PAC Mode Estimation using PPR Martingale Confidence Sequences

NeurIPS 2021 Submission Number 8167

Abstract

We consider the problem of correctly identifying the *mode* of a discrete distribution $\mathcal{P}$ with sufficiently high probability by observing a sequence of i.i.d. samples drawn according to $\mathcal{P}$. This problem reduces to the estimation of a single parameter when $\mathcal{P}$ has a support set of size $K = 2$. Noting the efficiency of prior-posterior-ratio (PPR) martingale confidence sequences [1] for handling this special case, we propose a generalisation to mode estimation, in which $\mathcal{P}$ may take $K \geq 2$ values. We observe that the "one-versus-one" principle yields a more efficient generalisation than the "one-versus-rest" alternative. Our resulting stopping rule, denoted PPR-ME, is optimal in its sample complexity up to a logarithmic factor. Moreover, PPR-ME empirically outperforms several other competing approaches for mode estimation. We demonstrate the gains offered by PPR-ME in two practical applications: (1) sample-based forecasting of the winner in indirect election systems, and (2) efficient verification of smart contracts in permissionless blockchains.

1 Introduction

We investigate the problem of estimating the mode of a given, arbitrary discrete probability distribution $\mathcal{P} = (p, v, K)$ by observing a sequence of i.i.d. samples drawn according to $\mathcal{P}$. Here $\mathcal{P}$ takes values from the support set $\{v_1, v_2, \ldots, v_K\}$ for some $K \geq 2$; for $1 \leq i \leq K$, the probability of obtaining v_i from $\mathcal{P}$ is p_i. We assume that $\mathcal{P}$ has a unique mode, and without loss of generality, $p_1 > p_2 \geq p_3 \geq p_4 \geq \cdots \geq p_K$ (which makes v_1 the mode).

Our aim is to provide a procedure $\mathcal{L}$ to identify the mode of $\mathcal{P}$. At each step $t \geq 1$, $\mathcal{L}$ can either ask for a sample $x^t \sim \mathcal{P}$ or it can terminate and declare its answer. For "mistake probability" $\delta \in (0, 1)$, L is said to be δ-correct if for every qualifying discrete distribution $\mathcal{P}$, L terminates with probability 1 and correctly identifies the mode of $\mathcal{P}$ with probability at least $1 - \delta$. If $\mathcal{L}$ terminates after observing the sequence of samples $x^1, x^2, \ldots, x^T$ for some $T \geq 1$, we may assume that its answer is the most frequent value of $\mathcal{P}$ in this sequence, since it can be argued that no other choice can decrease the mistake probability across all problem instances. Hence, it is convenient to view $\mathcal{L}$ simply as a *stopping rule*, which only needs to decide when to terminate. We aim to devise a δ-correct stopping rule $\mathcal{L}$ with low *sample complexity*—informally the number of samples T observed before stopping.

In order to make our problem "properly" PAC, we could introduce a tolerance parameter ϵ, with the implication that any returned value with associated probability at least $p_1 - \epsilon$ will be treated as correct. We omit this generalisation, noting that it can be handled quite easily by the methods proposed in the paper. In fact, our version with $\epsilon = 0$ exactly matches the problem defined by Shah et al. [2], whose algorithms and analysis are our primary baseline. Shah et al. [2] effectively "settle" the PAC mode estimation problem from a theoretical perspective. First they show the following lower bound.

Theorem 1 (Lower bound [2]). *Fix $\delta \in (0, 1)$, $K \geq 2$, and a δ-correct stopping rule $\mathcal{L}$. On each problem instance $\mathcal{P} = (p, v, K)$, the expected number of samples observed by $\mathcal{L}$ is at least*

$$\frac{p_1}{(p_1 - p_2)^2} \ln\left(\frac{1}{2.4\delta}\right).$$

They also give a stopping rule, $\mathcal{A}_1$, whose sample complexity is within a logarithmic factor.

Theorem 2 ($\mathcal{A}_1$ upper bound [2])**.** *Fix $\delta \in (0,1)$, $K \geq 2$, and problem instance $\mathcal{P} = (p, v, K)$. When $\mathcal{A}_1$ is run on $\mathcal{P}$, with probability $1-\delta$, the number of samples it observes is at most*

$$\frac{592}{3}\frac{p_1}{(p_1-p_2)^2}\ln\left(\frac{592}{3}\sqrt{\frac{K}{\delta}}\frac{p_1}{(p_1-p_2)^2}\right).$$

In this paper, we approach the PAC mode estimation problem from a different perspective. In recent work, Waudby-Smith and Ramdas [1] propose prior-posterior-ratio (PPR) martingale confidence sequences, a novel framework to obtain "anytime" confidence bounds on unknown parameters of a probability distribution. The resulting stopping rule is simple, with no need for tuning, and yet works surprisingly well in practice, for example to estimate the parameter of a Bernoulli distribution by sampling. Encouraged by this empirical finding, we undertake the generalisation of the PPR martingale test to mode estimation, wherein the K-valued input distribution $\mathcal{P}$ has $K-1$ parameters, for some $K \geq 2$. We denote our generalised stopping rule PPR-ME (PPR applied to mode estimation).

In Section 2, we upper-bound the sample complexity of PPR-ME for the base case of $K = 2$, showing it to be optimal up to a logarithmic factor for the well-studied "Bernoulli" setting. This finding might independently bear significance to the push for tighter confidence bounds in the bandits literature [3, 4]. In Section 3, we describe PPR-ME for $K \geq 2$ and derive an upper bound on its sample complexity. Interestingly, this upper bound also matches the lower bound in Theorem 1 up to a logarithmic factor, and marginally improves upon the constant factor in the upper bound from Theorem 2. In sections 2 and 3, we also provide empirical comparisons between PPR-ME and other approaches to PAC mode estimation. We find strong support for using "one-versus-one" comparisons for mode estimation (as in PPR-ME), as opposed to a "one-versus-rest" approach (as in $\mathcal{A}_1$ [2]).

