A Guide to Conducting Behavioral Research on Amazon's Mechanical Turk

Article *in* Behavior Research Methods · June 2011

DOI: 10.3758/s13428-011-0124-6 · Source: PubMed

2 authors:

Winter Mason
Facebook
41 PUBLICATIONS **5,099** CITATIONS
SEE PROFILE

Siddharth Suri
Microsoft
53 PUBLICATIONS **4,803** CITATIONS
SEE PROFILE

Behav Res
DOI 10.3758/s13428-011-0124-6

Conducting behavioral research on Amazon's Mechanical Turk

Winter Mason · Siddharth Suri

Abstract Amazon's Mechanical Turk is an online labor market where requesters post jobs and workers choose which jobs to do for pay. The central purpose of this article is to demonstrate how to use this Web site for conducting behavioral research and to lower the barrier to entry for researchers who could benefit from this platform. We describe general techniques that apply to a variety of types of research and experiments across disciplines. We begin by discussing some of the advantages of doing experiments on Mechanical Turk, such as easy access to a large, stable, and diverse subject pool, the low cost of doing experiments, and faster iteration between developing theory and executing experiments. While other methods of conducting behavioral research may be comparable to or even better than Mechanical Turk on one or more of the axes outlined above, we will show that when taken as a whole Mechanical Turk can be a useful tool for many researchers. We will discuss how the behavior of workers compares with that of experts and laboratory subjects. Then we will illustrate the mechanics of putting a task on Mechanical Turk, including recruiting subjects, executing the task, and reviewing the work that was submitted. We also provide solutions to common problems that a researcher might face when executing their research on this platform, including techniques for conducting synchronous experiments, methods for ensuring high-quality work, how to keep data private, and how to maintain code security.

Keywords Crowdsourcing · Online research · Mechanical turk

W. Mason (✉) · S. Suri
Yahoo! Research,
New York, USA
e-mail: m@winteram.com

S. Suri
e-mail: suri@yahoo-inc.com

Published online: 30 June 2011

Introduction

The creation of the Internet and its subsequent widespread adoption has provided behavioral researchers with an additional medium for conducting studies. In fact, researchers from a variety of fields, such as economics (Hossain & Morgan, 2006; Reiley, 1999), sociology (Centola, 2010; Salganik, Dodds, & Watts, 2006), and psychology (Birnbaum, 2000; Nosek, 2007), have used the Internet to conduct behavioral experiments.[1] The advantages and disadvantages of online behavioral research, relative to laboratory-based research, have been explored in depth (see, e.g., Kraut et al., 2004; Reips, 2000). Moreover, many methods for conducting online behavioral research have been developed (e.g., Birnbaum, 2004; Gosling & Johnson, 2010; Reips, 2002; Reips & Birnbaum, 2011). In this article, we describe a tool that has emerged in the last 5 years for conducting online behavioral research: crowdsourcing platforms. The term *crowdsourcing* has its origin in an article by Howe (2006), who defined it as a job outsourced to an undefined group of people in the form of an open call. The key benefit of these platforms to behavioral researchers is that they provide access to a persistently available, large set of people who are willing to do tasks—including participating in research studies—for relatively low pay. The crowdsourcing site with one of the largest subject pools is Amazon's Mechanical Turk[2] (AMT), so it is the focus of this article.

[1] This is clearly not an exhaustive review of every study done on the Internet in these fields. We aim only to provide some salient examples.

[2] The name "Mechanical Turk" comes from a mechanical chess-playing automaton from the turn of the 18th century, designed to look like a Turkish "sorcerer," which was able to move pieces and beat many opponents. While it was a technological marvel at the time, the real genius lay in a diminutive chess master hidden in the workings of the machine (see http://en.wikipedia.org/wiki/The_Turk). Amazon's Mechanical Turk was designed to hide human workers in an automatic process; hence, the name of the platform.

Originally, Amazon built Mechanical Turk specifically for *human computation tasks*. The idea behind its design was to build a platform for humans to do tasks that are very difficult or impossible for computers, such as extracting data from images, audio transcription, and filtering adult content. In its essence, however, what Amazon created was a labor market for microtasks (Huang, Zhang, Parkes, Gajos, & Chen, 2010). Today, Amazon claims hundreds of thousands of workers and roughly ten thousand employers, with AMT serving as the meeting place and market (Ipeirotis, 2010a; Pontin, 2007). For this reason, it also serves as an ideal platform for recruiting and compensating subjects in online experiments. Since Mechanical Turk was initially invented for human computation tasks, which are generally quite different than behavioral experiments, it is not a priori clear how to conduct certain types of behavioral research, such as synchronous experiments, on this platform. One of the goals of this work is to exhibit how to achieve this.

Mechanical Turk has already been used in a small number of online studies, which fall into three broad categories. First, there is a burgeoning literature on how to combine the output of a small number of cheaply paid workers in a way that rivals the quality of work by highly paid, domain-specific experts. For example, the output of multiple workers was combined for a variety of tasks related to natural language processing (Snow, O'Connor, Jurafsky, & Ng, 2008) and audio transcription (Marge, Banerjee, & Rudnicky, 2010) to be used as input to other research, such as machine-learning tasks. Second, there have been at least two studies showing that the behavior of subjects on Mechanical Turk is comparable to the behavior of laboratory subjects (Horton, Rand, & Zeckhauser, in press; Paolacci, Chandler, & Ipeirotis, 2010). Finally, there are a few studies that have used Mechanical Turk for behavioral experiments, including Eriksson and Simpson (2010), who studied gender, culture, and risk preferences; Mason and Watts (2009), who used it to study the effects of pay rate on output quantity and quality; and Suri and Watts (2011), who used it to study social dilemmas over networks. All of these examples suggest that Mechanical Turk is a valid research environment that scientists are using to conduct experiments.

Mechanical Turk is a powerful tool for researchers that has only begun to be tapped, and in this article, we offer insights, instructions, and best practices for using this tool. In contrast to previous work that has demonstrated the validity of research on Mechanical Turk (Buhrmester, Kwang, & Gosling, in press; Paolacci et al., 2010), the purpose of this article is to show how Mechanical Turk can be used for behavioral research and to demonstrate best practices that ensure that researchers quickly get high-quality data from their studies.

There are two classes of researchers who may benefit from this article. First, there are many researchers who are not aware of Mechanical Turk and what is possible to do with it. In this guide, we exhibit the capabilities of Mechanical Turk and several possible use cases, so researchers can decide whether this platform will aid their research agenda. Second, there are researchers who are already interested in Mechanical Turk as a tool for conducting research but may not be aware of the particulars involved with and/or the best practices for conducting research on Mechanical Turk. The relevant information on the Mechanical Turk site can be difficult to find and is directed toward human computation tasks, as opposed to behavioral research, so here we offer a detailed "how-to" guide for conducting research on Mechanical Turk.

Why Mechanical Turk?

There are numerous advantages to online experimentation, many of which have been detailed in prior work (Reips, 2000, 2002). Naturally, Mechanical Turk shares many of these advantages, but also has some additional benefits. We highlight three unique benefits of using Mechanical Turk as a platform for running online experiments: (1) subject pool access, (2) subject pool diversity, and (3) low cost. We then discuss one of the key advantages of online experimentation that Mechanical Turk shares: faster iteration between theory development and experimentation.

Subject pool access Like other online recruitment methods, Mechanical Turk offers access to subjects for researchers who would not otherwise have access, such as researchers at smaller colleges and universities with limited subject pools (Smith & Leigh, 1997) or nonacademic researchers, with whom recruitment is generally limited to ads posted online (e.g., study lists, e-mail lists, social media, etc.) and flyers posted in public areas. While some research necessarily requires subjects to actually come into the lab, there are many kinds of research that can be done online.

Mechanical Turk offers the unique benefit of having an existing pool of potential subjects that remains relatively stable over time. For instance, many academic researchers experience the drought/flood cycle of undergraduate subject pools, with the supply of subjects exceeding demand at the beginning and end of a semester and then dropping to almost nothing at all other times. In addition, standard methods of online experimentation, such as building a Web site containing an experiment, often have "cold-start" problems, where it takes time to recruit a panel of reliable subjects. Aside from some daily and weekly seasonalities, the subject availability on Mechanical Turk is fairly stable

(Ipeirotis, 2010a), with fluctuations in supply largely due to variability in the number of jobs available in the market.

The single most important feature that Mechanical Turk provides is access to a large, stable pool of people willing to participate in experiments for relatively low pay.

Subject pool diversity Another advantage of Mechanical Turk is that the workers tend to be from a very diverse background, spanning a wide range of age, ethnicity, socio-economic status, language, and country of origin. As with most subject pools, the population of workers on AMT is not representative of any one country or region. However, the diversity on Mechanical Turk facilitates cross-cultural and international research (Eriksson & Simpson, 2010) at a very low cost and can broaden the validity of studies beyond the undergraduate population. We give detailed demographics of the subject pool in the Workers section.

Low cost and built-in payment mechanism One distinct advantage of Mechanical Turk is the low cost at which studies can be conducted, which clearly compares favorably with paid laboratory subjects and comparably to other online recruitment methods. For example, Paolacci et al. (2010) replicated classic studies from the judgment and decision-making literature at a cost of approximately $1.71 per hour per subject and obtained results that paralleled the same studies conducted with undergraduates in a laboratory setting. Göritz, Wolff, and Goldstein (2008) showed that the hassle of using a third-party payment mechanism, such as PayPal, can lower initial response rates in online experiments. Mechanical Turk skirts this issue by offering a built-in mechanism to pay workers (both flat rate and bonuses) that greatly reduces the difficulties of compensating individuals for their participation in studies.

Faster theory/experiment cycle One implicit goal in research is to maximize the efficiency with which one can go from generating hypotheses to testing them, analyzing the results, and updating the theory. Ideally, the limiting factor in this process is the time it takes to do careful science, but all too often, research is delayed because of the time it takes to recruit subjects and recover from errors in the methodology. With access to a large pool of subjects online, recruitment is vastly simplified. Moreover, experiments can be built and put on Mechanical Turk easily and rapidly, which further reduces the time to iterate the cycle of theory development and experimental execution.

Finally, we note that other methods of conducting behavioral research may be comparable to or even better than Mechanical Turk on one or more of the axes outlined above, but taken as a whole, it is clear that Mechanical Turk can be a useful tool for many researchers.

Validity of worker behavior

Given the novel nature of Mechanical Turk, most of the initial studies focused on evaluating whether it could effectively be used as a means of collecting valid data. At first, these studies focused on whether workers on Mechanical Turk could be used as substitutes for domain-specific experts. For instance, Snow et al. (2008) showed that for a variety of natural language processing tasks, such as affect recognition and word similarity, combining the output of just a few workers can equal the accuracy of expert labelers. Similarly, Marge et al. (2010) compared workers' audio transcriptions with domain experts and found that after a small bias correction, the combined outputs of the workers were of a quality comparable to that of the experts. Urbano, Morato, Marrero, and Martin (2010) crowdsourced similarity judgments on pieces of music for the purposes of music information retrieval. Using their techniques, they obtained a partially ordered list of similarity judgments at a far cheaper cost than hiring experts, while maintaining high agreement between the workers and the experts. Alonso and Mizzaro (2009) conducted a study in which workers were asked to rate the relevance of pairs of documents and topics and compared this with a gold standard given by experts. The output of the Turkers was similar in quality to that of the experts.

Of greater interest to behavioral researchers is whether the results of studies conducted on Mechanical Turk are comparable to results obtained in other online domains, as well as offline settings. To this end, Buhrmester et al. (in press) compared Mechanical Turk subjects with a large Internet sample with respect to several psychometric scales and found no meaningful differences between the populations, as well as high test–retest reliability in the Mechanical Turk population. Additionally, Paolacci et al. (2010) conducted replications of standard judgment and decision-making experiments on Mechanical Turk, as well as with subjects recruited through online discussion boards and subjects recruited from the subject pool at a large Midwestern university. The studies they replicated were the "Asian disease" problem to test framing effects (Tversky & Kahneman, 1981), the "Linda" problem to test the conjunction fallacy (Tversky & Kahneman, 1983), and the "physician" problem to test outcome bias (Baron & Hershey, 1988). Quantitatively, there were only very slight differences between the results from Mechanical Turk and subjects recruited using the other methods, and qualitatively, the results were identical. This is similar to the results of Birnbaum (2000), who found that Internet users were more logically consistent in their decisions than were laboratory subjects.

There have also been a few studies that have compared Mechanical Turk behavior with laboratory behavior. For

example, the "Asian disease" problem (Tversky & Kahneman, 1981) was also replicated by Horton et al. (in press), who also obtained qualitatively similar results. In the same study, the authors found that workers "irrationally" cooperated in the one-shot Prisoner's Dilemma game, replicating previous laboratory studies (e.g., Cooper, DeJong, Forsythe, & Ross, 1996). They also found, in a replication of another, more recent laboratory study (Shariff & Norenzayan, 2007), that providing a religious prime before the game increased the level of cooperation. Suri and Watts (2011) replicated a public goods experiment that was conducted in the classroom (Fehr & Gachter, 2000), and despite the difference in context and the relatively lower pay on Mechanical Turk, there were no significant differences from a prior study conducted in the classroom (Fehr & Gachter, 2000).

