Earning While Learning: How to Run Batched Bandit Experiments

Jan Kemper
University of Mannheim, ZEW
jan.kemper@zew.de

Davud Rostam-Afschar
University of Mannheim, IZA, GLO
rostam-afschar@uni-mannheim.de

Abstract. Researchers collect data in most experiments not all at once but sequentially over a period of time. This allows to observe outcomes early and adapt the treatment assignment to reduce the costs of inferior treatments. This article discusses multi-armed-bandit-type adaptive experimental designs and algorithms for balancing exploration of treatment effects and exploitation of better treatments. By design, bandits break usual asymptotics and make inference difficult. We show how a batched bandit design allows for valid confidence intervals and compare coverage of the Batched Bandit estimator in Monte Carlo simulations. We introduce **bbandits** in Stata, a tool for easily running Monte Carlo simulations to assist the design and implementation of experiments before data are collected, interactively running own bandit experiments, and analysing adaptively collected data. **bbandits** implements three popular treatment assignment algorithms: ε-first, ε-greedy, and Thompson sampling. **bbandits** facilitates estimation, inference, and visualization.

JEL Classification: C1, C11, C12, C13, C15, C18, C8, C87, C88, C9, D83

Keywords: Randomized controlled trial, causal inference, multi-armed bandits, experimental design, machine learning

1 Introduction

One of the most popular research designs is to randomly assign a fixed share of participants to one of several treatment arms with the goal to learn about the treatment effects or to select the best treatment. Such randomized controlled trials are considered the gold standard for causal inference but often come with prohibitively high costs. If the treatments aim to maximize some outcome of participants like productivity, health, or wealth, cost such as financial or time losses, forgone opportunities, or ethical costs may arise with a one-shot experiment, because some participants would receive a treatment that may be suboptimal more often than necessary to learn about its effect size. For example, a drug that reduces the probability of death more effectively could be chosen with higher frequency, if based on observations collected so far. Other examples include experiments to find optimal dosages for pharmaceuticals, most effective labor market programs, optimal schooling track choices, best tax policies, optimal portfolio choice, selecting the best ad campaigns, hiring best applicants, supply chain optimization, selecting best pricing policies, etc. In such cases, researchers ex-post regret having assigned an inferior treatment but could not know which treatment is inferior ex-ante. Since researchers typically collect data from experiments not all at once but sequentially or in batches over a period of time, they face a trade-off between exploring the treatment effects and exploiting knowledge from previous observations about the performance of the arms. The methods discussed in this paper allow to assign treatments to participants in a data-driven way, balancing earning and learning.

In settings, where treatments are costly to carry out or participants are costly to recruit, traditional inference with fixed and balanced treatment assignment may be hardly feasible because of a too low number of observations per treatment arm, precluding the researcher to explore treatment arms that appear less promising a priori. For example, for testing effect sizes of two or more methods for heart surgery there are often only few observations available. Even if interest is only in testing treatment effect sizes ex-post, the most common constraint is a fixed budget or a fixed time horizon that limits the number of treatment arms. For conclusive causal inference with sufficiently many observations per arm, researchers thus need to preselect a set of treatments, which may be undesirable already in vignette studies with several hundred profiles and even more severe in settings like testing for dosages of pharmaceuticals or ad campaigns, where the number of treatments is typically in the millions or infinitely many. Therefore, researchers may require conclusive results from causal inference to be obtained faster than in traditional experimental designs and a data-driven selection of treatments instead of a preselection based on assumptions.

Sampling algorithms to minimize a measure of regret in the exploration-exploitation trade-off have been studied in the statistical and machine learning literature as part of the broader concept of reinforcement learning, and increasingly often also in medicine (Lei et al. 2022), economics (Camerer et al. 2024; Hirano and Porter 2023; Chen and Andrews 2023; Kasy and Sautmann 2021; Hadad et al. 2021; Avivi et al. 2021; Athey and Imbens 2019) political science (Offer-Westort et al. 2021), survey methods research (Gaul et al. 2024), education (Rafferty et al. 2019), psychology (Schulz et al. 2020)

etc. The advantages of such algorithms have also led to a rapid growth of interest by practitioners (Hill et al. 2017; Scott 2015; Agarwal et al. 2014; Chapelle and Li 2011; Scott 2010; Graepel et al. 2010). Most sequential experimental designs can be thought of as multi-armed bandits. The prime example to illustrate multi-armed bandit problem is a collection of slot machines, each with a different but unknown fixed payoff, where a player that wants to maximize her wealth must play, repeatedly, to learn about the slot machine with the highest success probability and to exploit her knowledge. Equivalently she may minimize her regret of having played a slot machine with non-optimal reward.

In this paper, we discuss the trade-off between exploration and exploitation using popular algorithms for sequential treatment assignment, including ε-first, ε-greedy, and Thompson sampling. Compared to traditional experimental designs, optimizing sampling algorithms introduce bias and standard hypothesis tests become invalid due to unequal assignment of participants to the treatment arms. Obtaining correct estimates and standard errors for causal inference is therefore not straightforward (Zhang et al. 2021; Bibaut et al. 2021). We study data from sequential experiments conducted in batches and show how correct estimation and causal inference can be carried out with the batched ordinary least squares (BOLS) regression. In a Monte Carlo study we compare the BOLS estimator to the ordinary least squares (OLS) estimator for configurations with few and many observations per batch, different number of batches, and various treatment effects sizes.

These Monte Carlo simulations reflect cases often encountered by applied researchers. We show that a batched adaptive experiment requires sufficiently many observations per batch to obtain unbiased estimates. The number of batches is less important. In a two arm scenario at least 20 observations per batch were necessary to come close to an unbiased estimate. Furthermore, standard asymptotic OLS inference is invalid for small effect margins because then the OLS estimator is asymptotically not normally distributed which leads to overrejections. By small margin we mean effect sizes which are close to zero with substantial noise. The BOLS estimator that we analyze in this article can fix this problem but is inefficient in large margin cases, that is, standard errors are larger in comparison to the OLS standard errors. Our Monte Carlo simulations suggest that in practice, though, the overrejection rates of the OLS estimator are not too far from the true rejection rates. OLS also performs well in large margin cases. Both OLS and BOLS are consistent estimators and their point estimates are similar, even in small samples. For applied researchers, we recommend to always compute both the OLS and the BOLS estimates and their corresponding confidence intervals. Our Monte Carlo simulations suggest to rely on BOLS inference in the small margin case and on OLS inference in the large margin case.

We introduce the new Stata command **bbandits** for batched bandit experiments. **bbandits** provides simple routines for simulating, interactively running, and analysing batched bandit experiments. This includes Monte Carlo simulations of treatment assignments with ε-first, ε-greedy, and Thompson sampling to assist the design and implementation of adaptive experiments. With **bbandits** researchers can implement their own batched bandit experiments. **bbandits** provides valid statistical inference and correct coverage and a wide range of statistics and illustrations to analyse adaptively

collected data. A number of examples illustrate how to use **bbandits**.

In the next section, we introduce the epsilon-greedy and Thompson sampling algorithms and describe the data structure of batched experiments. In Section 3 , we discuss batched regression as a method for correct causal inference. In Section 4, we show results from Monte Carlo simulations to assess the properties of the BOLS. In Section 5, we describe the syntax and options of the new **bbandits** command and provide a step-by-step guide for how to run batched bandit experiments. We provide empirical applications in Section 6 and show how to run own bandit experiments with **bbandits** in section 7. Section 8 concludes.

2 Methods

2.1 Adaptive experimental designs and algorithms

The simplest version of a discrete (Bernoulli) multi-armed bandit problem is the choice between treatments with stochastic, stationary, independent and identically distributed (i.i.d.) rewards with no contextual information or covariates. Given are k treatment groups, an outcome variable R, and an indicator for the batch t after which treatment groups can be assigned based on R from period $t-1$. Consider a policy maker that asks a researcher to design an experiment to find out which of several information campaigns about vaccines effectively reduces infections. The question is now how to learn which arm reduces infections most effectively without exploring treatment arms that are less effective too often.[1]

Common algorithms to design experiments with exploration-exploitation trade-off are epsilon-first, epsilon-greedy and Thompson sampling algorithms (see Burtini et al. (2015) for an overview).

ε-first. Epsilon-first is widely known as A/B testing and often applied to two-armed bandits. The first share of trials epsilon serve as exploration or burn-in phase. In this phase, researchers assign an equal share of participants to each arm and estimate each arm's outcome serving as a prediction for future expected outcomes. In the remaining $(1-\varepsilon)$ share of trials, the exploitation phase, only the arm with the best empirical estimate is selected.

ε-greedy. The epsilon-greedy approach is another very common strategy that explores for the entire number of trials of the experiment. For a constant or decreasing share $(1-\varepsilon)$ of all trials the best treatment is selected and for the remaining trials, all treatment arms are randomly assigned with equal probability. Even after having learned about the average outcome of each arm, constant epsilon-greedy continue to explore some epsilon fraction of the trials and thus asymptotically do not converge to the optimal

1. Such an algorithm is greedy in the sense that an action is chosen solely to maximize immediate reward.

arm. To alleviate this problem one can decrease the exploration share ε for each batch.

An ε-first strategy is superior to an ε-greedy strategy when the total number of trials of the experiment is fixed and known ex-ante and rewards are stationary, since it is based on a larger exploration phase. ε-first is, however, more vulnerable to non-stationary of the reward distribution, because the entire exploration is limited to the beginning of the experiment.

Thompson sampling. A classical heuristic known as Thompson sampling was developed for allocating experimental effort in bandit problems arising in clinical trials (Thompson 1933, 1935). Thompson sampling models uncertainty about the shape of the distribution from which the observed outcomes R are drawn and the expected outcome R explicitly.

Figure 1 shows the cumulative reward of playing the optimal choice from the beginning compared to Thompson sampling and to a traditional experimental design with fixed and balanced arms. Thompson sampling sacrifices some reward to explore the reward distributions but much less than in a traditional experiment.

Thompson sampling is also a principled way to model learning, since the distribution (and not only the expectation) is updated according to Bayes' rule as observations are gathered. Assuming a parametric distribution over a parameter set of success rates θ, we can compute the probability at time t of a given arm k providing optimal reward as

$$P(E_\theta[R_k,t] = \max\{E_\theta[R_1],\ldots,E_\theta[R_K]\}|R_t) = \int_0^1 \mathbb{1}\left[S_t = \arg\max_{k=1,\ldots,K} E_\theta[R_k,t]\right] P(\theta|R_t)d\theta,$$

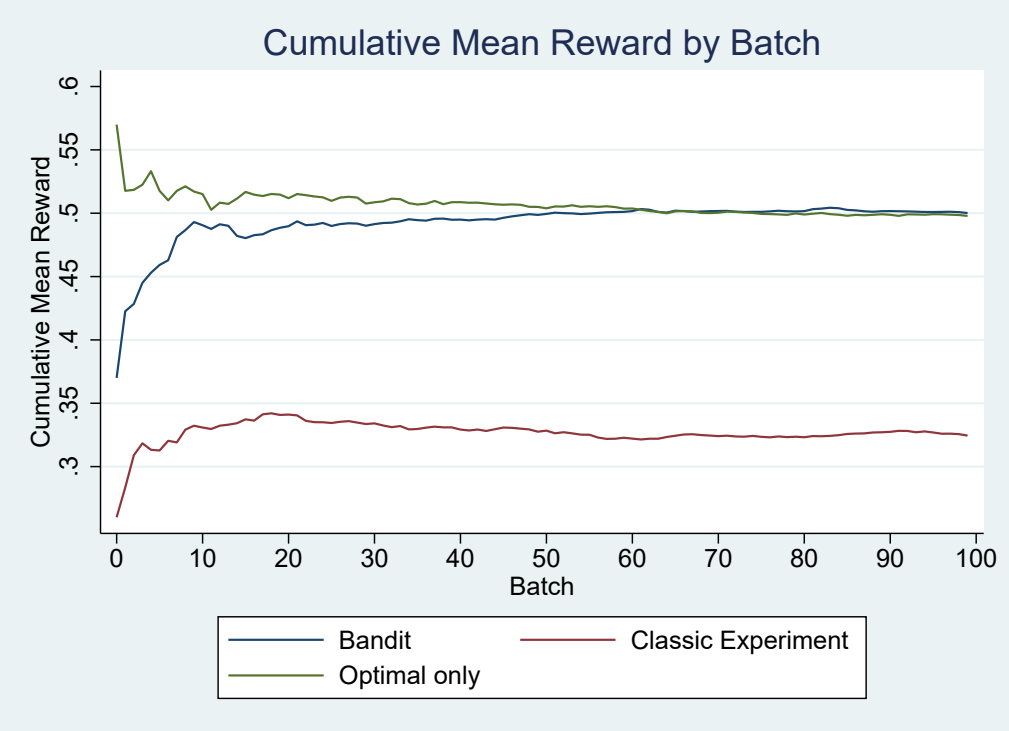
where S_t is the single arm at trial t, such that the expected reward given the sampled parameters and treatment is maximized.

