NBER WORKING PAPER SERIES

MONOPSONY IN ONLINE LABOR MARKETS

Arindrajit Dube
Jeff Jacobs
Suresh Naidu
Siddharth Suri

Working Paper 24416
http://www.nber.org/papers/w24416

NATIONAL BUREAU OF ECONOMIC RESEARCH
1050 Massachusetts Avenue
Cambridge, MA 02138
March 2018

We thank Gary Hsieh and Panos Ipeirotis for sharing data as well as Bentley Macleod, Aaron Sojourner, and Glen Weyl for helpful comments. The views expressed herein are those of the authors and do not necessarily reflect the views of the National Bureau of Economic Research.

At least one co-author has disclosed a financial relationship of potential relevance for this research. Further information is available online at http://www.nber.org/papers/w24416.ack

Monopsony in Online Labor Markets
Arindrajit Dube, Jeff Jacobs, Suresh Naidu, and Siddharth Suri
NBER Working Paper No. 24416
March 2018
JEL No. J01,J42

ABSTRACT

On-demand labor platforms make up a large part of the "gig economy." We quantify the extent of monopsony power in one of the largest on-demand labor platforms, Amazon Mechanical Turk (MTurk), by measuring the elasticity of labor supply facing the requester (employer) using both observational and experimental variation in wages. We isolate plausibly exogenous variation in rewards using a double-machine-learning estimator applied to a large dataset of scraped MTurk tasks. We also re-analyze data from 5 MTurk experiments that randomized payments to obtain corresponding experimental estimates. Both approaches yield uniformly low labor supply elasticities, around 0.1, with little heterogeneity.

Arindrajit Dube
Department of Economics
Crotty Hall
412 North Pleasant Street
University of Massachusetts
Amherst, MA 01002
and IZA
adube@econs.umass.edu

Jeff Jacobs
Columbia University
420 W 118th street
New York, NY 10027
jpj2122@columbia.edu

Suresh Naidu
Columbia University
420 West 118th Street
New York, NY 10027
and NBER
sn2430@columbia.edu

Siddharth Suri
Microsoft Research, NYC
641 Avenue of the Americas, 7th Floor
New York, NY 10011
suri@microsoft.com

1 Introduction

Online platforms are becoming an increasingly important feature of the labor market. For example, Katz and Krueger (2016) measured a roughly 50% increase in flexible work arrangements in the U.S. economy between 2005 and 2015 and estimated that this increase accounts for "94% of the net employment growth in the U.S. economy" during this time. Katz and Krueger define flexible work arrangements as involving "temporary help agency workers, on-call workers, contract workers, and independent contractors or freelancers", which includes work done via digital labor markets such as Uber, TaskRabbit or Amazon Mechanical Turk (MTurk). Other recent surveys corroborate the importance of online labor platforms (Smith (2016)). Taken as a whole, these studies show that short-term jobs allocated by online labor platforms have risen as a share of employment in recent years and will continue to do so in the near future. This raises a basic question: how competitive are online labor markets?

While a number of prior works have suggested that employers in online labor markets have a surprising degree of of market power, they have stopped short of quantifying it.[1] In this paper, we rigorously estimate the degree of requester market power in a widely-used online labor market – Amazon Mechanical Turk. This is the most popular online micro-task platform, allowing requesters (employers) to post jobs which workers can complete for pay.

We provide initial evidence regarding how sensitive the duration of task vacancies are to task rewards, using data from a near-universe of tasks scraped from MTurk. This evidence thus provides us with an estimate of wage-setting (monopsony) power facing task requesters (Manning (2003); Card et al. (2016)). We isolate plausibly exogenous variation in rewards using a double-machine-learning (Chernozhukov et al. (2017)) method, which controls for a highly predictive function of observables generated from the textual and numeric metadata of each task.

We then present results from a number of independent experiments on the sensitivity of workers' acceptance of tasks to the level of pay offered. We analyze data from 5 previous experiments that randomized wages of MTurk subjects, with the full list of experiments we surveyed given in Appendix B. While the previous experimenters had randomly varied the wage, none except (Dube et al. (2017)) recognized that they had estimated a task-specific labor supply curve, nor noticed that this reflected monopsony power on the MTurk marketplace. We empirically estimate both a "recruitment" elasticity where workers see a reward and associated task as part of their normal browsing for jobs, and a "retention" elasticity where workers, having

[1] Kingsley et al. (2015) describe a variety of reasons why market power persists in the MTurk marketplace. They document (i) an information asymmetry where employers have more information on workers than *vice versa*, (ii) a serious amount of market concentration where 10% of the requesters post over 98% of the tasks, and (iii) *ex ante* wage rates set by the requester.

already accepted a task, are given an opportunity to perform additional work for a randomized bonus payment. The recruitment elasticity is based on a novel "honeypot" experimental design, where randomly-varied wage offers were made observable only to random subsets of MTurk workers.

Together, these very different pieces of evidence provide a remarkably consistent estimate of the labor supply elasticity facing MTurk requesters, indicating the robustness of our results. The three experiments with a "honeypot" design suggest a *recruitment* elasticity between 0.05 and 0.11. Similarly, *retention* probabilities do not increase very much as a function of reward posted, with implied retention elasticities in the 0.1 to 0.5 range for the two experiments using that design. The precision-weighted average experimental requester's labor supply elasiticity is 0.14, and in particular the pooled recruitment elasticity is 0.06, remarkably close to the corresponding 0.08 estimate produced by our preferred double-ML specification. The estimates are uniformly small across subsamples, with little heterogeneity by reward amount. This close agreement suggests that the constant elasticity specification commonly used in the literature may not be a bad approximation in this context. As a further contribution, our paper provides an independent – and favorable – assessment of the double-ML estimator against an experimental benchmark.

The rest of this paper is structured as follows. Section 2 outlines a model of monopsony in a task market. Section 3 uses double-machine-learning to isolate plausibly-exogenous variation in wages when estimating the requester's labor supply function from observational data, then uses these double-ML estimates to assess external validity of the experimental estimates. Section 4 provides experimental evidence on the extent of labor market power on MTurk. Section 5 concludes.

2 Monopsony in A Task Market

Monopsony is characterized by two features: wage-setting power and inability to wage-discriminate. MTurk, with its task-posting structure, did not offer many margins for wage-discrimination until very recently (after our sample period). In our sample period, requesters could only restrict the set of eligible workers based on prior acceptance rates (the rates at which previous requesters had deemed their work satisfactory) or location (e.g., India or the United States)

Monopsony power may arise due to a small number of employers on the platform, from search frictions in locating higher paying tasks, or from idiosyncratic preferences over task characteristics. Given the sizable number of requesters per worker, and the fact that tasks are centrally posted on MTurk, we posit that a model based on discrete choice over differentiated tasks – as in Card et al. (2016) – provides the most

compelling explanation for the presence of monopsony power in online platforms. In their model, individuals have idiosyncratic, privately observed tastes over each job, creating an upward sloping labor supply curve facing each firm.

In Appendix A we present a nested logit model of the MTurk market, with the first nest being the "recruitment" margin, that brings workers into a HIT (Human Intelligence Task) batch, and the second nest being the "retention" margin that incentivizes workers to complete tasks within the batch. The model illustrates how requesters will jointly set both initial wages and bonus wages, and shows why these two wages need independent experiments in order to estimate the two elasticities. The model also yields the formula for the sum of both wages as a markdown on productivity that depends on both the recruitment elasticity as well as the retention elasticity.

While we proceed with a random utility interpretation of the nested logit, Fosgerau et al. (2016) show a generic equivalence between rational inattention and random utility based discrete choice models (in particular the nested logit can be expressed as a rational inattention model with a generalized Shannon entropy cost of information processing). While MTurk makes many work options available and easy to find, employers may have outsized market power either due to idiosyncratic tastes of workers for particular tasks (random utility) or due to costly information processing that makes it difficult to discern which task is best (rational inattention).

3 Observational Evidence on Recruitment Elasticity from MTurk

3.1 Data and empirical strategy

For our observational analysis, we use two primary sources of scraped MTurk data. The first dataset was obtained from Ipeirotis (2010), and covers the January 2014 to February 2016 period. The data consists of over 400,000 scraped HIT batches from the Mechanical Turk Tracker web API[2]. This scraper downloaded the newest 200 HIT batches posted to MTurk every six minutes, then the status page for each discovered HIT batch was checked every minute until the page reported that all HITs in the batch had been accepted.

Beginning in May 2016 we launched our own scraper, which took snapshots of all HIT batches on MTurk every 30 minutes, later increased to every 10 minutes beginning in March 2017. This scraping strategy may miss batches that are posted and filled too quickly for the scraper to detect (i.e. duration less than 30 or 10 minutes). This scraping strategy yielded over 300,000 HIT batches, but stopped working on August 22,

[2] `http://crowd-power.appspot.com/#/general`

2017, and we have been unable to collect more data since then. We show results separately for these two datasets, and find broadly similar results. Further details on the data are in Appendix C, including densities of the log durations showing similar support.