In summary, there are numerous studies that show correspondence between the behavior of workers on Mechanical Turk and behavior offline or in other online contexts. While there are clearly differences between Mechanical Turk and offline contexts, evidence that Mechanical Turk is a valid means of collecting data is consistent and continues to accumulate.

Organization of this guide

In the following sections, we begin with a high-level overview of Mechanical Turk, followed by an exposition of methods for conducting different types of studies on Mechanical Turk. In the first half, we describe the basics of Mechanical Turk, including who uses it and why, and the general terminology associated with the platform. In the second half, we describe, at a conceptual level, how to conduct experiments on Mechanical Turk. We will focus on new concepts that come up in this environment that may not arise in the laboratory or in other online settings around the issues of ethics, privacy, and security. In this section, we also discuss the online community that has sprung up around Mechanical Turk. We conclude by outlining some interesting open questions regarding research on Mechanical Turk. We also include an appendix with engineering details required for building and conducting experiments on Mechanical Turk, for researchers and programmers who are building their experiments.

Mechanical Turk basics

There are two types of players on Mechanical Turk: requesters and workers. Requesters are the "employers," and the workers (also known as *Turkers* or *Providers*) are the "employees"—or more accurately, the "independent contractors." The jobs offered on Mechanical Turk are referred to as Human Intelligence Tasks (HITs). In this section, we discuss each of these concepts in turn.

Workers

In March of 2007, the *New York Times* reported that there were more than 100,000 workers on Mechanical Turk in over 100 countries (Pontin, 2007). Although this international diversity has been confirmed in many subsequent studies (Mason & Watts, 2009; Paolacci et al., 2010; Ross, Irani, Silberman, Zaldivar, & Tomlinson, 2010), as of this writing the majority of workers come from the United States and India, because Amazon allows cash payment only in U.S. dollars and Indian Rupees—although workers from *any country* can spend their earnings on Amazon.com.

Over the past 3 years, we have collected demographics for nearly 3,000 unique workers from five different studies (Mason & Watts, 2009; Suri & Watts, 2011). We compiled these studies, and of the 2,896 workers, 12.5% chose not to give their gender, and of the remainder, 55% reported being female and 45% reported being male. These demographics agree with other studies that have reported that the majority of U.S. workers on Mechanical Turk are female (Ipeirotis, 2010b; Ross et al., 2010). The median reported age of workers in our sample is 30 years old, and the average age is roughly 32 years old, as can be seen in Fig. 1; the overall shape of the distribution resembles reported ages in other Internet-based research (Reips, 2001). The different studies we compiled used different ranges when collecting information about income, so to summarize we classify workers by the top of their declared income range, which can be seen in Fig. 2. This shows that the majority of workers earn roughly U.S. $30 k per annum, although some respondents reported earning over $100 k per year.

Having multiple studies also allows us to check the internal consistency of these self-reported demographics. Of the 2,896 workers, 207 (7.1%) participated in exactly two studies, and of these 207, only 1 worker (0.4%) changed the answer on gender, age, education, or income. Thus, we conclude that the internal consistency of self-reported demographics on Mechanical Turk is high. This agrees with Rand (in press), who also found consistency in self-reported demographics on Mechanical Turk, and with Voracek, Stieger, and Gindl (2001), who compared the gender reported in an online survey (not on Mechanical Turk) conducted at the University of Vienna with that in the school's records and found a false response rate below 3%.

Given the low wages and relatively high income, one may wonder why people choose to work on Mechanical Turk at all. Two independent studies asked workers to indicate their reasons for doing work on Mechanical Turk. Ross et al. (2010) reported that 5% of U.S. workers and 13% of Indian workers said "MTurk money is always

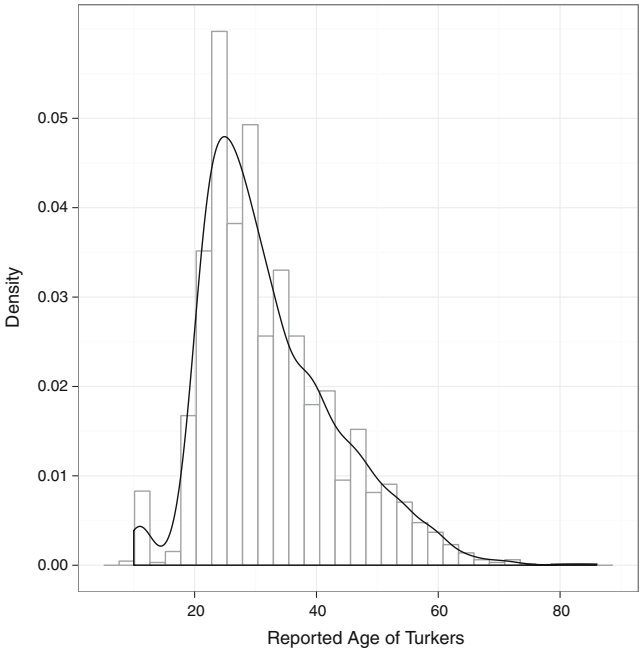

Fig. 1 Histogram (gray) and density plot (black) of reported ages of workers on Mechanical Turk

necessary to make basic ends meet." Ipeirotis (2010b) asked a similar question but delved deeper into the motivations of the workers. He found that 12% of U.S. workers and 27% of Indian workers reported that "Mechanical Turk is my primary source of income." Ipeirotis (2010b) also reported that roughly 30% of both U.S. and Indian workers indicated that they were currently

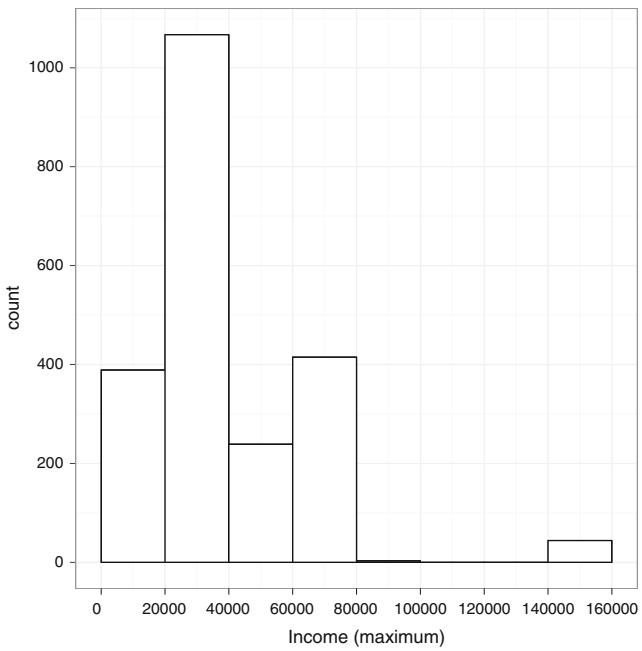

Fig. 2 Distribution of the maximum of the income (in U.S. dollars) interval self-reported by workers

unemployed or held only a part-time job. At the other end of the spectrum, Ross and colleagues asked how important money earned on Mechanical Turk was to them: Only 12% of U.S. workers and 10% of Indian workers indicated that "MTurk money is irrelevant," implying that the money made through Mechanical Turk is at least relevant to the vast majority of workers. The modal response for both U.S. and Indian workers was that the money was simply nice and might be a way to pay for "extras." Perhaps the best summary statement of why workers do tasks on Mechanical Turk is the 59% of Indian workers and 69% of U.S. workers who agreed that "Mechanical Turk is a fruitful way to spend free time and get some cash" (Ipeirotis, 2010b). What all of this suggests is that most workers are not trying to scrape together a living using Mechanical Turk (fewer than 8% reported earning more than $50/week on the site).

The number of workers available at any given time is not directly measurable. However, Ipeirotis (2010a) has tracked the number of HITs created and available every hour (and recently, every minute) over the past year and has used these statistics to infer the number of HITs being completed. With this information, he has determined that there are slight seasonalities with respect to time of day and day of week. Workers tend to be more abundant between Tuesday and Saturday, and Huang et al. (2010) found faster completion times between 6 a.m. and 3 p.m. GMT, (which resulted in a higher proportion of Indian workers). Ipeirotis (2010a) also found that over half of the HIT groups are completed in 12 hours or less, suggesting a large active worker pool.

To become a worker, one must create a worker account on Mechanical Turk and an Amazon Payments account into which earnings can be deposited. Both of these accounts merely require an e-mail address and a mailing address. Any worker, from anywhere in the world, can spend the money he or she earns on Mechanical Turk on the Amazon.com Web site. As was mentioned before, to be able to withdraw their earnings as cash, workers must take the additional step of linking their Payments account to a verifiable U.S. or Indian bank account. In addition, workers can transfer money between Amazon's Payment accounts. While having more than one account is against Amazon's Terms of Service, it is possible, although somewhat tedious, for workers to earn money using multiple accounts and transfer the earnings to one account to either be spent on Amazon.com or withdrawn. Requesters who use external HITs (see The Anatomy of a HIT section) can guard against multiple submissions by the same worker by using browser cookies and tracking IP addresses, as Birnbaum (2004) suggested in the context of general online experiments.

Another important policy forbids workers from using programs ("bots") to automatically do work for them. Although infringements of this policy appear to be rare (but

see McCreadie, Macdonald, & Ounis, 2010), there are also legitimate workers who could best be described as *spammers*. These are individuals who attempt to make as much money completing HITs as they can, without regard to the instructions or intentions of the requester. These individuals might also be hard to discriminate from bots. Surveys are favorite targets for these spammers, since they can be completed easily and are plentiful on Mechanical Turk. Fortunately, Mechanical Turk has a built-in reputation system for workers: Every time a requester rejects a worker's submission, it goes on their record. Subsequent requesters can then refuse workers whose rejection rate exceeds some specified threshold or can block specific workers who previously submitted bad work. We will revisit this point when we describe methods for ensuring data quality.

Requesters

The requesters who put up the most HITs and groups of HITs on Mechanical Turk are predominantly companies automating portions of their business or intermediary companies that post HITs on Mechanical Turk on the behalf of other companies (Ipeirotis, 2010a). For example, search companies have used Mechanical Turk to verify the relevance of search results, online stores have used it to identify similar or identical products from different sellers, and online directories have used it to check the accuracy and "freshness" of listings. In addition, since businesses may not want to or be able to interact directly with Mechanical Turk, intermediary companies have arisen, such as Crowdflower (previously called Dolores Labs) and Smartsheet.com, to help with the process and guarantee results. As has been mentioned, Mechanical Turk is also used by those interested in machine learning, since it provides a fast and cheap way to get labeled data such as tagged images and spam classifications (for more market-wide statistics of Mechanical Turk, see Ipeirotis, 2010a).

In order to run studies on Mechanical Turk, one must sign up as a requester. There are two or three accounts required to register as a requester, depending on how one plans to interface with Mechanical Turk: a requester account, an Amazon Payments Account, and (optionally) an Amazon Web Services (AWS) account.

One can sign up for a requester account at https://requester.mturk.com/mturk/beginsignin.[3] It is advisable to use a unique e-mail address for running experiments, preferably one that is associated with the researcher or the research group, because workers will interact with the researcher through this account and this e-mail address. Moreover, the workers will come to learn a reputation and possibly develop a relationship with this account on the basis of the jobs being offered, the money being paid, and, on occasion, direct correspondence. Similarly, we recommend using a name that clearly identifies the researcher. This does not have to be the researcher's actual name (although it could be) but also should be sufficiently distinctive that the workers know who they are working for. For example, the requester name "University of Copenhagen" could refer to many research groups, and workers might be unclear about who is actually doing the research; the name "Perception Lab at U. Copenhagen" would be better.

To register as a requester, one must also create an Amazon Payments account (https://payments.amazon.com/sdui/sdui/getstarted) with the same account details as those provided for the requester account. At this point, a funding source is required, which can be either a U.S. credit card or a U.S. bank account. Finally, if one intends to interact with Mechanical Turk programatically, one must also create an AWS account at https://aws-portal.amazon.com/gp/aws/developer/registration/index.html. This provides one with the unique digital keys necessary to interact with the Mechanical Turk Application Programming Interface (API), which is discussed in detail in the Programming interfaces section of the Appendix.

Although Amazon provides a built-in mechanism for tracking the reputation of the workers, there is no corresponding mechanism for the requesters. As a result, one might imagine that unscrupulous requesters could refuse to pay their workers, irrespective of the quality of their work. In such a case, there are two recourses for the aggrieved workers. One recourse is to report this to Amazon. If repeated offenses have occurred, the requester will be banned. Second, there are Web sites where workers share experiences and rate requesters (see the Turker community section for more details). Requesters that exploit workers would have an increasingly difficult time getting work done because of these external reputation mechanisms.