It is convenient to work with beta distributions where $B(R_{k,t}|\alpha_k,\beta_k)$ denotes the density of the beta distribution for random variable R_t with parameters α_k and β_k. This beta distribution has a mean $\alpha_k/(\alpha_k+\beta_k)$ and variance $\frac{\alpha_k\beta_k}{(\alpha_k+\beta_k)^2(\alpha_k+\beta_k+1)}$, and the distribution becomes more concentrated as $\alpha_k+\beta_k$ grows. An advantage of the beta distribution is that it has a conjugate prior, that is, each treatments' posterior distribution $P(\theta|R_t)$ is also beta with parameters that can be updated according to a simple rule:

$$(\alpha_k,\beta_k) = \begin{cases} (\alpha_k,\beta_k) & \text{if chosen arm} \neq k, \\ (\alpha_k,\beta_k) + (R_t, 1-R_t) & \text{if chosen arm} = k. \end{cases} \tag{1}$$

Starting from $\alpha_k = \beta_k = 1$, where the prior is uniform over $[0,1]$ reflecting no prior knowledge about the reward probability. α_k or β_k increases by one with each observed success or failure, respectively, such that the distribution becomes more peaked with each trial. Rather than computing the integral, simply sampling once from the estimated reward distribution at each round for each arm and select the arm with the

Figure 1: Cumulative Mean Reward



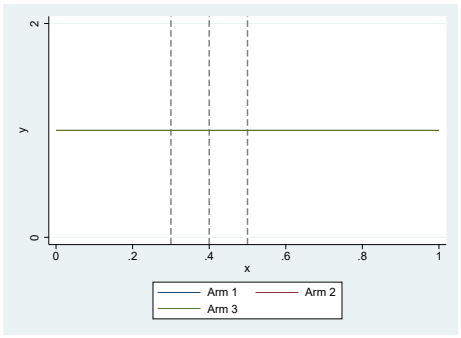
Notes: The simulated experiment consists of three arms with the following true success rates: Arm 1 = 0.5, Arm 2 = 0.4, Arm 3 = 0.3. The Bernoulli Thompson Sampling algorithm with a clipping rate of 0.05 and a decay rate of 0.9 was applied. The simulation for the figure is generated by the following command: `bbandit_sim 0.5 0.4 0.3 , size(100) batch(100) clipping(0.05) decay(0.9) Thompson`.

highest reward draw is sufficient, because the distribution will be peaked (Thompson 1933; Agrawal and Goyal 2013; Wang and Chen 2018; Perrault et al. 2020).[2]

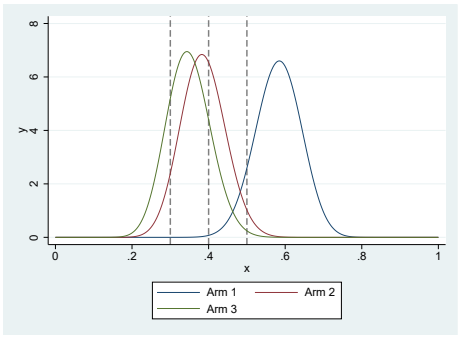
Figure 2 shows how the reward distribution for a given arm becomes more peaked with additional batches. Kalkanli and Ozgur (2020) show that if run for sufficiently large number of batches, the Thompson sampling algorithm can be used to discover the optimal treatment.

2. Note that for pure exploration, i.e. to find the best arm as fast as possible, Thompson sampling needs to sample several sets of samples, in order to be informative about whether there is an arm that is the optimal one with high probability and which arm needs exploration (Wang and Zhu 2022).

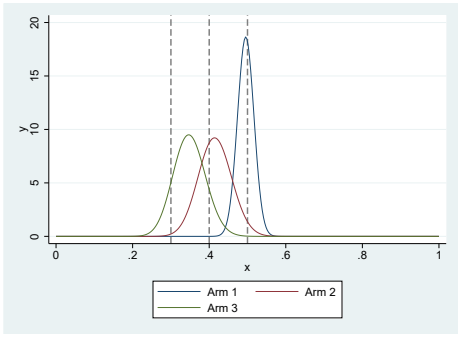
Figure 2: Treatment assignment under Beta-Bernoulli Thompson sampling



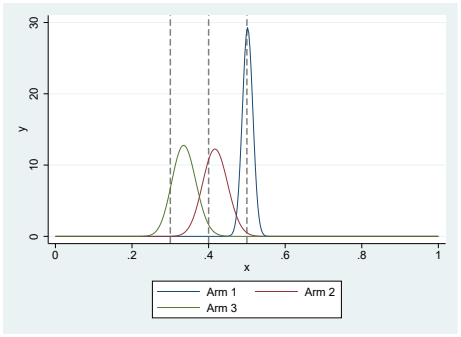
(a) Batch 1



(b) Batch 2



(c) Batch 5



(d) Batch 10

Notes: Thompson sampling with Bernoulli-beta-reward distributions with true success rates of 0.5 for arm 1, 0.4 for arm 2, and 0.3 for arm 3. The figure shows that the uncertainty with which the success rate is estimated becomes smaller. The vertical line indicates the true success rate. The figure was generated running `bbandit_sim 0.5 0.4 0.3, size(200) batch(10) clipping(0.1) Thompson plot_Thompson`.

2.2 Data structure of adaptive experiments

Given are two treatment groups $k \in \{A, B\}$ with individuals i, an indicator for the batch t, and an outcome variable $R_{i,t}$. Treatment groups are assigned based on R from period $t-1$. For example, think of an information campaign that informs individuals that are assigned to arm A about vaccines and those assigned to arm B to receive no information about vaccines, weeks of the information campaign as batches, and infection with disease as outcome variable. One key question now is whether the mean outcome difference, also called margin, is large enough to statistically differ from the arm that provides no information. Further information on the derivation of the formula can be found in appendix 11.

The regression equation for each batch is then:

$$E_t[R_{i,t}] = \beta_0 + \Delta_t \times \mathbb{1}_{k,t} + u_{i,t},$$

where Δ_t is an unbiased and consistent estimate of the margin in each batch t. u_t is an error term.

The adaptive assignment of treatments to participants makes the aggregation of the batchwise margins difficult. An OLS estimator that pools batches can lead to asymptotic non-normality and, thus, to invalid statistical tests (Zhang et al. 2021).[3]

Based on the results for a large class of bandit algorithms from Zhang et al. (2020), we propose the batched OLS (BOLS) estimator that weights batchwise estimates such that the aggregate margins are consistent and asymptotically normally distributed.

$$\Delta^{\text{BOLS}} = \frac{\sum_t^T \omega_t \times \Delta_t}{\sum_t^T \omega_t} = \sum_{t=1}^{T} \gamma_t \Delta_t,$$

where $\omega_t = \sqrt{\frac{N_{t,k} \times N_{t,b}}{N_{t,k} + N_{t,b}}}$ and $\gamma_t = \frac{\omega_t}{\sum_{t=1}^T \omega_t}$.

Δ_t is just the OLS estimator for the margin between two arms in each batch t. As the observations are independently drawn within each batch, standard OLS with all its useful properties can be applied. N_k is the number of observations of arm k, while N_b is the number of observation of the baseline arm b. The weight ω_t becomes bigger if both arms have more equal shares of observations. γ_t is the normalized weight which is between 0 and 1 and adds up to one when summed over all t. It captures how much Δ_t contributes to Δ^{BOLS}. It shows that Δ^{BOLS} is a weighted average of the batch OLS coefficients.[4]

Intuitively, this estimator puts higher weight on batches where both treatment arms are more balanced. Vice versa, batches where one treatment arm dominates receives less weight. Hence, batches with lots of new information on the treatment effect are

3. In the Monte Carlo study in section 4, we show that when the batch size is sufficiently large the estimates from Batched OLS and OLS are both consistent.
4. The **bbandits** command also returns these weights and batchwise estimated OLS coefficients.

Table 1: Stylized data structure. Here $k = A$ and $b = B$.

Obs	Selected arm	Batch	Reward	True Expected Reward	OLS	Batch-Wise OLS	ω_t
1	A	0	0	0.5	0.600	0.500	$\sqrt{\frac{2\times2}{2+2}}$
2	B	0	0	0.2	0.167	0.000	$\sqrt{\frac{2\times2}{2+2}}$
3	A	0	1	0.5	0.600	0.500	$\sqrt{\frac{2\times2}{2+2}}$
4	B	0	0	0.2	0.167	0.000	$\sqrt{\frac{2\times2}{2+2}}$
5	A	1	0	0.5	0.600	0.500	$\sqrt{\frac{2\times2}{2+2}}$
6	B	1	1	0.2	0.167	0.500	$\sqrt{\frac{2\times2}{2+2}}$
7	A	1	1	0.5	0.600	0.500	$\sqrt{\frac{2\times2}{2+2}}$
8	B	1	0	0.2	0.167	0.500	$\sqrt{\frac{2\times2}{2+2}}$
9	A	2	0	0.5	0.600	0.667	$\sqrt{\frac{1\times3}{1+3}}$
10	A	2	1	0.2	0.600	0.667	$\sqrt{\frac{1\times3}{1+3}}$
11	A	2	1	0.5	0.600	0.667	$\sqrt{\frac{1\times3}{1+3}}$
12	B	2	0	0.2	0.167	0.000	$\sqrt{\frac{1\times3}{1+3}}$
13	A	3	1	0.5	0.600	0.667	$\sqrt{\frac{1\times3}{1+3}}$
14	A	3	0	0.2	0.600	0.667	$\sqrt{\frac{1\times3}{1+3}}$
15	A	3	1	0.5	0.600	0.667	$\sqrt{\frac{1\times3}{1+3}}$
16	B	3	0	0.2	0.167	0.000	$\sqrt{\frac{1\times3}{1+3}}$
OLS					$\widehat{\text{Reward}} = 0.6 - 0.433 \times \mathbb{1}_{\text{arm B}}$		
BOLS					$-0.443 = \frac{0.5\times1+0\times1+\sqrt{\frac{1\times3}{1+3}}\times0.667+\sqrt{\frac{1\times3}{1+3}}\times0.667}{1+1+\sqrt{\frac{1\times3}{1+3}}+\sqrt{\frac{1\times3}{1+3}}}$		
					$\widehat{\text{Reward}} = 0.6 - 0.443 \times \mathbb{1}_{\text{arm B}}$		

Note: The table shows fictional data in a stylized structure for adaptive experiments. There are two arms A and B. The first batch is batch 0 where the arms would be randomly assigned. Afterwards treatment would be assigned according to one of the bandit algorithms (e.g. Thompson sampling).

up-weighted. Zhang et al. (2021) show that this procedure yields asymptotic normality although the observations which are drawn by a bandit algorithm are not independent anymore.

In table 1 a stylized data structure of a batched adaptive experiment is shown. The columns 1-4 show the standard data structure of a batched adaptive experiment which is also the data structure required for the **bbandits** command. The "Selected arm" column just indicates which of the arms was played, the batch variable captures for each row the current batch and the reward is the realized outcome after this arm was played. In this simple example there are two arms A and B. The expected reward of arm A is 0.5 while the expected reward of arm B is 0.2. There are four batches starting with batch 0. The data generating process is adaptive because the treatment assignment changes based on the observations in the previous batches. The column *Batch OLS* shows the OLS coefficient calculated for each batch. Because the observations are independently drawn within each batch, OLS estimates for each batch are consistent and asymptotically normally distributed. In the next column entitled *OLS*, the pooled OLS regression coefficient is presented. In the final column the aggregate BOLS estimator is presented which is a weighted sum of the batched OLS estimates.

3 Inference about causal effects

From the batched OLS (BOLS) test statistic for K arms and T batches for comparing arm k against arm b (Zhang et al. 2020), we can construct confidence intervals:

$$Pr\Big(\Delta^{\text{BOLS}} - w \times c \times \sigma \leq \mu \leq \Delta^{\text{BOLS}} + w \times c \times \sigma\Big) = 1 - \alpha,$$

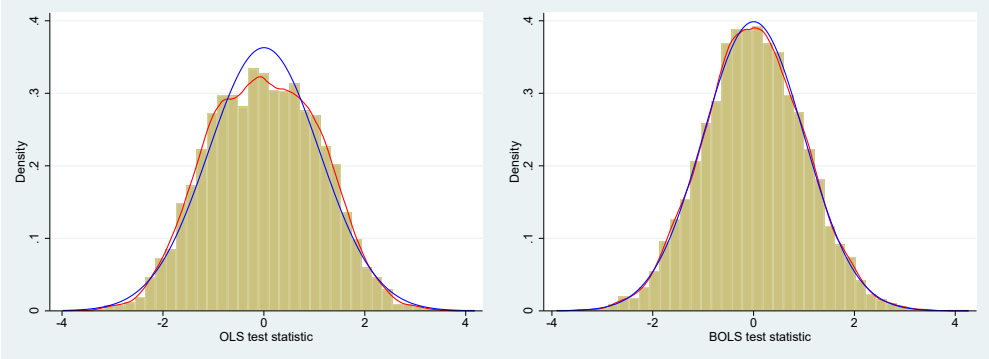
where Δ^{BOLS} is the weighted estimated marginal effect, that is, the differences in the average reward probability between comparison arm k and baseline arm b of OLS regressions for each batch t, μ is the hypothesized difference between means of the arms, c is a critical value, that is, for a $1 - \alpha = 95\%$ significance level, the $1 - \alpha/2 = 97.5$th percentile of the distribution Pr. σ reflects the sampling error, and w is a weight correcting for adaptive sampling, i.e. unequal sampling probabilities. The weight is

$$w = \frac{\sqrt{T}}{\sum_{t=1}^{T} \omega_t},$$

where T is the total number of batches, and ω_t includes the number of times that comparison arm k was played as assigned by the algorithm $N_{t,k}$, and the number of times that baseline arm b was played as assigned by the algorithm $N_{t,b}$.

