psiTurk: A framework for running behavioral experiments online

First Author · Second Author

Abstract Insert your abstract here.

1 Introduction

Online experiments are growing in popularity in the behavioral sciences. In comparison to traditional laboratory research, web-based experiments offer a number of benefits to investigators, including faster data collection and access to a more diverse pool of potential participants. At the same time, a new set of technical and experimental challenges must be met before a researcher can adopt the web-based approach.

psiTurk is a framework of software and web services that aims to address many of these challenges. It facilitates the creation, deployment, and sharing of web-based experiments that rely on Amazon Mechanical Turk (AMT), which is currently the most common platform to run human experiments online. psiTurk solves a number of technical requirements including serving an experiment on the web, saving experimental data to a database, and restricting the participant pool according to the experimenter's needs. It also provides an interface with AMT to create and test new projects as well as to manage and pay participants. Finally, in the spirit of "open science," it provides an *experiment exchange* for sharing experiments that can be easily replicated and extended by other researchers.

F. Author
first address
Tel.: +123-45-678910
Fax: +123-45-678910
E-mail: fauthor@example.com

S. Author
second address

The principal goal of this framework is thus to handle common technical challenges so that researchers can focus on the development and dissemination of online experiments. In this paper we present a high-level overview of the psiTurk framework and its advantages for running web-based experiments. We begin with an overview of online behavioral research, the AMT platform, and the specific challenges that psiTurk is designed to address. We then step through the core functionality of psiTurk and describe its role in an experimenter's workflow. Finally, we consider some future directions for psiTurk and online experimentation in general.

2 Web-based behavioral research

It is easy to see why online experiments are an attractive option for behavioral scientists. Compared to traditional methods of experimentation in the laboratory they allow researchers to collect large data sets in a fraction of the time and at much lower cost. At the same time, they also open up a completely new set of web-based tools that can be used to design the look and flow of an experiment and that far exceed the capabilities of traditional experiment software. They also cater to a wide range of needs from simple surveys to more complicated interactive interfaces, or even multi-player games.

Still, there exists a range of hurdles that researchers need to tackle when running experiments online, some of which might altogether bar some scientists from adopting online experimentation. To find out more the specific needs and challenges we conducted a survey in early 2014 of behavioral researchers regarding online experimentation.

2.1 Survey of researchers' needs

We collected responses from 201 researchers in psychology (58%), linguistics (16%), marketing (7%), neuroscience (6%), and economics (4%), most of whom had some experience collecting behavioral data online in their labs (85%).

These researchers showed a clear interest in online data collection. Nearly all listed fast data collection (98%) and large sample sizes (93%) as benefits of online data collection, with most interested in the potential for lower costs (75%) and a more diverse population (60%) as well. Nearly 40% stated that they treat papers reporting data collected online identically to that collected in a lab.

Respondents also felt that online data collected presented a set of unique challenges, as shown in Figure 1. Some felt that data collected online was unreliable (35%) or the population unrepresentative (25%). Many felt that experiment designs they were interested in did not work well online (50%), or that web technology is too complex (25%), and researchers based outside the US reported difficulty using services like Amazon Mechanical Turk (23%).

The majority of respondents (56%) listed Qualtrics, a service for conducting online surveys, as the tool they were currently most likely to use, but most (79%) expressed interest in being able to run full experiments including multiple, trials, fixation crosses, etc. The vast majority (94%) reported interest in new tools that simplified online data collection.

2.2 Amazon Mechanical Turk

Amazon Mechanical Turk (AMT) is an online platform that lets you post a wide variety of tasks to a large pool of participants. People who complete tasks, so called *HITs* (Human Intelligence Tasks), are called *workers*, people who post tasks are called *requesters*. Instead of spending weeks to run experiments in the lab, on AMT data from a large number of workers can be collected within a couple of hours. Beyond providing a platform to connect requesters and workers, AMT places few restrictions on the type of work that can be requested and the amount paid per HIT. Workers are paid a fixed amount for each HIT they complete which is determined by the requester.

From a worker's perspective, the AMT website provides an up-to-date listing of HITs that are available, with a title, brief description, and payment amount. A worker can click on a HIT to see its full *advertisement* (an embedded webpage that is either hosted by Amazon

or the requester) and decide to accept the HIT. Accepting a HIT reserves the ability to complete it within a timeframe specified by the requester. The HIT is then be completed either on a requester's own hosted service (i.e., a webpage external to the AMT site), or on the AMT website itself through the use of customized templates. Upon completion, the worker "submits" the HIT, after which point the completed work must be approved by the requester before they are compensated. In addition, some HITs offer bonus payments that are determined by the requester based on a worker's performance.