The Anatomy of a HIT

All of the tasks available on Mechanical Turk are listed together on the site in a standardized format that allows the workers to easily browse, search, and choose between the jobs being offered. An example of this is shown in Fig. 3. Each job posted consists of many HITs of the same "HIT type," meaning that they all have the same characteristics. Each HIT is displayed with the following information: the title of the HIT, the requester who created the HIT, the wage being offered, the number of HITs of this type available to be worked on, how much time the requester has allotted for

[3] The Mechanical Turk Web site can be difficult to search and navigate, so we will provide URLs whenever possible.

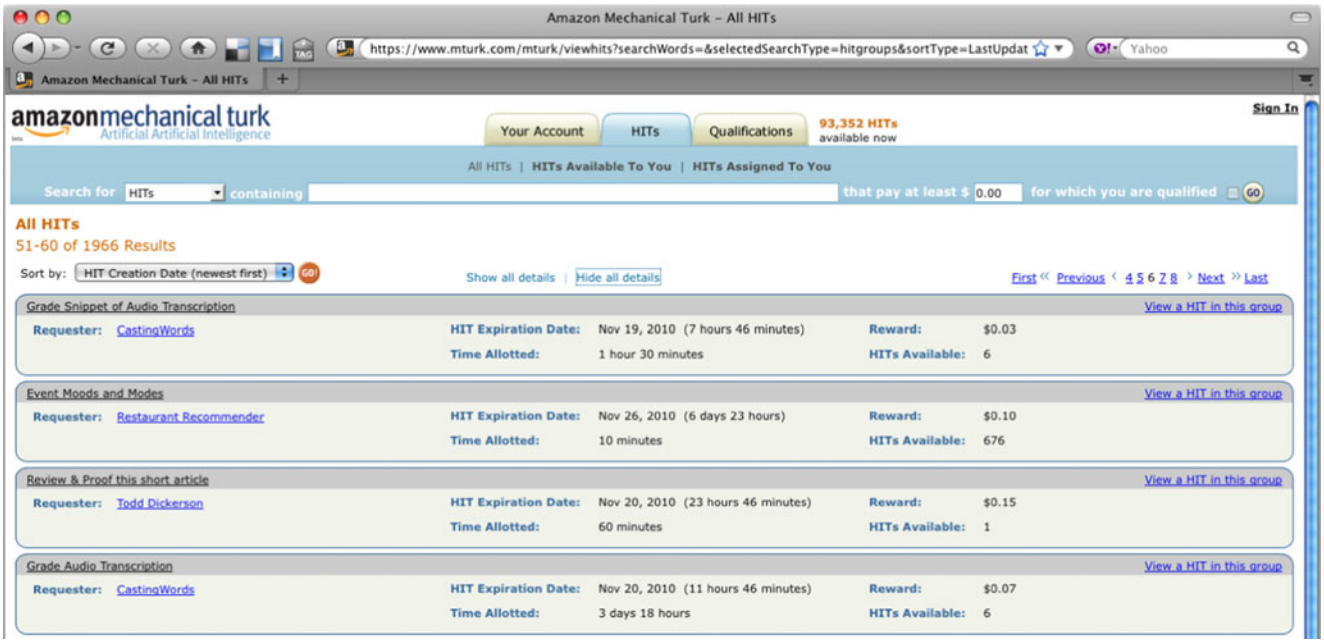

Fig. 3 Screenshot of the Mechanical Turk marketplace

completing the HIT, and when the HIT expires. By clicking on a link for more information, the worker can also see a longer description of the HIT, keywords associated with the HIT, and what qualifications are required to accept the HIT. We elaborate on these qualifications later, which restrict who can work on a HIT and, sometimes, who can preview it. If the worker is qualified to preview the HIT, he or she can click on a link and see the preview, which typically shows what the HIT will look like when he or she works on the task (see Fig. 4 for an example HIT).

All of this information is determined by the requester when creating the HIT, including the qualifications needed to preview or accept the HIT. A very common qualification requires that over 90% of the assignments a worker has completed have been accepted by the requesters. Another common type of requirement is to specify that workers must reside in a specific country. Requesters can also design their own qualifications. For example, a requester could require the workers to complete some practice items and correctly answer questions about the task as a prerequisite to working on the actual assignments. More than one of these qualifications can be combined for a given HIT, and workers always see what qualifications are required and their own value for that qualification (e.g., their own acceptance rate).

Another parameter the requester can set when creating a HIT is how many "assignments" each HIT has. A single HIT can be made up of one or more assignments, and a worker can do only one assignment of a HIT. For example, if the HIT were a survey and the requester only wanted each worker to do the survey once, he or she would make one HIT with many assignments. As another example, if the task was labeling images and the requester wanted three different workers to label every image (say, for data quality purposes), the requester would make as many HITs as there are images to be labeled, and each HIT would have three assignments.

When browsing for tasks, there are several criteria the workers can use to sort the available jobs: how recently the HIT was created, the wage offered per HIT, the total number of available HITs, how much time the requester allotted to complete each HIT, the title (alphabetical), and how soon the HIT expires. Chilton, Horton, Miller, and Azenkot (2010) showed that the criterion most frequently used to find HITs is the "recency" of the HIT (when it was created), and this has led some to periodically add available HITs to the job in order to make it appear as though the HIT is always fresh. While this undoubtedly works in some cases, Chilton and colleagues also found an outlier group of recent HITs that were rarely worked on—presumably, these are the jobs that are being continually refreshed but are unappealing to the workers.

The offered wage is not often used for finding HITs, and Chilton et al., (2010) found a slight negative relationship at the highest wages between the probability of a HIT being worked on and the wage offered. This finding is reasonably explained by unscrupulous requesters using high wages as bait for naive workers—which is corroborated by the finding that higher paying HITs *are* more likely to be worked on, once the top 60 highest paying HITs have been excluded.

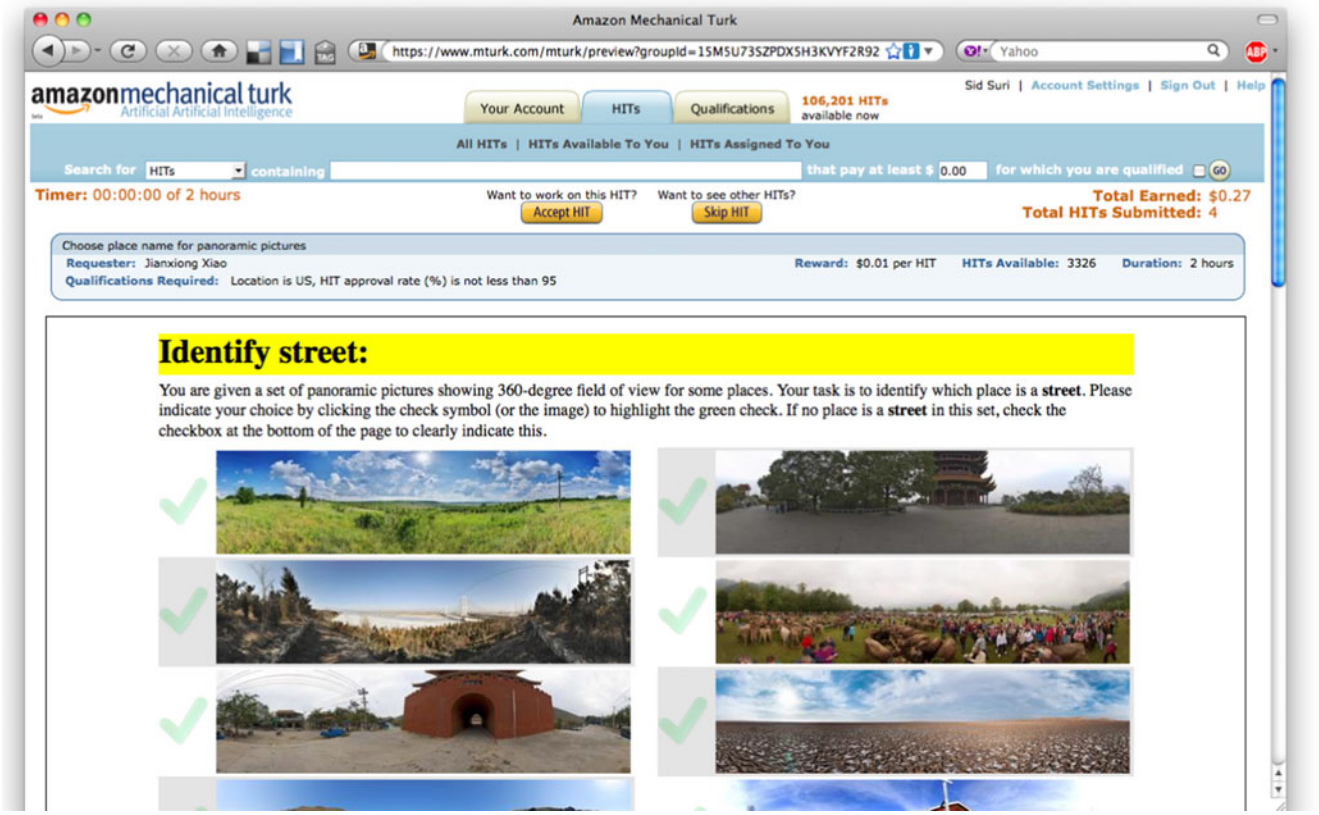

Fig. 4 Screenshot of an example image classification HIT

Internal or external HITs Requesters can create HITs in two different ways, as internal or external HITs. An internal HIT uses templates offered by Amazon, in which the task and all of the data collection are done on Amazon's servers. The advantage of these types of HITs is that they can be generated very quickly and the most one needs to know to build them is HTML programming. The drawback is that they are limited to be single-page HTML forms. In an external HIT, the task and data are kept on the requester's server and are provided to the workers through a frame on the Mechanical Turk site, which has the benefit that the requester can design the HIT to do anything he or she is capable of programming. The drawback is that one needs access to an external server and, possibly, more advanced programming skills. In either case, there is no explicit cue that the workers can use to differentiate between internal and external HITs, so there is no difference from the workers' perspective.

Lifecycle of HIT The standard process for HITs on Amazon's Mechanical Turk begins with the creation of the HIT, designed and set up with the required information. Once the requester has created the HIT and is ready to have it worked on, the requester posts the HIT to Mechanical Turk. A requester can post as many HITs and as many assignments as he or she wants, as long as the total amount owed to the workers (plus fees to Amazon) can be covered by the balance of the requester's Amazon Payments account.

Once the HIT has been created and posted to Mechanical Turk, workers can see it in the listings of HITs and choose to accept the task. Each worker then does the work and submits the assignment. After the assignment is complete, requesters review the work submitted and can accept or reject any or all of the assignments. When the work is accepted, the base pay is taken from the requester's account and put into the worker's account. At this point requesters can also grant bonuses to workers. Amazon charges the requesters 10% of the total pay granted (base pay plus bonus) as a service fee, with a minimum of $0.005 per HIT.

If there are more HITs of the same type to work on after the workers complete an assignment, they are offered the opportunity to work on another HIT of the same type. There is even an option to automatically accept HITs of the same type after completing one HIT. Most HITs have some kind of initial time cost for learning how to do the task correctly, and so it is to the advantage of workers to look for tasks with many HITs available. In fact, Chilton et al. (2010) found that the second most frequently used criterion for sorting is the number of HITs offered, since workers look for tasks where the investment in the initial overhead will pay off with lots of work to be done. As was mentioned, the

requester can prevent this behavior by creating a single HIT with multiple assignments, so that workers cannot have multiple submissions.

The HIT will be completed and will disappear from the list on Mechanical Turk when either of two things occurs: All of the assignments for the HIT have been submitted, or the HIT expires. As a reminder, both the number of assignments that make up the HIT and the expiration time are defined by the requester when the HIT is created. Also, both of these values can be increased by the requester while the HIT is still running.

Reviewing work Requesters should try to be as fair as possible when judging which work to accept and reject. If a requester is viewed as unfair by the worker population, that requester will likely have a difficult time recruiting workers in the future. Many HITs require the workers to have an approval rating above a specified threshold, so unfairly rejecting work can result in workers being prevented from doing other work. Most importantly, whenever possible requesters should be clear in the instructions of the HIT about the criteria on which work will be accepted or rejected.

One typical criterion for rejecting a HIT is if it disagrees with the majority response or is a significant outlier (Dixon, 1953). For example, consider a task where workers classify a post from Twitter as spam or not spam. If four workers rate the post as spam and one rates it as not spam, this may be considered valid grounds for rejecting the minority opinion. In the case of surveys and other tasks, a requester may reject work that is done faster than a human could have possibly done the task. Requesters also have the option of blocking workers from doing their HIT. This extreme measure should be taken only if a worker has repeatedly submitted poor work or has otherwise tried to illicitly get money from the requester.