The BOLS estimate Δ_t is asymptotically normally distributed (Zhang et al. 2020), implying that in each batch Δ_t is a consistent estimate of the true Δ_t. Figure 3 compares the BOLS test statistic to the corresponding OLS test statistic in a Monte Carlo simulation, where the margin is zero for Thompson sampling. OLS inference leads to an

Figure 3: OLS and BOLS under Beta-Bernoulli two-arm Thompson Sampling with batch size $N_t = 100$ at batch $t = 25$



(a) Empirical distribution of standardized OLS estimator for the margin

(b) Empirical distribution of standardized BOLS estimator for the margin

Notes: The results are based on a Monte Carlo simulation with 10000 trials. Both treatment arms have have a true expected value of 0.5, hence the margin is zero ($\beta_1 = 0$). The figure was generated running `bbandit_sim 0.5 0.5, monte_carlo Thompson clipping(0.1) n(10000)`.

overrejection of the null hypothesis because more mass is in the tails. In the Appendix in Figure 10 the empirical distribution for the Epsilon Greedy algorithm is presented, which also shows that OLS is not normally distributed. If there are no reasons, e.g., nonstationarity of rewards, for Δ_t to change over batches, then $\Delta_t \to \Delta^{\text{BOLS}}$.[5]

An estimate for σ_t is obtained from the residuals of regressions of the treatment indicators on reward indicators for each batch with degrees of freedom correction $(N_{t,k} + N_{t,b}) - 2$. Zhang et al. (2020) show that $\hat{\sigma}_t$ is a consistent estimate of σ^2.

With two arms, the distribution Pr is the Student-t distribution with $n-2$ degrees of freedom that can be obtained by simulation of draws from the Student-t distribution.

In practice, the confidence intervals can be well approximated by

$$CI = \Delta^{\text{BOLS}} \pm w \times 1.96 \times \hat{\sigma}.$$

5. The noise variance σ^2 for two arms for the pooled OLS is $\hat{\sigma}^2 = \frac{1}{nT-2} \sum_{t=1}^{T} \sum_{i=1}^{n} \left(\varepsilon_k - \varepsilon_b\right)^2$, where ε_k is the residual of a regression of the outcome variable on an indicator that is one for arm k and zero for arm b. For the batched OLS it is $\hat{\sigma}_t^2 = \frac{1}{n-2} \sum_{i=1}^{n} \left(\varepsilon_k - \varepsilon_b\right)^2$. This reflects variance within batches only and for large n is consistent, such that $\hat{\sigma}_t^2 \xrightarrow{P} \sigma^2$.

4 Properties of Batched Bandit estimators: Evidence from Monte Carlo simulations

Nie et al. (2018) show that sample means from adaptively collected experiments are biased because of the complex dependencies introduced by the adaptive nature of the data generating process. Zhang et al. (2020) show that for *batched* bandits asymptotic normality and consistency can be proven which indicates that a batched approach mitigates the problems of unbatched bandits. The underlying reason is that the batched setting better resembles independent sampling procedures because within each batch the data is drawn independently. Within batch independence implies that all useful properties of sample means (e.g. central limit theorems) can be applied for sample mean estimation within the batch. This additional structure is used by Zhang et al. (2020) to prove asymptotic normality for across batch estimation. Nonetheless, they also show (see above) that the standard OLS estimator is non-normally distributed for small margins. Despite its practical relevancy, relatively little is known about the small sample properties of the OLS and the BOLS estimator under adaptively collected data.

To fill this gap, we ran a Monte Carlo simulation where we simulated relevant scenarios for applied researchers and compare the performance of the OLS and the BOLS estimator. Specifically, we simulated 10,000 iterations for each scenario. In our first setting, we simulate experiments with few observations per batch but with many batches. In our second setting, we examine data generating processes with few batches but a relatively high number of observations per batch. Each sample is drawn from a Bernoulli distribution and the Bernoulli Thompson sampling algorithm with varying clipping rates is applied.[6] There are only two arms and the outcome of interest is the difference between the arms (e.g. treatment vs. control group). In our first simulation study, we vary the number of batches between 10 and 100 and increase the sample size per batch from 5 to 100. As Zhang et al. (2020) have shown, the effect margin plays an important role for inference. Therefore, we simulate a 0, 0.1 and 0.2 margin.

4.1 Bias

When only one observation per batch is available the experiment corresponds to the standard multi-armed bandit approach which leads to biased estimates (Nie et al. 2018). In this simulation, we examine how OLS and BOLS perform when the batch size is very small under various numbers of batches. The results of the simulation are presented in table 4. The presented estimates are the means of the OLS/BOLS coefficients over 10,000 simulated iterations. In parentheses are the type-I error rejection rates for the 95 percent significance level. If the estimators were asymptotically standard normally distributed, the H0 hypothesis would be rejected 5 percent of the time even though the H0 is correct.

There are two main results. First, the bias of the unbatched multi-armed bandit

6. In the Appendix in table 6 we report the results for the epsilon-greedy algorithm where the reward is drawn from a normal distribution.

algorithms is also prevalent in the batched case with few observations per batch. This can be seen in the second row of table 4. With only 5 observations per batch the estimates are biased for the margin 0.1 and 0.2 cases. The coefficient value does not correspond to the true margin of 0.1 or 0.2. On average, the estimates are about 30 to 40 percent off. Both OLS and BOLS approaches are affected by the bias. The bias originates from the violation of the independence assumption (Nie et al. 2018). Second, inference should not be trusted with small numbers of observations per batch. In the first row, the rejection rates deviate significantly from the expected 0.05, indicating that the normality assumption is not warranted. Increasing the number of batches does not solve the problem which can be seen in the high deviations from 0.05 for the cases with 100 batches and only 5 observations per batch. In a standard setting 500 observations (5 per 100 rounds) is in many cases sufficient to approximate a normal distribution but in an adaptive setting it is not, because asymptotics go over large N_t and not T. The results show that a sufficient number of observations per batch is required. The underlying reason for why the number of observations in each batch rather than the total number of batches is decisive, is that the observations within batches are independent, while there are complex dependencies between batches. With a batch size of 20 the bias becomes smaller and the type-I error rates are closer to 0.05. We conclude that 20 observations per batch is a minimum requirement in a two arm scenario in the Bernoulli case.

To summarize, the simulation demonstrates that an adaptive experiment requires sufficiently many observations per batch to obtain unbiased estimates. The number of observations per batch, rather than the number of batches, is decisive.

4.2 Inference

In our second simulation study, we vary the number of batches between 1 and 8 and increase the sample size per batch from 100 to 1000. We simulate a 0, 0.1 and 0.2 margin.

In table 3 the results of the second simulation study are presented. The first column N_t indicates the number of observations per batch. For instance, in a clinical trial, the experiment might involve an initial batch of 100 participants, with some receiving the treatment and others serving as control units, followed by subsequent batches of 100 participants in successive phases of the trial. In the second column, $\Delta[R]$ is the true difference between the two arms. The following four columns present the results for 1, 3, 5 and 8 batches. The parameter of interest is the mean of the 10,000 calculated differences between the two arms ($\hat{\Delta}$). In the parenthesis below are the type-I error rates which indicate how often the H0 hypothesis was rejected on the 95 percent level. Under normality the H0 should be rejected in 5 percent of the times. The four columns on the right show the same results for the BOLS estimator.

The first observation is that the OLS and the BOLS estimator are equal when there is only a single batch. In this case, there is no adaptive sampling and thus, also no weighting of the different batches. In this setting, the standard OLS procedure is

Table 2: Bernoulli Thompson Sampling - Clipping 0.1

		OLS/Batch				BOLS/Batch			
N_t	$\Delta[R_{t,k}]$	10	25	50	100	10	25	50	100
5	0.0	0.003	-0.002	0.000	-0.001	0.003	-0.003	0.001	-0.001
		(0.134)	(0.146)	(0.139)	(0.140)	(0.006)	(0.004)	(0.005)	(0.003)
5	0.1	0.130	0.141	0.145	0.145	0.141	0.148	0.149	0.147
		(0.143)	(0.130)	(0.118)	(0.111)	(0.012)	(0.009)	(0.008)	(0.009)
5	0.2	0.260	0.268	0.270	0.268	0.280	0.281	0.276	0.271
		(0.140)	(0.124)	(0.108)	(0.106)	(0.013)	(0.009)	(0.007)	(0.003)
10	0.0	-0.001	-0.001	0.000	0.001	-0.001	-0.001	0.000	0.001
		(0.082)	(0.070)	(0.068)	(0.064)	(0.055)	(0.054)	(0.050)	(0.048)
10	0.1	0.114	0.113	0.111	0.106	0.107	0.104	0.104	0.102
		(0.078)	(0.064)	(0.058)	(0.048)	(0.060)	(0.054)	(0.055)	(0.052)
10	0.2	0.22	0.211	0.205	0.202	0.208	0.202	0.201	0.200
		(0.061)	(0.048)	(0.051)	(0.050)	(0.061)	(0.063)	(0.069)	(0.068)
20	0.0	0.000	0.000	-0.000	-0.000	0.001	0.000	-0.000	0.000
		(0.072)	(0.067)	(0.064)	(0.062)	(0.054)	(0.051)	(0.049)	(0.048)
20	0.1	0.113	0.110	0.106	0.103	0.106	0.105	0.102	0.101
		(0.063)	(0.053)	(0.047)	(0.045)	(0.053)	(0.053)	(0.054)	(0.056)
20	0.2	0.212	0.204	0.201	0.201	0.206	0.200	0.200	0.200
		(0.054)	(0.048)	(0.05)	(0.05)	(0.065)	(0.064)	(0.070)	(0.069)
100	0.0	-0.001	-0.000	0.000	-0.000	-0.001	-0.000	0.000	0.000
		(0.067)	(0.062)	(0.063)	(0.06)	(0.054)	(0.048)	(0.048)	(0.049)
100	0.1	0.105	0.102	0.100	0.100	0.102	0.101	0.100	0.100
		(0.048)	(0.041)	(0.049)	(0.045)	(0.053)	(0.046)	(0.055)	(0.049)
100	0.2	0.201	0.201	0.200	0.200	0.200	0.200	0.200	0.200
		(0.052)	(0.051)	(0.051)	(0.047)	(0.069)	(0.072)	(0.069)	(0.063)

Note: The Monte Carlo simulation is based on 10000 iterations. In the parenthesis is the Type-I error (α) for the 95-percentile. The true 95-percentile of the standard normal distribution implies a rejection rate of 0.05. $\Delta[R_{t,k}]$ is the true margin between the two arms.

consistent.[7] Second, in the zero margin case, when we increase the number of batches, the rejection rate increases to 0.072. In our Bernoulli simulation, it is not a severe deviation but it shows the issues of the non-normality of the OLS estimator. The H0 hypothesis is rejected too often because an asymptotic standard normal distribution is assumed but this assumption is false. This can be seen by analyzing the zero margin case for a large number of observations. Even when every batch contains 1000 observations and there are 8 batches, the rejection rate does not converge to the correct value of 0.05. The BOLS estimator resolves this problem and reports correct rejection rates which stem from the fact that the normality assumption is asymptotically correct for the BOLS estimator. Also for small sample cases, like the 100 observations per batch in the first row, the normality assumption is supported by the correct type-I rejection rates. In line with the results in Zhang et al. (2021), the OLS inference is correct for the case where the margin is not close to zero.[8] For the BOLS in the Thompson-Bernoulli simulation with a large margin, convergence seems to be a bit slower with the small clipping size of 0.05 as there is a minor over rejection rate. With a higher clipping rate the the rejection rate is closer to 0.05 (see table 5) which shows that the clipping rate is an important factor for inference. In general, a higher clipping rate leads to more exploration and hence also to more informative signals, which can be used in the estimation of poorly performing treatment arms. In the Appendix in table 6 Monte Carlo simulation results for the same setting with the epsilon-greedy algorithm with an epsilon rate of 0.2 are presented. The results are similar. There is a minor over-rejection of about 0.005 when the margin is zero. Although, the over rejection is minor, the distribution of the OLS estimator under epsilon-greedy is clearly not normally distributed. BOLS fixes the issue and performs well in all settings. When the margin is not zero, the OLS estimator is normally distributed and standard OLS inference is valid. BOLS and OLS are consistent.

We report standard errors in parentheses in table 7 and table 8 in the appendix. The results suggest that OLS yields more precise estimates, since the OLS standard errors are smaller compared to those of the BOLS estimates.

