human perception of differences in the correlation of objects in graphical models [16, 23].

Approximate Query Processing: There are two broad categories of algorithms for approximate query processing: online and offline. We consider related work for both categories.

Online AQP is most closely related to our work presented in this paper because PFunk-H is an online algorithm. The main idea behind online aggregation [18] is the interactive and on-the-fly construction and refinement of confidence intervals for estimates of aggregate functions (AVG, SUM, and so on). Online aggregation is particularly appealing for the interactive query processing functionality it provides. Users are provided with an interactive tool that can be used to stop processing aggregate groups when the users deem fit. Thus, the onus rests on the user to the approximate processing. This interface, although very intuitive for users that have knowledge of statistics, could be daunting for lay users with no background in math. On the other hand, PFunk-H stops processing automatically when the query answers intended for visualization are perceptually indiscernible from the true answers. In this manner, our answers are computed approximately, yet satisfy formal visual guarantees. IFocus [6] is another online sampling algorithm which provides a formal *probabilistic* guarantee different from what PFunk-H gives: the visual property of correct ordering, whereby groups or bars in a visualization are ordered correctly.

As per offline AQP, Garofalakis et al. [13] surveys the area. Examples of systems that support offline AQP inlude BlinkDB [2] and Aqua [1]. In an online AQP system, samples are typically chosen a-priori (i.e., via Neyman Allocation [11]) and are tailored to a workload or set of predictable queries [3, 7, 8, 22].

Scalable and Interactive Visualization Tools: A slew of interactive visualization tools and frameworks have been introduced by both the database and visualization communities. Examples include Tableau, SeeDB, and SnapToQuery [15, 31, 21].

Online perceptually-aware approximate query processing algorithms like PFunk-H can, in theory, be incorporated into any of such systems so that the visualizations produced can be approximate to enhance interactivity and scalability yet guarantee certain perceptual properties.

Statistical Estimators: Statistical sampling theory is a well-established field with many formulas and theorems that database researchers have leveraged to make sampling-based AQP systems [26, 32, 27].

The core of PFunk-H is based on the Hoeffding-Serfling concentration inequality which works well in practice when the variance of the population is unknown. The concentration inequality provides confidence intervals that makes no distributional assumptions [20]. The estimated confidence intervals could be tighter if the variance were known or even approximated. The Hoeffding-Bernstein concentration inequality uses the empirical variance of a subset of the dataset and produces tighter confidence intervals than Hoeffding-Serfling for certain probability distributions [5].

In future iterations of PFunk-H, we hope to incorporate information from other estimators (such as empirical variance) during online processing.

6. CONCLUSION & FUTURE DIRECTIONS

In this paper, we introduced a method of building and testing simple perceptual models for sampling-based approximate query processing. Specifically, we used the visual property of perceptual indiscernibility of objects to train monotonically increasing models (log-linear and linear). We presented an algorithm that can use a class of perceptual models (i.e. perceptual functions) to obtain approximate answers with formal guarantees on the confidence interval for the approximation. PFunk-H uses a univariate perceptual function to ensure that the approximate answer differs from the true answer by a perceptually indiscernible amount.

By using perceptual models in approximating query answers intended for visualization, we ensure that our process of approximation is not oblivious to its intended use case and ecological validity. In the near future, we hope to provide a method of modifying any existing AQP system to use visual perceptual models during the process of online query approximation.

References

[1] S. Acharya, P. B. Gibbons, and V. Poosala. Aqua: A Fast Decision Support Systems Using Approximate Query Answers. *International Conference on Very Large Databases (VLDB)*, pages 754–757, 1999.

[2] S. Agarwal, B. Mozafari, A. Panda, H. Milner, S. Madden, and I. Stoica. BlinkDB: queries with bounded errors and bounded response times on very large data. In *EuroSys '13*, 2013.

[3] B. Babcock, S. Chaudhuri, and G. Das. Dynamic sample selection for approximate query processing. In *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data*, SIGMOD '03, pages 539–550, New York, NY, USA, 2003. ACM.

[4] J. C. Baird. *Psychophysical analysis of visual space*. Pergamon Press Oxford, New York, [1st ed.] edition, 1970.

[5] R. Bardenet and O.-A. Maillard. Concentration inequalities for sampling without replacement. *Bernoulli*, 21(3):1361–1385, 08 2015.

[6] E. Blais, A. Kim, P. Indyk, and S. Madden. Rapid Sampling for Visualizations with Ordering Guarantees. In *41st International Conference on Very Large Data Bases*, volume 8, pages 521–532, 2015.

[7] S. Chaudhuri, G. Das, M. Datar, R. Motwani, and V. Narasayya. Overcoming limitations of sampling for aggregation queries. In *Data Engineering, 2001. Proceedings. 17th International Conference on*, pages 534–542, 2001.