Improving HIT efficiency

How much to pay One of the first questions asked by new requesters on Mechanical Turk is how much to pay for a task. Often, rather than anchoring on the costs for online studies, researchers come with the prior expectation based on laboratory subjects, who typically cost somewhat more than the current minimum wage. However, recent research on the behavior of workers (Chilton et al., 2010) demonstrated that workers had a reservation wage (the least amount of pay for which they would do the task) of only $1.38 per hour, with an average effective hourly wage of $4.80 for workers (Ipeirotis, 2010a).

There are very good reasons for paying more in lab experiments than on Mechanical Turk. Participating in a lab-based experiment requires aligning schedules with the experimenter, travel to and from the lab, and the effort required to participate. On Mechanical Turk, the effort to participate is much lower since there are no travel costs, and it is always on the worker's schedule. Moreover, because so many workers are using AMT as a source of extra income using free time, many are willing to accept lower wages than they might otherwise. Others have argued that because of the necessity for redundancy in collecting data (to avoid spammers and bad workers), the wage that might otherwise go to a single worker is split among the redundant workers.[4] We discuss some of the ethical arguments around the wages on Mechanical Turk in the Ethics and privacy section.

A concern that is often raised is that lower pay leads to lower quality work. However, there is evidence that for at least some kinds of tasks, there seems to be little to no effect of wage on the quality of work obtained (Marge et al., 2010; Mason & Watts, 2009). Mason and Watts used two tasks in which they manipulated the wage earned on Mechanical Turk, while simultaneously measuring the quantity and quality of work done. In the first study, they found that the number of tasks completed increased with greater wages (from $0.01 to $0.10) but that there was no difference in the quality of work. In the second study, they found that subjects did more tasks when they received pay than when they received no pay per task but saw no effect of actual wage on quantity or quality of the work.

These results are consistent with the findings from the survey paper of Camerer and Hogarth (1999), which showed that for most economically motivated experiments, varying the size of the incentives has little to no effect. This survey article does, however, indicate that there are classes of experiments, such as those based on judgments and decisions (e.g., problem solving, item recognition/recall, and clerical tasks) where the incentive scheme has an effect on performance. In these cases, however, there is usually a change in behavior going from paying zero to some low amount and little to no change in going from a low amount to a higher amount. Thus, the norm on Mechanical Turk of paying less than one would typically pay laboratory subjects should not impact large classes of experiments.

Consequently, it is often advisable to start by paying less than the expected reservation wage, and then increasing the wage if the rate of completed work is too low. Also, one way to increase the incentive to subjects without drastically increasing the cost to the requester is to offer a lottery to subjects. This has been done in other online contexts (Göritz, 2008). It is worth noting that requesters can post HITs that pay nothing, although these are rare and unlikely to be worked on unless there is some additional motivation

[4] http://behind-the-enemy-lines.blogspot.com/2010/07/mechanical-turk-low-wages-and-market.html.

(e.g., benefiting a charity). In fact, previous work has shown that offering subjects financial incentives increases both the response and retention rates of online surveys, relative to not offering any financial incentive (Frick, Bächtiger, & Reips, 2001; Göritz, 2006).

Time to completion The second most often asked question is how quickly work is completed. Of course, the answer to the question depends greatly on many different factors: how much the HIT pays, how long each HIT takes, how many HITs are posted, how enjoyable the task is, the reputation of the requester, and so forth. To illustrate the effect of one of these variables, the wage of the HIT, we posted three different six-question multiple-choice surveys. Each survey was one HIT with 500 assignments. We posted the surveys on different days so that we would not have two surveys on the site at the same time. But we did post them on the same day of the week (Friday) and at the same time of day (12:45 p.m. EST). The $0.05 version was posted on August 13, 2010; the $0.03 version was posted on August 27, 2010; and the $0.01 version was posted on September 17, 2010. We held the time and day of week constant because, as was mentioned earlier, both have shown to have seasonality trends (Ipeirotis, 2010a). Figure 5 shows the results of this experiment. The response rate for the $0.01 survey was much slower than those for the $0.03 and $0.05 versions, which had very similar response rates. While this is not a completely controlled study and is just meant for illustrative purposes, Buhrmester et al. (in press) and Huang et al. (2010) found similar increases in completion time with greater wages. Looking across these studies, one could conclude that the relationship between wage and completion time is positive but nonlinear.

Attrition Attrition is a bigger concern in online experiments than in laboratory experiments. While it is possible for subjects in the lab to simply walk out of an experiment, this happens relatively rarely, presumably because of the social pressure the subjects might feel to participate. In the online setting, however, user attrition can come from a variety of sources. A worker could simply open up a new browser window and stop paying attention to the experiment at hand, he or she could walk away from their computers in the middle of an experiment, a user's Web browser or entire machine could crash, or his or her Internet connectivity could cut out.

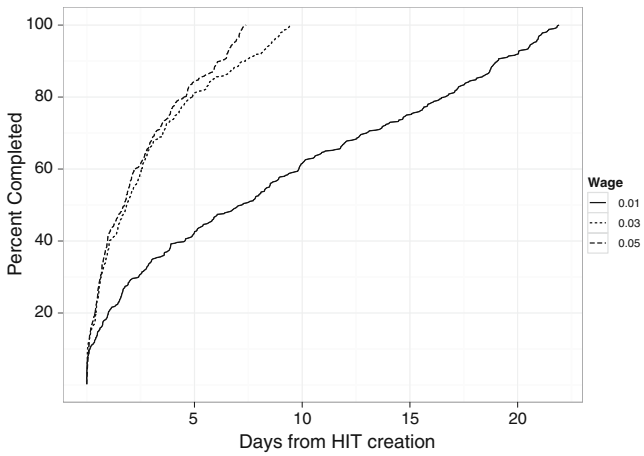

Fig. 5 Response rate for three different six-question multiple-choice surveys conducted with different pay rates

One technique for reducing attrition in online experiments involves asking subjects how serious they are about completing the experiment and dropping the data from those whose seriousness is below a threshold (Musch & Klauer, 2002). Other techniques involve putting anything that might cause attrition, such as legal text and demographic questions, at the beginning of the experiment. Thus, subjects are more likely to drop out during this phase than during the data-gathering phase (see Reips, 2002, and follow-up work by Göritz & Stieger, 2008). Reips (2002) also suggested using the most basic and widely available technology in an online experiment to avoid attrition due to software incompatibility.

Conducting studies on Mechanical Turk

In the following sections, we show how to conduct research on Mechanical Turk for three broad classes of studies. Depending on the specifics of the study being conducted, experiments on Mechanical Turk can fall anywhere on the spectrum between laboratory experiments and field experiments. We will see examples of experiments that could have been done in the lab but were put on Mechanical Turk. We will also see examples of what amount to online field experiments. We outline the general concepts that are unique to doing experiments on Mechanical Turk throughout this section and elaborate on the technical details in the Appendix.

Surveys

Surveys conducted on Mechanical Turk share the same advantages and disadvantages as any online survey (Andrews, Nonnecke, & Preece, 2003; Couper, 2000). The issues surrounding online survey methodologies have been studied extensively, including a special issue of *Public Opinion Quarterly* devoted exclusively to the topic (Couper & Miller, 2008). The biggest disadvantage to conducting surveys online is that the population is not representative of any geographic area or segment of population, and Mechanical Turk is not even particularly representative of the online population.

Methods have been suggested for correcting these selection biases in surveys generally (Berk, 1983; Heckman, 1979), and the appropriate way to do this on Mechanical Turk is an open question. Thus, as with any sample, whether it be online or offline, researchers must decide for themselves whether the subject pool on Mechanical Turk is appropriate for their work.

However, as a tool for conducting pilot surveys or for surveys that do not depend on generalizability, Mechanical Turk can be a convenient platform for constructing surveys and collecting responses. As was mentioned in the Introduction, relative to other methodologies, Mechanical Turk is very fast and inexpensive. However, this benefit comes with a cost: the need to validate the responses to filter out bots and workers who are not attending to the purpose of the survey. Fortunately, validating responses can be managed in several relatively time- and cost-effective ways, as outlined in the Quality assurance section. Moreover, because workers on Mechanical Turk are typically paid after completing the survey, they are more likely to finish it once they start (Göritz, 2006).

Amazon provides a HIT template to aid in the construction of surveys (Amazon also provides other templates, which we discuss in the HIT templates section of the Appendix). Using a template means that the HIT will run on an Amazon machine. Amazon will store the data from the HIT, and the requester can retrieve the data at any point in the HIT's lifecycle. The HIT template gives the requester a simple Web form where he or she defines all the values for the various properties of the HIT, such as the number of assignments, pay rate, title, and description (see the Appendix for a description of all of the parameters of an HIT). After specifying the properties for the HIT, the requester then creates the HTML for the HIT. In the HTML, the requester specifies the type of input and content for each input type (e.g., survey question), and for multiple-choice questions, the value for each choice. The results are given back to the requester in a column-separated file (.csv). There is one row for each worker and one column for each question, where the worker's response is in the corresponding cell. Requesters are allowed to preview the modified template to ensure that there are no problems with the layout.

Aside from standard HTML, HIT templates can also include variables that can have different values for each HIT, which Mechanical Turk fills in when a worker previews the HIT. For example, suppose one did a simple survey template that asked one question: What is your favorite ${object}? Here, ${object} is a variable. When designing the HIT, a requester could instantiate this variable with a variety of values by uploading a .csv file with ${object} as the first column and all the values in the rows below. For example, a requester could put in values of color, restaurant, and song. If done this way, three HITs would be created, one for each of these values. Each one of these three HITs would have ${object} replaced with color, restaurant, and song, respectively. Each of these HITs would have the same number of assignments as specified in the HIT template.

Another way to build a survey on Mechanical Turk is to use an external HIT, which requires you to host the survey on your own server or use an outside service. This has the benefit of increased control over the content and aesthetics of the survey, as well as allowing one to have multiple pages in a survey and, generally, more control over the form of the survey. This also means the data is secure because it is never stored on Amazon's servers. We will discuss external HITs more in the next few sections.

It is also possible to integrate online survey tools such as SurveyMonkey and Zoomerang with Mechanical Turk. One may want to do this instead of simply creating the survey within Mechanical Turk if one has already created a long survey using one of these tools and would simply like to recruit subjects through Mechanical Turk. To integrate with a premade survey on another site, one would create a HIT that provides the worker with a unique identifier, a link to the survey, and a submit button. In the survey, one would include a text field for the worker to enter their unique identifier. One could also direct the worker to the "dashboard" page (https://www.mturk.com/mturk/dashboard) that includes their unique worker ID, and have them use that as their identifier on the survey site. The requester would then know to approve only the HITs that have a survey with a matching unique identifier.

Random assignment

The cornerstone of most experimental designs is random assignment of subjects to different conditions. The key to random assignment on Mechanical Turk is ensuring that every time the study is done, it is done by a new worker. Although it is possible to have multiple accounts (see the Workers section), it is against Amazon's policy, so random assignment to unique Worker IDs is a close approximation to uniquely assigning individuals to conditions. Additionally, tracking worker IP addresses and using browser cookies can help ensure unique workers (Reips, 2000).

One way to do random assignment on Mechanical Turk is to create external HITs, which allows one to host any Web-based content within a frame on Amazon's Mechanical Turk. This means that any functionality one can have with Web-based experiments—including setups based on JavaScript, PHP, Adobe Flash, and so forth—can be done on Mechanical Turk. There are three vital components to random assignment with external HITs. First, the URL of the landing page of the study must be included in the parameters for the external HIT so Mechanical Turk will know where the code for the experiment resides. Second, the code for the experiment must capture three variables passed to it from Amazon when a worker accepts the HIT:

the "HITId," "WorkerId," and "AssignmentId." Finally, the experiment must provide a "submit" button that sends the Assignment ID (along with any other data) back to Amazon (using the externalSubmit URL, as described in the Appendix).

For a Web-based study that is being hosted on an external server but delivered on Mechanical Turk, there are a few ways to ensure that subjects are being assigned to only one condition. The first way is to post a single HIT with multiple assignments. In this way, Mechanical Turk ensures that each assignment is completed by a different worker: each worker will see only one HIT available. Because every run through the study is done by a different person, random assignment can be accomplished by ensuring that the study chooses a condition randomly every time a worker accepts a HIT.

While this method is relatively easy to accomplish, it can run into problems. The first arises when one has to rerun an experiment. There is no built-in way to ensure that a worker who has already completed a HIT will not be able to return the next time a HIT is posted and complete it again, receiving a different condition assignment the second time around. Partially, this can be dealt with by careful planning and testing, but some experimental designs may need to be repeated multiple times while ensuring that subjects are receiving the same condition each time. A simple but more expensive way to deal with repeat workers is to allow all workers to complete the HIT multiple times and disregard subsequent submissions. A more cost-effective way is to store the mapping between a Worker ID (passed to the site when the worker accepts the HIT) and that worker's assigned condition. If the study is built so that this mapping is checked when a worker accepts the HIT, the experimenter can be sure that each worker experiences only a single condition. Another option is to simply refuse entry to workers who have already done the experiment. In this case, requesters must clearly indicate in the instructions that workers will be allowed to do the experiment only once.