We use the time it takes for a posted task to disappear as a measure of the probability of acceptance, and regress the duration of the task posting on the observed reward to obtain an estimate of η_1, the recruitment elasticity. We take advantage of the vast amount of available online crowdsourcing data to estinate η, using high-dimensional regression adjustment applied to numeric and textual characteristics of the tasks to control for possible sources of endogenous task characteristics.

The resulting linear specification is estimated on observations of HIT batch durations and rewards, and is given by:

$$\ln(duration_h) = -\eta_1 \ln(reward_h) + \nu_h + \epsilon_h \tag{1}$$

Where ν is a nuisance parameter that is correlated with both rewards and durations, and ϵ is an error term that is conditionally independent of durations, so $E[\epsilon|\nu] = 0$. An unbiased estimate of η_1 requires that we correctly control for ν, the determinants of duration that are correlated with rewards, in particular labor demand. The virtue of the experimental estimates in the fourth section is that randomization ensures that ν is independent of $\log(reward)$. With observational data, we must rely on a sufficiently rich set of observables to control for ν, and it is impossible to be completely confident that all possible sources of omitted variable bias have been eliminated. However, the large and high-dimensional nature of the observational MTurk data lets us push the limits of observational analysis. We use two different approaches for this analysis – fixed effects regression and double-machine-learning.

3.2 Fixed-Effects Regression

In our first strategy, we control for requester and time fixed effects, along with fixed effects for deciles of the time allotted by the requester, with the idea that much variation in labor demand is idiosyncratic to a given requester asking a given task in a distinct time period. Formally, we assume that $\nu = \rho_r + \tau_t + \delta_d$. This says that the unobserved relative task attractiveness is captured by the identity of the employer, the time the task is first posted, and the decile of the time allotted for the task d.

We can then estimate a standard fixed-effects regression:

$$\ln(duration) = -\eta_1 \ln(reward) + \rho_r + \tau_t + \delta_d + \epsilon \tag{2}$$

3.3 Double Machine Learning

As our second approach, we implement a "double-machine-learning"(double-ML) estimator recently developed by Chernozhukov et al. (2017), which in our case uses an ensemble machine learning approach to model the unobserved ν.

In particular, we suppose that ν in equation 1 is equal to $g_0(Z)$, an unknown function of a high-dimensional vector of observable variables Z. We further suppose that variation in rewards is generated by another function of Z so that $\log(rewards) = m_0(Z) + \mu$. Combining these two equations we get:

$$\ln(duration) = -\eta_1 \ln(reward) + g_0(Z) + \epsilon,\ E[\epsilon|Z, \ln(reward)] = 0 \tag{3}$$

$$\ln(reward) = m_0(Z) + \mu,\ E[\mu|Z] = 0, \tag{4}$$

and we obtain an unbiased and $\sqrt{n}$-consistent estimator of η_1 via the double-ML procedure Chernozhukov et al. (2017). The benefit of this procedure stems from the fact that it allows us to utilize any number of state-of-the-art machine learning methods, such as neural nets or random forests, to obtain estimates of the conditional expectation functions $\widehat{g_0}(Z) = E[\widehat{log(duration)}|Z]$ and $\widehat{m_0}(Z) = E[\widehat{log(rewards)}|Z]$ which are then "de-biased" to obtain our desired estimator $\check{\eta_1}$. Specifically, from our machine learning-estimated $\widehat{g_0}(Z)$ and $\widehat{m_0}(Z)$ we can compute the residuals from (3) and (4) as $\widehat{\mu} = log(reward) - \widehat{m_0}(Z)$ and $\widehat{\epsilon} = log(duration) - \widehat{g_0}(Z)$, respectively, and use these residuals to compute the final estimator as

$$\check{\eta_1}^0 = \left(\frac{1}{n}\sum_{i=1}^{n} \widehat{\mu_i}^2\right)^{-1} \frac{1}{n}\sum_{i=1}^{n} \widehat{\mu_i}\widehat{\epsilon_i}, \tag{5}$$

The bias from overfitting will not asymptotically go to 0 if the same data is used to estimate $g_0(Z)$ and $m_0(Z)$ and η_1. However, if a different sample is used to estimate $g_0(Z)$ and $m_0(Z)$ and $\check{\eta_1}$ is averaged over multiple folds, then the estimator is consistent and unbiased.

The intuition behind this estimator is similar to the classic partial regression formula. In equation 1 the partial regression formula implies that η_1 could be recovered from a regression of $E[\ln(duration)|\nu]$ on $E[\ln(reward)|\nu]$. The double-ML estimator uses machine learning to form proxies for ν that fit both conditional expectations very well, implying that the resulting residuals have "partialled out" a very flexible function of all covariates that capture as much of the variation as possible.

Double-machine-learning allows us to leverage a large number of covariates for identifying causal effects, using whichever prediction algorithm has highest goodness-of-fit (e.g. R^2) in held-out data. We construct a large set of covariates, in this case n-grams and other textual features derived from the HIT batch titles, descriptions, and keywords, as well as numeric features such as the number of HIT tasks posted in the batch, time allotted, and time posted. We experiment with a number of methods, discussed below, with parameters tuned via 5-fold cross-validation.

3.3.1 HIT Features Used For Prediction

We use both non-textual and textual features as inputs to the double-ML procedure. First, we use non-textual features from the HIT including information about the batch size, time allotted for each HIT in the group by the requester, time remaining before expiration of the HIT group, required qualifications (e.g., worker acceptance rate required to be above $x\%$), the volume of HIT groups posted by the requester across the marketplace, and so on (the full set of features is described in Appendix D.3).

Additionally, we generate four distinct types of textual features from each HIT group's description, title, and list of keywords: n-grams, topic distributions, Doc2Vec embeddings, and hand-engineered features. The details can be found in Appendix D.

3.3.2 Double Machine Learning Procedure

To satisfy the sample-splitting requirement of the double-ML estimator (Chernozhukov et al., 2017), the full set of HIT groups is split into two equally-sized subsets, A and B. Each subset is further split into training and validation sets, with 80% of the observations in A going into A_{train} and 20% into A_{val}, and similarly for B_{train} and B_{val}. The machine learning then proceeds in two "stages".

In the first stage, the n-gram features are computed for A_{train} and B_{train}, and two series of learning algorithms are run, the first with A_{train} as training data and A_{val} as test data, the second with B_{train} as training data and B_{val} as test data. For each series and each dataset (Ipeirotis (2010) and our own scraped data) the algorithm which achieves the highest total validation score (here the sum of validation scores for reward prediction and duration prediction) is selected as the "final" algorithm to be used for the remainder of the procedure[3].

[3]In every case, scikit-learn's RandomForestRegressor achieved the highest score out of {AdaBoostRegressor, BaggingRegressor, ExtraTreesRegressor, GradientBoostingRegressor, RandomForestRegressor, SVR (SupportVectorRegressor)}. The random forest regression constructs a series of decision trees, each of which is built based on a random subset of all available features, and takes the mean prediction over all of these trees to be the estimate (thus random forest regressions are a type of ensemble method). For more on random forest regression, see Breiman (2001), Section 11.

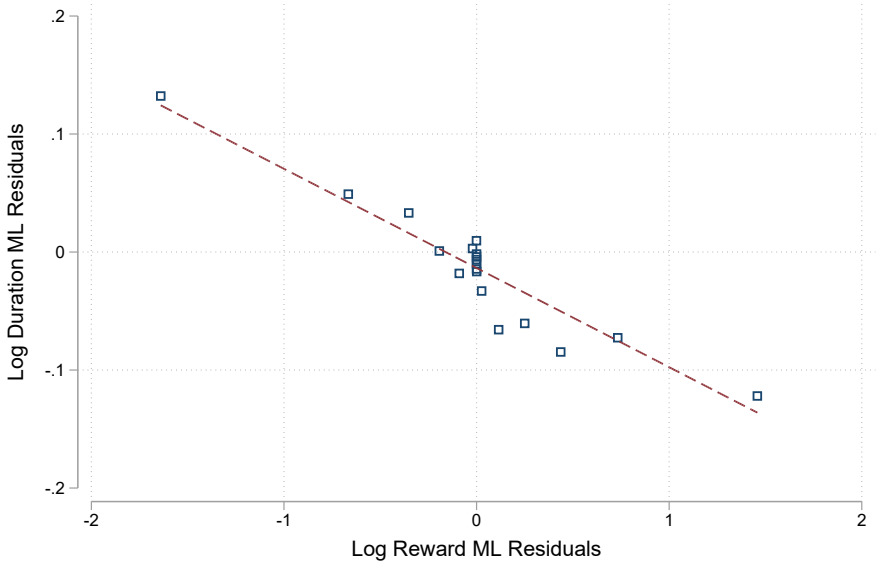

Figure 1: Binned scatterplot (20 ventiles) for double-ML residuals of log duration and log rewards. Residuals are calculated as difference between observed value and predicted value from a random forest trained on a held-out sample, as described in Section 3.3.2.