Regarding inference in multi-armed bandit experiments, these results have the following implications for empirical researchers. When the margin is close to zero, asymptotic normality is an incorrect assumption for the OLS estimates and the resulting inference is based on the wrong distributions. In practice, for hypothesis testing, the deviations were relatively small from the correct rejection rates in the simulated settings. Nonetheless, we recommend using the BOLS inference in small margin cases. For large margins, the OLS estimates are normally distributed and standard OLS inference is valid. Furthermore, the OLS standard errors are smaller. To keep it consistent, we recommend calculating both OLS and BOLS confidence intervals in all settings.

Regarding statistical consistency, all OLS and BOLS estimators in all analyzed finite sample settings seem to be consistent. All estimates are very close to the true margin. The bias problem of the unbatched multi-armed bandits algorithms do not occur in this

7. Small deviations come from rounding errors in the estimation.
8. Additionally, we conducted a Jarque-Bera test and normality could not be rejected anymore.

batched setting with a batch size of at least 100. The reported small sample results provide evidence that a batched approach helps to circumvent the multi-armed bandit bias problem. Hence, batching, with a large enough batchwise sample, might be a straightforward and feasible solution to address small sample bias and at the same time allow for correct inference. The downside of the batched approach is that the improved statistical properties come at the costs of fewer possibility to maximize rewards.

Table 3: Bernoulli Thompson Sampling - Clipping 0.05

		OLS/Batch				BOLS/Batch			
N_t	$\Delta[R]$	1	3	5	8	1	3	5	8
100	0.0	-0.000	-0.000	-0.000	-0.001	-0.000	-0.001	-0.000	-0.001
		(0.055)	(0.061)	(0.071)	(0.072)	(0.056)	(0.048)	(0.053)	(0.050)
100	0.1	0.101	0.107	0.109	0.110	0.101	0.106	0.107	0.106
		(0.057)	(0.054)	(0.054)	(0.052)	(0.057)	(0.053)	(0.054)	(0.058)
100	0.2	0.201	0.204	0.204	0.204	0.201	0.204	0.204	0.203
		(0.056)	(0.046)	(0.054)	(0.047)	(0.057)	(0.062)	(0.070)	(0.065)
200	0.0	0.001	-0.000	-0.001	-0.0	0.001	-0.000	-0.001	-0.000
		(0.055)	(0.063)	(0.064)	(0.075)	(0.055)	(0.051)	(0.048)	(0.054)
200	0.1	0.1	0.105	0.105	0.105	0.1	0.105	0.103	0.103
		(0.054)	(0.051)	(0.045)	(0.042)	(0.054)	(0.052)	(0.055)	(0.050)
200	0.2	0.200	0.201	0.200	0.200	0.200	0.201	0.2	0.200
		(0.052)	(0.047)	(0.05)	(0.051)	(0.052)	(0.06)	(0.064)	(0.064)
500	0.0	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
		(0.053)	(0.058)	(0.064)	(0.072)	(0.053)	(0.052)	(0.046)	(0.052)
500	0.1	0.100	0.101	0.101	0.101	0.100	0.101	0.101	0.101
		(0.054)	(0.050)	(0.046)	(0.046)	(0.054)	(0.054)	(0.056)	(0.054)
500	0.2	0.200	0.200	0.200	0.200	0.200	0.200	0.199	0.200
		(0.051)	(0.049)	(0.049)	(0.049)	(0.051)	(0.059)	(0.06)	(0.064)
1000	0.0	-0.000	-0.000	-0.000	0.000	-0.000	-0.000	-0.000	0.000
		(0.048)	(0.062)	(0.067)	(0.069)	(0.049)	(0.053)	(0.051)	(0.050)
1000	0.1	0.100	0.100	0.100	0.100	0.100	0.100	0.100	0.100
		(0.052)	(0.052)	(0.048)	(0.048)	(0.052)	(0.055)	(0.056)	(0.053)
1000	0.2	0.200	0.200	0.200	0.200	0.200	0.200	0.200	0.200
		(0.053)	(0.050)	(0.047)	(0.051)	(0.052)	(0.063)	(0.066)	(0.068)

Note: The Monte Carlo simulation is based on 10000 iterations. In the parenthesis is the Type-I error (α) for the 95-percentile. The true 95-percentile of the standard normal distribution implies a rejection rate of 0.05. $\Delta[R_{t,k}]$ is the true margin between the two arms.

To conclude, the Monte Carlo simulation study shows that an adaptive experiment requires sufficiently many observations per batch to obtain unbiased estimates. Note that the number of observations per batch and not the number of batches is decisive. Furthermore, OLS inference for small margins is invalid because the asymptotic normality assumption is unwarranted but the deviations from the true rejection rate for

hypothesis testing were small. The BOLS estimator can fix this problem but is inefficient in large margin cases. OLS performs well in large margin cases. We recommend, to always compute both estimates and their corresponding confidence intervals. Our Monte Carlo simulations suggest to rely on BOLS inference in the small margin case and on OLS inference in the large margin case.

5 The bbandits commands

The **bbandits** package contains four commands which assist to analyze, simulate and conduct adaptive experiments with batched bandits.

5.1 Analyzing bbandit data

The **bbandits** command provides the BOLS and OLS estimates and correct confidence intervals. Additionally, it provides a graphical analysis.

Syntax

Batched bandit analysis can be implemented in Stata using the following command syntax:

```
bbandits reward assignedarm batch [ , reference_arm(#) test_value(#)
    plot_thompson stacked twooptions_thompson(string)
    twooptions_bols(string) twooptions_ols(string)
    twooptions_sharebybatch(string) twoptions_stackedsharebybatch(string)
    twooptions_cumsharesbyybatch(string) ]
```

The **bbandits** syntax requires three variables. It takes the dependent variable *reward* also known as outcome or regret, typically a measure of success or failure, as the first input from the data. The second input requires a categorical variable, *assignedarm* that indicates for which of k arms the reward has been observed. As a third input, **bbandits** takes a categorical variable, which indicates in which t of T batches the reward has been observed.

Options

`reference_arm(#)` specifies which arm is taken as a reference arm (control group). The treatment effects are then calculated with respect to the reference arm. The default reference arm is arm 0 (first of K arms).

`test_value(#)` specifies the null-hypothesis. The default value is 0.

`plot_thompson` plots beta distributions for each of the treatments.

stacked plots beta distributions for each of the treatments but vertically stacked.

twooptions_thompson(string) takes user-specific two-way options for the twoway Thompson plot.

twooptions_bols(string) takes user-specific two-way options for the plot of the BOLS treatment effects.

twooptions_ols(string) takes user-specific two-way options for the plot of the BOLS and the OLS treatment effects.

twoptions_sharebybatch(string) takes user-specific two-way options for the plot of the shares assigned to each treatment arm by batch.

twoptions_stackedsharebybatch(string) takes user-specific two-way options for the plot of the shares assigned to each treatment arm by batch but stacked as an area.

twoptions_cumsharesbyybatch(string) takes user-specific two-way options for the plot of the cumulative shares assigned to each treatment arm by batch stacked as an area.

Returned results

Scalars
- e(N) number of observations

Matrices
- e(res) matrix with all output results
- e(batch_ols_coefficients) matrix with all batched OLS coefficients
- e(batched_ols_weights) matrix containing the BOLS weights

The matrix $e(res)$ includes for each arm the OLS margins, the BOLS margins, the z statistics, the p-values, the BOLS 95% confidence intervals, the observations of the reference arm, the observations of the treatment arm, the treatment arm indicator and the OLS 95% confidence intervals. In addition, all returned results from **regress** are returned.

5.2 Simulate bandit experiments

Syntax

The following command can simulate a bandit experiment to explore the data structure and mechanism of the algorithm. Additionally, it allows the user to conduct a simple Monte Carlo study to examine the inference in different scenarios.

bbandit_sim *true_expected_values* [, batch(#) size(#) eps(#) decay(#) clipping(#) exploration_phase(#) greedy thompson plot_thompson

`twopts(string) stacked monte_carlo n(#) reference_arm(#) arm(#) ]`

The **bbandit_sim** syntax requires the scalar parameters *true_expected_values*, for example, success parameters 0.2 0.3 0.4, to specify the reward distributions.[9] Two arms are required but it could be more. A series of options are allowed to further calibrate the algorithm. The two popular classes of bandit algorithms, epsilon-greedy and Thompson sampling, can be chosen in the options. The default algorithm is epsilon-greedy. The default case is to simulate a single bandit experiment and the return of the simulated data set. Users can also run a simple Monte Carlo study which is specified by the respective option.

Options

`batch(#)` specifies the number of batches. The default is 25 batches.

`size(#)` specifies the batch size. The default is 100 observations per batch.

`eps(#)` specifies the epsilon parameter for the epsilon-greedy algorithm. The default epsilon is 10%.

`decay(#)` is a linear decay rate for the epsilon-greedy algorithm. The default value is 1.

`clipping(#)` specifies the clipping rate for the Bernoulli Thompson algorithm. The default value is 5%.

`exploration_phase(#)` specifies the number of batches where the algorithm assigns the treatment uniformly. The default value is 0.

`greedy` specifies the epsilon-greedy algorithm instead of the Bernoulli-Thompson algorithm. The default is epsilon-greedy.

`thompson` specifies the Bernoulli-Thompson Sampling algorithm instead of the epsilon-greedy treatment assignment algorithm.

`plot_thompson` generates plots of the beta distribution under the Thompson sampling algorithm.

`twopts(string)` takes user-specific two-way options for the twoway Thompson plot.

`stacked` plots the beta distribution under the Thompson sampling algorithm but vertically stacked.

`monte_carlo` specifies a Monte Carlo simulation with 1,000 repetitions. Only two arms can be compared to each other even if multiple arms are specified.

`N(#)` specifies the number of repetitions for the Monte Carlo simulation.

9. For the epsilon-greedy algorithm rewards are drawn from a normal distribution with the specified expected value and a standard deviation of 1. For Thompson Sampling the rewards are drawn from a Bernoulli distribution.

reference_arm(#) specifies the reference arm for the Monte Carlo simulation.

arm(#) specifies the arm which is compared against the reference arm in the Monte Carlo simulation. The Monte Carlo simulation function only allows to compute the test statistic for two arms but the data can be drawn from a multi-arm simulation.

Return

The simulation returns a simulated data set. The Monte Carlo simulation returns the test statistics for the OLS and BOLS. Additionally, the stored result e(decay_rate) is returned. It captures the epsilon or clipping rate for each batch and should become smaller when a decay rate is determined.

Matrices
e(decay_rate) Epsilon or clipping rate for each batch.

5.3 Conducting bandit experiments interactively

To easily conduct a bandit experiment, our package provides two commands. Below, in the section 7, we provide an example how to use these commands. First, the **bbandit_initialize** command helps to set up a compatible data structure for an adaptive experiment and the package. Second, the **bbandit_update** function implements the bandit algorithms and returns which treatment arms to assign for the next batch based on the chosen bandit algorithm. Both commands are briefly explained below.

Syntax

bbandit_initialize [, batch(#) arms(#) exploration_phase(#)]

For **bbandit_initialize** the user must load a data set where each row is one of the potentially treated units.

Options

batch(#) specifies the number of batches. The default number is 3.

arms(#) specifies the number of treatment arms. The default is 2.

exploration_phase(#) specifies the number of batches where the algorithm assigns the treatment uniformly. The default value is 1.

Return

This command generates the following three new variables. First, the variable *reward* is generated but containing missings, because the experiment has not yet started. The

user must fill this variable later with the experimental data. Second, the variable *chosen_arm* is generated. This variable contains the assigned treatment arms. For *exploration_phase*, treatment arms are assigned uniformly and during the adaptive portion of the experiment algorithmically. Third, the variable *batch* indicates the respective batch. **bbandit_initialize** generates batches with equal batch sizes.

Syntax

The **bbandit_update** command has the following syntax:

```
bbandit_update [ , thompson greedy clipping(#) epsilon(#)
    excel("path") ]
```

During the adaptive portion of the experiment, the bandit algorithm chosen by the user assigns the treatment for the next batch. After having carried out the treatment, the user records the *reward*. The user then enters the newly observed rewards into the data set and runs **bbandit_update** again to start the next iteration of the algorithm that assigns units again to treatments for the next batch based on the history of rewards. Once all units are treated and the rewards are recorded, the **bbandits** command can be used to analyze the data.

Options

`thompson(#)` specifies Bernoulli Thompson sampling algorithm.

`greedy(#)` specifies the epsilon-greedy algorithm.

`clipping(#)` specifies the clipping rate for the Bernoulli Thompson algorithm. The default value is 0.05.

`epsilon(#)` specifies the epsilon rate for the epsilon-greedy algorithm. The default value is 0.1.

`excel("path")` indicates that the updated data is saved as an Excel file under the specified path. The saved file can be used to impute the newly observed rewards.

Return

The **bbandit_update** command returns a data set which includes the assigned treatment arms for the next batch according to the bandit algorithms. It also generates a variable *chosen_arm_numeric* which is a numeric label which corresponds to the assigned arms.

Scalars

`e(current_batch)`	Returns the current batch.

Matrices

`e(arm_labels)`	Numeric label
`e(probabilities)`	Share that the corresponding arm should be played in the next batch.

The returned matrix *e(arm_labels)* is a numeric label which corresponds to the numeric label for the assigned arm from the newly generated variable *chosen_arm_numeric*. The order of e(probabilities). corresponds to the *e(arm_labels)*. So, the first value is the share of arm 0, the second is the share of arm 1, and so on.