Mapping the Worker ID to the condition assignment does not, of course, rule out the possibility that the workers will discuss their condition assignments. As we discuss in the Turker community section, workers are most likely to communicate about the HITs on which they worked in the online forums focused on Mechanical Turk. It is possible that these conversations will include information about their condition assignments, and there is no way to prevent subjects from communicating. This can also be an issue in general online experiments and in multisession offline experiments. Mechanical Turk has the benefit that these conversations on the forums can be monitored by the experimenter.

When these methods are used, the preview page must be designed to be consistent with all possible condition assignments. For instance, Mason and Watts (2009) randomized the pay the subjects received. Because the wage offered per HIT is visible before the worker even previews the HIT, the different wage conditions had to be done through bonuses and could not be revealed until after the subject had accepted the HIT.

Finally, for many studies, it is important to calculate and report intent-to-treat effects. Imagine a laboratory study that measures the effect of blaring noises on reading comprehension that finds the counterintuitive result that the noises improve comprehension. This result could be explained by the fact that there was a higher dropout rate in the "noises" condition and the remainder either had superior concentration or were deaf and, therefore, unaffected. In the context of Mechanical Turk, one should be sure to keep records of how many people accepted and how many completed the HIT in each condition.

Synchronous experiments

Many experimental designs have the property that one subject's actions can affect the experience and, possibly, the payment of another subject. Mechanical Turk was designed for tasks that are asynchronous in nature, in which the work can be split up and worked on in parallel. Thus, it is not a priori clear how one could conduct these types of experiments on Mechanical Turk. In this section, we describe one way synchronous participation can be achieved: by building a subject panel, notifying the panel of upcoming experiments, providing a "waiting room" for queuing subjects, and handling attrition during the experiment. The methods discussed here have been used successfully by Suri and Watts (2011) in over 100 experimental sessions, as well as by Mao, Parkes, Procaccia, and Zhang (2011).

Building the panel An important part of running synchronous experiments on Mechanical Turk is building a panel of subjects to notify about upcoming experiments. We recommend building the panel by either running several small, preliminary experiments or running a different study on Mechanical Turk and asking subjects whether they would like to be notified of future studies. In these preliminary experiments, the requester should require that all workers who take part in the experiment be first-time players, indicate this clearly in the instructions, and build it into the design of the HIT. Since the default order in which workers view HITs is by time of creation, with the newest HITs first, a new HIT is seen by quite a few workers right after it has been created (Chilton et al., 2010). Thus, we found requiring only 4 to 8 subjects works well, since this ensures that the first worker to accept the HIT will not have to wait too long before the last worker accepts this HIT and the session can begin.

At the end of the experiment, perhaps during an exit survey, the requester can ask the workers whether they

would like to be notified of future runs of this or other experiments. When subjects are asked whether they would like to be notified of future studies, we recommend making the default option to not be notified and asking the workers to opt in. Since most tasks on Mechanical Turk are rather tedious, even a moderately interesting experiment will have a very high opt-in rate. For example, the opt-in rate was 85% for Suri and Watts (2011). In addition, since the workers are required to be fresh (i.e., never having done the experiment before), this method can be used to grow the panel fairly rapidly. Figure 6 shows the growth of one panel using this method, and we have seen even faster growth in subsequent studies. It should be clear to the subjects joining the panel whether they are being asked to do more studies of the same type or studies of a different type from the same requester. If they agree to the latter, the panels can be reused from experiment to experiment. Göritz et al. (2008) showed that paying individuals between trials of an experiment can increase response and retention rates, although their results were attenuated by the fact that their subjects had to take the time to sign up for a PayPal account, which is unnecessary on Mechanical Turk.

In our experience, small preliminary experiments have a benefit beyond growing the panel: they serve to expose bugs in the experimental system. Systems where users concurrently interact can be difficult to test and debug, since it can be challenging for a single person to get the entire system in a state where the bug reveals itself. Also, it is better for problems to reveal themselves with a small number of workers in the experiment than with a large number.

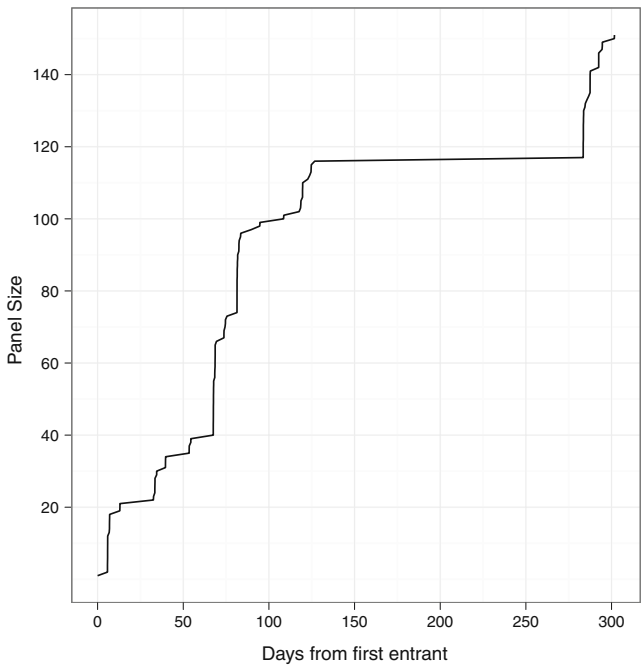

Fig. 6 Rate of growth of panel from Suri and Watts (2011). Periods without growth indicate times between experimental runs

Notifying workers Now that we have shown how to construct a panel, we next show how to take advantage of it. Doing so involves a method that Mechanical Turk provides for sending messages to workers. Before the experiment is to run, a requester can use the NotifyWorkers API call to send workers a message indicating the times at which the next experiment(s) will be run (see the Appendix for more details, including how to ensure that the e-mails are delivered and properly formatted). We found that sending a notification the evening before an experiment was sufficient warning for most workers. We also found that conducting experiments between 11 a.m. and 5 p.m. EST resulted in the experiment filling quickly and proceeding with relatively few dropouts. Also, if one wants to conduct experiments with n subjects simultaneously, experience has shown us that one needs a panel with $3n$ subjects in it. Using this rule of thumb, we have managed to run as many as 45 subjects simultaneously. If the panel has substantially more than $3n$ subjects, many workers might get shut out of the experiment, which can be frustrating to them. In this case, one could either alter the experiment to allow more subjects or sample $3n$ subjects from the panel.

Waiting room Since the experiment is synchronous, all of the workers must begin the experiment at the same time. However, there will inevitably be differences in the time that workers accept the HIT. One way to resolve this issue is to create an online "waiting room" for the workers. As more workers accept the HIT, the waiting room will fill up until the requisite number of workers have arrived and the experiment can begin. We have found that indicating to the workers how many people have joined and how many are required provides valuable feedback on how much time they can expect to wait. Once one instance of the experiment has filled up and begun, the waiting room can then either inform additional prospective workers that the experiment is full and they should return the HIT or funnel them into another instance of the experiment. The waiting room and the message that the experiment is full are good opportunities to recruit more subjects into the study and/or advertise future runs of the experiment.

Attrition In the synchronous setting, it is of paramount importance to have a time-out after which, if a subject has not chosen an action, the system chooses one for him or her. Including this time-out and automated action avoids having an experiment stall, with all of the subjects waiting for a missing subject to take an action. Because experiments on Mechanical Turk are inexpensive, an experimenter can simply throw out trials with too much attrition. Alterna-

tively, the experimenter can use the dropouts as an opportunity to have a (dummy) confederate player act in a prescribed way to observe the effect on the subjects. In the work of Suri and Watts (2011), the authors discarded experiments where fewer than 90% of the actions were done by humans (as opposed to default actions chosen by the experimental system). Out of 94 experiments run with 20–24 players, 21 had to be discarded using this criterion.

Quality assurance

The downside to fast and cheap data is the potential for low quality. From the workers' perspective, they will earn the most money by finding the fastest and easiest way to complete HITs. As was mentioned earlier, most workers are not motivated primarily by the financial returns and genuinely care about the quality of their work, but nearly all of them also care, at least a little, about how efficiently they are spending their time. However, there are a few workers who do not care about the quality of the work they put out as long as they earn money (they are typically characterized as *spammers*). Moreover, there are reports of programs (*bots*) designed to automatically complete HITs (McCreadie et al., 2010), and these are essentially guaranteed to provide bad data.

To ensure that the instructions for the HIT are clear, requesters can add a text box to their HIT asking whether any part of it was confusing. In addition, there has been a significant amount of research put into methods for improving and assuring data quality. The simplest and, probably most commonly used method is obtaining multiple responses. For many of the common tasks on Mechanical Turk, this is a very effective and cost-efficient strategy. For instance, Snow and colleagues compared workers on Mechanical Turk with expert labelers for natural language tasks and determined how many Mechanical Turk worker responses were required to get expert-level accuracy (Snow et al., 2008), which ranged from two to nine with a simple majority rule and one or two with more sophisticated learning algorithms. Sheng, Provost, and Ipeirotis (2008) used labels acquired through Mechanical Turk as input to a machine-learning classifier and showed over 12 data sets that, using the "majority vote" label obtained from multiple labels, improved classification accuracy in all cases. In follow-up work, Ipeirotis, Provost, and Wang (2010) developed an algorithm that factors both per-item classification error and per-worker biases to reduce error with even fewer workers and labels.

However, for most survey and experimental data, where individual variability is an important part of the data obtained, receiving multiple responses may not be an option for determining "correct" responses. For surveys and some experimental designs, one option is to include a question designed to discourage spammers and bots, something that requires human knowledge and the same amount of effort as other questions in the survey but has a verifiable answer that can be used to vet the submitted work. Kittur, Chi, and Suh (2008) had Mechanical Turk workers rate the quality of Wikipedia articles and compared them with experts. They found a significant increase in the quality of the data obtained when they included additional questions that had verifiable answers: The proportion of invalid responses went from 48.6% to 2.5%, and the correlation of responses to expert ratings became statistically significant. If you include these "captcha" or "reverse Turing test" questions, it is advisable to make it clear that workers will not be paid if the answers to the verifiable questions are not answered correctly. Also, if the questions are very incongruent with the rest of the study, it should be clear that they are included to verify the legitimacy of the other answers. Two examples of such questions are "Who is the president of the United States?" and "What is 2 + 2?" We asked the former question as a captcha question in one of the surveys described in Fig. 5. Out of 500 responses, only six people got the question wrong, and three people did not answer the question.

In some cases, it may be possible to have the workers check their own work. If responses in a study do not have *correct* answers but do have *unreasonable* answers, it may be possible to use Mechanical Turk workers to vet the responses of others' work. For instance, if a response to a study requires a free-text response, one could create another HIT for the purpose of validating the response. It would be a very fast and easy task for workers (and therefore, inexpensive for requesters) to read these responses and verify that they are a coherent and reasonable response to the question asked. Little, Chilton, Goldman, and Miller (2010) found that this sort of self-correction can be a very efficient way of obtaining good data.

Finally, another effective way of filtering bad responses is to look at the patterns of responses. Zhu and Carterette (2010) looked at the pattern of responses on surveys and found that low-quality responses had very low-entropy patterns of response—always choosing one option (e.g., the first response to every question) or alternating between a small number of options in a regular pattern (e.g., switching between the first and the last responses). The time spent completing individual tasks can also be a quick and easy means of identifying poor/low-effort responses—so much so that filtering work by time spent is built into the Mechanical Turk site for reviewing output. When Kittur et al. (2008) included verifiable answers in their study, they found that the time spent completing each survey went up from 1.5 min to over 4 min. It is usually possible to determine a lower bound on the amount of time required to

actually participate in the study and to filter responses that fall below this threshold.

Security

As was stated above, the code for an external HIT typically resides on the requester's server. The code for the HIT is susceptible to attacks from the general Internet population, because it must be executable by any machine on the Internet to work on Mechanical Turk. Here, we provide a general overview of some security issues that could affect a study being run as an external HIT and ways to mitigate the issues. In general, it is advisable to consult an expert in computer security when hosting a public Web site.

To begin with, we advocate that requesters make an automated nightly backup of the work submitted by the workers. In order to ensure the integrity of the data gathered, a variety of security precautions are necessary for external HITs. Two of the most common attacks on Web-based applications are database (most commonly SQL) injection attacks and Cross Site Scripting (XSS) attacks. A database injection attack can occur on any system that uses a database to store user input and experiment parameters, which is a common way to design Web-based software. A database injection attack can occur at any place where the code takes user input. There are a variety of inputs that a malicious user could give that would trick the database underlying the requester's software to run it. Such code could result in the database executing an arbitrary command specified by the malicious user, and some commands could compromise the data that have been stored. Preventing this type of attack is a relatively straightforward matter of scrubbing user input for database commands—for instance, by removing characters recognized by the database as a command. There are a variety of software libraries in many programming languages that will aid in this endeavor specific to the particular implementation of the database and software that can be found for free online.