From a requester's perspective, AMT provides a web-based interface for creating/managing HITs and approving work that has been submitted. In addition, they offer a "sandbox" version of the platform in which HITs can be tested without paying live participants. For basic tasks like surveys and open text fields, AMT offers a range of built-in templates that can be used to create experiments within the AMT website. However, the degree of customization is limited for these templates. In particular, they do not support advanced experimental logic that is common to many psychology experiments (e.g., adapting the presentation of stimuli based on performance, or counterbalancing across a number of experimental variables). In addition, they are limited to relatively simple forms of interaction and stimulus presentation.

For more complex tasks, a requester can provide a link to their own hosted service which runs the experiment, saves the data, and only interfaces with AMT at the beginning of the task (receiving basic information about the worker) and at the end of the task (sending confirmation that the task has been completed). Thus, these "external HITs" enable a much wider range of experiments at the cost of additional technical requirements. psiTurk is designed to handle many of these requirements so that researchers can efficiently run experiments that are hosted externally.

2.3 What problems does psiTurk solve?

The psiTurk platform is comprised of four key services which address the needs of behavioral researchers, and in general mitigates the technical complexity that was cited by survey participants as a key hurdle to online data collection.

2.3.1 Interface to AMT

psiTurk provides an easy to use interactive command line tool that can be used to interact with AMT without ever having to use the AMT website. The command
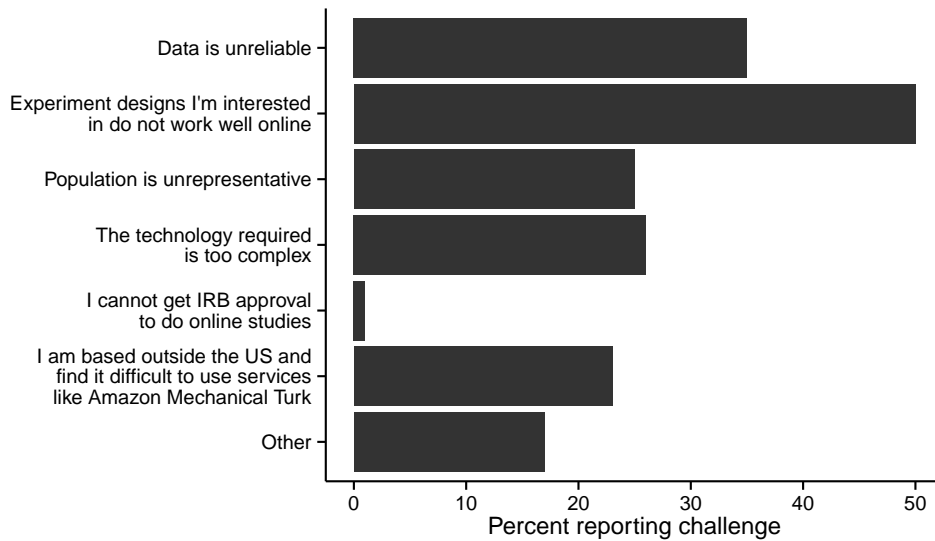
Fig. 1 Challenges faced by 201 researchers who completed our survey on collecting behavioral data online.

line tool can be used to create and manage HITs, both live and in the AMT sandbox (AMT's testing environments for HITs before they go live), and to pay workers for participating. Besides the basic payment that every worker receives for completing a HIT it also has the capability to give automated bonus payments to participants based on a function designed by the experimenter. This feature is particularly helpful since the AMT website requires bonuses to be entered by hand for each participant individually.

2.3.2 Technical complexity and security

psiTurk reduces some of the main technical challenges that researchers face by providing a web server to host an experiment (built using the Flask web framework, http://flask.pocoo.org) and basic database to store participants' data. It also solves one very specific problem by providing a secure ad server that can be used to serve the advertisements that workers see on the AMT website. Without a secure server ads would not display on AMT and thus the ad server acts as a bridge between AMT and a researcher's local computer. While providing a basic server and database, psiTurk also allows researchers to add custom server routes or alternative databases that can be tailored to their specific needs.