Over the years, the mode estimation problem has received attention in many different contexts [5, 6, 7]. We illustrate the gains obtained from PPR-ME in two contrasting real-life applications. In Section 4, we show that PPR-ME, used as a subroutine, can reduce the sample complexity of winner-forecasting in indirect elections [8]. In Section 5, we illustrate the relevance of PPR-ME for probabilistic verification in permissionless blockchains [9]. We conclude with a summary in Section 6.

2 PPR Martingale Confidence Sequences

In this section, we consider the "base case" of mode estimation, in which $\mathcal{P}$ takes exactly $K = 2$ values. Notice that $\mathcal{P}(p, v, 2)$ is a Bernoulli distribution that generates v_1 with probability p_1 and v_2 with probability $p_2 = 1 - p_1$. Treating $p_1 \in [0,1]$ as the sole parameter of the distribution, our task is to devise a δ-correct stopping rule to test if $p_1 > \frac{1}{2}$. Since p_1 may be arbitrarily close to $\frac{1}{2}$, it is not possible to decide beforehand how many samples suffice for the test to succeed. An unfortunate consequence of having a *random* stopping time is that it cannot be used directly within concentration inequalities such as Chernoff bounds. Rather, stopping rules invariably go through a union bound over all possible stopping times, dividing the mistake probability δ among them [10, 11]. Although there has been progress towards optimising this apportioning of δ [4, 12], resulting methods still have tunable parameters in their "decay rates" that govern the stopping time.

The recent development of "time-uniform" or "anytime" Chernoff bounds [13] relieve the experimenter of the burden of parameter-tuning. Arising from this line of research is the framework of prior-posterior-ratio (PPR) martingale confidence sequences [1], which result in a simple and intuitive stopping rule. Although the rule may be applied to a wide range of single-parameter families, we restrict our upcoming discussion to the Bernoulli case at hand: that is, to test whether $p_1 > \frac{1}{2}$.

To apply the PPR martingale framework, we maintain a belief distribution π for p_1 over its range $[0,1]$, and update π according to Bayes' rule as samples are observed. Our aim is still to provide a frequentist guarantee of that holds for all possible values of p_1 (δ-correctness). To this end we must ensure that the prior distribution π^0 gives non-zero density to all possible values of p_1. We do so by adopting the uniform prior $\pi^0(\hat{p}_1) = 1$ for $\hat{p}_1 \in [0,1]$. For $t \geq 1$, we update our belief distribution after observing sample x^t, as follows.

$$\pi^t(\hat{p}_1) = \frac{\pi^{t-1}(\hat{p}_1)\cdot(\hat{p}_1)^{\mathbf{1}[x^t=v_1]}\cdot(1-\hat{p}_1)^{\mathbf{1}[x^t=v_2]}}{\int_{\rho=0}^{1}\pi^{t-1}(\rho)\cdot(\rho)^{\mathbf{1}[x^t=v_1]}\cdot(1-\rho)^{\mathbf{1}[x^t=v_2]}d\rho}.$$

The prior-posterior-ratio (PPR) at $\hat{p}_1 \in [0,1]$ is given by $R^t(\hat{p}_1) = \frac{\pi^0(\hat{p}_1)}{\pi^t(\hat{p}_1)}$. Making use of the fact that the PPR evaluated at the *true* parameter value, p_1, is a martingale, it can be shown that the sequence of sets $(C^t)_{t=0}^{\infty}$, where $C^t \overset{\text{def}}{=} \{\hat{p}_1 : R^t(\hat{p}_1) < \frac{1}{\delta}\}$, is a $(1-\delta)$-confidence sequence for p_1 [1]. In other words, we have the "anytime" guarantee that

$$\mathbb{P}\{\, \exists\, t \geq 0 : p_1 \notin C^t\} \leq \delta. \tag{1}$$

The belief distribution π^t and hence the PPR R^t assume a convenient form if initialised with the uniform prior. Suppose the sequence of samples up to time t is $x^1, x^2, \dots, x^t$, which contains s_1^t occurrences of v_1 and $s_2^t = t - s_1$ occurrences of v_2. Then for $t \geq 0$ and $\hat{p}_1 \in [0,1]$, we obtain $\pi_t(\hat{p}_1) = \text{Beta}(\hat{p}_1; s_1^t + 1, s_2^t + 1)$. We can terminate as soon as the $(1-\delta)$-confidence sequence on p_1 does not contain $\frac{1}{2}$. For easy readability, let us define indices first(t) and second(t), where $(\text{first}(t), \text{second}(t)) \in \{(1,2),(2,1)\}$ satisfy $s^t_{\text{first}(t)} \geq s^t_{\text{second}(t)}$. We obtain the following simple simple stopping rule, applied at each time step $t \geq 1$.

$$\boxed{\text{PPR-Bernoulli: Stop and declare } v_{\text{first}(t)} \text{ as mode iff Beta}\left(\tfrac{1}{2}; s^t_{\text{first}(t)} + 1, s^t_{\text{second}(t)} + 1\right) \leq \delta.}$$

The δ-correctness of this stopping is immediate from (1), itself a proven result [1]. However, we are not aware of any existing analysis of its stopping time. We obtain the following upper bound by expanding out the Beta density function at $\frac{1}{2}$ and bounding tail probabilities using Chernoff bounds. The detailed working, which also involves steps to optimise constants, is provided in Appendix A.

Lemma 3 (PPR-Bernoulli upper bound)**.** *Fix $\delta \in (0,1)$ and problem instance $\mathcal{P} = (p, v, 2)$. When PPR-Bernoulli is run on $\mathcal{P}$, with probability $1-\delta$, the number of samples it observes is at most*

$$\frac{20.775 p_1}{(p_1 - \frac{1}{2})^2} \ln\left(\frac{2.49}{(p_1 - \frac{1}{2})^2 \delta}\right).$$

Like the result in Theorem 2, the lemma upper-bounds the sample complexity only on the event that a mistake is not made. However, we can obtain an upper bound on the *expected* sample complexity, with only the constants changed, by adopting a doubling trick—wherein the procedure is repeatedly performed with increasing budgets and decreasing mistake probabilities, until it terminates. The "p_1" in the numerator of the upper bound above can be ignored as a constant factor since $p_1 \in (\frac{1}{2}, 1]$, but the explicit form is convenient for our generalisation to $K \geq 2$, which we take up in the next section.