Cross-site scripting attacks (XSS) are another type of code injection attack. Here, a malicious user would try to inject arbitrary scripting code, such as malicious JavaScript code, into the input in an attempt to get the requester's server to run the code. Here again, one of the main methods for preventing this type of attack is input validation. For example, if the input must be a number, the requester's code should ensure that the only characters in the input are numbers, a plus or minus sign, or a decimal point. Another preventative measure is to "HTML escape" the user input, which ensures that any code placed in the input by a malicious user will not be executed. We caution prospective requesters who use external HITs to take these measures seriously.

Code security is not the only type of security necessary for experiments on Mechanical Turk. The protocol that the requester uses to run the experiment must also be secure. We demonstrate this with an example. The second author of this article attempted a synchronous experiment that was made up of many HITs. The first part of the HIT was to take a quiz to ensure understanding of the experiment. If a worker passed the quiz, he or she would enter the waiting room and then eventually go into the experiment. Workers were paid \$0.50 for passing the quiz, along with a bonus, depending on their actions in the experiment. Two malicious workers then accepted as many HITs as they could at one time. Meanwhile, the benevolent workers accepted one HIT each, passed the quiz, went into the waiting room, and eventually began the experiment. After accepting as many as possible, the malicious workers then filled out the quiz correctly for each HIT, submitting them after the experiment began. Thus, the malicious workers were paid for their quizzes and were not allowed into the experiment. The second author got bilked out of roughly \$200. The fix was simply to make the experiment one HIT with many assignments, so that each Turker could accept only one HIT at a time.

Ethics and privacy

As with any research involving human subjects, care must be taken to ensure that subjects are treated in an ethical manner and that the research follows standard guidelines such as the Belmont Report (Ryan et al., 1979). While oversight of human research is typically managed by the funding or home institution of the researcher, it is the researcher's responsibility to ensure that appropriate steps are taken to conduct ethical research.

Mechanical Turk and other crowdsourcing sites define a relatively new ethical and legal territory, and therefore the policies surrounding them are open to debate. Felstiner (2010) reviews many of the legal grounds and ethical issues related to crowdsourcing and is an excellent starting point for the discussion. There are also many ethical issues that apply to online experimentation in general. While this has been covered extensively elsewhere (Barchard & Williams, 2008), we felt that it would be helpful to the reader to highlight them here. In the following section, we touch on issues relevant to Institutional Review Boards (IRBs) when proposing research on Mechanical Turk.

Informed consent

Informed consent of subjects is nearly always a requirement for human subject research. One way to obtain consent on

Mechanical Turk is to have a statement on the preview page of the HIT that explains the purpose of the study, the risks and benefits of the research (to the extent that they can be explained), and a means by which the subjects can contact the researcher (and/or the human subjects review board) about problems they may experience in the course of participating in the study. This way, the potential subjects have all of the information they need to make an informed decision about whether they want to participate before accepting the HIT. Alternatively, the initial preview page can be thought of as the "call for participation," and the informed consent statement can be provided after they have accepted the HIT, followed by an option to continue or return the HIT. Which method one employs likely depends on the constraints of the research and the human subjects review board.

Debriefing

Similarly, it is important to ensure that at the end of participation, the workers understand the purpose of the experiment and are reminded how to contact the researcher in the event of questions or complaints. Providing a debriefing statement is even more important if there is any deception or undisclosed information in the study. For these cases especially, we suggest presenting the debriefing statement after the participation is completed but before the submit button is made available to the workers, to ensure that they see it before they can be paid.

Additionally, there is nothing built into Mechanical Turk that prevents researchers from using deception. Some researchers may wish to avoid having their subject pool "contaminated" with subjects who have gone through an experiment that uses deception. To mitigate this issue, a researcher could create his or her own panel of workers and guarantee to them that they will never be deceived by that researcher's experiment. This would help foster a norm of trust between the researcher and the subjects in his or her panel.

Restricted populations

Another issue that must be considered is the possibility of minors or other restricted populations participating in the experiment. Although reported demographics of workers under 18 years of age are very low, there are no guarantees that the workers will be adults, and therefore, precautions must be taken to validate the age of the workers. Unfortunately, there are no built-in means of checking the age of the workers or whether they fall into any other restricted population, such as convicted felons or mentally disabled individuals. The best we can suggest, as with any online research, is to have an initial screening with voluntarily provided information that prevents restricted populations from participating (Barchard & Williams, 2008).

Compensation

One frequently heard complaint about the ethics of using Mechanical Turk centers around the low wages the workers receive. Legally, the workers on Mechanical Turk are considered "independent contractors" and, therefore, fall outside the minimum wage laws; there is an established contract between the requester and worker to do the work at the agreed wage independent of the time required to do the task. In the United States, requesters are required to provide an IRS Form 1099 if any single worker earns over the IRS tax reporting threshold (currently $600[5]), and workers are required to report their income to the IRS if they earn more than the IRS threshold. Because of the low wages on Mechanical Turk, however, this rarely happens.

Although some issues remain (such as the enforcement of Amazon's stated policies), there are some reasonable arguments for the low wages on Mechanical Turk. From the employer's perspective, some have argued[6] that because Mechanical Turk is effectively a "market for lemons" (Akerlof, 1970), the equilibrium wage is lower than if the requesters could more easily check the quality of work before compensating the workers. From the worker's perspective, as was mentioned earlier, most workers are not relying on the wages earned on Mechanical Turk for necessities. More important, the working conditions and hours are wholly determined by the worker. There is absolutely no direct or indirect obligation or constraint on the workers to do any work on Mechanical Turk. In other words, the decision to engage in the contract is completely at the worker's liberty, a situation that rarely, if ever, exists in other employment situations.

Confidentiality

Short of falsifying the information submitted during the requester sign-up period, it is not possible for a requester to remain anonymous on Mechanical Turk. That being said, it is possible for a requester to use the name of an institution or company or to provide a fake name, although these uses are discouraged or disallowed because it makes it harder for the workers to track the reputation of the requester. In contrast, it is the norm for workers to remain anonymous on Mechanical Turk. Worker IDs are anonymized strings and do not contain personally identifiable information. Howev-

[5] http://www.irs.gov/instructions/i1099msc/ar02.html.

[6] http://behind-the-enemy-lines.blogspot.com/2010/07/mechanical-turk-low-wages-and-market.html.

er, if a requester were to send a note to a worker using the NotifyWorkers API call and the worker were to reply, the reply would go from the worker's e-mail address to the requester's. The e-mail address of the worker would therefore be revealed to the requester.

There are also privacy issues concerning where the data gathered on Mechanical Turk are stored. On a template HIT, Amazon has access to the data, and although they state that they will not look at the data, it may still be a concern for experiments or behavioral research that gather personally sensitive data. For example, suppose that a requester did a survey asking whether a worker has a sexually transmitted disease. If this were done using an internal HIT, Amazon would have a list of Worker IDs, along with their account information and their answer to the survey. One advantage of the external HIT, therefore, is that the data go straight from the worker to the external server managed by the requester, so the data are never available to Amazon. In addition, a requester can use the https protocol to ensure that the data that are transferred between a worker's browser and the requester's server running an external HIT are encrypted (Schmidt, 2007).

Turker community

A rich online community has sprung up around Mechanical Turk, much of which focuses on the reputation of requesters. There is an asymmetry in the reputations of workers and requesters on Mechanical Turk. Requesters can reject (i.e., refuse to pay for) any or all work done by a worker without giving a reason. Moreover, any requester can choose to refuse workers whose percentage of work rejected is higher than some threshold. These features make the reputation of workers, which is encoded by their acceptance rate, a fundamental feature of Mechanical Turk. However, there is no systematic reputation mechanism for requesters. As a result, off-site reputation systems have been developed, including Turkopticon[7] and Turker Nation.[8] Turkopticon is a site that allows workers to rate requesters along four axes: communicativity, generosity, fairness, and promptness. Turker Nation is an online bulletin board where workers routinely comment on requesters and communicate about individual HITs. It is strongly encouraged that new requesters "introduce" themselves to the Mechanical Turk community by first posting to Turker Nation before putting up HITs. These external sites can have a strong effect on the acceptance rate of HITs and therefore serve effectively as a watchdog on abusive requesters. Moreover, the forums allow one to monitor workers' reactions to the study, which at times can provide insight into one's methods or even the substantive focus of the research itself.

There are many instances where requesters could find themselves interacting directly with workers. The Mechanical Turk interface allows workers to send the requester of a HIT a message. For instance, workers may wish to contact requesters if part of their HIT is unclear or confusing. Similarly, workers may post comments on Turker Nation regarding either positive or negative aspects of a HIT. We advocate that requesters keep a professional rapport with their workers as if they were company employees. This will benefit the requester by maintaining a high reputation among workers, leading more workers to do their HITs in the future.

Finally, we note that there are a number of blogs where researchers who either conduct experiments using Mechanical Turk or study Mechanical Turk itself often post. These sites—"A Computer Scientist in a Business School,"[9] "Experimental Turk,"[10] "Deneme,"[11] and "Crowdflower"[12]—are useful for researchers interested in keeping up on the latest Mechanical Turk research.

Conclusion

In this article, we have described a tool for behavioral researchers to conduct online studies: Amazon's Mechanical Turk. This crowdsourcing platform provides researchers with access to a massive subject pool available 365 days a year, freeing academic scientists from the boom-and-bust semester cycle. The workers on Mechanical Turk generally come from a more diverse background than the typical college undergraduate, and in numbers that equal or exceed the size of even large universities' subject pools. Furthermore, since the reservation wage of workers is only $1.38 per hour (Chilton et al., 2010) (with an effective wage of roughly $4.80; Ipeirotis, 2010a), the subjects tend to be less comparable to or expensive than subjects recruited through other means. There have also been a number of studies that validate the behavior of workers, as compared with offline behavior.

In an overview of the basics of Mechanical Turk, we described the two roles on the site, *requesters* and *workers*, and the jobs they perform, called *human intelligence tasks*. We then explained how to conduct three types of studies on Mechanical Turk: surveys, standard random assignment experiments, and synchronous experiments.

We hope that this guide opens doors for behavioral research of all kinds, from traditional laboratory studies, to

[7] http://turkopticon.com.
[8] http://www.turkernation.com.
[9] http://behind-the-enemy-lines.blogspot.com.
[10] http://experimentalturk.wordpress.com.
[11] http://groups.csail.mit.edu/uid/deneme.
[12] http://blog.crowdflower.com.

field experiments, to novel research on the crowdsourcing platform itself.

Author Note Both authors contributed equally to this work. We would like to thank Duncan J. Watts and Daniel G. Goldstein for encouraging us to write this article and the many people who provided helpful feedback on earlier drafts.

Appendix

In this appendix, we describe the technical details involved in engineering HITs on Mechanical Turk, to provide guidance to the researchers actually building the studies on Mechanical Turk beyond what is available on the Mechanical Turk site and specifically directed toward behavioral researchers.

HIT parameters

In order to create a HIT, the requester must specify the following parameters:

Wage This is the amount the worker will receive for each completed and approved assignment.

Title This is text that briefly describes the task.

Description This is longer text that provides more information about the HIT.

Keywords These are some comma-separated words that workers can use to search for the HIT.

Access Key, Secret Key These are the digital keys that identify the requester. They were provided when the requester account was created.

Assignment duration This is the allotted amount of time the worker has to complete the HIT after accepting it. Anecdotal reports suggest that workers use the amount of time allotted for the assignment to gauge whether this is a relatively long or short HIT, as compared with the other HITs available.

Lifetime This is the maximum amount of time the HIT will be available for workers to accept after it is posted on Mechanical Turk by the requester if all of the HITs are not first completed.

Question field The Question field is an XML string. For internal HITs, it specifies the content of the HTML form. For external HITs, it specifies the URL for the content of the HIT and the height of the frame in pixels to display the HIT. For applications that require the entire frame to be in view, we recommend a maximum frame height of 600 pixels.[13]

Qualifications These are the requirements the worker must have to work on the HIT. Multiple Qualifications can be required for a single HIT.

Max assignments This is the number of assignments for each HIT.

Auto approval delay This is the amount of time after an Assignment is submitted before Amazon automatically approves the work and pays the worker.

Requester annotation These are notes the requester can provide when making an API call. For instance, one could tag a HIT with an arbitrary string that could be used for tracking purposes.