6 Examples with empirical application

In this section, we present applications of the **bbandit** command to analyse and conduct inference about effect sizes ex-post using data from adaptive experiments. The applications differ mainly in terms of treatment assignment algorithm and the number of treatment arms.

The first application is taken from Kasy and Sautmann (2021) and uses exploration sampling (explained below) for six treatment arms. The second application is based on Gaul et al. (2024) and uses Thompson sampling for 32 treatment arms.

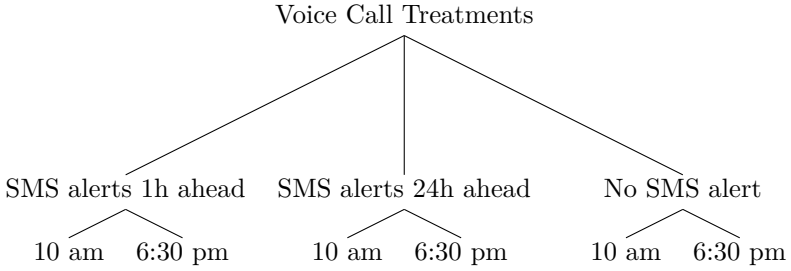
6.1 Six call methods to enroll rice farmers (Kasy and Sautmann 2021)

Kasy and Sautmann (2021) designed an experiment using *exploration sampling* (see below and Section 4 in their paper) to help Precision Agriculture for Development (PAD) choose among a variety of different call methods to enroll rice farmers in one state in India. PAD is an NGO that works with government partners to provide a phone-based personalized agricultural extension service to farmers.

The outcome (reward) is a binary variable describing call completion:

$$\text{call completed} = \begin{cases} 1 & \text{if call recipient answered five questions asked during call,} \\ 0 & \text{otherwise.} \end{cases}$$

PAD tested six automated voice calls treatments. Call time varied to be either in the morning (10 am) or in the evening (6:30 pm). The calls were made with SMS text message alerts sent 1 hour ahead, 24 hours ahead, or without SMS alerts.



Kasy and Sautmann (2021) base their algorithm on Thompson sampling and propose a modified treatment assignment algorithm which they call *exploration sampling*. In Thompson sampling, the probability that arm k is best in terms of reward at time t $p_t^k = P(E_\theta[R_k, t] = \max\{E_\theta[R_1], \ldots, E_\theta[R_K]\}|R_t)$ can be used for treatment assignment. *Exploration sampling* replaces the Thompson assignment shares

$$(p_t^1, p_t^2, ..., p_t^k)$$

with the following transformed shares $q_t^k = \frac{p_t^k(1-p_t^k)}{\sum_k p_t^k(1-p_t^k)}$.

This modification shifts weight away from the best performing option to its close competing treatments. Since there is at most one k for which $p_t^k > 1/2$, q_t^k is monotonically increasing and concave in p_t^k.

A list of 10,000 valid phone numbers were randomly assigned to one of 16 batches with 600 numbers each (and one with 400). Starting on June 3, 2019, a new experimental wave was started every other day and completed the next day.

The success rate of each treatment arm was estimated starting with a uniform prior in order to determine the assignment frequencies for each consecutive wave using exploration sampling.

```
.
. use "example data\kasy_sautmann_2021.dta", clear
. bbandits outcome treatment date

Number of obs                     =       10000
Est. Rewards only best arm        =        1926    Mean reward best arm      =      0.1926
Actual total reward               =        1804    Actual mean reward        =      0.1804
Est. reward uniformly chosen arms =        1709    Mean reward uniform       =      0.1709
```

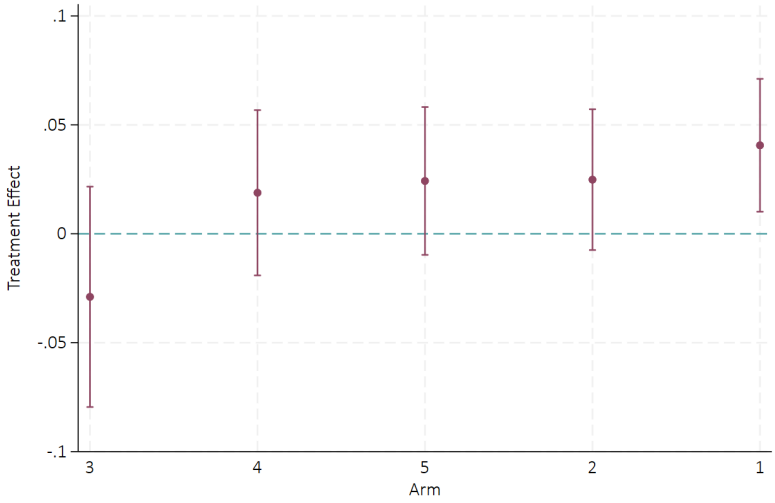
Arm b	Mean Reward						Share arm b
	0.1606						0.0903
k v. b	Margin OLS	Margin BOLS	z	P>|z|	[95% Conf.	Interval]	Share arm k
1-0	0.0320	0.0406	2.61	0.009	0.0101	0.0711	0.3931
2-0	0.0185	0.0249	1.51	0.132	-0.0075	0.0572	0.2234
3-0	-0.0158	-0.0289	-1.12	0.262	-0.0795	0.0216	0.0366
4-0	0.0078	0.0188	0.97	0.330	-0.0191	0.0568	0.1081
5-0	0.0192	0.0243	1.40	0.161	-0.0097	0.0582	0.1485

The results can be interpreted as follows: Calling farmers at 10 am after a text message an hour ahead of time is with over 75% probability the treatment with the greatest success rate, estimated to be 19.3% (16.06%+3.20%). Across treatments, higher success rates are associated with a higher number of observations, and correspondingly smaller posterior standard deviation. Figure 4 shows the treatment margins for each treatment compared to the reference treatment (10 am without SMS alert).

Cumulatively, 39.31% of farmers received the most successful type of call. The least successful call (at 6:30 pm without a text message alert) was received by only 3.66%. Based on the posterior estimated success rates, exploration sampling not only improved learning, but also increased overall success rates within the experiment (18.04%) compared to a standard design with equal assignment to treatment arms (where the estimated success rate based on posterior means would be 17.15%).

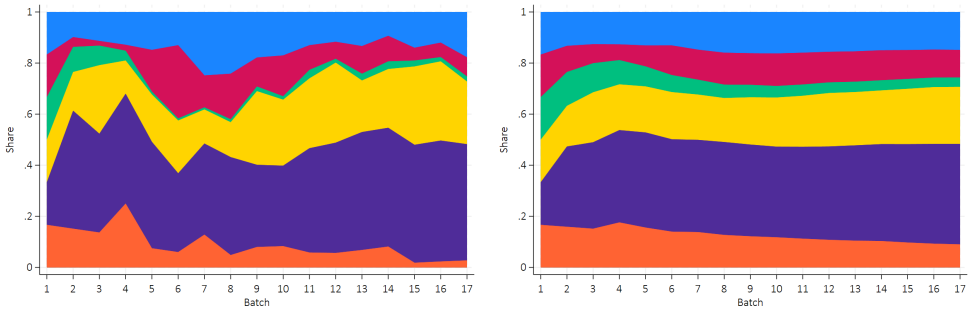
The Figures 5a and 5b show that one treatment was assigned to the most participants from wave 2 onwards, but some closely competing treatments got a high share of observations, especially in early waves. The number of observations per wave assigned to each of the treatments did stabilize towards later waves.

Figure 4: BOLS Treatment effect for the six call methods to enroll rice farmers



Notes: The Figure shows the BOLS estimates of margins relative to the control arm with their respective asymptotic 95% confidence intervals. The estimated margin for arm 1 is 3.20% (OLS) and 4.06% (BOLS). The figure was generated using `kasy_sautmann_2021.dta` and running `bbandits outcome treatment date`.

Figure 5: Treatment assignment of the six call methods to enroll rice farmers over batches



(a) Batchwise shares (b) Cumulative shares

Notes: This figure shows the shares assigned over the experiment to each of the 32 arms in each batch and cumulatively by batch. The treatment assignment algorithm is Exploration sampling. Arm 1 is selected most frequently with 39.31% overall. The cumulative shares show a quite balanced exploration. The figure was generated using `kasy_sautmann_2021.dta` and running `bbandits outcome treatment date`.

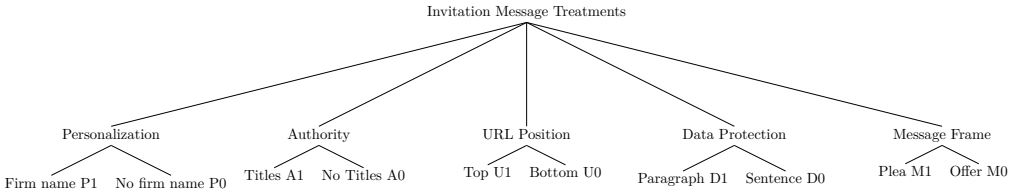
6.2 32 invitation messages for business surveys (Gaul et al. 2024)

Gaul et al. (2024) designed an experiment using *Thompson sampling* (see Section 2.1 and Section 3 in their paper) to help support the German Business Panel (GBP) to select among a variety of different invitation messages to survey firm decision makers in Germany. The GBP is a web-based survey study of firm decision makers in Germany that invites participants each work day (see Bischof et al. (2024) for details).

The outcome (reward) is a binary variable describing start of the survey:

$$\text{survey started} = \begin{cases} 1 & \text{if email invitation recipient started the survey,} \\ 0 & \text{otherwise.} \end{cases}$$

The GBP tested five components of invitation letters and their full interactions in $2^5 = 32$ treatments. The first treatment varies personalization by mentioning or not mentioning the firm name, the second varies the authority of the sender by listing the official full academic titles along with the senders' names or their names only, the third varies the position of the URL to start the survey to be at the top or at the bottom of the invitation message, the fourth emphasizes data protection in a separate paragraph with two strongly phrased sentences or in a single sentence, finally, the fifth varies the message frame by including phrases that plea for support in the survey's cause or to simply offer to participate in the survey.



Gaul et al. (2024) apply the Thompson sampling algorithm and use the assignment rule based on the probability that arm k is best in terms of reward at time t.

11,000 randomly selected contacts from public and private firms in Germany were assigned to each of 15 batches from a list of 176,000 contacts. Each batch corresponds to a week between August 16, 2022 and November 25, 2022. The first four batches used fixed and balanced burn-in phase, in which each treatment was assigned with probability 1/32. The success rates of each treatment arm was estimated weekly and starting from batch 5, the frequencies with which of the 32 treatments was assigned were calculated using the Thompson assignment rule for each consecutive batch.[10]

10. Based on the poor performance in the burn-in phase, treatment P0A0U0D1M0 was not sent out in batch 5. This can be prevented with setting a sufficiently high clipping rate.

```
. use "example data\gaul_et_al_2024.dta", clear
. bbandits reward selected trial

Number of obs                     =      176000
Est. Rewards only best arm        =        8623    Mean reward best arm      =      0.0490
Actual total reward               =        7833    Actual mean reward        =      0.0445
Est. reward uniformly chosen arms =        7430    Mean reward uniform       =      0.0422
```