2.3.3 Coding the Experiment

In general, the psiTurk framework is *not* designed as a programming interface for building the actual experiment. That is, it requires that the experimenter knows some basic JavasScipt, HTML, and CSS to code the

logic and looks of a new study. Having said that, psiTurk does include its own JavaScript library, which provides helpful functions for recording data and sending it back to the server, preloading pages and images, as well as some basic functions for looping through instruction screens. These features should alleviate some of the most painful aspects of coding your own experiment and allow the researcher to focus on the task logic and design instead. psiTurk also ships with a complete working example task that can be used as a template for new projects.

2.3.4 Recruiting, filtering, and instructing participants

Finally, psiTurk has a range of features of particular interest to behavioral researchers running controlled experiments. For example, it offers basic capabilities for assigning workers to different experimental conditions and counterbalancing features of the experiment. It also provides a range of templates for common experiment building blocks (e.g., consent form, debriefing, instructions). In addition, it allows experimenters to filter workers by: excluding certain devices, or browsers; restricting the geographic location based on their Amazon account or by their IP address; or preventing a worker from participating in the same experiment more than once.

2.4 Remaining challenges

Some of the concerns raised by survey respondents are inherent to online data collection or the AMT platform

and therefore psiTurk does not address them. It is worth pointing out, however, that the common concern about data reliability may be somewhat exaggerated based on recent studies that have successfully replicated a wide range of classic cognitive psychology findings using AMT data (Crump et al, 2013). Interestingly, this study also found that increasing worker payment had no effect on reliability, suggesting that even at low payment levels data quality was high.

On the other hand, there clearly exist experimental protocols that simply are not amenable to online experimentation. Experiments that require very fine-grained temporal control over stimulus presentation, for example, may be unsuited because browsers will not be able to reliably display content fast enough if the presentation time becomes too short (Crump et al, 2013). Similarly, any experiment that requires control over a participant's screen size, resolution, or distance from the screen will be problematic due to the nature of the web browsing environment. One advantage of psiTurk is that it automatically collects data on worker's interaction with the experiment window, that is, if and when the window was resized and if and when the user switched tabs or windows. This kind of data can be used to evaluate a worker's level of engagement while completing the task.

We will continue to discuss these challenges at the end of the paper.

3 Overview of the psiTurk system

The structure of the psiTurk system is shown in Figure X.

In this section we highlight important functions of each component. Complete documentation and tutorials are available at http://psiturk.org/docs, including instructions for installing the software.

3.1 Configuration and project structure

After creating an Amazon Web Services account, a researcher must obtain credentials (a *key ID* and a *secret key*) that are then added to a global configuration file `.psiturkconfig`. When psiTurk runs for the first time this file is automatically created in the user's home directory, but the user's AWS credentials must be added in order for psiTurk to interact with AWS. Additionally, an account (and associated pair of keys) on the psiTurk.org website is required to allow posting of ads on the Ad server. These credentials (obtained from a user's psiTurk profile page) are also placed in `.psiturkconfig`.

This global configuration is used whenever psiTurk is run, allowing multiple experiments (each residing in its own directory somewhere on the user's system) to rely on the same AWS and psiTurk accounts, or to share other common configuration options as desired.

A new psiturk project can be initialized in the current directory using:

```
$ psiturk-setup-example
```

which creates a new directory containing an example experiment. The project directory typically includes the following:

`config.txt`: This file contains settings for the current experiment, including:

- Metadata about the experiment that is displayed to workers on the AMT website (title, description)
- Restrictions on which workers can accept the HIT, including geographic location (e.g., limiting to US workers only), AMT-specific qualifications (e.g., a worker's proportion of past work that has been approved), and web browsers that are permitted.
- Database setup (Sqlite by default)
- Server parameters (URL, port number)
- Task parameters (e.g., number of conditions)

`custom.py`: Within Flask web framework, possible to add custom routes so that server-side functionality can be added to the psiTurk web server.

`participants.db`: If Sqlite is used (on by default), this file will be the database.

`server.log`: A log file containing any messages from the experiment server (not from the actual experiment code!)

`static/` directory: Experiment files, including Javascript, CSS, and images.

`templates/` directory: HTML files associated with experiment

Setting up a new experiment thus entails 1) editing the settings in `config.txt`, and 2) modifying the contents of the `static` and `templates` directories to program the experiment. The remaining files are only necessary in order to debug server errors (`server.log`), add new server-side functionality (`custom.py`), or to access saved data (`participants.db`).