To begin the second stage of the procedure, we select the 100 textual features which best predicted the reward values in the first stage, along with the 100 which best predicted the duration values, and set these as the first 200 columns of our second-stage feature matrix. The additional text features described in Section 3.3.1 are then appended to the matrix, along with numeric features described in Appendix D. The "final" algorithm discovered in stage one is then run twice, the first time with the HIT groups in A used as training data and groups in B used as test data, and the second with the training and test sets reversed. These two values are then averaged (so as to satisfy the sample-splitting requirement of the double-ML estimator) as specified in equation 5 to produce the final estimate $\check{\eta_1}^0$ (along with its standard error) for each dataset.

3.4 Results

In Table 1 we present basic OLS results, fixed effects regressions, and the double-ML regressions (with and without fixed effects). Column 1 shows the simple bivariate regression of log duration on log reward. Unsurprisingly this regression is inconclusive, likely because of extensive omitted variables that are correlated with task attractiveness and the intensity of requester demand, both of which would be correlated with both the reward posted as well as the time until the HIT is filled.

Column 2 implements the fixed-effects specification, controlling for deciles of time allotted for the task as

Table 1: Duration Elasticities from Observational MTurk Data

	(1)	(2)	(3)	(4)	(5)	(6)	(7)
Log Reward	0.186	-0.0708					
	(0.0947)	(0.0672)					
Log Reward-ML res.			-0.0841	-0.0589	-0.0762	-0.169	-0.0373
			(0.00820)	(0.0108)	(0.0169)	(0.0250)	(0.00429)
N	644873	629756	644873	629756	93775	292746	258352
Clusters	41167	26050	41167	26050	6962	18340	24923
Type	OLS	FE	ML	ML-FE	ML	ML	ML
Data	Pooled	Pooled	Pooled	Pooled	2017	2016-2017	2014-2016

Notes: This Table presents OLS, FE, and Double-ML estimates using the data obtained from scraping MTurk between Jan. 2014 and Feb. 2016 (from Ipeirotis (2010)), May 2016-March 2017 (scraped every 30 minutes), and March-August 2017 (scraped every 10 minutes). FE indicates fixed-effects for the hour of the first appearance of the HIT, requester, and time allotted fixed effects. Standard errors are clustered at the requester level.

well as fixed effects for requester and the time posted described above. The coefficient on log reward is -0.07. Columns 3 through 7 show the results from the double-ML estimator. Column 3 shows the bivariate OLS regression of residualized durations on residualized rewards, and here the coefficient on residualized rewards is a strongly significant -0.08. Figure 1 shows the corresponding binned scatterplot, which shows the binned residuals falling quite close to the linear fit implied by a constant elasticity. Column 4 in Table 1 adds the fixed effects from Column 2 to the ML specification, and obtains a quite similar estimate of -0.06, suggesting that the double-ML procedure is effectively purging the effects of observable variables omitted from Column 1 (as a large change in the coefficient would suggest that there were other unobserved variables confounding the regression). Columns 5-7 show the double-ML specifications for the different scraped samples. While there is some heterogeneity, the implied elasticities are uniformly small.

4 Experimental Evidence on Labor Supply Elasticity Facing Requesters on MTurk

The observational evidence is quite suggestive of a low requester's recruitment elasticity, η_1, but even in the double-ML estimates concerns about omitted variable bias may linger. It is possible that not all task-relevant characteristics have been adequately controlled for, despite the high predictive power of our conditional expectation functions above. If we have experimental (random) variation in rewards, we can estimate the following regression:

$$Pr(Accept) = \alpha + \beta reward + \epsilon \tag{6}$$

and an estimate of η would be recovered by $\eta = \beta \times \frac{E[reward]}{Pr(Accept)}$ with the expectation taken over the population of workers in the sample. We can compare this estimate of η to the double-ML estimate from the observational data above to bolster our confidence in our estimate: if both estimates yield similar results it suggests that the double-ML estimator is indeed adequately controlling for unobserved variation, and that the experimental estimates are externally valid. In our framework above, however, the η are differentiated depending on whether the acceptance decision is the initial "recruitment" (η_1), or the subsequent "retention" decision (η_2). Experimentally estimated η_1's are the most directly comparable to the Double-ML estimates, but experimental estimates of η_2 are available from high-powered experiments, as detailed in the next subsection. We then describe how "honeypot" experiments can be engineered to estimate η_1 experimentally and show results from existing experiments.

4.1 Experimental Retention Elasticities

Horton et al. (2011) and Dube et al. (2017) both run variants of the following experiment. A simple uniformly priced (say, 10 cent) HIT is posted. Subjects give demographic information and perform a simple task (e.g., tagging an image). The subjects are then asked if they would like to perform a given number of additional identical tasks for a randomized bonus wage. The change in the probability of acceptance as a function of the wage gives the responsiveness of requester's labor supply to random wage posting, with low values suggesting a great deal of market power. This is a "retention" elasticity, η_2, as workers have already been drawn into a HIT i when asked whether they wish to continnue.

Experiment 1 was conducted by Horton et al. (2011), and was among the earliest attempts to estimate economic parameters from MTurk. The authors aimed to elicit the labor-supply elasticity of online workers to the market, but this design does not elicit the market labor supply, but rather the requester's labor supply (i.e., the supply to the experimenter/requester for the particular task). The task in this experiment was transcribing Tagalog translations of paragraphs from Adam Smith's *The Theory of Moral Sentiments*.

Experiment 2 was conducted by Dube et al. (2017) in 2016, deliberately emulating the design of the Horton et al. (2011) study with the aim of testing for left-digit bias in the requester's labor supply of online workers. Hence the rewards are substantially lower, between 5 and 15 cents, but the sample sizes are correspondingly larger. The task here was tagging sheets of the 1850 US census slave schedules for the presence of marks in the fugitive slave columns.

We show results for both the full sample and sophisticates (defined as working more than 10 hours on MTurk and primarily for money). The resulting requester's labor supply elasticities are shown in Columns

Table 2: Offer Acceptance and Offered Rewards from Retention Experiments

	(1)	(2)	(3)	(4)
	Panel A: Horton et al. 2011 Probability of Accepting Offer			
Reward	0.127	0.140	0.0861	0.0973
	(0.0219)	(0.0241)	(0.0292)	(0.0333)
N	328	307	125	107
η_2	0.234	0.241	0.192	0.202
SE	0.0334	0.0364	0.0594	0.0664
Avg. Reward	11.60	11.63	11.37	11.50
Sophisticated	No	No	Yes	Yes
Controls	No	Yes	No	Yes
	Panel B: Dube et al. 2017 Probability of Accepting Offer			
Reward	0.0267	0.0486	0.0764	0.0782
	(0.0171)	(0.0202)	(0.0348)	(0.0329)
Controls	No	Yes	No	Yes
N	5184	5017	1702	1618
η_2	0.052	0.077	0.118	0.114
SE	0.0333	0.0322	0.0534	0.0479
Avg. Reward	9	9	9	9
Sophisticated	No	No	Yes	Yes

Notes: Coefficients from logit regressions of accept indicator on log reward from "retention" experiments, and calculated elasticities, assessed at the specification sample mean. Robust standard errors in parenthesis.

1-4 of Table 2. The implied η_2 from the Horton et al. estimates are quite low, between 0.19 and 0.25, while implied η_2 from the Dube et al. estimates are even lower, always below 0.12. Besides differences in the tasks, one likely reason for the very slight difference is the different support of the reward variation (Dube et al. randomize between 5 and 15 cents, while Horton et al. randomize between 10 and 25 cents), and the composition of workers and requesters likely changed considerably between 2011 and 2016. But broadly the estimates are quite comparable, despite these differences.

4.2 Experimental Recruitment Elasticities

Engineering an experiment to test the recruitment elasticity is much more challenging than estimating the retention elasticity. We take advantage of three pieces of prior work – Ho et al. (2015), Hsieh and Kocielnik (2016), and Yin et al. (2018) – that presented tasks with varying pay rates to random subsets of the MTurk population such that workers assigned one pay rate could not see the tasks available to other workers who had a different pay rate. We stress that none of the papers actually estimated a labor supply elasticity using this random variation in pay.