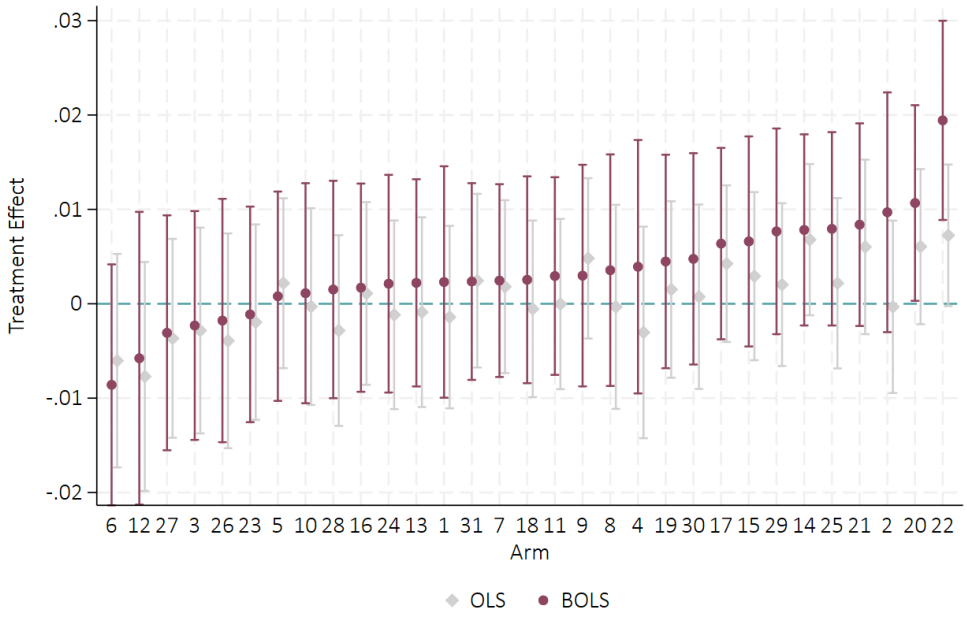
Arm b	Mean Reward						Share arm b
	0.0417						0.0181
k v. b	Margin OLS	Margin BOLS	z	P>|z|	[95% Conf.	Interval]	Share arm k
1-0	-0.0014	0.0023	0.37	0.712	-0.0100	0.0146	0.0220
2-0	-0.0003	0.0097	1.50	0.135	-0.0030	0.0224	0.0288
3-0	-0.0028	-0.0023	-0.37	0.710	-0.0144	0.0098	0.0137
4-0	-0.0030	0.0039	0.57	0.567	-0.0095	0.0174	0.0125
5-0	0.0022	0.0008	0.14	0.888	-0.0103	0.0119	0.0312
6-0	-0.0060	-0.0086	-1.32	0.187	-0.0214	0.0042	0.0121
7-0	0.0018	0.0025	0.47	0.638	-0.0078	0.0127	0.0284
8-0	-0.0003	0.0036	0.57	0.570	-0.0087	0.0158	0.0141
9-0	0.0048	0.0030	0.50	0.619	-0.0088	0.0147	0.0444
10-0	-0.0003	0.0011	0.19	0.851	-0.0105	0.0128	0.0162
11-0	-0.0000	0.0029	0.55	0.583	-0.0075	0.0134	0.0308
12-0	-0.0077	-0.0058	-0.73	0.466	-0.0213	0.0097	0.0097
13-0	-0.0009	0.0022	0.40	0.692	-0.0088	0.0132	0.0186
14-0	0.0068	0.0078	1.51	0.130	-0.0023	0.0180	0.0715
15-0	0.0029	0.0066	1.16	0.245	-0.0045	0.0177	0.0331
16-0	0.0011	0.0017	0.30	0.762	-0.0093	0.0127	0.0219
17-0	0.0042	0.0064	1.23	0.218	-0.0038	0.0165	0.0527
18-0	-0.0005	0.0025	0.45	0.650	-0.0084	0.0135	0.0255
19-0	0.0015	0.0045	0.78	0.438	-0.0068	0.0158	0.0256
20-0	0.0061	0.0107	2.02	0.044	0.0003	0.0210	0.0571
21-0	0.0060	0.0084	1.53	0.126	-0.0024	0.0191	0.0271
22-0	0.0072	0.0194	3.61	0.000	0.0089	0.0300	0.1840
23-0	-0.0020	-0.0011	-0.19	0.847	-0.0126	0.0103	0.0166
24-0	-0.0012	0.0021	0.36	0.718	-0.0094	0.0137	0.0190
25-0	0.0022	0.0079	1.52	0.129	-0.0023	0.0182	0.0308
26-0	-0.0039	-0.0018	-0.27	0.787	-0.0147	0.0111	0.0119
27-0	-0.0037	-0.0031	-0.48	0.628	-0.0155	0.0094	0.0155
28-0	-0.0028	0.0015	0.26	0.797	-0.0100	0.0130	0.0183
29-0	0.0020	0.0077	1.38	0.168	-0.0032	0.0186	0.0400
30-0	0.0007	0.0048	0.83	0.405	-0.0064	0.0160	0.0210
31-0	0.0024	0.0024	0.44	0.658	-0.0081	0.0128	0.0278

The results can be interpreted as follows: Inviting firm decision makers using personalization, no authority, a top position for the URL, no emphasis on data protection, and pleading for support the treatment with the greatest success rate (arm 22, P1A0U1D0M1), estimated to be 6.11% (4.17%+1.94%) with BOLS 4.89% (4.17%+0.72%) with OLS. Across treatments, higher success rates are associated with a higher number of observations, and correspondingly smaller posterior standard deviation. Figure 6 shows the margins compared to the reference arm P0A0U0D0M0.

Cumulatively, 18.40% of firm decision makers received the most successful invita-

tion letter. The least successful invitation letter (arm 3, P0A0U0D1M0) was received by only 1.21%. Based on the posterior estimated success rates including the burn-in phase, Thompson sampling improved learning and also increased overall success rates within the experiment (4.88%) compared to a standard design with equal assignment to treatment arms (where the estimated success rate based on posterior means would be 4.50%).

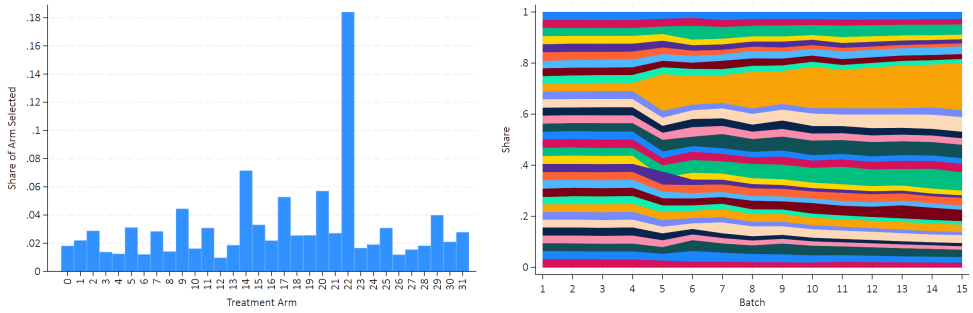
Figure 6: BOLS and OLS Treatment effects for 32 invitation messages



Notes: This figure shows estimates BOLS along with corresponding OLS estimated margins relative to the reference arm for each of the 32 arms. The treatment assignment algorithm is Thompson sampling. The estimated margin for arm 22 (P1A0U1D0M1) is 0.72% (OLS) and 1.94% (BOLS). OLS confidence bands tend to be smaller than BOLS with a standard error of xx (BOLS) and 0.38% (OLS) for arm 22. The figure was generated using `gaul_et_al_2024.dta` and running `bbandits reward selected trial`.

Focusing on the five components instead of interaction effects from each invitation messages, personalization increases the starting rate significantly by 0.41 percentage points (p-value: <0.0001). High authority and pleading for help also increase the starting rate by 0.16 (p-value: 0.0394) and 0.25 percentage points (p-value: 0.0036). Given the overall low baseline starting rate, these effects are also economically meaningful. For example, relative to the marginal mean with an unpersonalized invite of 4.12%, personalizing the message increases the propensity to start answering the questionnaire by 9.95%.

Figure 7: Treatment assignments for 32 invitation messages



(a) Total frequency of treatment assignment (b) Cumulative shares

Notes: This figure shows the shares assigned over the experiment to each of the 32 arms and the cumulative shares assigned to each arms by batch. The treatment assignment algorithm is Thompson sampling. Arm 22 (P1A0U1D0M1) is selected most frequently with 18.40% overall. The cumulative shares show that arm 22 was selected more frequently, right after the burn-in phase, from batch 4 on. The figure was generated using `gaul_et_al_2024.dta` and running `bbandits reward selected trial`.

The figures 7 show that one treatment (arm 22, P1A0U1D0M1) was assigned to the most participants from batch 5 onwards, but some closely competing treatments got a high share of observations, especially in early waves. Thompson sampling assigned increasingly more observations per batch to the better performing arms.

7 Designing and conducting bandit experiments

A suggested workflow run own adaptive experiments with **bbandits** includes the following steps:

Initialization

1. Create list of all potentially treated units
2. Separate them into batches (**bbandit_initialize**)
3. Determine number of batches for exploration phase
4. Decide for number of treatments
   → randomly assign treatments in exploration phase
5. Enter rewards from first iteration
6. Configure updating algorithm (**bbandit_update**): specify clipping rate for Thompson sampling

After initialization

1. Assign treatment according to weights from algorithm
2. Enter rewards + update
3. Repeat process until rewards are observed for all units
4. Analyze the adaptively generated data with **bbandit**

7.1 Initialization

The **bbandits** package provides two functions to run batched bandits experiments. To illustrate the standard workflow think of a setting like in Duflo et al. (2012) where the outcome variable is the absence of a school teacher and the two treatment arms are financial incentives for attending class and video surveillance. A control arm receives no incentives. Instead of running a classic field experiment as Duflo and co-authors do, one could use the **bbandits** package to conduct an adaptive experiment.

To start the experiment, compiling a list of all participating units is recommended. In the illustrative example, these would be school teachers where some are control units while others receive one of the two treatment arms (surveillance or financial incentives). The rewards will be simulated here. Once the list of all participating subjects is loaded into Stata, the **bbandit_initialize** command can be used to generate the required data structure and separate the list of subjects into equally sized batches.[11] Additionally the number of treatment arms and the exploration phase has to be specified. The exploration phase defines the number of batches where the treatment is just uniformly distributed.

The **bbandit_initialize** function will randomly assign the treatment arms in the specified exploration phase. For our simulated example, consider 1,000 school teachers, which will be separated into ten batches with an exploration phase of two batches. In Figure 8 the generated data structure is shown. Each ID corresponds to a level of the variable *chosen_arm*. Since we have not collected any data for the reward yet, the variable *reward* is set to missing. Once observations come in, this variable holds the realized rewards from the assigned treatments. For example, treatment 3 (surveillance camera) is assigned to *school_1*. The observed reward (attendance of the teacher) needs to be recorded in the variable *reward* column for *school_1*.

```
. bbandit_initialize, batch(10) arms(3) exploration_phase(2)
```

7.2 After initialization

In the next step the user has to run the first batches of the experiment and observe the rewards (outcome). To produce a new assignment scheme based on the realized

11. If the user would like to deviate from equal batch sizes, the data structure can easily be generated manually by applying standard Stata commands.

Figure 8: Initial data structure

	ID	reward	chosen_arm	batch
1	school_1	.	3	1
2	school_2	.	3	1
3	school_3	.	3	1
4	school_4	.	2	1
5	school_5	.	3	1
6	school_6	.	1	1
7	school_7	.	1	1
8	school_8	.	2	1
9	school_9	.	1	1
10	school_10	.	3	1

rewards so far, the data including the newly observed rewards need to be reloaded into Stata.[12] If one does not want to specify an exploration phase, the rewards and the chosen treatment arms have to be manually specified.

The **bbandit_update** function can now be used to assign the treatment according to a bandit algorithm for the next batch. In our example, the outcome variable is binary, so we could specify the Bernoulli Thompson Sampling algorithm with the option *Thompson*. Additionally, the *clipping* parameter has to be specified for the Thompson Sampling algorithm. For the epsilon-greedy algorithm specifying the parameter epsilon is required.

```
. bbandit_update reward chosen_arm batch, thompson clipping(0.2)
> excel("mypath")
```

The *clipping rate* sets the minimum probability that an arm is played. [13] Regardless of the chosen algorithm, the **bbandit_update** function will return the treatment arms which should be played in the next batch (1) and the according treatment probabilities (2) for each arm. Now the user has to assign the treatment to the treatment arms according to the algorithm and observe the results. Then the observed rewards are read into Stata. The easiest way to implement this procedure is to saves the data set into a file and then add the rewards into the file. This can either be done in Stata or in a different program. The **bbandit_update** function has the option *Excel* which saves the data set into an Excel file in the specified folder. For small-scale experiments one convenient option would be to record the rewards in an Excel file and then read the

12. An exploration phase of at least one batch is required. In the Bernoulli Thompson Sampling case starting with an exploration phase of one is equal to immediately starting with the algorithm because in the first batch the prior is a uniform distribution.
13. Without setting a clipping rate inference can become a problem because it could occur that in some batches some treatment arms are not assigned which leads to a non-defined margin in these batches.

Figure 9: Data structure for updating

	A	B	C	D	E	F
1	ID	rand	reward	chosen_arm	batch	chosen_arm_numeric
192	school_19	1	0	1	2	2
193	school_19	1	1	3	2	0
194	school_19	0	1	3	2	0
195	school_19	1	0	3	2	0
196	school_19	0	1	2	2	1
197	school_19	0	1	1	2	2
198	school_19	1	0	2	2	1
199	school_19	1	0	3	2	0
200	school_19	0	1	2	2	1
201	school_20	0	1	1	2	2
202	school_20	1		3	3	0
203	school_20	1		3	3	0
204	school_20	1		3	3	0
205	school_20	1		2	3	1
206	school_20	0		2	3	1

file back into Stata. Figure 9 shows the data structure exported in Excel where the rewards can be recorded in the yellow marked column. The *chosen_arm* column in the figure displays which treatment arm is assigned according to the bandit algorithm. The *chosen_arm_numeric* column will automatically be generated and entails an integer which corresponds to the "chosen_arm" category.[14] The user has to fill-in the observed rewards into the variable *rewards* and then load the file back into Stata.

Subsequently, the **bbandit_update** is applied again and all steps are repeated until the experiment is completed. Finally, the results can be analyzed with the **bbandits** command.

8 Conclusion

Bandits and other adaptive experiments offer the opportunity to reduce costs of experiments by balancing learning about treatment effects and earning from assigning the best treatment arm. This way, outcomes for participants can be improved and better policies can be identified more quickly than in traditional experiments with fixed and balanced assignment to arms over the entire course of the experiment. This has led to a push to use more bandits in medicine, economics, political science, survey methods research, education, psychology, sociology, and other fields.

Some popular algorithms for sequential treatment assignment include ε-first, ε-greedy, and Thompson sampling. Before running bandit experiments, it is desirable to try different configurations and algorithms in simulations. When sequentially running a bandit experiment, analysis of the collected data so far and systematic assignment to treatments is required. Because of non-uniform treatment assignment, bandits break by design usual asymptotics and therefore standard inference is not always possible.