It is worth noting that the LHS of the PPR-Bernoulli stopping rule can be evaluated *exactly* as a rational using integer arithmetic, requiring a only lightweight incremental update after each sample. This implementation of the PPR-Bernoulli stopping rule is arguably simpler than the numerical optimisation needed by many other stopping rules, as we see shortly.

2.1 Empirical comparison with other Stopping Rules for $K = 2$

We present experimental results comparing PPR-Bernoulli with other stopping rules from the literature. The predominant approach is to construct lower and upper confidence bounds on p_1 that hold with probability $1 - \delta_t$ for each $t \geq 1$, satisfying $\sum_{t=0}^{\infty} \delta_t \leq \delta$. The δ-correctness of the procedure is ensured by terminating only when the lower confidence bound exceeds $\frac{1}{2}$ or the upper confidence bound falls below $\frac{1}{2}$. A common choice is to set $\delta_t = k\frac{\delta}{t^{\alpha}}$, with constants k and α tuned for efficiency while ensuring δ-correctness. As representatives of this approach, we pick the LUCB and KL-LUCB algorithms of Kaufmann and Kalyanakrishnan [11]. The former inverts Hoeffding's inequality to obtain lower and upper confidence bounds, while the latter uses a tighter Chernoff bound. Although these algorithms themselves are meant for bandit applications, their efficiency crucially depends on the tightness of the confidence bounds applied to each arm. The tuned confidence bounds [11] therefore become suitable baselines for our comparison. Details of our implementation of the LUCB and KL-LUCB confidence bounds, as well as forthcoming ones, are provided in Appendix C. Links to code and related instructions are given in Appendix D.

With the intent of avoiding a naïve union bound over time, Garivier applies a so-called *peeling* argument to divide time into increasingly-sized slices. He obtains confidence regions by associating the random stopping time with a self-normalised process. The resulting stopping rule, which we denote KL-SN, still has a tunable parameter "c", which we set as recommended by Garivier [12].

Although the $\mathcal{A}_1$ algorithm of Shah et al. [2] is designed specifically for mode estimation, we include it in this comparison to observe its performance when $K = 2$. In this special case, the algorithm reduces to an application of an empirical Bernstein bound given by Maurer and Pontil [14].

In Figure 1a, we observe the stopping times of the different rules as p_1 is varied, for mistake probability $\delta = 0.01$. KL-LUCB shows a marginal improvement over (Hoeffding) LUCB, while KL-SN clearly outperforms both. However, PPR-Bernoulli is significantly more efficient than even KL-SN. Given its simplicity and non-reliance on parameter-tuning, PPR-Bernoulli therefore becomes an attractive proposition for stopping problems. Surprisingly, in spite of using variance information, $\mathcal{A}_1$ registers the worst performance among all the methods compared in this experiment. We attribute this result to slack in the constants used in its stopping rule, which future work can possibly tighten, while still ensuring δ-correctness. Figure 1b affirms the trend among the stopping rules when they are applied to the same distribution ($p_1 = 0.65$), but with varying mistake probability δ.

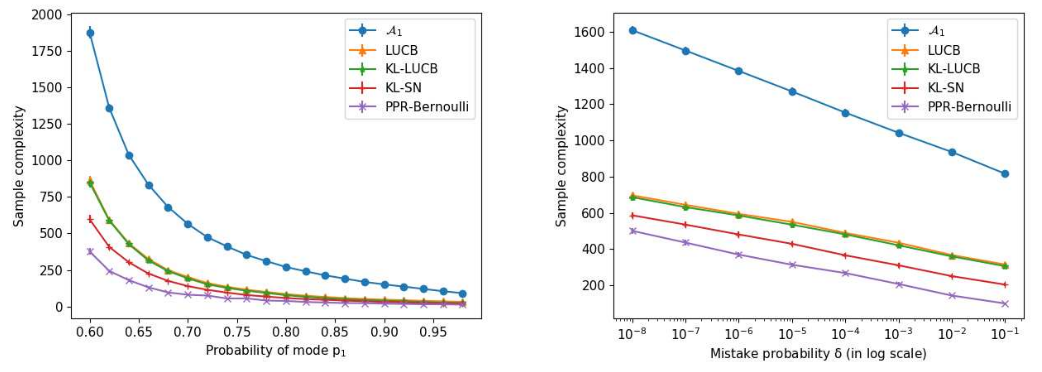
3 PPR-ME Stopping Rule for Mode Estimation

In this section, we present our generalisation of the PPR-Bernoulli stopping rule to the problem of mode estimation: that is, when the support set is of size $K \geq 2$. In the broader machine learning literature, the most common approaches for generalising 2-class problems to more classes are "one-versus-one" (denoted 1v1) and "one versus rest" (denoted 1vr). We investigate both approaches, and present them in turn. Interestingly, we observe overwhelming evidence in favour of using the 1v1 approach for mode estimation. Thus, the PPR-ME stopping rule that we propose based on the investigation in this paper is the same as PPR-1v1, which we describe next.

3.1 One-versus-one (1v1) Approach

In the first $t \geq 1$ samples, let the number of occurrences of value v_i be $s_i(t)$, $1 \leq i \leq K$. The 1v1 generalisation proceeds based on the idea that if v_i is to be declared the mode, we need to be sufficiently sure that v_i is more probable than v_j for $i, j \in \{1, 2, \ldots, K\}, i \neq j$. In keeping, we simultaneously run PPR-Bernoulli tests on each (i, j) pair with mistake probability $\frac{\delta}{K-1}$. Each (i, j) test relies solely on the number of occurrences of v_i and v_j, disregarding other values. Hence it amounts to observing samples from a Bernoulli variable with parameter $\frac{p_i}{p_i+p_j}$, and verifying which side of $\frac{1}{2}$ its mean lies. The overall procedure stops when some $i \in \{1, 2, \ldots, K\}$ has won each of its tests. By a union bound, we obtain that with probability at least $1 - \delta$, the (true) mode v_1 will not ever lose a test. Thus, upon termination, v_1 is returned with probability at least $1 - \delta$.