All of these experiments use a two-phase "honeypot" design. In the first phase a generic HIT is posted at a fixed pay rate. In this simple task, workers are asked a couple of survey questions including whether they would like to be notified of future work opportunities. The IDs of the workers who said yes are then randomized into treatment conditions. During the second phase of the experiment HITs corresponding to the different treatment conditions are launched with identical tasks but varying rewards. This design uses a relatively obscure piece of the MTurk API that lets a requester make a HIT group visible to only a subset of workers. Thus each HIT group can only be seen by and accepted by those treated, and it appears as a regular HIT group in the MTurk interface for them. This design, which first appeared in Section 5 of Ho et al. (2015) and was later refined in Yin et al. (2018), replicates the search environment workers are facing before having said yes to the task.

In the first experiment (Ho et al. (2015)), 800 people were recruited via a 0.05 cent "honeypot" HIT, and then randomly split into four treatment groups of 200 workers each. Three groups were given \$0.50 to complete the HIT (one control, one with a surprise \$1.00 bonus, and one with a performance based bonus), and a fourth group was given \$1.50. We drop the group that was given an additional performance-based incentive to focus on the recruitment elasticity.

In the second experiment (Yin et al. (2018)), 1,800 workers recruited using the same "honey pot" protocol were randomly split into three treatment groups, with rewards for the additional task of \$0.03, \$0.04, and \$0.05, respectively. For the task itself, users were asked to categorize an Amazon.com review as positive or negative. Of the 600 in each group, 357 in the \$0.03 group accepted, 351 in the \$0.04 group accepted, and 371 in the \$0.05 group accepted.

In the third experiment (Hsieh and Kocielnik (2016)), 927 workers were recruited via a similar design, with the task being to brainstorm the "number of uses of a brick" (a measure of creative thinking) and given one of 7 random rewards: 0 cents, 5 cents, 25 cents, 1% chance of \$5, 1% chance of \$25, and 25 and 50 cent donation to charity. We drop the lottery and charity treatments and examine only the variation in rewards (0, 5 or 25 cents) , which leaves us with 338 observations. Of these, 131 were in the 0 cent reward group (68 accepted), 89 were in the 5 cent group (52 accepted), and 118 were in the 25 cent group (82 accepted). We made a synthetic dataset based on these numbers in communication with the authors, as the replication data was unavailable.

Neither of the first two experiments asked demographic characteristics, and replication data for the third is unavailable, so there is limited capacity to control for observables. However, the randomized assignment of the reward mitigates any role for covariates besides improving precision. Table 3 shows the simple OLS

Table 3: Recruitment Elasticities From Three Experiments

	(1)	(2)	(3)	(4)
Reward	0.00186	0.0451	0.0287	0.00744
	(0.00188)	(0.0587)	(0.0104)	(0.00385)
N	600	1800	338	2738
η_1	0.0497	0.0724	0.115	0.0610
SE	0.0503	0.0944	0.0417	0.0290
Avg. Reward	83.33	4	10.04	22.13
Experiment	Spot Diff.	Tag Ads	Brainstorming	Pooled

Notes: Coefficients from logit regressions of an accept indicator on reward from "recruitment" experiments, and calculated elasticities, assessed at the experimental sample mean. The pooled specification includes experiment fixed effects, and is weighted by the inverse of the standard deviation of rewards within each experiment. Robust standard errors in parentheses.

regression results using the same logit specification as equation 6, separately by experiment, and then pooled, controlling for experiment fixed effects and weighting by the inverse of the standard deviation of rewards within each experiment.

While not strongly significant, even when all experiments are pooled, the point estimates are remarkably similar despite the very different wage levels at which the experiments were run, and close to the very small estimates obtained from the double-ML procedure above. The implied recruitment elasticity from the pooled three experiments is 0.06 and is distinguishable from 0 at 5% significance. While the first 2 experiments have insignificant elasticities, in the third experiment we obtain a statistically significant, but still small elasticity, despite a smaller sample size, possibly due to the more attractive nature of the ex-post task relative to the other two.

4.3 Comparison of Estimates

Figure 2 shows the estimates from each of the experiments, together with the double-ML estimates obtained from the two samples, each split by quintiles of the reward distribution in each of the two scraped samples. The consistency of the estimates is remarkable, and generally implies a low labor supply elasticity facing requesters on MTurk, with some estimates unable to rule out 0 with 95% confidence.

While the double-ML estimates are again quite consistent with the experimental retention elasticities, the more natural comparison is with the experimental recruitment elasticities. In this domain the larger sample from the 2014-2016 period is numerically quite close to the experimental elasticities (which were run in this period), while the 2016-2017 estimate is slightly larger.

The graph also plots the precision-weighted mean experimental elasticity (weighted by the inverse of the

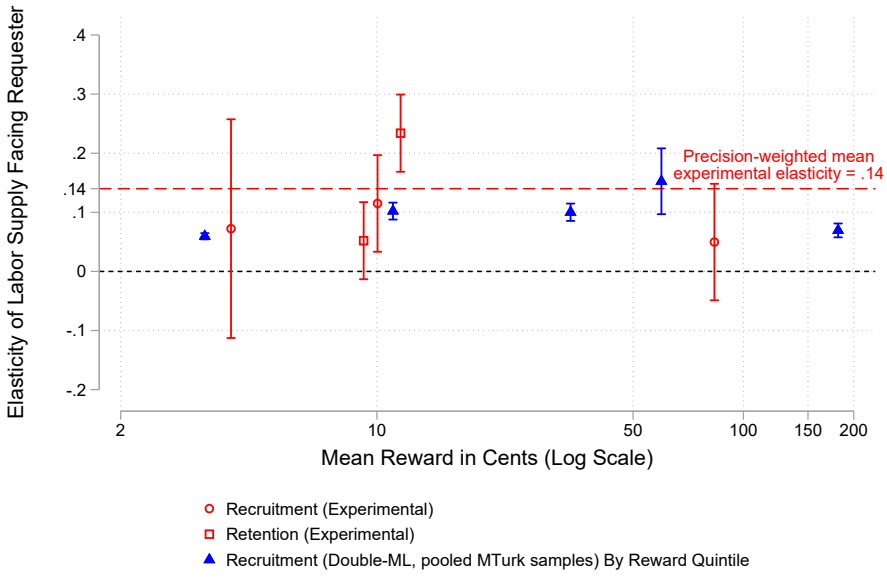

Figure 2: Baseline estimates from both "recruitment" and "retention" experimental designs (Column 1 of Table 2 and Columns 1-3 of Table 3), as well as Double-ML recruitment elasticities from observational data, plotted by mean reward of sample.

variance of the estimated elasticities) of 0.14. The double-ML estimates are all very close to this line, despite being estimated using very different sources of variation in the rewards.

We can use our estimates to infer the distribution of MTurk surplus between workers and requesters, following the formula in Appendix A. The markdowns are quite large, with workers paid less than 20% of their productivity. These are close to the markdowns implied by firm's labor supply elasticities estimated for nurses by Staiger et al. (2010), among the lowest in the literature.

5 Discussion and Conclusion

In his review of Manning's 2003 book Monopsony in Motion, Peter Kuhn made the following conjecture. "[U]pward-sloping labor supply curves—whether induced by search or other factors—seem unlikely to me to be a serious constraint for most firms. This seems even more likely to be the case in the near future, as ... information technology has the potential to reduce search frictions." (Kuhn, 2004, pp. 376). The emergence of online platforms represents an idealized environment where search costs are presumably very low. Despite this, we find a highly robust and surprisingly high degree of market power even in this large and diverse spot labor market.

While monopsony power over small stakes tasks on MTurk may seem unimportant, our results have important implications for platform design and the distribution of gains from digital labor markets, which are likely to become more important over time. If one believes that requesters are fully exploiting their monopsony power (as the calibration of requester misoptimization in Dube et al. (2017) suggests), then the markdown of productivity in the wage is considerable, with workers paid less than 20% of their productivity. This suggests that much of the surplus created by this online labor market platform is captured by employers.

The source of the monopsony power on MTurk likely lies in the information and market environment presented to workers and requesters, together with the absence of bargaining or many margins of wage discrimination. In particular, the tastes different workers have for a given task may be quite dispersed and not easily discerned by requesters, which induces requesters posting a wage to trade-off the probability of acceptance against a lower wage. Further, this may be exacerbated by the information environment facing workers, which makes searching for alternative jobs difficult. Jobs are highly heterogeneous in time required, entertainment value ("fun") to the worker, and the reliability of the requester in approving payments (Benson et al., 2017). There is no single dimensional index of job quality that can be used to order HIT groups while searching: workers can't sort HIT groups by the *real* wage. Potentially this could be easily remedied: indeed our double-ML predictions of HIT duration could be used to discount rewards by a summary measure of task attractiveness.

Why does this inefficient market structure persist? While answering this is beyond the scope of this paper, one possibility is that MTurk itself has a considerable amount of market power as a platform for crowdsourcing. Amazon's market power as a platform may allow it to underinvest in market design, allowing inefficient market structures to persist. However, the recent addition of "premium" categories of workers for requester targeting may signal Amazon's attempt to eliminate inefficiencies created by requester market power, while preserving the unequal distribution of the surplus between requesters and workers. If requesters are the more elastic side of the crowdsourcing market, it may be optimal for Amazon to provide a platform that enables the bulk of the surplus to be captured by employers rather than workers.