HIT types All HITs with the same title, description, keywords, reward, assignment duration, auto-approval delay, and qualifications will automatically be assigned to the same HIT Type ID, or requesters can manually create a HIT Type by specifying those parameters. Requesters can also use a HIT Type to create HITs, so that the new HITs automatically inherit the parameters defined by the HIT Type. HITs with the same HIT Type ID are assigned the same URL, which makes it easier for requesters to direct workers to their HIT. Organizing similar HITs under the same HIT Type also signals to workers that the tasks are similar, attracting workers who are already familiar with the task.

Sandbox

Mechanical Turk provides a "sandbox" (http://requestersandbox.mturk.com) that requesters can use to design, build, and fine tune their HITs before making them "live" and available to the workers by putting the HIT on the production site. The sandbox allows extensive testing of the entire Mechanical Turk process before actually launching the final product. The process of building the HIT on the sandbox and production sites is identical, and once the HIT is completed on the sandbox, it is trivial to move it over to production. If using templates, the code can simply be copied and pasted into a HIT template existing on the production site. If using the command-line tools or the API, it simply requires modifying a line in a configuration file to

[13] See http://www.w3schools.com/browsers/browsers_display.asp for the distribution of screen sizes on the Web.

shift the code to the production site (e.g., from "http://requestersandbox.mturk.com" to "http://www.mturk.com"). Once requesters have gained some experience with building HITs they may wish to put their HITs directly on the production site, but we encourage new requesters to use the sandboxes.

HIT templates

Perhaps the easiest method for creating a HIT is to use the HIT templates that Mechanical Turk provides. Any HIT that is a single-page HTML form can be constructed using one of these templates. In this section, we describe the workflow for designing, testing, debugging, and launching such a HIT.

Amazon provides 13 different templates that give basic HTML layouts for many common tasks on Mechanical Turk, such as image labeling, data correction, and search relevance. The requesters can then modify these basic templates to suit their needs. For example, one template gives the outline of a simple survey, which we will use as a running example throughout this section. Another useful template is the blank template into which any HTML code can be inserted.

Before writing the actual survey questions and answers, Mechanical Turk provides a simple Web form where the experimenter can define all the values for the various properties of the HIT described in the HIT parameters section. After specifying the properties for the HIT, the requester then provides Mechanical Turk the HTML specifying the content of the HIT, which can be done with the rudimentary HTML editor provided by Amazon. Requesters are shown a preview of the template to ensure that there are no bugs in the layout.

As discussed in the section on surveys, Mechanical Turk also allows requesters to specify variables in their HTML code. Variables provide an easy way to generate a variety of HITs that all have the same general format but differ in the values of the variables. For example if one offered a HIT that asked people which of two images they preferred, the HIT might have two variables, ${pic1}$ and ${pic2}$. Then, in a separate .csv file, there would be one column corresponding to the URLs for the image to be shown as ${pic1}$ and one column for the URLs to be shown as ${pic2}$. There would be as many HITs as there are pairs of pictures. The rest of the properties of the HIT, such as pay rate, number of assignments, and so forth, would be the same as those specified in the template.

After requesters are satisfied with the state of their HIT, they can publish their HIT. If the HIT was created on requestersandbox.mturk.com, the HIT can be worked on at workersandbox.mturk.com. This is useful for debugging in a number of ways. First, requesters can provide the URL for the HIT to colleagues to get feedback on the clarity of the questions and answers. Second, once colleagues have done the HIT, requesters can download the results file to ensure that the questions and answers are encoded properly by their HTML. A good practice for testing multiple choice answers is to ensure that all possible answers have been given in test runs and ensure that they are encoded properly in the data output.

After testing the HIT on the sandbox, the requester can copy and paste the HTML form from the sandbox site to the production site. Whether or not the HIT was placed on the sandbox or on the production site, Mechanical Turk provides the requester with the ability to download all the results delivered up to that point in time. Also, the requester can choose to extend the amount of time the HIT will appear on Mechanical Turk, expire the HIT early, or add more assignments to the HIT.

After the HIT is launched, workers will accept the HIT and submit their work. As they do so, the results become available for the requester to review. When the HIT is made via a template, Mechanical Turk provides the requester with a simple Web form that shows the results of the workers, along with a check box indicating whether to accept or reject the work.

One way to view the results is through the Mechanical Turk site, which presents the results in a spreadsheet-like format. This way of viewing them allows one to filter by various features, such as how quickly the HIT was completed, and select groups of results to approve or reject. Additionally, the data can be returned in a column-separated format (.csv). This file contains the HIT ID, Assignment ID, and Worker ID for each assignment completed, as well as the properties of the HIT as specified by the requester and the worker's answers to each question in separate columns. Furthermore, the date and time the worker accepted and submitted the HIT and difference between the two—the number of seconds the worker worked on the HIT—are also returned. If the amount of time the worker spent on the HIT is shorter than could possibly be achieved by a human, the requester may wish to reject this work.

Command-line tools The most basic tools for creating and managing HITs on your own computer (rather than a Web-based interface) are the command-line tools (CLTs) offered by Amazon's Mechanical Turk, which can be found at http://developer.amazonwebservices.com/connect/entry.jspa?externalID=694 and are available for Windows and Unix operating systems (including Mac OS X and Linux). The CLTs allow one to interact with the Mechanical Turk APIs with simple text files and text-based commands, providing a wrapper to the Java API interface that does not require an understanding of Java or the API.

The command line tools are a set of scripts that, when executed on the command line, interact with the Amazon API using existing files to populate the parameters for the commands executed. All of these scripts are located in the "bin" folder—as is the most important input file: mturk.properties. All of the CLTs look in this file to find the requester's Access Key and Secret Key necessary for signing the API calls (assigned when the requester account is created; see the Requesters section). The file also has the URL for the API, which can be changed to create the HITs on the requester Sandbox or the production Mechanical Turk site.

The majority of users of the CLTs will prefer to work off the examples. There is an example for every type of HIT available in the templates, as well as the external HIT, which populates the HIT with content from an external server. Each example folder contains all of the necessary files to create and load HITs of that type. Every one of these examples uses three key files: *yourtask*.properties, *yourtask*.question, and *yourtask*.input. It is important that the prefix *yourtask* is the same across all three files, but otherwise any valid file name can be used.

The first file, *yourtask*.properties, describes the parameters of the HIT (described in the HIT parameters section). The file is a standard form with variables for each of the parameters that can be freely modified. The second file, *yourtask*.question, is an XML-formatted file that defines what kind of HIT you want to create and what you want to put in it. The structure of this XML file depends on the type of HIT being created, which is why it is useful to work off the example file in the corresponding folder. The third file, *yourtask*.input, lets you set variables to be used in your HIT, which are used in the same way as in the templates.

Also in each folder are the scripts necessary to load the HITs onto Mechanical Turk, review the results, and approve and delete the HIT. The script to load the HITs is called run.sh or run.cmd (depending on whether you are using the Unix or the Windows CLTs) but needs to be modified before being executed. The key part of the file is the line that calls the loadHITs script, which looks for the three key input files. It is important that these files are correctly described and that the "label" flag is also *yourtask*, the same prefix as the input files. The two output files created by the scripts will use this label in the filename, and they are necessary for reviewing and approving the HITs.

After modifying the mturk.properties file, the run script, and the input files, the script is nearly ready to be executed to generate the HITs. All of the scripts depend on two environmental variables: MTURK_CMD_HOME, which is the path to the folder containing the scripts and the Java programs, and JAVA_HOME, which is the path to Java on the machine. Once these are appropriately set, run can be called.

This will post the HITs on Mechanical Turk and create a file, *yourtask*.success, that indicates that the HIT has been successfully created and stores a list of the HIT IDs that were created by the script. This file is the only place these HIT IDs are stored (other than Amazon's servers) and, therefore, are your only key to reviewing, approving, and deleting the HITs using the CLTs. It is good practice to create a backup of *yourtask*.success described by the creation date. When run is executed again, *yourtask*.success is overwritten, and any HITs not yet managed are now accessible only through the Web interface and can be managed only one at a time, which is extremely inconvenient if you created many HITs.

To download the results obtained from the Turkers, the getResults script must be called. As with run, the names of the files being called must be modified to *yourtask*, the prefix of the input and success files. Once called, the script generates a file called *yourtask*.results. This file contains the complete information about every submitted assignment from the HITs listed in *yourtask*.success, including the Worker ID, Assignment ID, and any fields included in the form submitted by the worker.

The results file, *yourtask*.results, is a column-separated file just like the results file you can download with the template HITs, making it easily accessible with spreadsheet programs. Additionally, if you are using the CLTs to create HITs in one of the "template" formats, you can run the generateResultsSummary script, which will analyze the results and create a tab-delimited file called *yourtask*.summary. This file will contain the majority answer for each question, the percentage of workers who selected this answer, and how many workers selected the answer out of the number who worked on the HIT.

The *yourtask*.results file also includes a column labeled "reject." If you are manually reviewing the workers' submissions, you would enter something in this column to indicate that assignment is to be rejected. This field is blank by default, meaning that all submitted work will be accepted unless this field is changed.

Once you have reviewed the work in *yourtask*.results, you must execute the script reviewResults. This rejects the assignments that have something in the "reject" field and approves the rest, thereby paying those workers from your account (as well as the 10% fee to Amazon). An alternative way of reviewing work is to split into separate files containing only the accepted or rejected assignments, and separately call approveWork and rejectWork (in the "bin" folder) with the flag "-approvefile" or "-rejectfile" for each.

Finally, once you are finished collecting data and reviewing the work, you can run the approveAndDeleteResults script. This will delete the HITs listed in the *yourtask*.success file and automatically approve any submitted assignments not yet reviewed. Both reviewResults and approveAndDeleteResults can be run multiple times (if, for instance, additional work is

completed after running the files) as long as the *yourtask.success* file exists. Running reviewResults a second time will print errors for assignments already reviewed but will otherwise ignore them and continue to work correctly for unreviewed assignments.

Programming interfaces

The most general and powerful way to interact with Mechanical Turk is programmatically through their APIs. Using the APIs, one can create HITs, create qualifications, approve and reject assignments, grant bonuses, send messages to the workers, and block or unblock specific workers from accepting your HITs. For an overview of the API and the general concepts involved, see http://docs.amazonwebservices.com/AWSMechTurk/latest/AWSMechanicalTurkRequester/. For the list of functions in the API, their descriptions, their inputs and their outputs see http://docs.amazonwebservices.com/AWSMturkAPI/2008-08-02/. In this section, we will describe the six basic operations (five of which are explicitly part of the API) necessary to create a HIT and pay the workers.[14]

All of the API calls have a few fields in common. For example they all have a field called Service which is set to "AWSMechanicalTurkRequester" which specifies which of the Amazon APIs this request is a part of. In addition, there is a field called Operation that specifies what method of the API is to be executed.

Each API call also requires the system time of the machine making the API call. The next two fields are designed to authenticate each API call to a specific requester. Each call requires the requester to pass in the AWS Access Key that was assigned when the requester registered for the account (described in the Requesters section). Furthermore, each call must be signed by computing a hash of the Service, Operation, and time fields with their AWS Secret Key. Since this is considered a secret, unique to each requester, it ensures that no one can falsely pose as a requester.

CreateHIT This function is a programmatic way of setting all the parameters of a HIT described in the HIT parameters section. When a HIT is created successfully, Amazon returns two identifiers. The first is the HIT ID. This can be used in other API calls to extend the life of the HIT, expire the HIT early, disable the HIT, as well as other possible operations on a HIT. The second identifier that is returned is the HIT Type ID (see the HIT parameters section).

[14] There are two main protocols by which one can make calls to the Mechanical Turk API, SOAP, and REST. While both protocols have their advantages and disadvantages, in this section we focus on making API calls using REST due to its simplicity.

SubmitAssignment Submitting an assignment is not technically part of the API, but when a worker finishes a task, this method must be invoked to let Mechanical Turk know that the assignment has been completed. Executing this method amounts to submitting an HTML form to the URL http://www.mturk.com/mturk/externalSubmit with the Assignment ID as a hidden POST variable. Alternatively, one can either use a GET variable or simply redirect the worker to a URL of the form http://www.mturk.com/mturk/externalSubmit/assignmentId=XXX&foo=bar. The perhaps unexpected "foo = bar" is there as a workaround to a bug in the Mechanical Turk API. Without some key–value pair in its place, the assignment will not be submitted.

ApproveAssignment, RejectAssignment The only required field in these API calls is the Assignment ID. An optional field is a string that is a message from the requester to the worker that provides some feedback for this assignment. When ApproveAssignment is executed, the money for the HIT will be transferred from the requester's account to the worker's account. Also, the 10% surcharge will be transferred to Amazon. When RejectAssignment is called, the assignment will be rejected, and no money will change hands.

GrantBonus This call takes as input the Worker ID, Assignment ID, Bonus Amount (in U.S. dollars), and a Reason. This command results in the bonus amount being transferred from the requester's account to the worker's account. This command can be executed only on assignments that have been submitted and approved.