Whereas the description above suggests we need to monitor $\binom{K}{2}$ tests at each step, closer inspection reveals that a much lighter implementation is possible. As before, let first(t) denote the index of the



(a) p_1 varied; $\delta = 0.01$.

(b) δ varied; $p_1 = 0.65$.

Figure 1: Comparison of stopping rules for the Bernoulli case ($K = 2$). Both plots show sample complexity: in (a) as p_1 is varied, and in (b) as δ is varied. The results are averages from 100 runs. Error bars show one standard error (in both plots very small).

most-frequently occurring value (with arbitrary tie-breaking) after t samples: that is, $s^t_{\text{first}(t)} \geq s^t_i$ for $i \in \{1, 2, \dots, K\}$. Now, if at all a winner is identified after t samples, clearly it must be $v_{\text{first}(t)}$, which has as many occurrences as any other value. Hence, we only need to track tests involving $v_{\text{first}(t)}$. Now, it is also immediate that $v_{\text{first}(t)}$ wins all its tests if and only if it defeats the second most frequently occurring value, which we denote second(t): that is, second$(t) \in \{1, 2, \dots, K\}$, second$(t) \neq$ first(t) satisfies $s^t_{\text{second}(t)} \geq s^t_i$ for $i \in \{1, 2, \dots, K\} \setminus \{\text{first}(t)\}$. Hence, we may implement our stopping rule, denoted PPR-1v1, using a *single* PPR-Bernoulli test at each step $t \geq 1$.

$$\boxed{\text{PPR-1v1: Stop and declare } v_{\text{first}(t)} \text{ as mode iff Beta}\left(\tfrac{1}{2}; s^t_{\text{first}(t)} + 1, s^t_{\text{second}(t)} + 1\right) \leq \tfrac{\delta}{K-1}.}$$

Whereas a $\Theta(K)$ initialisation step is needed to initialise all counts to zero, observe that the computation needed at each step $t \geq 1$ is only $\Theta(1)$—which suffices for identifying first(t) and second(t) based on their previous values, and incrementing the count of the x_t. Interestingly, we are able to show a sample complexity upper bound for this simple procedure to improve upon the one from Theorem 2 [2], albeit by a small constant factor.

Theorem 4 (PPR-1v1 upper bound). *Fix $\delta \in (0, 1)$, $K \geq 2$, and problem instance $\mathcal{P} = (p, v, K)$. When PPR-ME is run on $\mathcal{P}$, with probability $1 - \delta$, the number of samples it observes is at most*

$$t^\star = \frac{194.07 p_1}{(p_1 - p_2)^2} \ln\left(\sqrt{\frac{79.68(K-1)}{\delta}} \frac{p_1}{(p_1 - p_2)}\right).$$

Proof. Fix arbitrary $t > t^\star$ and $j \geq 2$. Our strategy is to show that with sufficiently high probability, v_1 and v_j will have separated before t pulls. First, let Z_t denote the sum of the number of occurrences of v_1 and v_j in the first t samples: that is, $Z^t = s^t_1 + s^t_j$. Clearly Z_t is a binomial random variable with parameters t and $(p_1 + p_j)$, We argue that Z^t cannot fall too far below its mean. Concretely, take $\delta' = \frac{\delta}{2(K-1)}$ and $l = \sqrt{\frac{2\ln(\frac{1}{\delta'})}{(p_1+p_2)t}}$. A Chernoff bound yields

$$\mathbb{P}\{Z_t \leq (1 - l)(p_1 + p_j)t\} \leq \exp\left(-\frac{l^2(p_1 + p_j)t}{2}\right) \leq \delta'$$

for our choice of l. Thus, with probability at least $1 - \delta'$, v_1 and v_j together have more than $(1 - l)(p_1 + p_j)t$ samples. Now, the test to separate v_1 and v_j is PPR-Bernoulli on a Bernoulli variable with parameter $q_1 = \frac{p_1}{p_1+p_j} > \frac{1}{2}$. Although PPR-ME runs this test with mistake probability $\frac{\delta}{K-1}$, imagine running it with mistake probability $\delta' = \frac{\delta}{2(K-1)}$. The latter test would necessarily incur equal or more samples on *every* run. Yet, from Lemma 3, we know that with probability at least $1 - \delta'$, the latter will terminate after at most $u = \frac{20.775 q_1}{(q_1 - \frac{1}{2})^2} \ln\left(\frac{2.49}{(q_1 - \frac{1}{2})^2 \delta'}\right)$ samples. It can be verified that for $t > t^\star$, $u < (1 - l)(p_1 + p_j)t$ (calculation shown in Appendix B). In other words, the probability that v_1 and v_j have *not* separated before t pulls is at most $2\delta'$.

Since the argument above holds for arbitrary $j \geq 2$, a union bound establishes that with probability at least $1 - \delta$, v_1 must have separated from all other values—implying termination—before t pulls. □

3.2 One-versus-rest (1vr) Approach

Notice that under PPR-1v1, sample x^t at each step $t \geq 1$ contributes to only $K - 1$ tests; $\binom{K-1}{2}$ tests corresponding to values other than x^t receive no information. The 1vr approach becomes an alternative to address this *apparent* wastage of information. Under the 1vr scheme, we associate a Bernoulli variable B_i with each value v_i, $1 \leq i \leq K$, which has probability p_i of generating v_i, and probability $1 - p_i$ of generating its *negation* "$\neg v_i$". Consequently, each sample of $\mathcal{P}$ adds to one of the outcomes of B_i for *each* $i \in \{1, 2, \dots, K\}$. We draw an anytime confidence sequence for B_i with mistake probability $\frac{\delta}{K}$, and terminate after $t \geq 1$ samples if the confidence set of $B_{\text{first}(t)}$ does not overlap with any of the others. Invoking the PPR martingale confidence sequence, we note that with probability at least $1 - \frac{\delta}{K}$, p_i will lie in all the intervals $(\text{LCB}^t_i, \text{UCB}^t_i)$, $t \geq 1$, where

$$\text{LCB}^t_i = \min\left\{\hat{p} \in [0, 1] : \pi^t(\hat{p}) = \frac{\delta}{K}\right\} \text{ and } \text{UCB}^t_i = \max\left\{\hat{p} \in [0, 1] : \pi^t(\hat{p}) = \frac{\delta}{K}\right\}$$

can be computed numerically. The δ-correctness of the 1vr rule, summarised below, is established by applying a union bound on the mistake probabilities of each B_i, $1 \leq i \leq K$.