[8] S. Chaudhuri, G. Das, and V. Narasayya. Optimized stratified sampling for approximate query processing. *ACM Trans. Database Syst.*, 32(2), June 2007.

[9] W. Cleveland and R. McGill. Graphical Perception and Graphical Methods for Analyzing Scientific Data. *Science*, 229(4716):828, 1985.

[10] W. S. Cleveland and R. McGill. Graphical perception: Theory, experimentation, and application to the development of graphical methods. *Journal of the American Statistical Association*, 79(387):pp. 531–554, 1984.

[11] W. Cochran. *Sampling Techniques.* Wiley, third edition, 1977.

[12] Ç. Demiralp, M. S. Bernstein, and J. Heer. Learning perceptual kernels for visualization design. *IEEE Trans. Vis. Comput. Graph.*, 20(12):1933–1942, 2014.

[13] M. N. Garofalakis and P. B. Gibbon. Approximate query processing: Taming the terabytes. In *Proceedings of the 27th International Conference on Very Large Data Bases*, VLDB '01, pages 725–, 2001.

[14] M. Gleicher, M. Correll, C. Nothelfer, and S. Franconeri. Perception of average value in multiclass scatterplots. *IEEE Trans. Vis. Comput. Graph.*, 19(12):2316–2325, 2013.

[15] P. Hanrahan. Analytic database technologies for a new kind of user: The data enthusiast. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, SIGMOD '12, pages 577–578, New York, NY, USA, 2012. ACM.

[16] L. Harrison, F. Yang, S. Franconeri, and R. Chang. Ranking visualizations of correlation using weber's law. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):1943–1952, Dec 2014.

[17] J. Heer and M. Bostock. Crowdsourcing graphical perception: using mechanical turk to assess visualization design. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 203–212. ACM, 2010.

[18] J. M. Hellerstein, P. J. Haas, and H. J. Wang. Online aggregation. *Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data*, pages 171–182, 1997.

[19] W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963.

[20] W.-C. Hou, G. Ozsoyoglu, and B. Taneja. Statistical Estimators for Relational Algebra Expressions. In *Proceedings of the 7th ACM SIGACT-SIGMOD-SIGART Symposium on PODS*, pages 276–287, 1988.

[21] L. Jiang and A. Nandi. SnapToQuery : Providing Interactive Feedback during Exploratory Query Specification. 8(11):1250–1261, 2015.

[22] S. Joshi and C. M. Jermaine. Robust stratified sampling plans for low selectivity queries. In *Proceedings of the 24th International Conference on Data Engineering, ICDE 2008, April 7-12, 2008, Cancún, México*, pages 199–208, 2008.

[23] M. Kay and J. Heer. Beyond weber's law: A second look at ranking visualizations of correlation. *IEEE Trans. Visualization & Comp. Graphics (Proc. InfoVis)*, 2016.

[24] D. Kersten, P. Mamassian, and A. Yuille. Object Perception as Bayesian Inference. *Department of Statistics, UCLA*, pages 1–34, 2011.

[25] A. Kim, E. Blais, A. Parameswaran, P. Indyk, S. Madden, and R. Rubinfeld. Rapid sampling for visualizations with ordering guarantees. *Proc. VLDB Endow.*, 8(5):521–532, Jan. 2015.

[26] A. Kleiner, A. Talwalkar, and S. Agarwal. A general bootstrap performance diagnostic. *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 419–427, 2013.

[27] F. Olken and D. Rotem. Simple random sampling from relational databases. In *Proceedings of the 12th International Conference on Very Large Data Bases*, VLDB '86, pages 160–169, San Francisco, CA, USA, 1986. Morgan Kaufmann Publishers Inc.

[28] R. J. Serfling. Probability inequalities for the sum in sampling without replacement. *Ann. Statist.*, 2(1):39–48, 01 1974.

[29] J. Talbot, V. Setlur, and A. Anand. Four Experiments on the Perception of Bar Charts. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):2152–2160, Nov. 2014.

[30] R. Teghtsoonian. *The American Journal of Psychology*, 88(4):677–684, 1975.

[31] M. Vartak, S. Madden, A. Parameswaran, and N. Polyzotis. Seedb: Automatically generating query visualizations. *Proc. VLDB Endow.*, 7(13):1581–1584, Aug. 2014.

[32] J. S. Vitter. Random sampling with a reservoir. *ACM Trans. Math. Softw.*, 11(1):37–57, Mar. 1985.

[33] E. Wu and A. Nandi. Towards Perception-aware Interactive Data Visualization Systems. 2015.

[34] J. Zacks, E. Levy, B. Tversky, and D. J. Schiano. Reading bar graphs: Effects of extraneous depth cues and graphical context. *Journal of Experimental Psychology: Applied*, 4(2):119 – 138, 1998.