Creating the experiment requires at least some basic web programming skills (especially using HTML, CSS,

and JavaScript). Once you mastered the basics, you can take advantage of the vast number of libraries and tools that can help you to build sharp and sophisticated experiments with the support of a large community of users.

3.2 Command line interface: Managing HITs and serving the experiment

PsiTurk runs as a command line interface (CLI) within a standard terminal window. There are several advantages to designing psiTurk as a simple CLI beyond its efficiency of use. This design makes the user-interface code clear and easy to read and write, allowing newcomers to quickly understand and contribute to the open-source project. Integrating a new feature into the interface is as simple as describing the syntax and functioning of a new command. The CLI also ensures that psiTurk is easy to interact with not just on a laptop or desktop but also on a remote server or in the cloud, where users may have terminal-only access.

Tab completion makes it quick to type commands, and the `help` command can be used to show details on the usage of any command. The straightforward and consistent syntax of the CLI allows users to perform a wide variety of tasks–from creating HITs and paying workers, to launching Amazon Web Services database instances, to opening an experiment in a browser for debugging–that would otherwise be spread across a number of websites and programs.

Entering `psiturk` at the command line in any directory containing a psiTurk project launches the psiTurk CLI. The psiTurk CLI features a colorized prompt that provides important information at a glance, including whether the server is running, the current mode ("live" or "sandbox", and the number of HITs currently running on AMT:

```
[psiTurk server:off mode:sdbx #HITs:0]$
```

In some examples below, this prompt is truncated as `[psiTurk]$` to indicate commands that occur within the psiTurk CLI, while commands that are entered on the command line are preceded simply by `$`.

3.2.1 Managing HITs

Commands are organized into groups based on their function, following a general "`command subcommand arguments`" format. For example, one can create a HIT by typing

```
[psiTurk]$ hit create <# assignments> <$
  amount> <duration>
```

What happens after you create a HIT?
List any existing HITs with:

```
[psiTurk]$ hit list --active
```

3.2.2 Serving an experiment

The experiment server is started using

```
[psiTurk]$ server on
```

This opens the port specified in the project configuration file. The experiment server will then respond to incoming requests, assuming that the port is publicly accessible. This requires a static IP to prevent the experiment's URL from changing. Users without a static IP address can use a dynamic DNS service to forward requests to their dynamic IP. If the system running psiTurk is behind a router, the router must be configured to forward requests on the same port.

During the development of an experiment, the current code can be tested using:

```
[psiTurk]$ debug
```

which will open a new browser window in which the current experiment can be tested.

Can restart the server with `server restart` or open the server log with `server log`.

The sandbox mode is active by default, which means that calling `hit create` will submit a HIT to the AMT sandbox website. The experimenter can then test the experiment from the perspective of a worker before making it publicly available. Once the experiment is ready for real workers, the `mode` command can be used to switch to "live" mode, after which newly created HITs will be submitted to the main AMT site.

In most cases, a user of the psiTurk CLI will never have to log into the MTurk website except to add money to their MTurk requester account.

3.3 The Ad server

PsiTurk provides a cloud service at `psiturk.org` for serving ads to workers. This service serves the ads with appropriate SSL certificates, something which can be

difficult on many server configurations.[1] The Ad Server removes need to deal with complex web security issues (https, data mining on workers) Participants recruited via Mechanical Turk first interact with your task via ads. Ads are simply the digital version of hanging a poster or flyer around your university building in order to recruit participants. Technically, ads are snippets of HTML code that describe what your task is about and what you're offering for compensation. As a result, they are the front line for any subject recruitment online. It's easy to overlook the importance of a good ad, and making that ad visible to as many participants as possible.

3.4 Javascript library *psiTurk.js*

The javascript library *psiTurk.js* enables interaction with the server from the client-side (javascript) experiment code. The goal of the library is to handle the most common functionality of psiTurk-based web experiments, without imposing any additional requirements on the structure or design of the experiments themselves. Researchers can draw on a vast array of javascript libraries to design their experiment, while using psiTurk.js to save data and notify the server of changes in a participant's status.

3.4.1 Initialization

Figure X shows javascript code from an experiment script (`static/js/task.js`). Initialization. Preloading of HTML and images. Basic structure for presenting instructional pages.