$$\boxed{\text{PPR-1vr: Stop and declare } v_{\text{first}(t)} \text{ as mode iff for } 1 \leq i \leq K,\, i \neq \text{first}(t),\, \text{LCB}^t_{\text{first}(t)} \geq \text{UCB}^t_i.}$$

3.3 Empirical Comparisons with other Stopping Rules for Mode Estimation

The $\mathcal{A}_1$ algorithm of Shah et al. [2] is essentially a 1vr approach that uses Empirical Bernstein confidence bounds [14]. Noting that it can just as well be implemented in a 1v1 form, we include such a variant, denoted $\mathcal{A}_1$-1v1, in our experimental comparisons with PPR. For good measure, we also include 1v1 and 1vr variants based on the KL-SN confidence bound [12], which finished second to PPR-Bernoulli for $K = 2$ (see Section 2). Table 1 shows the performance of 1v1 and 1vr versions of PPR, $\mathcal{A}_1$, and KL-SN on a variety of distributions. We observe the same trend on *each* problem instance: (1) The 1v1 variant of each stopping rule outperforms the corresponding 1vr variant, and (2) PPR is most sample-efficient, followed by KL-SN and $\mathcal{A}_1$. In line with the upper bound from theorems 2 and 4, it does appear that the sample complexity is largely determined by p_1 and p_2 (identical for instances $\mathcal{P}_5$ and $\mathcal{P}_6$) in the table. The logarithmic dependence on K results in a slightly larger number of samples for all these methods on $\mathcal{P}_6$.

Interestingly, not only do the 1v1 variants of each method perform better than 1vr in aggregate, we observe that they terminate *before* the 1vr variants on *every* single run. Intuitively, it appears that although the 1vr variants update all their K tests with each sample, the condition to separate any two variables becomes weaker than that under the corresponding 1v1 variant. We discuss this effect in Appendix E. It is also worth noting that PPR-1v1, which can be implemented with a constant number of arithmetic operations, is *computationally* the fastest among the methods presented in this section.

4 Application: Winner-forecasting in Indirect Elections

Opinion polls—deployed to gauge public perception of competing products in a market, or to forecast the winner of an upcoming election—are a natural application of mode estimation. In fact, the algorithms discussed in Section 3 can all be applied with only minor alterations to *plurality* systems, wherein the task is precisely that of determining the choice preferred by the largest fraction of the target population. Waudby-Smith and Ramdas [1] illustrate the use of PPR martingale confidence sequences precisely on this application. Specifically they consider the construction of confidence sequences when polls receive *without-replacement* samples from finite populations.

Table 1: Sample complexity comparison for mode estimation, run with mistake probability $\delta = 0.01$. The number after the $\times$ symbol indicates the multiplicity of that particular probability value in the distribution; thus $\mathcal{P}_1$ has $p = (0.5, 0.25, 0.25)$. The values reported are averages from 100 or more runs, and show one standard error.

Distribution	K	Type	$\mathcal{A}_1$ [2]	KL-SN [12]	PPR
$\mathcal{P}_1$: .5, .25 $\times$ 2	3	1vr	1344±20	444±14	262±12
		1v1	1157±19	346±13	**218±11**
$\mathcal{P}_2$: .4, .2 $\times$ 3	4	1vr	1919±29	662±18	397±15
		1v1	1515±25	472±15	**298±13**
$\mathcal{P}_3$: .2, .1 $\times$ 8	9	1vr	5082±51	3767±68	1201±29
		1v1	3340±43	1138±32	**789±28**
$\mathcal{P}_4$: .1, .05 $\times$ 18	19	1vr	12014±129	4540±80	2850±55
		1v1	7351±88	2554±56	**1840±53**
$\mathcal{P}_5$: .35, .33, .12, .1 $\times$ 2	5	1vr	155277±2356	63739±2238	38001±1311
		1v1	117988±2078	47205±1291	**33660±1125**
$\mathcal{P}_6$: .35, .33, .04 $\times$ 8	10	1vr	158254±2442	66939±2241	41963±1330
		1v1	121150±2183	49576±1341	**36693±1185**

In parliamentary democracies such as India [15] and the United Kingdom [16], individuals in each *constituency* (or *seat*)—typically a geographically contiguous region—elect a party based on plurality; the party winning the most seats forms the government. This two-level structure is followed both to elect governments in each state, and for the entire country. Forecasting the winning party in such an *indirect* voting system calls for a more sophisticated sampling procedure. Whereas it would *suffice* to separately identify the winner from each seat by sampling, it might be wasteful to do so when the overall winning party has a clear majority in its number of seats. Rather, one could potentially use the results currently available from *all* the constituencies to decide which ones to query next.

In this section, we consider a procedure that (1) keeps track of the current winners and leaders at the aggregate level, and (2) at each step samples the constituencies that appear most promising to confirm the aggregate trend. In principle, this algorithm, denoted DCB (for "Difference in Confidence Bounds") can be coupled with any algorithm that uses confidence bounds for mode estimation. Yet, we obtain the best results when DCB uses PPR-ME as a subroutine, thereby highlighting the relevance of PPR-ME not only as a stopping rule, but also as an input to on-line decision making.

DCB takes cue from the LUCB algorithm for best-arm identification in bandits [10]. At each step t, it identifies two parties, a^t and b^t, that appear the most promising to win the overall election: these parties are picked based on their current number of wins and "leads" in individual constituencies. Subsequently the algorithm chooses a constituency each for a^t and b^t, samples from which could "most" help distinguish the tally of the two. We provide a detailed specification of DCB in Appendix F.

4.1 Performance when Coupled with DCB

We compare DCB with a round-robin strategy for picking the next constituency to sample. Both approaches can be implemented with different stopping rules, which are also varied. Table 2 shows our results on two recent (and contrasting) elections conducted in India.[1] In the 2014 national elections, the winning party secured 282 seats from among 543, giving it a very large victory over the second-place party, which won 44 seats. The second problem instance is from a closer contest in the state of Bihar, in which the top three parties won 80, 71, and 53 seats, of a total 242.