MTurk workers and their advocates have long noted the asymmetry in market structure amongst themselves. Others (e.g. Arrieta-Ibarra et al. (2017)) have pointed out that monopsony in data markets may lead to inefficiently small and low-quality sample sizes in machine learning or survey applications. Both efficiency and equality concerns have led to the rise of competing, "worker-friendly" platforms such as Stanford's Dynamo, mechanisms for sharing information about good and bad requesters and HITs such as Turkopticon and online discussion fora. Scientific funders such as Russell Sage have instituted minimum wages for crowd-

sourced work. Our results suggest that these sentiments and policies may have an economic justification.

References

Abernethy, Jacob, Yiling Chen, Chien-Ju Ho, and Bo Waggoner, "Low-Cost Learning via Active Data Procurement," in "Proceedings of the Sixteenth ACM Conference on Economics and Computation" EC '15 ACM New York, NY, USA 2015, pp. 619–636.

Arrieta-Ibarra, Imanol, Leonard Goff, Diego Jiménez Hernández, Jaron Lanier, and E Weyl, "Should We Treat Data as Labor? Moving Beyond'Free'," 2017.

Benson, Alan, Aaron Sojourner, and Akhmed Umyarov, "The value of employer reputation in the absence of contract enforcement: A randomized experiment," 2017.

Berinsky, Adam J, Gregory A Huber, and Gabriel S Lenz, "Evaluating Online Labor Markets for Experimental Research: Amazon.com's Mechanical Turk," *Political Analysis*, 2012, *20* (3), 351–368.

Blei, David M, Andrew Y Ng, and Michael I Jordan, "Latent Dirichlet Allocation," *Journal of Machine Learning Research*, mar 2003, *3*, 993–1022.

Breiman, Leo, "Random Forests," *Mach. Learn.*, October 2001, *45* (1), 5–32.

Buhrmester, Michael, Tracy Kwang, and Samuel D. Gosling, "Amazon's Mechanical Turk: A New Source of Inexpensive, Yet High-Quality, Data?," *Perspectives on Psychological Science*, 2011, *6* (1), 3–5.

Callison-Burch, Chris, "Crowd-workers: Aggregating information across turkers to help them find higher paying work," in "Second AAAI Conference on Human Computation and Crowdsourcing" 2014.

Card, David, Ana Rute Cardoso, Jörg Heining, and Patrick Kline, "Firms and labor market inequality: Evidence and some theory," Technical Report, National Bureau of Economic Research 2016.

Chandler, Dana and John Horton, "Labor Allocation in Paid Crowdsourcing: Experimental Evidence on Positioning, Nudges and Prices," 2011.

Chernozhukov, Victor, Denis Chetverikov, Mert Demirer, Esther Duflo, Christian Hansen, Whitney Newey, and James Robins, "Double/debiased machine learning for treatment and structural parameters," *The Econometrics Journal*, 2017.

Crump, Matthew J. C., John V. McDonnell, and Todd M. Gureckis, "Evaluating Amazon's Mechanical Turk as a Tool for Experimental Behavioral Research," *PLOS ONE*, 03 2013, *8* (3), 1–18.

Dasgupta, Anirban and Arpita Ghosh, "Crowdsourced Judgement Elicitation with Endogenous Proficiency," in "Proceedings of the 22Nd International Conference on World Wide Web" WWW '13 ACM New York, NY, USA 2013, pp. 319–330.

Deng, Jia, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei, "Imagenet: A large-scale hierarchical image database," in "Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on" IEEE 2009, pp. 248–255.

Difallah, Djellel Eddine, Michele Catasta, Gianluca Demartini, and Philippe Cudré-Mauroux, "Scaling-Up the Crowd: Micro-Task Pricing Schemes for Worker Retention and Latency Improvement," in "HCOMP" 2014.

_ , _ , _ , Panagiotis G. Ipeirotis, and Philippe Cudré-Mauroux, "The Dynamics of Micro-Task Crowdsourcing: The Case of Amazon MTurk," in "Proceedings of the 24th International Conference on World Wide Web" WWW '15 International World Wide Web Conferences Steering Committee Republic and Canton of Geneva, Switzerland 2015, pp. 238–247.

Doerrenberg, Philipp, Denvil Duncan, and Max Löffler, "Asymmetric labor-supply responses to wage-rate changes: Evidence from a field experiment," 2016, (16-006).

Dube, Arindrajit, Alan Manning, and Suresh Naidu, "Monopsony, Misoptimization, and Round Number Bunching in the Wage Distribution," 2017.

Fosgerau, Mogens, Emerson Melo, and Matthew Shum, "Discrete choice and rational inattention: A general equivalence result," 2016.

Gadiraju, Ujwal, Ricardo Kawase, and Stefan Dietze, "A Taxonomy of Microtasks on the Web," in "Proceedings of the 25th ACM Conference on Hypertext and Social Media" HT '14 ACM New York, NY, USA 2014, pp. 218–223.

Heer, Jeffrey and Michael Bostock, "Crowdsourcing Graphical Perception: Using Mechanical Turk to Assess Visualization Design," in "Proceedings of the SIGCHI Conference on Human Factors in Computing Systems" CHI '10 ACM New York, NY, USA 2010, pp. 203–212.

Ho, Chien-Ju, Aleksandrs Slivkins, Siddharth Suri, and Jennifer Wortman Vaughan, "Incentivizing High Quality Crowdwork," in "Proceedings of the 24th International Conference on World Wide Web" WWW '15 International World Wide Web Conferences Steering Committee Republic and Canton of Geneva, Switzerland 2015, pp. 419–429.

Hofmann, Thomas, Bernhard Schölkopf, and Alexander J. Smola, "Kernel methods in machine learning," *Ann. Statist.*, 06 2008, *36* (3), 1171–1220.

Honnibal, Matthew and Mark Johnson, "An Improved Non-monotonic Transition System for Dependency Parsing," in "Conference on Empirical Methods in Natural Language Processing" 2015.

Horton, John J, David G Rand, and Richard J Zeckhauser, "The online laboratory: Conducting experiments in a real labor market," *Experimental Economics*, 2011, *14* (3), 399–425.

Horton, John Joseph and Lydia B. Chilton, "The Labor Economics of Paid Crowdsourcing," in "Proceedings of the 11th ACM Conference on Electronic Commerce" EC '10 ACM New York, NY, USA 2010, pp. 209–218.

Hsieh, Gary and Rafał Kocielnik, "You get who you pay for: The impact of incentives on participation bias," in "Proceedings of the 19th ACM Conference on Computer-Supported Cooperative Work & Social Computing" ACM 2016, pp. 823–835.

Huang, Eric, Haoqi Zhang, David C. Parkes, Krzysztof Z. Gajos, and Yiling Chen, "Toward Automatic Task Design: A Progress Report," in "Proceedings of the ACM SIGKDD Workshop on Human Computation" HCOMP '10 ACM New York, NY, USA 2010, pp. 77–85.

Ipeirotis, Panagiotis G., "Analyzing the Amazon Mechanical Turk Marketplace," *XRDS*, December 2010, *17* (2), 16–21.

Katz, Lawrence F. and Alan B. Krueger, "The Rise and Nature of Alternative Work Arrangements in the United States, 1995-2015," Working Paper 22667, National Bureau of Economic Research September 2016.

Kingsley, Sara Constance, Mary L. Gray, and Siddharth Suri, "Accounting for Market Frictions and Power Asymmetries in Online Labor Markets," *Policy & Internet*, 2015, *7* (4), 383–400.

Kuhn, Peter, "Is monopsony the right way to model labor markets? a review of Alan Manning's monopsony in motion," *International Journal of the Economics of Business*, 2004, *11* (3), 369–378.

Le, Quoc and Tomas Mikolov, "Distributed Representations of Sentences and Documents," in Eric P. Xing and Tony Jebara, eds., *Proceedings of the 31st International Conference on Machine Learning*, Vol. 32 of *Proceedings of Machine Learning Research* PMLR Bejing, China 22–24 Jun 2014, pp. 1188–1196.

Lindley, Dennis V, "The choice of sample size," *Journal of the Royal Statistical Society: Series D (The Statistician)*, 1997, *46* (2), 129–138.

Manning, Alan, *Monopsony in motion: Imperfect competition in labor markets*, Princeton University Press, 2003.

Manyika, James, Susan Lund, Kelsey Robinson, John Valentino, and Richard Dobbs, "A labor market that works: Connecting talent with opportunity in the digital age," *McKinsey Global Institute*, 2015.