NotifyWorkers This call can take a list of up to 100 Workers IDs, a subject line, and a message. The message (input to the method as a string) will be sent from the Amazon system to the e-mail address associated with each worker account with the specified subject line. If delivering this message to any one of the workers returns an error, the command returns an error, even if the rest of the notifications went through. Thus, we advocate that requesters put the Worker ID of an account they own in each batch of Worker IDs to ensure that the messages go through and were properly formatted.

References

Akerlof, G. A. (1970). The market for "lemons": Qualitative uncertainty and the market mechanism. *Quarterly Journal of Economics, 84,* 488–500.

Alonso, O., & Mizzaro, S. (2009). Can we get rid of TREC assessors? Using Mechanical Turk for relevance assessment. In S. Geva, J.

Kamps, C. Peters, T. Saka, A. Trotman, & E. Voorhees (Eds.), *Proceedings of the SIGIR 2009 Workshop on the Future of IR Evaluation* (pp. 15–16). Amsterdam: IR Publications.

Andrews, D., Nonnecke, B., & Preece, J. (2003). Electronic survey methodology: A case study in reaching hard-to-involve Internet users. *International Journal of Human–Computer Interaction, 16,* 185–210.

Barchard, K. A., & Williams, J. (2008). Practical advice for conducting ethical online experiments and questionnaires for United States psychologists. *Behavior Research Methods, 40,* 1111–1128.

Baron, J., & Hershey, J. (1988). Outcome bias in decision evaluation. *Journal of Personality and Social Psychology, 54,* 569–579.

Berk, R. A. (1983). An introduction to sample selection bias in sociological data. *American Sociological Review, 48,* 386–398.

Birnbaum, M. H. (Ed.). (2000). *Psychological experiments on the Internet*. San Diego, CA: Academic Press.

Birnbaum, M. H. (2004). Human research and data collection via the Internet. *Annual Review of Psychology, 55,* 803–832.

Buhrmester, M. D., Kwang, T., & Gosling, S. D. (in press). Amazon's Mechanical Turk: A new source of inexpensive, yet high-quality, data? *Perspectives on Psychological Science*.

Camerer, C. F., & Hogarth, R. M. (1999). The effects of financial incentives in experiments: A review and capital-labor-production framework. *Journal of Risk and Uncertainty, 19,* 7–42.

Centola, D. (2010). The spread of behavior in an online social network experiment. *Science, 329,* 1194.

Chilton, L. B., Horton, J. J., Miller, R. C., & Azenkot, S. (2010). *Task search in a human computation market. In Proceedings of the ACM SIGKDD Workshop on Human Computation* (pp. 1–9). New York: ACM.

Cooper, R., DeJong, D. V., Forsythe, R., & Ross, T. W. (1996). Cooperation without reputation: Experimental evidence from prisoner's dilemma games. *Games and Economic Behavior, 12,* 187–218.

Couper, M. P. (2000). Web surveys: A review of issues and approaches. *Public Opinion Quarterly, 64,* 464–494.

Couper, M. P., & Miller, P. V. (2008). *Web survey methods. Public Opinion Quarterly, 72,* 831.

Dixon, W. J. (1953). Processing data for outliers. *Biometrics, 9,* 74–89.

Eriksson, K., & Simpson, B. (2010). Emotional reactions to losing explain gender differences in entering a risky lottery. *Judgment and Decision Making, 5,* 159–163.

Fehr, E., & Gachter, S. (2000). Cooperation and punishment in public goods experiments. *American Economic Review, 90,* 980–994.

Felstiner, A. L. (2010). *Working the crowd: Employment and labor law in the crowdsourcing industry.* Retrieved from http://works.bepress.com/alek_felstiner/1/

Frick, A., Bächtiger, M.-T., & Reips, U.-D. (2001). Financial incentives, personal information, and drop out in online studies. In U.-D. Reips & M. Bosnjak (Eds.), *Dimensions of Internet science* (pp. 209–219). Lengerich: Pabst Science.

Göritz, A. S. (2006). Incentives in Web studies: Methodological issues and a review. *International Journal of Internet Science, 1,* 58–70.

Göritz, A. S. (2008). The long-term effect of material incentives on participation in online panels. *Field Methods, 20,* 211–225.

Göritz, A. S., & Stieger, S. (2008). The high-hurdle technique put to the test: Failure to find evidence that increasing loading times enhances data quality in Web-based studies. *Behavior Research Methods, 40,* 322–327.

Göritz, A. S., Wolff, H. G., & Goldstein, D. G. (2008). Individual payments as a longer-term incentive in online panels. *Behavior Research Methods, 40,* 1144–1149.

Gosling, S. D., & Johnson, J. A. (Eds.). (2010). *Advanced methods for conducting online behavioral research*. Washington, DC: American Psychological Association.

Heckman, J. J. (1979). Sample selection bias as a specification error. *Econometrica, 47,* 153–161.

Horton, J. J., Rand, D. G., & Zeckhauser, R. J. (in press). The online laboratory. *Experimental Economics*.

Hossain, T., & Morgan, J. (2006). Plus shipping and handling: Revenue (non)equivalence in field experiments on eBay. *Advances in Economic Analysis & Policy, 6,* 3.

Howe, J. (2006). The rise of crowdsourcing. *Wired Magazine, 14,* 1–4.

Huang, E., Zhang, H., Parkes, D. C., Gajos, K. Z., & Chen, Y. (2010). *Toward automatic task design: A progress report. In Proceedings of the ACM SIGKDD Workshop on Human Computation* (pp. 77–85). New York: ACM.

Ipeirotis, P. G. (2010a). Analyzing the Amazon Mechanical Turk marketplace. *ACM XRDS, 17,* 16–21.

Ipeirotis, P. G. (2010b). *Demographics of Mechanical Turk (Tech. Rep. No. CeDER-10-01)*. New York: New York University. Retrieved from http://hdl.handle.net/2451/29585. March.

Ipeirotis, P. G., Provost, F., & Wang, J. (2010). *Quality management on Amazon Mechanical Turk. In Proceedings of the ACM SIGKDD Workshop on Human Computation* (pp. 64–67). New York: ACM.

Kittur, A., Chi, E. H., & Suh, B. (2008). Crowdsourcing user studies with Mechanical Turk. In M. Czerwinski & A. Lund (Eds.), *Proceeding of the Twenty-Sixth Annual SIGCHI Conference on Human Factors in Computing Systems* (pp. 453–456). New York: ACM.

Kraut, R., Olson, J., Banaji, M., Bruckman, A., Cohen, J., & Couper, M. (2004). Psychological research online: Opportunities and challenges. *American Psychologist, 59,* 105–117.

Little, G., Chilton, L. B., Goldman, M., & Miller, R. C. (2010). *Exploring iterative and parallel human computation processes. In Proceedings of the ACM SIGKDD Workshop on Human Computation* (pp. 68–76). New York: ACM.

Mao, A., Parkes, D. C., Procaccia, A. D., & Zhang, H. (2011). Human computation and multiagent systems: An algorithmic perspective. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence*. San Francisco.

Marge, M. , Banerjee, S., & Rudnicky, A. I. (2010). Using the Amazon Mechanical Turk for transcription of spoken language. In J. Hansen (Ed.), *Proceedings of the 2010 IEEE Conference on Acoustics, Speech and Signal Processing* (pp. 5270–5273). IEEE.

Mason, W. A., & Watts, D. J. (2009). *Financial incentives and the performance of crowds. In Proceedings of the ACM SIGKDD Workshop on Human Computation* (pp. 77–85). New York: ACM.

McCreadie, R. M. C., Macdonald, C., & Ounis, I. (2010). Crowdsourcing a news query classification dataset. In M. Lease, V. Carvalho, & E. Yilmaz (Eds.), *Proceedings of the ACM SIGIR 2010 Workshop on Crowdsourcing for Search Evaluation (CSE 2010)* (pp. 31–38). Geneva, Switzerland. July 23.

Musch, J., & Klauer, K. C. (2002). Psychological experimenting on the World Wide Web: Investigating content effects in syllogistic reasoning. In M. B. B. Batinic & U.-D. Reips (Eds.), *Online social sciences* (pp. 181–212). Göttingen: Hogrefe.

Nosek, B. A. (2007). Implicit–explicit relations. *Current Directions in Psychological Science, 16,* 65–69.

Paolacci, G., Chandler, J., & Ipeirotis, P. G. (2010). Running experiments on Amazon Mechanical Turk. *Judgment and Decision Making, 5,* 411–419.

Pontin, J. (2007). Artificial intelligence, with help from the humans. *New York Times*. March.

Rand, D. G. (in press). The promise of Mechanical Turk: How online labor markets can help theorists run behavioral experiments. *Journal of Theoretical Biology.*

Reiley, D. (1999). Using field experiments to test equivalence between auction formats: Magic on the Internet. *American Economic Review, 89,* 1063–1080.

Reips, U. D. (2000). The Web experiment method: Advantages, disadvantages and solutions. In M. H. Birnbaum (Ed.), *Psychological experiments on the Internet* (pp. 89–114). San Diego: Academic Press.

Reips, U. D. (2001). The Web experimental psychology lab: Five years of data collection on the Internet. *Behavior Research Methods, Instruments, & Computers, 33,* 201–211.

Reips, U. D. (2002). Standards for Internet-based experimenting. *Experimental Psychology, 49,* 243–256.

Reips, U. D., & Birnbaum, M. H. (2011). Behavioral research and data collection via the internet. In R. W. Proctor & K.-P. L. Vu (Eds.), *The handbook of human factors in web design* (pp. 563–585). Mahwah: Erlbaum.

Ross, J., Irani, L., Silberman, M. S., Zaldivar, A., & Tomlinson, B. (2010). Who are the crowdworkers? Shifting demographics in Amazon Mechanical Turk. In K. Edwards & T. Rodden (Eds.), *Proceedings of the ACM Conference on Human Factors in Computing Systems* (pp. 2863–2872). New York: ACM.

Ryan, K. J., Brady, J., Cooke, R., Height, D., Jonsen, A., King, P., et al. (1979). *The Belmont report: Ethical principles and guidelines for the protection of human subjects of research*. Washington, DC: National Commission for the Protection of Human Subjects of Biomedical and Behavioral Research.

Salganik, M. J., Dodds, P. S., & Watts, D. J. (2006). Experimental study of inequality and unpredictability in an artificial cultural market. *Science, 311,* 854–856.

Schmidt, W. C. (2007). Technical considerations when implementing online research. In A. Joinson, K. McKenna, T. Postmes, & U.-D. Reips (Eds.), *The Oxford handbook of internet psychology* (pp. 461–472). Oxford: Oxford University Press.

Shariff, A. F., & Norenzayan, A. (2007). *God is watching you. Psychological Science, 18,* 803.

Sheng, V. S., Provost, F., & Ipeirotis, P. G. (2008). *Get another label? Improving data quality and data mining using multiple, noisy labelers. In Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 614–622). New York: ACM.

Smith, M., & Leigh, B. (1997). Virtual subjects: Using the Internet as an alternative source of subjects and research environment. *Behavior Research Methods, 29,* 496–505.

Snow, R., O'Connor, B., Jurafsky, D., & Ng, A. Y. (2008). Cheap and fast—but is it good? Evaluating non-expert annotations for natural language tasks. In M. Lapata & H. T. Ng (Eds.), *Proceedings of the Conference on Empirical Methods in Natural Language Processing* (pp. 254–263). New York: ACM.

Suri, S., & Watts, D. J. (2011). Cooperation and contagion in Web-based, networked public goods experiments. *PLoS One, 6*(3), e16836.

Tversky, A., & Kahneman, D. (1981). The framing of decisions and the psychology of choice. *Science, 211,* 453–458.

Tversky, A., & Kahneman, D. (1983). Extensional versus intuitive reasoning: The conjuction fallacy in probability judgement. *Psychological Review, 90,* 293–315.

Urbano, J., Morato, J., Marrero, M., & Martín, D. (2010). Crowdsourcing preference judgments for evaluation of music similarity tasks. In M. Lease, V. Carvalho, & E. Yilmaz (Eds.), *Proceedings of the ACM SIGIR 2010 Workshop on Crowdsourcing for Search Evaluation (CSE 2010)* (pp. 9–16). Geneva, Switzerland.

Voracek, M., Stieger, S., & Gindl, A. (2001). Online replication of evolutionary psychology evidence: Sex differences in sexual jealousy in imagined scenarios of mate's sexual versus emotional infidelity. In U.-D. Reips & M. Bosnjak (Eds.), *Dimensions of Internet science* (pp. 91–112). Lengerich: Pabst Science.

Zhu, D., & Carterette, B. (2010). An analysis of assessor behavior in crowdsourced preference judgments. In M. Lease, V. Carvalho, & E. Yilmaz (Eds.), *Proceedings of the ACM SIGIR 2010 Workshop on Crowdsourcing for Search Evaluation (CSE 2010)* (pp. 21–26). Geneva, Switzerland.