While it is not the central feature of this paper, it is worth noting that the DCB strategy indeed improves over round-robin polling by roughly a factor of two, regardless of the stopping rule. As intended, it does not waste samples on constituencies that are inconsequential to the overall result (observed in the "seats resolved" columns). Of more direct relevance to the theme of the paper is that even when embedded within a decision-making outer loop, PPR continues to outperform $\mathcal{A}_1$ and KL-SN, and the 1v1 approach still dominates 1vr.

It must be emphasised that in practice, election surveys typically have the financial budget only for a few thousands of samples, and the time only for a single-stage (non-sequential) poll [8]. Opinion

[1]Election results are in the public domain; the authors accessed them at `https://www.indiavotes.com/`.

Table 2: Sample complexity of various stopping rules when coupled with (1) round-robin (RR) polling of constituencies and (2) DCB. All experiments are run with mistake probability $\delta = 0.01$. Values shown are averages from 10 runs, and show one standard error. "Seats resolved" indicates the number of constituencies in which a winner was identified before the overall procedure terminated.

Algorithm	India-2014 (543 seats) Samples	India-2014 (543 seats) Seats Resolved	Bihar-2015 (242 seats) Samples	Bihar-2015 (242 seats) Seats Resolved
RR-$\mathcal{A}_1$-1v1	1578946 ± 10255	239 ± 2	4201429 ± 43932	221 ± 1
RR-$\mathcal{A}_1$-1vr	1767464 ± 15828	234 ± 1	4841406 ± 58188	222 ± 1
RR-KLSN-1v1	610181 ± 7072	240 ± 2	2198678 ± 44514	221 ± 1
RR-KLSN-1vr	726512 ± 6186	236 ± 2	2668729 ± 47535	222 ± 2
RR-PPR-1v1	$\mathbf{471661 \pm 7373}$	241 ± 3	$\mathbf{1813213 \pm 42081}$	221 ± 1
RR-PPR-1vr	560815 ± 7778	238 ± 2	2054171 ± 40394	221 ± 1
DCB-$\mathcal{A}_1$-1v1	856678 ± 3935	182 ± 1	2301936 ± 36399	135 ± 1
DCB-$\mathcal{A}_1$-1vr	951684 ± 5246	180 ± 1	2481552 ± 21483	134 ± 1
DCB-KLSN-1v1	325265 ± 2096	186 ± 2	1127963 ± 22191	139 ± 2
DCB-KLSN-1vr	376108 ± 4067	181 ± 1	1312027 ± 22892	139 ± 1
DCB-PPR-1v1	$\mathbf{256911 \pm 2096}$	188 ± 1	$\mathbf{883389 \pm 15581}$	142 ± 2
DCB-PPR-1vr	296580 ± 2372	184 ± 1	993495 ± 18859	139 ± 1

polls must also cope with difficulties in obtaining truthful votes and samples drawn uniformly at random from a population [17]. We have conveniently assumed a more abstract setting primarily to tease apart the efficiency of different stopping rules for mode estimation. We believe the numerical superiority of PPR-1v1 makes it a good choice for embedding within more complex and realistic models, although we leave the exercise of doing so to future work.

5 Application: Verifying Smart Contracts in Permissionless Blockchains

Our second application of PAC mode estimation is in a domain of growing contemporary relevance. Permissionless blockchains such as Bitcoin [18] and Ethereum [19] allow uncertified agents to join a pool of service providers, also called *nodes*. A recent feature that has emerged in such blockchains is the execution of "smart contracts" [9, 19], which could include, for example, running computationally-heavy jobs such as machine learning algorithms. In an ideal world, a client who requires a computation to be performed can simply enter into a smart contract with some particular node in the blockchain, and pay a transaction cost for the service. Unfortunately, there is no guarantee that nodes in a permissionless blockchain are honest. A "Byzantine" (or *malicious*) node could easily return a quick-to-compute, incorrect output, to the detriment of the client.

In recent work, Das et al. [9] propose an approach for the *probabilistic verification* of smart contracts. Abstractly, assume that the computation to be performed for the client is deterministic, and it has a (yet unknown) output o_{correct}. The proposed model accommodates any blockchain in which the fraction of Byzantine nodes is at most $f_{\max} \in [0, \frac{1}{2})$. With this assumption, it becomes feasible to give a probabilistic guarantee on obtaining the correct output. For any fixed mistake probability $\delta \in (0, 1)$, the client could ship out the computation to $\Theta\left(\frac{1}{(\frac{1}{2}-f_{\max})^2}\log(\frac{1}{\delta})\right)$ nodes, and take their majority response as the answer, thereby ensuring δ-correctness. Unfortunately, transaction costs can be substantial, especially those for computationally-intensive contracts. Hence, it is in the client's interest to minimise the number of nodes queried to achieve the same probabilistic guarantee. For example, a sequential procedure could potentially query fewer nodes if $f \ll f_{\max}$.

5.1 Comparison with SPRT

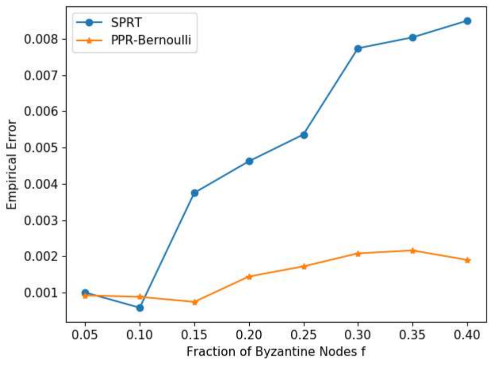
Our contribution in the context of this application is to propose PPR-ME as an alternative to Wald's Sequential Probability Ratio Test (SPRT) [20], which is used by Das et al. [9] for their verification procedure. This classical test finds use in many other engineering applications [21, 22], some of which could also benefit from the advantages of PPR-ME over SPRT.