Marge, Matthew, Satanjeev Banerjee, and Alexander I Rudnicky, "Using the Amazon Mechanical Turk for transcription of spoken language," in "Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on" IEEE 2010, pp. 5270–5273.

Mason, Winter and Duncan J. Watts, "Financial Incentives and the "Performance of Crowds"," in "Proceedings of the ACM SIGKDD Workshop on Human Computation" HCOMP '09 ACM New York, NY, USA 2009, pp. 77–85.

_ **and Siddharth Suri**, "Conducting behavioral research on Amazon's Mechanical Turk," *Behavior research methods*, 2012, *44* (1), 1–23.

Radanovic, Goran and Boi Faltings, "Learning to scale payments in crowdsourcing with properboost," in "Fourth AAAI Conference on Human Computation and Crowdsourcing" 2016.

Robinson, Peter M., "Root-N-Consistent Semiparametric Regression," *Econometrica*, 1988, *56* (4), 931–954.

Rogstadius, Jakob, Vassilis Kostakos, Aniket Kittur, Boris Smus, Jim Laredo, and Maja Vukovic, "An Assessment of Intrinsic and Extrinsic Motivation on Task Performance in Crowdsourcing Markets," 2011.

Smith, Aaron, "Gig work, online selling and home sharing," *Pew Research Center*, 2016.

Sorokin, Alexander and David Forsyth, "Utility data annotation with amazon mechanical turk," in "Computer Vision and Pattern Recognition Workshops, 2008. CVPRW'08. IEEE Computer Society Conference on" IEEE 2008, pp. 1–8.

Staiger, Douglas O, Joanne Spetz, and Ciaran S Phibbs, "Is there monopsony in the labor market? Evidence from a natural experiment," *Journal of Labor Economics*, 2010, *28* (2), 211–236.

Yin, Ming, Mary L. Gray, and Siddharth Suri, "Running Out of Time: The Impact and Value of Flexibility in On-Demand Work," *Under Review at The ACM CHI Conference on Human Factors in Computing Systems 2018*, 2018.

Appendix A A Nested Logit Model of Mechanical Turk

Many crowdsourcing tasks and experiments that we will consider have a 2-part structure, which we model via a nested logit. First a wage w_1 is posted to draw subjects into an experimental protocol, then a second wage w_2 is randomized. Suppose agents face N HIT batches and each HIT batch i is made up of T_i individual tasks, indexed by j, each of which pay can pay a different wage. Agents get an initial wage w_{1i} and a task specific wage w_{2ij}. Utility from doing task j in batch i is (suppressing covariates):

$$U_{ij} = \eta_1 \ln(w_{1i}) + \eta_2 \ln(w_{2ij}) + \epsilon_{ij} \tag{7}$$

where ϵ_{ij} has cumulative distribution $F(\epsilon_{ij}) = \exp\left(-\sum_{i=1}^{N}(\sum_{j=1}^{T_i}\exp\left(\frac{\epsilon_{ij}}{\rho_i}\right)\right)^{\rho_i}$. $1-\rho_i$ measures the correlation in the errors within a batch i, but errors are independent across batches. Different batches can have differing numbers of tasks T_i, each of which pays a different ("bonus") wage w_{ij}, with differing elasticity of substitution between the tasks within a batch ρ_i.

Within a batch i, the probability of choosing task j is $P_{j|i} = \frac{e^{\frac{\eta_2 \ln(w_{ji}^2)}{\rho_i}}}{\sum_{k=1}^{T_i} e^{\frac{\eta_2 \ln(w_{ki}^2)}{\rho_i}}} = \frac{w_{2ij}^{\frac{\eta_2}{\rho_i}}}{\sum_{k=1}^{T_i} w_{ik}^{\frac{\eta_2}{\rho_i}}}$ so the "retention" elasticity is $\frac{\eta_2}{\rho_i}$.

Given the task-specific wages and number of tasks in the batch i, the probability of choosing HIT i is $P_i = \frac{w_{1i}^{\eta_1}(\sum_j^{T_i} w_{2ij}^{\frac{\eta_2}{\rho_i}})^{\rho_i}}{\sum_{h=1}^{N} w_{1h}^{\eta_1}(\sum_{k=1}^{T_h} w_{2hk}^{\frac{\eta_2}{\rho_i}})^{\rho_i}}$.

For expositional ease suppose $w_{2ij} = w_{2i}$ for all j within a batch i. We can represent our different identification strategies in this setup. Taking logs we get an equation for the "recruitment" probability:

$$\log(P_i) = \eta_1 \log(w_{1i}) + \frac{\eta_2}{\rho_i}\log(w_{2i}) + \frac{1}{\rho_i}\log T_i + \mu_i \tag{8}$$

Where μ_i is an error term that is possibly correlated with the other terms on the right hand side of the equation. Note that the recruitment probability depends on the (possibly expected) second-stage wages and number of tasks. Thus a simple regression of acceptance probability on posted batch wages would likely be confounded by properties of the batch tasks and the bonus wages paid.

Taking a percent change approximation to the log we get:

$$\eta_1 \approx \frac{dP_i}{dlog(w_i)}\frac{1}{\bar{P}_i} \tag{9}$$

and

$$\frac{\eta_2}{\rho_i} \approx \frac{dP_{j|i}}{dlog(w_{2i})} \frac{1}{\bar{P_{j|i}}} \tag{10}$$

Identification of η_1 is obtained from 8 by a) using machine-learned functions to control for batch properties (i.e. w_{2i}, ρ_i, and T_i as well as any other characteristics of the task that influence worker utility) in the double-ML approach and b) using the "honeypot" design described below to randomize w_{1i} holding the batch properties constant. Identification of $\frac{\eta_2}{\rho_i}$ is obtained from the "retention" experiments described below, with possibly heterogeneous elasticities coming from differences in ρ_i across batches. The nested logit structure captures the difference between initial "recruitment" elasticities that measure the responsiveness of workers into a HIT batch and secondary "retention" elasticities that measure the completion of tasks within the batch.

Consider a decision maker or researcher who needs a given volume of tasks (e.g., images labeled) that they value at p_i each. The requester must choose wages so as to maximize $(p_i - w_{1i} - w_{2i})P(i|w_{1i})P(j|i, w_{2i})$.

The optimal choice of the wages w_{1i} and w_{2i} are given by the first-order conditions to this problem, which can be manipulated to give:

$$w_{1i} = \frac{\rho_i \eta_1}{\eta_2} w_{2i} \tag{11}$$

$$\frac{p_i - w_{1i} - w_{2i}}{w_{1i} + w_{2i}} = \frac{\rho_i}{\rho_i \eta_1 + \eta_2} \tag{12}$$

The first expression shows that it is difficult to isolate variation in w_{1i} that is not also correlated with w_{2i}, as both are determined by task productivity p_i. Thus in tasks with a 2-part structure, it is important to control for w_{2i} when estimating η_1 with variation in w_{1i}. The second expression gives the percentage gap between the value of the task and the total wage being paid, or markdown, as a variant of the Lerner condition. As either η_1 or $\frac{\eta_2}{\rho_i}$ become large, the gap between wages and productivity goes to 0, indicating perfect competition. In the crowdsourcing context, the recruitment and retention elasticities, together with the assumption that requesters are optimizing, is a measure that simultaneously captures the distribution of crowdsourcing value between requesters and labelers, as well as the inefficiencies that result.

Appendix B Other Experiments Surveyed

We surveyed a large number of MTurk experiments, shown in 4. However, we did not include those that did not randomize the wage within the same batch. In a large number of MTurk studies, researchers will

Study	Included	Reason
Berinsky et al. (2012)	No	HIT groups posted sequentially (not randomized)
Buhrmester et al. (2011)	No	HIT groups posted sequentially (not randomized)
Callison-Burch (2014)	No	Unable to obtain data
Chandler and Horton (2011)	No	Unable to obtain data
Crump et al. (2013)	No	HIT groups posted sequentially (not randomized)
Doerrenberg et al. (2016)	No	Piecemeal wage, non-honeypot setup
Dube et al. (2017)	Yes	Replicated
Heer and Bostock (2010)	No	HIT groups posted sequentially (not randomized)
Ho et al. (2015)	Yes	Randomized "honeypot" design
Horton and Chilton (2010)	No	Labor supply elasticity (0.34) imputed, not estimated directly
Horton et al. (2011)	Yes	Replicated
Huang et al. (2010)	No	Unable to obtain data
Hsieh and Kocielnik (2016)	Yes	Randomized "honeypot" design
Marge et al. (2010)	No	HIT groups posted sequentially (not randomized)
Mason and Watts (2009)	No	Piecemeal wage, non-honeypot setup
Rogstadius et al. (2011)	No	Common HIT pool creates non-independence of accept/reject decisions
Sorokin and Forsyth (2008)	No	HIT groups posted sequentially (not randomized)
Yin et al. (2018)	Yes	Randomized "honeypot" design

Table 4: All Experiments Surveyed

issue batches of HITs sequentially, with each batch being given a different wage[4] The majority of these are not randomized and thus we cannot use them to recover even quasi-experimental requester's labor supply elasticities. See Table 4 for full explanations of the inclusion/exclusion criteria for each study.