3.4.2 Tracking a participant's progress in an experiment

psiTurk records changes in a participant's status as they move through an experiment. Some of these status changes are automatic, e.g., when a participant is assigned to a condition or if they quit an experiment early. Two additional status changes are initiated by the client-side experiment code using psiTurk.js functions. First, since experiments typically begin with an instructional phase, calling the function `psiturk.finishInstructions()` upon completion of this phase will save the participant's status as having begun the main experiment. If the participant attempts to quit the experiment after that point, they will receive a warning that they will not be able to restart the

experiment and may forgo payment. Second, successful completion of the experiment is signaled by calling `psiturk.completeHIT()`, which closes the experiment and redirects the participant to the AMT page to submit their HIT.

3.4.3 Saving experimental data

Experiment code can save data in two formats. `psiturk.recordTrialData` takes any list of values as input and appends it to a list of "trial" data. This data structure is meant for sequential data that may be collected over the course of multiple trials or blocks, where each line corresponds to a new measurement. However, the format of this data is defined by the experimenter and is saved in the database as a single JSON object.

In contrast, `psiturk.recordUnstructuredData` is used to record (key, value) pairs, where the key is uniquely defined within the experiment. This format is useful for survey questions or other one-time measurements, e.g., (*Age*: 24).

Importantly, both functions above simply record the data in the appropriate format on the client-side. When the experimenter wishes to save the data to the server they call `psiturk.saveData()`, at which point both sets of data are sent to the server to be recorded in the database.

3.4.4 Automatic recording of browser interaction

One general shortcoming of web experiments is greater uncertainty about a participant's testing environment and engagement with the experiment. Unlike lab computers where most undesired behaviors can be prevented, a web participant is always able to close a web browser, switch to different applications, or change other aspects of their experience. However, standard methods exist for recording many aspects of a user's interaction with a web browser, and this data can be useful for 1) tracking how an experiment was actually displayed, and 2) the level of a participant's engagement.

For example, although it is possible to set the initial size of a browser window, a web participant can change the dimensions of the window, potentially obscuring or altering how the experiment is displayed. *psiTurk.js* automatically records these changes in the size of the window. Similarly, the experiment can choose to switch focus away from the experiment window (e.g., to another browser window or a different application). *psiTurk.js* automatically records every time that the experiment window loses and gains the participant's focus. This "event data" is automatically recorded and saved to

---

[1] SSL certification has recently become a requirement for hosting content within an `iframe` on `mturk.com`.

Listing 1 Example experiment code (javascript)

```
1   // Initalize psiturk object with parameters passed from server (see templates/exp.html)
2   var psiTurk = new PsiTurk(uniqueId, adServerLoc, mode);
3
4   var mycondition = condition; // these two variables are passed by the psiturk server process
5   var mycounterbalance = counterbalance; // they tell you which condition you have been assigned to
6
7   // All HTML snippets to be loaded from templates directory
8   var pages = ["instructions/instruct-1.html", ...];
9   psiTurk.preloadPages(pages);
10
11  var StroopExperiment = function() {
12          ...
13          var response_handler = function(e) {
14                  ...
15                  if (response.length>0) {
16                          listening = false;
17                          var hit = response == stim[1];
18                          var rt = new Date().getTime() - wordon;
19
20                          psiTurk.recordTrialData({'phase':"TEST",
21                                          'word':stim[0],
22                                          'color':stim[1],
23                                          'relation':stim[2],
24                                          'response':response,
25                                          'hit':hit,
26                                          'rt':rt}
27                                      );
28                          remove_word();
29                          next();
30                  }
31          };
32          ...
33          // Load the stage.html snippet into the body of the page
34          psiTurk.showPage('stage.html');
35          ...
36  };
37
38  var Questionnaire = function() {
39          ...
40          record_responses = function() {
41                  psiTurk.recordTrialData({'phase':'postquestionnaire', 'status':'submit'});
42                  $('textarea').each( function(i, val) {
43                          psiTurk.recordUnstructuredData(this.id, this.value);
44                  });
45                  $('select').each( function(i, val) {
46                          psiTurk.recordUnstructuredData(this.id, this.value);
47                  });
48          };
49          ...
50          // Load the questionnaire snippet
51          psiTurk.showPage('postquestionnaire.html');
52          psiTurk.recordTrialData({'phase':'postquestionnaire', 'status':'begin'});
53
54          $("#next").click(function () {
55              record_responses();
56              psiTurk.saveData({
57                  success: function(){
58                          psiTurk.computeBonus('compute_bonus', function() {
59                              psiTurk.completeHIT(); // when finished computing bonus, quit
60                  });
61              },
62              error: prompt_resubmit});
63          });
64  };
```

the database whenever `psiturk.saveData` is invoked.