To apply SPRT for verifying smart contracts, Das et al. [9] assume that out of the total of N nodes in the blockchain, batches of size m, chosen uniformly at random, can be queried in sequence. For simplicity assume the answers returned are from the set $\{0, 1, 2, \dots\}$. Let $q = \frac{m}{N}$, and $c_{i,t}$ be the number of times answer i is reported in the t^{th} step, $i \geq 0, t \geq 1$. Defining $l_{i,T} = \sum_{t=1}^{T}(2c_{i,t} - m)m$, a derivation [9] establishes that δ-correct SPRT stops at time T, giving i as the answer if

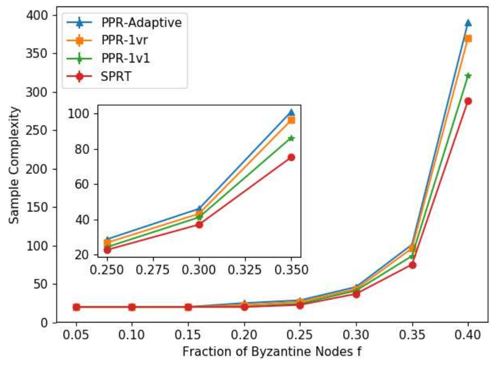
$$l_{i,T} > \ln\left(\frac{1-\delta}{\delta}\right)\frac{2q(1-q)N(1-f_{\max})f_{\max}}{1-2f_{\max}}.$$

The primary disadvantage of SPRT in this context is the need for the user to provide $f_{\max}$, which is used in the stopping rule. While a lower value of $f_{\max}$ will improve the efficiency of the rule, unfortunately δ-correctness no longer holds if f, the true fraction of Byzantine nodes, exceeds f_{max}. Figure 2a plots the empirical error made by SPRT (averaged over 50,000 runs) on a problem instance in which the Byzantine nodes all give the same (incorrect) answer. We fix $f_{\max} = 0.1$, and plot the error for different settings of f. The blockchain has $N = 1600$ nodes, of which SPRT samples $m = 20$ at a time. Although the test is run using mistake probability $\delta = 0.005$, observe that the empirical error exceeds δ when $f > f_{\max}$.

Since the verification task at hand is precisely that of PAC mode estimation, PPR-ME becomes a viable alternative, especially since it does not need the knowledge of $f_{\max}$. In fact, PPR-ME can identify the mode even if its associated probability is less than $\frac{1}{2}$ (although in this case, it can no longer be guaranteed that the mode is o_{correct}, since the Byzantine nodes may collude). Observe from Figure 2a that unlike SPRT, the empirical error rate of PPR-ME (equivalent to PPR-Bernoulli since we have set $K = 2$) remains within δ even for $f > f_{\max}$.



(a) $K = 2$ (single adversarial answer)



(b) $K = 10$

Figure 2: Comparisons with SPRT for the probabilistic verification of smart contracts, obtained with parameter settings $N = 1600$, $m = 20$, $\delta = 0.005$, $f_{max} = 0.1$. Plot (a) shows the empirical error rates of the algorithms as the true Byzantine fraction f is varied, taking $K = 2$. Plot (b) shows the sample complexity of various algorithms against varying f, on an instance with $K = 10$ answers.

In Figure 2b, we compare the sample complexities of PPR-ME and SPRT. Whereas other problem parameters (including f_{max}) stay the same as before, we consider an instance in which $K = 10$. The single correct answer is given by a $(1 - f)$-fraction of the nodes, while 9 different incorrect answers are given by the Byzantine nodes, each equally common. The version of SPRT used is a 1vr adaptation of the basic procedure [9] to $K = 10$. First, we observe that SPRT terminates before PPR-1v1 and PPR-1vr at all values of f. The PPR algorithms pay this price for having to assure δ-correctness at all values of $f < \frac{1}{2}$, unlike SPRT, which does so only for $f < f_{max}$. We show the performance of another PPR variant, denoted "PPR-Adaptive", in the same plot. In reality, we cannot be sure about the number of answers K that will be returned by the blockchain's nodes–and hence cannot use it in our stopping rule. Under PPR-Adaptive, which is a 1v1 strategy, the overall mistake probability δ is divided into the *infinite* sequence $k\frac{\delta}{1^2}, k\frac{\delta}{2^2}, k\frac{\delta}{3^2}, \ldots$ (with $k = \frac{6}{\pi^2}$). Whenever a new answer is revealed, it is inserted into the list of possible answers, and its pairwise tests given mistake probabilities from the unused portion of the sequence. In principle, PPR-Adaptive can accommodate any number of answers, incurring only a small increase in sample complexity, as visible from Figure 2b.

6 Conclusion

In this paper, we apply the framework of PPR martingale confidence sequences to the problem of PAC mode estimation. Our investigation follows two different dimensions that play a significant role in determining the efficiency of stopping rules. First is the tightness of the confidence bounds used internally in the stopping rule. By separately focusing on the Bernoulli case, we show that the PPR martingale stopping rule is sample-efficient both in theory and in practice. Our analysis as well as experiments may be of independent interest to research on pure exploration problems in stochastic bandits. The second aspect of mode estimation is the template applied to generalise from $K = 2$ to $K \geq 2$, which can typically be applied with any valid confidence bounds. Of the two major choices—"one-versus-one" (1v1) and "one-versus-rest" (1vr)—we find 1v1 to be the more efficient.

The PPR-ME algorithm that combines both ideas is simple, computationally efficient, yet significantly more sample-efficient not only than the recently-proposed $\mathcal{A}_1$ algorithm [2], but many other baselines (including an improved variant of $\mathcal{A}_1$ that we include in our experiments). To illustrate the practical relevance of PPR-ME, we present two separate real-world applications in which it can offer significant advantages over earlier approaches. In sample-based election forecasting, PPR-ME significantly reduces the sample complexity when paired with the DCB sampling strategy. Analysing the sample complexity of DCB in terms of its problem parameters remains an avenue for future work, which could contribute to a growing literature on sample-based techniques in social choice theory [23, 24]. Our second application, to probabilistic verification in blockchains, is based on PPR-ME being relatively assumption-free, which makes it a promising algorithm for many industrial applications.