[4]We examined the estimates in the following papers: Berinsky et al. (2012), Buhrmester et al. (2011), Crump et al. (2013), Doerrenberg et al. (2016), Heer and Bostock (2010), Horton and Chilton (2010), Marge et al. (2010), Mason and Watts (2009), Rogstadius et al. (2011), Sorokin and Forsyth (2008)

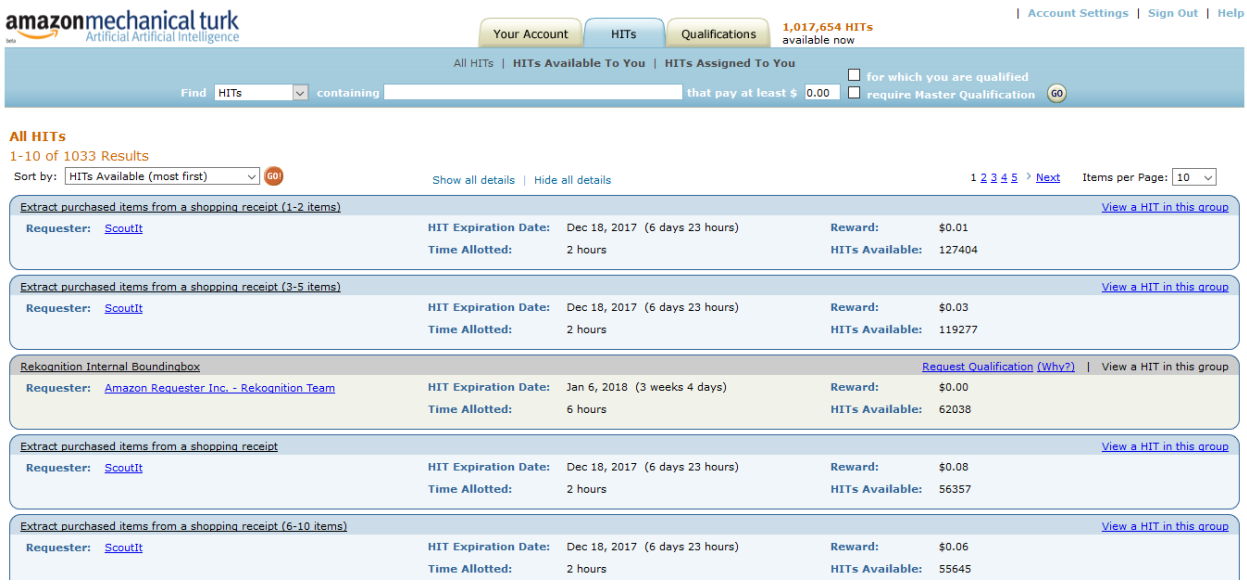

Figure 3: Sample interface page from Amazon Mechanical Turk

Appendix C Observational Data Appendix

Figure 3 shows a sample of the MTurk interface for workers. We use two different scraping strategies. Section Appendix C.1 describes data from Ipeirotis (2010) obtained via the Mechanical Turk Tracker API[5], and goes from January 2014 through February 2016, when the account was ended by Amazon. Beginning in May 2016, we ran our own scraper, which took snapshots of all HITs available to a worker with a US address every 30 minutes, though the frequency was increased to every 10 minutes beginning in May 2017. Data from this latter scrape is described in Section Appendix C.2.

Appendix C.1 Data for January 2014 – February 2016

Ipeirotis (2010) introduces the Mehcanical Turk Tracker, a web interface allowing researchers to view hourly market data (*e.g.*, number of HITs available) and demographic information (*e.g.*, proportion of workers who identify as male/female or who are from India/the United States) for the Amazon Mechanical Turk marketplace. An Application Programming Interface (API) is provided alongside the web interface, allowing for programmatic queries to be issued to the database. Using this API, we downloaded both "cross-sectional" data (*e.g.*, requester name, title, description, keywords) and "time series" data (number of HITs available in the group for each run of the scraper) for 410,284 HIT groups. Of these, 125,337 were either posted to the marketplace after February 1st, 2017 or had observations after this date, the date Amazon changed

[5] `https://crowd-power.appspot.com/#/general`

	(1) 2016-2017		(2) 2014-2016		(3) 2017	
	mean	sd	mean	sd	mean	sd
Duration	$2.080 \cdot 10^8$	$5.840 \cdot 10^8$	$2.020 \cdot 10^8$	$5.650 \cdot 10^8$	$1.370 \cdot 10^8$	$5.030 \cdot 10^8$
Reward	70.397	92.420	38.014	63.741	61.774	87.358
Log Reward Prediction	3.427	1.394	2.634	1.184	3.282	1.334
Log Duration Prediction	6.222	1.263	5.221	2.617	5.321	1.889
Log Reward Residual	0.002	0.501	0.004	0.707	0.003	0.486
Log Duration Residual	−0.012	1.420	−0.015	0.904	−0.017	0.837
Time Allotted	$3.573 \cdot 10^4$	$1.750 \cdot 10^5$	$4.668 \cdot 10^3$	$1.227 \cdot 10^4$	$2.607 \cdot 10^4$	$1.262 \cdot 10^5$
Observations	292746		258352		93775	

Table 5: The top panel presents from scraping MTurk between Jan. 2014 and Feb. 2016. The middle panel presents analogous estimates using data obtained from May. 2016 through August 2017.

its interface and the scraper ceased working, and thus were dropped from our analysis. Of the remaining 284,947, we dropped any that had

- Zero-valued reward,
- Only one observation (since we're unable to compute durations for these groups), or
- Rewards or durations greater than the 99.5th percentile of their respective distributions (approx. 90,000 minutes for durations and $10.00 for rewards),

leaving us with 263,213 "final" observations.

Appendix C.2 Data for May 2016 – August 2017

At the beginning of the project in May 2016, we set up a scraper which would log in to MTurk as a user with a US address and download all available information about each HIT group listed in the web interface as shown in Figure 3. The scraper ran every 30 minutes (on the hour and on the half-hour) starting at midnight EST on May 31st 2016, though this was increased to every 10 minutes beginning at midnight EST on May 31st 2017. The scraper was finally banned by Amazon on August 21st 2017 at 7:30pm EST. The every-30-minute scrapes from May 2016 to May 2017 produced 363,181 observations, while the every-10-minute scrapes from May 2017 to August 2017 produced 110,732.

Appendix D Full Double-ML Procedure

Appendix D.1 Data Loading/Merging

For each of our three datasets, the initial data processing proceeded as follows. First, a scraped panel dataset is loaded which contains, for each HIT group, the number of HITs available and the timestamp of each scrape in which the group was observed. This panel data then gets collapsed into a cross-sectional dataset consisting of several features derived from the distribution of the timestamps and HITs available – for example, into min(timestamp), max(timestamp), min(hits_available), and max(hits_available) for each HIT group. Then, a separate cross-sectional metadata file (containing, for example, the titles, descriptions, and requester names for each HIT group) is merged into the collapsed panel dataset via the unique Amazon-supplied group ID[6].

Appendix D.2 Data Cleaning

All observations with a reward greater than $5 or duration greater than 90,000 minutes (approximately two months) are dropped[7]. Then all observations with 0 reward or 0 duration values are dropped, to allow transformation of the dependent variables into log space.

Appendix D.3 Feature Selection and Test/Training Split

We transform the text scraped with each HIT batch into a large number of text features as follows:

- **N-grams**: An n-gram is an ordered sequence of n words. For example, if the full description for a HIT is "quick transcription task," this will produce three 1-grams "quick," "description" and "task"; two 2-grams "quick description" and "description task"; and a single 3-gram "quick description task." We use sliding windows of 1 to 3 words over all words within the title, HIT description and keyword list to form 1, 2 and 3-grams. The frequency of these n-grams in each HIT is then a feature used by the ML algorithm. We use the standard English stopword list in Scikit-learn to eliminate stopwords.
- **Topic Distributions**: Besides ordered sequence of words, sometimes sets of particular words ("topic") convey important information. A topic model is essentially an algorithm which searches for sets of words that tend to occur together in a corpus. For example, one of our topics identifies the words "image," "text," and "transcribe" as its top words. HITs requesting transcription of text from an image will tend

[6]This final cross-sectional file contains 411,196 observations for the Jan 2014 - Feb 2016 data, 363,181 for the May 2016 - May 2017 data, and 110,732 for the May 2017 - Aug 2017 data, as described in the previous section.