3.5 Experiment exchange

A significant advantage of web-based experiments is the potential for low-friction replication and extension. The *experiment exchange* (http://psiturk.org/ee) facilitates the sharing of experiments that have been built to run using psiTurk, acting as an "app store" for psiTurk-compatible designs. Once an experiment has been completed, a researcher can submit the following information to register their experiment on the exchange:

1. A Github repository containing the project code
2. Metadata about the experiment, including a title, description, and keywords
3. The DOI for the paper describing the results of the experiment (if any)
4. The version of psiTurk that was used to run the experiment

Metadata associated with an experiment allows other researchers to discover it through the psiTurk website. A publicly available experiment can then be downloaded using the following command on the command line,

```
$ psiturk-install <experiment-specific-id>
```

with the experiment-specific identifier found on the exchange page. If psiTurk is already configured on the user's system, the downloaded experiment will run without further changes, and can then act as a starting point for direct replication or extensions. Modified versions can then be re-run using the same population via AMT, thereby minimizing the potential for experimenter-specific biases.

Other initiatives (e.g., the Open Science Framework, https://osf.io) also aim to improve the transparency, reproducibility, and efficiency of research through centralized services, but are less focused on the specific technology involved in running online experiments. In contrast, the psiTurk experiment exchange links together experiments that can be run within a common framework. As a result, existing experiments can serve as examples to help new researchers learn to code for the web; they shorten the development time necessary (especially for popular experimental paradigms); and the exchange facilitates communication between researchers with similar interests.

4 Limitations and future directions (TMG)

As mentioned above, psiTurk faces some challenges inherent to all online data collection and may not be the right choice for researchers that require a very controlled experimental setup. But what are the limitations of psiTurk specifically?

One restriction is that it's currently designed to only work with Amazon Mechanical Turk. A common worry with AMT is the unrepresentativeness of its subject pool (in terms of demographics and geography), as well as the fact that certain experimental protocols are now too well-known to workers to guarantee naive participants (Chandler et al, 2014). These concerns are often not relevant for basic psychology research, but some experimenters may nevertheless look for alternatives to AMT. Many psiTurk features (i.e. the web server, data base, and psiturk.js) are general enough to facilitate running experiments on other platforms, too, but it would require some customization of certain AMT-specific functionality. Researchers who decide to use psiTurk on another platform could contribute such changes to the github repository (https://github.com/NYUCCL/psiTurk) to make them available to the community.

Another limitation is that psiTurk only runs on UNIX systems, like Linux and Mac OS, and can thus not be used on Windows computers. To meet this challenge, one option for Windows users is to use a cloud-based computing service to run psiTurk and host experiments. There exist multiple free or low-cost options, such as Amazon Web Services (http://aws.amazon.com), or OpenShift (https://www.openshift.com). The installation steps and setup may diverge slightly from the standard procedure but the psiTurk documentation already contains additional installation instructions for OpenShift. In the future, we hope to extend support and documentation to a wider range of cloud computing options.

Another future direction is to extend the counterbalancing capabilities of psiTurk. Currently, the built-in counterbalancing algorithm simply aims to assign participants equally to different cells. However, this equal assignment method can be problematic if, for instance, one group has a higher drop-out rate than the others (a problem rarely encountered in the lab). In that case the high drop-out group will need to receive more participants to equalize the difference and thus assignment probabilities will no longer be equal across groups. We are currently working on alternative counterbalancing algorithms that keep assignment probabilities equal (which may then lead to unequal number of participants) to avoid this problem. <mark>Not quite sure what the</mark>

current status is here. May need updating and maybe mention PlanOut if that's still relevant.

One exciting prospect of running online experiments is to let multiple participants interact to study coordination or group behavior. Currently psiTurk does not offer its own tool to facilitate such multi-person experiments, but we would like to add support for it in the future, using web-sockets or other protocols that enable communication between users.

Some small features not mentioned yet:

Automate catch trials? How far did the worker get? Do we count them in the N?

Limiting/throttling number of people doing the experiment at once

Video support

References

Chandler J, Mueller P, Paolacci G (2014) Nonnaïveté among amazon mechanical turk workers: Consequences and solutions for behavioral researchers. Behavior research methods 46(1):112–130

Crump MJ, McDonnell JV, Gureckis TM (2013) Evaluating amazon's mechanical turk as a tool for experimental behavioral research. PloS one 8(3):e57,410