References

[1] Ian Waudby-Smith and Aaditya Ramdas. Confidence sequences for sampling without replacement. In *Advances in Neural Information Processing Systems*, volume 33, pages 20204–20214. Curran Associates, Inc., 2020.

[2] Dhruti Shah, Tuhinangshu Choudhury, Nikhil Karamchandani, and Aditya Gopalan. Sequential mode estimation with oracle queries. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 5644–5651. AAAI Press, 2020.

[3] Emilie Kaufmann and Wouter Koolen. Mixture martingales revisited with applications to sequential tests and confidence intervals. *arXiv preprint arXiv:1811.11419*, 2018.

[4] Kevin Jamieson, Matthew Malloy, Robert Nowak, and Sébastien Bubeck. lil' ucb : An optimal exploration algorithm for multi-armed bandits. In *Proceedings of The 27th Conference on Learning Theory*, volume 35 of *Proceedings of Machine Learning Research*, pages 423–439. PMLR, 2014.

[5] Emanuel Parzen. On estimation of a probability density function and mode. *The Annals of Mathematical Statistics*, 33(3):1065–1076, 1962.

[6] Gurmeet Singh Manku and Rajeev Motwani. Approximate frequency counts over data streams. In *Proceedings of the 28th International Conference on Very Large Data Bases*, VLDB '02, page 346–357. VLDB Endowment, 2002.

[7] J. Misra and David Gries. Finding repeated elements. *Science of Computer Programming*, 2(2):143–152, 1982.

[8] Rajeeva Karandikar. Power and limitations of opinion polls in the context of Indian parliamentary democracy. In *Special Proceeding of 20th Annual Conference of SSCA*, pages 09 – 16. Society of Statistics, Computer and Applications, 2018.

[9] Sourav Das, Vinay Joseph Ribeiro, and Abhijeet Anand. YODA: enabling computationally intensive contracts on blockchains with byzantine and selfish nodes. In *26th Annual Network and Distributed System Security Symposium, NDSS*. The Internet Society, 2019.

[10] Shivaram Kalyanakrishnan, Ambuj Tewari, Peter Auer, and Peter Stone. PAC subset selection in stochastic multi-armed bandits. In *Proceedings of the 29th International Conference on International Conference on Machine Learning*, ICML'12, page 227–234. Omnipress, 2012.

[11] Emilie Kaufmann and Shivaram Kalyanakrishnan. Information complexity in bandit subset selection. In *Proceedings of the 26th Annual Conference on Learning Theory*, volume 30 of *Proceedings of Machine Learning Research*, pages 228–251. PMLR, 2013.

[12] Aurélien Garivier. Informational confidence bounds for self-normalized averages and applications. In *2013 IEEE Information Theory Workshop (ITW)*, pages 1–5. IEEE press, 2013.

[13] Steven R Howard, Aaditya Ramdas, Jon McAuliffe, Jasjeet Sekhon, et al. Time-uniform Chernoff bounds via nonnegative supermartingales. *Probability Surveys*, 17:257–317, 2020.

[14] Andreas Maurer and Massimiliano Pontil. Empirical Bernstein bounds and sample variance penalization. *arXiv preprint arXiv:0907.3740*, 2009.

[15] Rajeeva L. Karandikar, Clive Payne, and Yogendra Yadav. Predicting the 1998 Indian parliamentary election. *Electoral Studies*, 21(1):69–89, 2002.

[16] Clive Payne. Election forecasting in the UK: The BBC's experience. *Euramerica*, 33, 2003.

[17] Paul Perry. Certain problems in election survey methodology. *The Public Opinion Quarterly*, 43(3):312–325, 1979.

[18] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, http://bitcoin.org/bitcoin.pdf.

[19] Vitalik Buterin et al. A next-generation smart contract and decentralized application platform, https://cryptorating.eu/whitepapers/Ethereum/Ethereum_white_paper.pdf.

[20] A. Wald. Sequential Tests of Statistical Hypotheses. *The Annals of Mathematical Statistics*, 16(2):117 – 186, 1945.

[21] Kenny C. Gross and Keith E. Humenik. Sequential probability ratio test for nuclear plant component surveillance. *Nuclear Technology*, 93(2):131–137, 1991.

[22] R. Chen, J.-M. Park, and K. Bian. Robust distributed spectrum sensing in cognitive radio networks. In *IEEE INFOCOM 2008 - The 27th Conference on Computer Communications*, pages 1876–1884. IEEE Press, 2008.

[23] Palash Dey and Arnab Bhattacharyya. Sample complexity for winner prediction in elections. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, AAMAS '15, page 1421–1430. International Foundation for Autonomous Agents and Multiagent Systems, 2015.

[24] Arun Rajkumar, Suprovat Ghoshal, Lek-Heng Lim, and Shivani Agarwal. Ranking from stochastic pairwise preferences: Recovering condorcet winners and tournament solution sets at the top. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 665–673. PMLR, 2015.

[25] Aurélien Garivier and Emilie Kaufmann. Optimal best arm identification with fixed confidence. In *29th Annual Conference on Learning Theory*, volume 49 of *Proceedings of Machine Learning Research*, pages 998–1027. PMLR, 2016.

Checklist

The checklist follows the references. Please read the checklist guidelines carefully for information on how to answer these questions. For each question, change the default **[TODO]** to [Yes] , [No] , or [N/A] . You are strongly encouraged to include a **justification to your answer**, either by referencing the appropriate section of your paper or providing a brief inline description. For example:

- Did you include the license to the code and datasets? [Yes] See Section [a].
- Did you include the license to the code and datasets? [No] The code and the data are proprietary.
- Did you include the license to the code and datasets? [N/A]

Please do not modify the questions and only use the provided macros for your answers. Note that the Checklist section does not count towards the page limit. In your paper, please delete this instructions block and only keep the Checklist section heading above along with the questions/answers below.

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See section 4.1.
   (c) Did you discuss any potential negative societal impacts of your work? [No] We make an algorithmic contribution to an abstract, well-established problem. We do not perceive any negative societal impact.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes] Proofs are the appendix.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] in the supplemental material
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Appendix C, D
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] code, proofs, and implementation details in the supplemental material
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]