[7]These values correspond approximately to the 99.5th percentiles of the original distribution.

to have high feature values for this topic and lower values for other topics. The resulting features for each HIT is then the distribution over topics found in that HIT's title, description, and keyword list.[8] We use the NLTK English stopword corpus to drop stopwords. The top 5 words for each topic model run with $K \in \{5, 10, 15, 20\}$ are available online at `textlab.econ.columbia.edu/topicwords.pdf`.

- **Doc2Vec Embeddings**: Unlike LDA which tries to generate features by splitting *documents* into discrete human-interpretable topics, the goal of Doc2Vec is to generate a vector space in which vectors for *words* which are semantically similar are close together, and then infer a document-level vector within this same vector space via amalgamation of the learned vectors for its constituent words. For example, since "survey" and "questionnaire" are semantically similar in the sense that they are used in similar contexts ("a short [survey/quetionnaire]", "fill out this [survey/questionnaire]"), their vectors will be close together in the constructed vector space, and this will "pull" the document-level vectors for descriptions containing either word closer together. [9]

- **Hand-Engineered Features**: Finally, we use a set of custom regular-expression-based features, which are generally binary variables describing the presence or absence of certain salient keywords (*e.g.*, "survey", "transcribe"), but also real-valued variables capturing (for example) time estimates given in the titles/descriptions (*e.g.*, "5-minute survey"). The bulk of these features are derived from the explicit features described in Difallah et al. (2015), and the HIT taxonomy scheme developed in Gadiraju et al. (2014), described more fully in Appendix Appendix D. The hand-engineered features are as follows:

  - Based on common patterns we observed in HIT titles, descriptions, and keywords, dummy variables were created indicating the presence or absence of the following regular expressions: *easy*, *transcr** (capturing, e.g., "transcription" or "transcribe"), *writ** (capturing, e.g., "written", "write", or "writing"), *audio*, *image/picture*, *video*, *bonus*, *copy*, *search*, *ident** (capturing, e.g., "identify"), *text*, *date*, *fun*, *simpl**, *summar**, *only*, *improve*, *five/5*, *?*, and *!*.
  - Based on the HIT taxonomy scheme developed in Gadiraju et al. (2014), a numerical category was assigned to each HIT group via the following regular expressions:
    - **Information Finding (IF)**: *find*

[8] We run a Latent Dirichlet Allocation (LDA) topic model model (Blei et al. (2003)) on all descriptions. LDA requires the choice of a parameter K which determines how many topics the algorithm should try to discover: we estimate models with $K \in \{5, 10, 15, 20\}$.

[9] We run Doc2Vec model Le and Mikolov (2014) on all titles, descriptions, and keywords in the data, producing a 50-dimensional semantic information vector for each.

    - **Verification and Validation (VV)**: *check*, *match*
    - **Interpretation and Analysis (IA)**: *choose*, *categor**
    - **Content Creation (CC)**: *suggest*, *translat**
    - **Surveys (S)**: *survey*
    - **Content Access (CA)**: *click*, *link*, *read*
- The following numeric features were extracted, some of which were derived from features used in Difallah et al. (2015):
    - `time_allotted`: The time a worker is given to complete a given HIT
    - `time_left`: The time remaining before the HIT group expires (expired HIT groups are removed from the marketplace)
    - `first_hits`: The number of HITs initially posted to the marketplace
    - `last_hits`: The number of HITs remaining to be completed in the group at the time it was last observed
    - `min_hits`: The minumum number of HITs available observed for the group across all scrapes
    - `max_hits`: The maximum number of HITs available observed for the group across all scrapes
    - `avg_hitrate`: The average rate (per hour) at which HITs within the group were filled by workers
    - `avg_hits_completed`: The average change in available HITs between subsequent observations of the group
    - `med_hits_completed`: The median change in available HITs between subsequent observations of the group
    - `min_hits_completed`: The minimum change in available HITs between subsequent observations of the group
    - `max_hits_completed`: The maximum change in available HITs between subsequent observations of the group
    - `num_zeros`: The number of observations for which the number of available HITs in the group was listed as 0
    - `req_mean_reward`: The average reward over all HITs posted by the requester
    - `req_mean_dur`: The average duration of all HIT groups posted by the requester

* `title_len`: The length of the HIT group's title
* `desc_len`: The length of the HIT group's description
* `keywords_len`: The sum of the lengths of the HIT group's keywords
* `num_keywords`: The number of keywords given for the HIT group
* `title_words`: The number of words in the HIT group's title
* `desc_words`: The number of words in the HIT group's description
* `minutes_title`: The number of minutes if a phrase including "X minutes" appears in the title
* `minutes_desc`: The number of minutes if a phrase including "X minutes" appears in the description
* `minutes_kw`: The number of minutes if a phrase including "X minutes" appears in the keyword list
* `qual_len`: The length of the string given in the HIT group's description which lists the qualifications
* `num_quals`: The number of qualifications required for the HIT group
* `custom_not_granted`: The number of custom qualifications required for the HIT group for which our "blank" account (an account which had never accepted or completed a HIT) was not qualified
* `custom_granted`: The number of custom qualifications required for the HIT group for which our "blank" account (an account which had never accepted or completed a HIT) was qualified
* `any_loc`: A dummy variable representing whether or not the HIT group had a location restriction (e.g., US only)
* `us_only`: A dummy variable which is 1 if the HIT group is restricted to US workers, and 0 otherwise
* `appr_rate_gt`: The lower bound on approval rate required for workers to be eligible for the HIT, coded as -1 if no lower bound was enforced
* `rej_rate_lt`: The upper bound on rejection rate required for workers to be eligible for the HIT, coded as 101 if no upper bound was enforced
* `appr_num_gt`: The lower bound on number of approvals required for workers to be eligible for the HIT, coded as -1 if no lower bound was enforced

* `rej_num_lt`: The upper bound on number of rejections required for workers to be eligible for the HIT, coded as 999 if no upper bound was enforced
* `adult_content`: A dummy variable which is 1 if the HIT group indicated that it contained adult content, and 0 otherwise

To save on computation time, we utilized a "two stage" double ML procedure as outlined in Section 3.3.2. Given the initial split of the data into A and B sets, and the subsequent split of these sets into A_{train}, A_{val}, B_{train}, and B_{val}, a given run of our procedure ($A \to B$ or $B \to A$; here we describe the $A \to B$ run without loss of generality) proceeds as follows. First, in the "feature selection" phase, the full set of n-gram features were generated as described in Section 3.3.1, and a "preliminary" run of the learning algorithm was performed using *only* these n-gram features as the feature matrix for A_{train}, with the goal of predicting the reward and duration values in A_{val}. Upon completion of this stage, we "threw away" all but the top 100 most predictive n-gram features for reward and top 100 most predictive n-gram features for duration, and for the remainder of the run only those n-gram features were included. For illustration, the top 100 most predictive reward and duration features for the $A \to B$ run on the Jan 2014 - Feb 2016 data were as follows:

Once this feature selection phase was complete, a second "full data" phase was performed, as outlined in Section 3.3.2. In this phase, the top 200 n-gram features from the feature selection phase are included in the feature matrix for A along with the LDA, Doc2Vec, and hand-engineered features, with the goal of predicting the reward and duration values in B.

Appendix D.4 Regression via Random Forests

Before computing predictions on the test set, the validation set was used to tune not only hyperparameters but also which learning method was chosen. Random forest regression, implemented by RandomForestRegressor in scikit-learn, greatly outperformed the other classifiers we employed: {AdaBoostRegressor, BaggingRegressor, ExtraTreesRegressor, GradientBoostingRegressor, RandomForestRegressor, SVR (Support-VectorRegressor)}, and thus a trained random forest regression was our choice for computing predictions for the test data. The random forest method constructs a series of individual decision tree estimators, where each regressor is trained on a subset of the full feature set, and then reports the mean prediction over all regressors. Based on two additional cross-validation procedures, for the first-stage feature selection an random forest regression with 40 decision tree estimators was used (with the number of estimators optimized over {10, 20, ..., 100}) while for the second-stage ML the model was run with 600 estimators (optimized

	Jan 2014 - Feb 2016		May 2016 - May 2017		May 2017 - Aug 2017	
	$A \to B$	$B \to$ A	$A \to B$	$B \to A$	$A \to B$	$B \to A$
R^2_{reward}	0.7527	0.7571	0.8878	0.8861	0.8872	0.8832
R^2_{duration}	0.8937	0.8944	0.4485	0.4479	0.5377	0.5304

Table 6: R^2 scores for each run of the Double-ML regressions

over $\{100, 200, \ldots, 1000\}$, with the increased order of magnitude made feasible due to the fact that in the second stage all but 200 of the approximately 800,000 total n-gram features are dropped).

Appendix D.5 Computing the Double-ML estimate

Once the ML algorithm has finished its runs and the predicted log duration and log reward values have been generated for each fold of the data, the estimated η value is computed straightforwardly via Equation 5 with $n = 2$.