14. As many users are likely to export the data into a different programming language for recording the rewards, an extra column instead of a value label is convenient for a transfer into other programs.

To support conclusions, however, researchers require valid confidence intervals. Using *batched* bandits solve several of the problems of standard multi-armed bandits and allows to construct valid confidence intervals easily.

Our Monte Carlo simulations suggest that in practice, researchers should always compute both the OLS and the BOLS estimates and their corresponding confidence intervals. Our Monte Carlo simulations suggest to rely on BOLS inference in the small margin case and on OLS inference in the large margin case. When designing adaptive experiments, at least 20 observations per batch are necessary for inference.

This paper introduces a new Stata command **bbandits** for batched bandit experiments. **bbandits** makes simulating, interactively running, and analysing batched bandit experiments easy. The command runs Monte Carlo simulations for three popular treatment assignment algorithms: ε-first, ε-greedy, and Thompson sampling. With **bbandits** researchers can implement their own batched bandit experiments. **bbandits** provides valid statistical inference and correct coverage and a wide range of statistics and illustrations to analyse adaptively collected data.

9 References

Agarwal, D., B. Long, J. Traupman, D. Xin, and L. Zhang. 2014. LASER: A Scalable Response Prediction Platform for Online Advertising. In *Proceedings of the 7th ACM International Conference on Web Search and Data Mining*, 173–182. WSDM '14. New York, NY, USA: Association for Computing Machinery.

Agrawal, S., and N. Goyal. 2013. Further Optimal Regret Bounds for Thompson Sampling. In *Proceedings of the Sixteenth International Conference on Artificial Intelligence and Statistics*, ed. C. M. Carvalho and P. Ravikumar. Vol. 31 of *Proceedings of Machine Learning Research*, 99–107. Scottsdale, Arizona, USA: PMLR.

Athey, S., and G. W. Imbens. 2019. Machine Learning Methods That Economists Should Know About. *Annual Review of Economics* 11(Volume 11, 2019): 685–725.

Avivi, H., P. Kline, E. Rose, and C. Walters. 2021. Adaptive Correspondence Experiments. *AEA Papers and Proceedings* 111: 43–48.

Bibaut, A., M. Dimakopoulou, N. Kallus, A. Chambaz, and M. van Der Laan. 2021. Post-Contextual-Bandit Inference. *Advances in Neural Information Processing Systems* 34: 28548–28559.

Bischof, J., P. Doerrenberg, D. Rostam-Afschar, D. Simons, and J. Voget. 2024. The German Business Panel: Firm-Level Data for Accounting and Taxation Research. *European Accounting Review* .

Burtini, G., J. Loeppky, and R. Lawrence. 2015. A Survey of Online Experiment Design with the Stochastic Multi-Armed Bandit. Technical report.

Camerer, C., D. Bose, R. Daviet, and T. Imai. 2024. Social Preference Estimation Using Adaptive Experimental Design. Unpublished manuscript.

Chapelle, O., and L. Li. 2011. An Empirical Evaluation of Thompson Sampling. In *Advances in Neural Information Processing Systems*, ed. J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Weinberger. Vol. 24. Curran Associates, Inc.

Chen, J., and I. Andrews. 2023. Optimal Conditional Inference in Adaptive Experiments.

Duflo, E., R. Hanna, and S. P. Ryan. 2012. Incentives Work: Getting Teachers to Come to School. *American Economic Review* 102(4): 1241–1278.

Gaul, J. J., F. Keusch, D. Rostam-Afschar, and T. Simon. 2024. Invitation Messages for Business Surveys A Multi-Armed Bandit Experiment. Unpublished manuscript.

Graepel, T., J. Q. Candela, T. Borchert, and R. Herbrich. 2010. Web-Scale Bayesian Click-Through Rate Prediction for Sponsored Search Advertising in Microsoft's Bing Search Engine. In *Proceedings of the 27th International Conference on International Conference on Machine Learning (ICML '10)*, 13–20. Omnipress, Madison, WI, USA.

Hadad, V., D. A. Hirshberg, R. Zhan, S. Wager, and S. Athey. 2021. Confidence Intervals for Policy Evaluation in Adaptive Experiments. *Proceedings of the National Academy of Sciences* 118(15): e2014602118.

Hill, D. N., H. Nassif, Y. Liu, A. Iyer, and S. Vishwanathan. 2017. An Efficient Bandit Algorithm for Realtime Multivariate Optimization. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '17. ACM.

Hirano, K., and J. R. Porter. 2023. Asymptotic Representations for Sequential Decisions, Adaptive Experiments, and Batched Bandits. Technical report.

Kalkanli, C., and A. Ozgur. 2020. Asymptotic Convergence of Thompson Sampling. Technical report.

Kasy, M., and A. Sautmann. 2021. Adaptive Treatment Assignment in Experiments for Policy Choice. *Econometrica* 89(1): 113–132.

Lei, H., Y. Lu, A. Tewari, and S. A. Murphy. 2022. An Actor-Critic Contextual Bandit Algorithm for Personalized Mobile Health Interventions.

Nie, X., X. Tian, J. Taylor, and J. Zou. 2018. Why Adaptively Collected Data Have Negative Bias and How to Correct for It. In *International Conference on Artificial Intelligence and Statistics*, 1261–1269. PMLR.

Offer-Westort, M., A. Coppock, and D. P. Green. 2021. Adaptive Experimental Design: Prospects and Applications in Political Science. *American Journal of Political Science* 65(4): 826–844.

Perrault, P., E. Boursier, M. Valko, and V. Perchet. 2020. Statistical Efficiency of Thompson Sampling for Combinatorial Semi-Bandits. In *Advances in Neural Information Processing Systems*, ed. H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin. Vol. 33, 5429–5440. Curran Associates, Inc.

Rafferty, A., H. Ying, and J. Williams. 2019. Statistical Consequences of using Multi-armed Bandits to Conduct Adaptive Educational Experiments. *Journal of Educational Data Mining* 11(1): 47–79.

Schulz, E., N. T. Franklin, and S. J. Gershman. 2020. Finding Structure in Multi-Armed Bandits. *Cognitive Psychology* 119: 101261.

Scott, S. L. 2010. A Modern Bayesian Look at the Multi-Armed Bandit. *Applied Stochastic Models in Business and Industry* 26(6): 639–658.

———. 2015. Multi-Armed Bandit Experiments in the Online Service Economy. *Applied Stochastic Models in Business and Industry* 31: 37–49. Special issue on actual impact and future perspectives on stochastic modelling in business and industry.

Thompson, W. R. 1933. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika* 25(3-4): 285–294.

———. 1935. On the Theory of Apportionment. *American Journal of Mathematics* 57(2): 450–456.

Wang, S., and W. Chen. 2018. Thompson Sampling for Combinatorial Semi-Bandits. In *Proceedings of the 35th International Conference on Machine Learning*, ed. J. Dy and A. Krause. Vol. 80 of *Proceedings of Machine Learning Research*, 5114–5122. PMLR.

Wang, S., and J. Zhu. 2022. Thompson Sampling for (Combinatorial) Pure Exploration.

Zhang, K., L. Janson, and S. Murphy. 2020. Inference for Batched Bandits. *Advances in Neural Information Processing Systems* 33: 9818–9829.

———. 2021. Statistical Inference with M-Estimators on Adaptively Collected Data. In *Advances in Neural Information Processing Systems*, ed. M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan. Vol. 34, 7460–7471. Curran Associates, Inc.

About the authors

Jan Kemper is a PhD student at the University of Mannheim and at the ZEW.

Davud Rostam-Afschar is Professor at the University of Mannheim, affiliated with IZA and GLO.

Acknowledgments

We thank Johannes Gaul, Michael Knaus, David Preinerstorfer, and Thomas Simon for valuable comments. We declare that we have no interests, financial or otherwise, that relate to the research described in this paper. We are grateful to the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) for financial support through CRC *TRR 266 Accounting for Transparency* (Project-ID 403041268).

10 Appendix

11 Derivation of Formulas

Starting point is the normality result derived by Zhang et al. (2020). Their formula for the test statistic (p. 7) is as follows when the hypothesized value of the test is the true margin Δ:

$$\frac{1}{\sqrt{T}} \sum_{t=1}^{T} \sqrt{\frac{(\sum 1 - A_{i,t}) \sum A_{i,t}}{n\sigma^2}} (\hat{\Delta}_t - \Delta^{BOLS}) \sim N(0,1). \tag{2}$$

To simplify the notation, we write $\sum 1 - A_{i,t} = N_{b,t}$ and $\sum A_{i,t} = N_{k,t}$ where k is the treatment arm and b is the base arm (reference arm). To underline that we only compare two arms, we write $n = n_t = N_{b,t} + N_{k,t}$.

The same formula with new notation is the following:

$$\frac{1}{\sqrt{T}} \sum_{t=1}^{T} \sqrt{\frac{N_{b,t} N_{k,t}}{n_t \sigma^2}} (\hat{\Delta}_t - \Delta^{BOLS}) \sim N(0,1). \tag{3}$$

To simplify notation, we define $\sqrt{\frac{N_{b,t} N_{k,t}}{n_t}} = \omega_t$. Now, we derive the the confidence interval for the critical percentile c of the normal distribution (e.g. for $\alpha = 0.05$ c would be 1.96):

$$Pr\bigg(-c \leq \frac{1}{\sqrt{T}\sigma} \sum_{t=1}^{T} \omega_t (\hat{\Delta}_t - \Delta^{BOLS}) \leq c \bigg) = 1 - \alpha \tag{4}$$

We first multiply with σ and $\sqrt{T}$

$$Pr\bigg(-c\sigma\sqrt{T} \leq \sum_{t=1}^{T} \omega_t (\hat{\Delta}_t - \Delta^{BOLS}) \leq c\sigma\sqrt{T} \bigg) = 1 - \alpha \tag{5}$$

Re-arrange terms:

$$Pr\bigg(-c\sigma\sqrt{T} \leq \sum_{t=1}^{T} \omega_t \hat{\Delta}_t - \sum_{t=1}^{T} \omega_t \Delta^{BOLS} \leq c\sigma\sqrt{T} \bigg) = 1 - \alpha \tag{6}$$

Now we subtract $\sum_{t=1}^{T} \omega_t \hat{\Delta}_t$

$$Pr\bigg(-c\sigma\sqrt{T} - \sum_{t=1}^{T} \omega_t \hat{\Delta}_t \leq -\sum_{t=1}^{T} \omega_t \Delta^{BOLS} \leq c\sigma\sqrt{T} - \sum_{t=1}^{T} \omega_t \hat{\Delta}_t \bigg) = 1 - \alpha \tag{7}$$

Multiply by -1:

$$Pr\Bigg(-c\sigma\sqrt{T} + \sum_{t=1}^{T} \omega_t \hat{\Delta}_t \leq \sum_{t=1}^{T} \omega_t \Delta^{BOLS} \leq c\sigma\sqrt{T} + \sum_{t=1}^{T} \omega_t \hat{\Delta}_t \Bigg) = 1 - \alpha \tag{8}$$

Now divide by $\sum_{t=1}^{T} \omega_t$ to isolate Δ. We can do so, because Δ does not depend on t.

$$Pr\Bigg(\frac{-c\sigma\sqrt{T}}{\sum_{t=1}^{T} w_t} + \frac{\sum_{t=1}^{T} \omega_t \hat{\Delta}_t}{\sum_{t=1}^{T} w_t} \leq \Delta^{BOLS} \leq \frac{c\sigma\sqrt{T}}{\sum_{t=1}^{T} \omega_t} + \frac{\sum_{t=1}^{T} \omega_t \hat{\Delta}_t}{\sum_{t=1}^{T} \omega_t} \Bigg) = 1 - \alpha \tag{9}$$

To further simplify the notation we define the following: $w = \frac{\sqrt{T}}{\sum_{t=1}^{T} \omega_t}$. Additionally, $\frac{\sum_{t=1}^{T} \omega_t \hat{\Delta}_t}{\sum_{t=1}^{T} \omega_t} = \hat{\Delta}^{BOLS}$ which is a weighted average of the batchwise estimated OLS estimates.

$$Pr(\hat{\Delta}^{BOLS} - c\sigma w \leq \Delta \leq \Delta^{\hat{BOLS}} + c\sigma w) = 1 - \alpha \tag{10}$$

11.1 BOLS estimation

Above we showed that the confidence interval is centered around

$$\frac{\sum_{t=1}^{T} \omega_t \hat{\Delta}_t}{\sum_{t=1}^{T} \omega_t} = \hat{\Delta}^{BOLS}. \tag{11}$$

As the sum in the denominator is a constant, we can write:

$$\sum_{t=1}^{T} \frac{\omega_t}{\sum_{t=1}^{T} \omega_t} \hat{\Delta}_t = \sum_{t=1}^{T} \gamma_t \hat{\Delta}_t = \hat{\Delta}^{BOLS}. \tag{12}$$

Where $\gamma_t = \frac{\omega_t}{\sum_{t=1}^{T} \omega_t}$ is a weight between 0 and 1 which indicates how much each $\hat{\Delta}_t$ contributes to $\hat{\Delta}^{BOLS}$. Additionally, it shows that $\hat{\Delta}^{BOLS}$ is a weighted average of the batchwise estimated OLS estimates $\hat{\Delta}_t^{BOLS}$ with weights between 0 and 1 which sum up to 1.

11.2 Monte Carlo Simulation

Notes: The results are based on a Monte Carlo simulation with 5000 repetitions. Both treatment arms have a true expected value of 1, hence the margin is zero ($\beta_1 = 0$). The rewards are drawn from a normal distribution with mean 1 and standard deviation 1.

Table 4: Bernoulli Thompson Sampling - Clipping 0.3

		OLS/Batch				BOLS/Batch			
N_t	$\Delta[R_{t,k}]$	10	25	50	100	10	25	50	100
500	0.0	0.001	0.0	0.0	0.0	0.001	0.0	0.0	0.0
		(0.051)	(0.053)	(0.051)	(0.053)	(0.051)	(0.051)	(0.049)	(0.052)
500	0.1	0.099	0.099	0.1	0.1	0.099	0.1	0.1	0.1
		(0.048)	(0.054)	(0.053)	(0.046)	(0.048)	(0.055)	(0.054)	(0.048)
500	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2
		(0.05)	(0.048)	(0.047)	(0.05)	(0.051)	(0.054)	(0.052)	(0.058)
1000	0.0	-0.0	0.0	0.0	0.0	-0.0	0.0	0.0	0.0
		(0.054)	(0.057)	(0.052)	(0.049)	(0.054)	(0.054)	(0.049)	(0.047)
1000	0.1	0.099	0.1	0.1	0.1	0.099	0.1	0.1	0.1
		(0.052)	(0.055)	(0.053)	(0.05)	(0.052)	(0.052)	(0.053)	(0.05)
1000	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2
		(0.05)	(0.054)	(0.05)	(0.051)	(0.05)	(0.061)	(0.052)	(0.059)

Note: The Monte Carlo simulation is based on 10000 iterations. In the parenthesis are the Type-I error rates. That is the probability that H0 is true but falsely rejected.

Table 5: Bernoulli Thompson Sampling - Clipping 0.4

		OLS/Batch				BOLS/Batch			
N_t	$\Delta[R_{t,k}]$	1	3	5	8	1	3	5	8
500	0.0	0.0	-0.0	0.0	0.0	0.0	-0.0	0.0	0.0
		(0.053)	(0.052)	(0.052)	(0.052)	(0.053)	(0.052)	(0.05)	(0.053)
500	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
		(0.051)	(0.051)	(0.049)	(0.05)	(0.051)	(0.051)	(0.05)	(0.051)
500	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2
		(0.048)	(0.05)	(0.05)	(0.048)	(0.048)	(0.052)	(0.055)	(0.052)
1000	0.0	-0.0	-0.0	-0.0	0.0	-0.0	-0.0	-0.0	0.0
		(0.051)	(0.052)	(0.054)	(0.048)	(0.051)	(0.052)	(0.052)	(0.048)
1000	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
		(0.052)	(0.053)	(0.052)	(0.053)	(0.052)	(0.054)	(0.053)	(0.054)
1000	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2
		(0.052)	(0.048)	(0.045)	(0.048)	(0.052)	(0.05)	(0.049)	(0.052)

Note: The Monte Carlo simulation is based on 10000 iterations. In the parenthesis are the Type-I error rates. That is the probability that H0 is true but falsely rejected.

Table 6: Epsilon Greedy - Epsilon 0.2

		OLS/Batch				BOLS/Batch			
N_t	$\Delta[R_{t,k}]$	1	3	5	8	1	3	5	8
100	0	-0.003	0.002	0.002	-0.0	-0.003	0.002	0.002	-0.0
		(0.057)	(0.058)	(0.057)	(0.057)	(0.058)	(0.049)	(0.047)	(0.047)
100	1	0.998	1.0	1.0	0.999	0.998	1.0	1.0	0.998
		(0.057)	(0.054)	(0.051)	(0.049)	(0.056)	(0.051)	(0.053)	(0.046)
100	2	2.004	2.0	2.0	2.0	2.004	1.999	2.0	1.999
		(0.053)	(0.053)	(0.056)	(0.056)	(0.052)	(0.05)	(0.057)	(0.054)
200	0	0.001	0.001	-0.001	-0.001	0.001	0.001	-0.0	-0.001
		(0.048)	(0.053)	(0.055)	(0.056)	(0.048)	(0.05)	(0.049)	(0.047)
200	1	1.0	0.999	1.001	1.001	1.0	0.998	1.001	1.001
		(0.057)	(0.052)	(0.053)	(0.053)	(0.056)	(0.048)	(0.051)	(0.053)
200	2	2.001	2.0	1.999	2.0	2.001	2.0	1.999	2.001
		(0.053)	(0.052)	(0.049)	(0.051)	(0.053)	(0.05)	(0.052)	(0.052)
500	0	-0.001	-0.001	-0.0	0.0	-0.001	-0.0	-0.001	0.0
		(0.048)	(0.055)	(0.055)	(0.056)	(0.049)	(0.05)	(0.046)	(0.048)
500	1	0.999	1.001	1.0	1.0	0.999	1.001	1.0	1.0
		(0.053)	(0.049)	(0.048)	(0.048)	(0.053)	(0.049)	(0.049)	(0.05)
500	2	2.0	2.0	2.001	2.0	2.0	2.0	2.001	2.0
		(0.054)	(0.049)	(0.052)	(0.052)	(0.054)	(0.052)	(0.056)	(0.049)
1000	0	0.0	0.0	0.0	0.001	0.0	-0.0	-0.0	0.001
		(0.05)	(0.054)	(0.058)	(0.053)	(0.05)	(0.05)	(0.053)	(0.054)
1000	1	1.001	1.0	1.0	1.0	1.001	0.999	1.0	1.0
		(0.05)	(0.052)	(0.051)	(0.05)	(0.05)	(0.051)	(0.052)	(0.051)
1000	2	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0
		(0.053)	(0.056)	(0.047)	(0.053)	(0.053)	(0.05)	(0.045)	(0.049)

Note: The Monte Carlo simulation is based on 10000 iterations. In the parenthesis are the Type-I error rates. The rewards are drawn from a Normal distribution with a standard deviation of 1.

Table 7: Bernoulli Thompson Sampling - Clipping 0.1

		OLS/Batch				BOLS/Batch			
N_t	$\Delta[R_{t,k}]$	1	3	5	8	1	3	5	8
100	0.0	0.001	-0.001	-0.0	0.0	0.001	-0.001	-0.0	0.001
		(0.1)	(0.067)	(0.056)	(0.047)	(0.1)	(0.071)	(0.058)	(0.047)
100	0.1	0.101	0.104	0.106	0.104	0.101	0.104	0.103	0.101
		(0.098)	(0.065)	(0.052)	(0.043)	(0.098)	(0.072)	(0.059)	(0.049)
100	0.2	0.199	0.202	0.201	0.201	0.199	0.201	0.2	0.201
		(0.098)	(0.064)	(0.055)	(0.047)	(0.098)	(0.074)	(0.063)	(0.053)
200	0.0	-0.001	-0.0	0.001	-0.0	-0.001	-0.0	0.001	-0.0
		(0.071)	(0.048)	(0.04)	(0.033)	(0.071)	(0.05)	(0.041)	(0.033)
200	0.1	0.099	0.103	0.103	0.102	0.099	0.103	0.102	0.1
		(0.071)	(0.045)	(0.037)	(0.032)	(0.071)	(0.051)	(0.043)	(0.037)
200	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2
		(0.069)	(0.046)	(0.04)	(0.034)	(0.069)	(0.054)	(0.046)	(0.038)
500	0.0	0.0	0.0	0.0	-0.0	0.0	0.0	0.0	-0.0
		(0.044)	(0.03)	(0.025)	(0.021)	(0.044)	(0.032)	(0.026)	(0.021)
500	0.1	0.1	0.101	0.101	0.101	0.1	0.101	0.101	0.1
		(0.044)	(0.029)	(0.025)	(0.022)	(0.044)	(0.034)	(0.029)	(0.024)
500	0.2	0.199	0.2	0.2	0.2	0.199	0.2	0.2	0.2
		(0.043)	(0.03)	(0.026)	(0.022)	(0.043)	(0.035)	(0.029)	(0.024)
1000	0.0	-0.0	-0.0	0.0	-0.0	-0.0	-0.0	0.0	-0.0
		(0.032)	(0.021)	(0.018)	(0.015)	(0.032)	(0.022)	(0.018)	(0.015)
1000	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
		(0.031)	(0.021)	(0.018)	(0.015)	(0.031)	(0.025)	(0.021)	(0.017)
1000	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2
		(0.03)	(0.021)	(0.018)	(0.015)	(0.03)	(0.024)	(0.02)	(0.017)

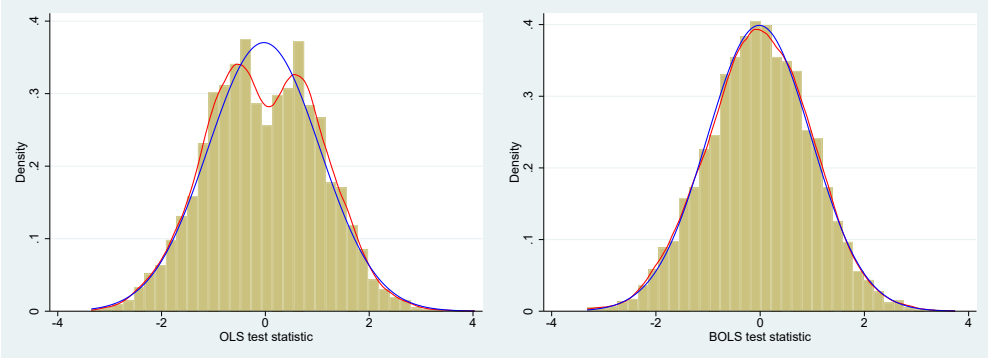
Note: The Monte Carlo simulation is based on 10000 iterations. In the parenthesis are the empirical standard errors. The rewards are drawn from a Bernoulli distribution.

Table 8: Epsilon Greedy - Epsilon 0.2

		OLS/Batch				BOLS/Batch			
N_t	$\Delta[R_{t,k}]$	1	3	5	8	1	3	5	8
100	0	-0.003	0.002	0.002	-0.0	-0.003	0.002	0.002	-0.0
		(0.202)	(0.137)	(0.115)	(0.097)	(0.202)	(0.158)	(0.132)	(0.108)
100	1	0.998	1.0	1.0	0.999	0.998	1.0	1.0	0.998
		(0.201)	(0.136)	(0.116)	(0.098)	(0.201)	(0.158)	(0.133)	(0.109)
100	2	2.004	2.0	2.0	2.0	2.004	1.999	2.0	1.999
		(0.2)	(0.137)	(0.119)	(0.1)	(0.2)	(0.158)	(0.136)	(0.11)
200	0	0.001	0.001	-0.001	-0.001	0.001	0.001	-0.0	-0.001
		(0.14)	(0.096)	(0.082)	(0.069)	(0.14)	(0.112)	(0.093)	(0.077)
200	1	1.0	0.999	1.001	1.001	1.0	0.998	1.001	1.001
		(0.143)	(0.096)	(0.083)	(0.071)	(0.143)	(0.111)	(0.094)	(0.078)
200	2	2.001	2.0	1.999	2.0	2.001	2.0	1.999	2.001
		(0.141)	(0.096)	(0.082)	(0.071)	(0.141)	(0.111)	(0.093)	(0.077)
500	0	-0.001	-0.001	-0.0	0.0	-0.001	-0.0	-0.001	0.0
		(0.089)	(0.061)	(0.052)	(0.043)	(0.089)	(0.071)	(0.059)	(0.048)
500	1	0.999	1.001	1.0	1.0	0.999	1.001	1.0	1.0
		(0.09)	(0.06)	(0.052)	(0.044)	(0.09)	(0.07)	(0.059)	(0.049)
500	2	2.0	2.0	2.001	2.0	2.0	2.0	2.001	2.0
		(0.091)	(0.061)	(0.052)	(0.044)	(0.091)	(0.071)	(0.059)	(0.049)
1000	0	0.0	0.0	0.0	0.001	0.0	-0.0	-0.0	0.001
		(0.063)	(0.043)	(0.037)	(0.031)	(0.063)	(0.05)	(0.042)	(0.035)
1000	1	1.001	1.0	1.0	1.0	1.001	0.999	1.0	1.0
		(0.063)	(0.043)	(0.037)	(0.031)	(0.063)	(0.05)	(0.042)	(0.035)
1000	2	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0
		(0.063)	(0.044)	(0.036)	(0.032)	(0.063)	(0.05)	(0.041)	(0.034)

Note: The Monte Carlo simulation is based on 10000 iterations. In the parenthesis are the empirical standard errors. The rewards are drawn from a Normal distribution with a standard deviation of 1.

Figure 10: OLS and BOLS under epsilon-greedy $N_t = 100$ at batch $t = 25$



(a) Empirical distribution of standardized OLS estimator for the margin

(b) Empirical distribution of standardized BOLS estimator for the margin