

Achiel's guide to (hacky) online experiments via Mturk

(Version 1)

Below you will find my step-by-step guide to walk you through the process of collecting data via online experiments. As a disclaimer: everything below is based on my personal experience and some hacky tricks I've cobbled together over the years. I cannot guarantee that there are easier/better/more efficient methods out there. Your mileage may vary.

A quick note on terminology: a HIT is short for Human Intelligence Task, the vernacular for "task" or "job" on Mturk. I will use "HIT" and "experiment" interchangeably throughout this guide. The same goes for the terms "worker" and "subject". Finally, on the Mturk platform you publish "batches" of HITs. A batch contains multiple individual worker's responses. Any one experiment can have multiple batches, as many as you see fit.

Step 0: Programming skills

To start off with the bad news: if you are reading this and have absolutely zero experience in programming, then you're definitely at a bit of a disadvantage. This is not to say that running online experiments is impossible without such an experience. In fact, if your experiment is simple enough to only require some basic randomization and questionnaire items then you will be just fine. However, anything more complex and your life would be a whole lot easier with some very basic programming skills.

Now for the good news: by no means will you need to be an expert coder. I have no formal training in programming apart from some first-year courses years' back, and my code shows it: it's hacky, inefficient, only works in a very narrow range of situations and is unfortunately more than a little buggy at times. However, this is absolutely not a deal-breaker for most experiments! As long as you know what to google (and copy the resulting code from Stack Overflow) and do enough bug-testing before running the experiment then you can scrape by with a very minimal knowledge of JavaScript/HTML.

So what's the best way to develop these skills? From personal experience, a website like code-academy¹ is a great way to learn the very basics. Learning these basics will probably cost you a few weeks to get started on. So if for some reason you're locked into your apartment for the foreseeable future, then this may be a good way to kill some time. It's a bit of an up-front investment in terms of time, but it'll make running online experiments a whole lot easier for every step of the process. After learning these basics, you'll hopefully understand enough of JavaScript/HTML to apply the single most powerful programming trick out there: googling your question, copying the code from Stack Overflow and search for additional comments to figure out why the example code isn't working the way you expected it to. At that point, it's just a matter of time and practice before you'll be an expert.

Some links to Code Academy to help you get started:

<https://www.codecademy.com/learn/learn-html>

¹ Important note: I learned most of my basics via code-academy years ago. Although it was great at the time, I have no idea how the platform developed in recent years. So no guarantees there, but it'll probably be the best place to check out if you want to get started.

<https://www.codecademy.com/learn/learn-css>

<https://www.codecademy.com/learn/introduction-to-javascript>

If you decide to dive into some programming, one additional tip: using an Integrated Development Environment (IDE) is like night and day. These are software programs that organize the HTML, CSS and JS files in your experiment, provide code completion, checks for a whole range of bugs, etc. Really, anything beyond a 3-line project should not be done outside of an IDE. Although there are many options available out there, my personal favorite is [Webstorm](#). If you have a university address (student addresses are also valid), you can get the educational license for free!

Step 1: Setting up an MTurk Account

There are two types of Mturk accounts: workers and requesters. If you want to conduct online experiments, you'll need the latter one. Creating a requester account is incredibly easy. First, go to <https://requester.mturk.com/> and click on "create an account" (small font, top-right of the screen). Amazon will then ask you for a name, email address and password. You will then be asked to fill in form containing your contact info, including a company name etc. Here, I just filled in the university's address, but overall I don't think this really does anything anyway. That's it, you now have a Requester account from which you can host experiments.

TIP: expect to get at least some emails from workers if your experiments are bug-free. Expected to get a lot of email if there's any bugs in your code. Expect to get a lot of angry emails if you are slow on payment, use some random device to determine subject's payoff or have any bugs influencing subject's payments. For these reasons, you may want to consider using a dedicated email address.

In order to hire subjects to perform HITs you will need to pre-load your account with funds. Go to "My Account" (again, top-right) and upload money from a credit card there. This money will directly go into your account. Any expenses (both the fixed fee as well as individual bonuses, more on that below) will be paid from this account. As far as I know it's a bit of a hassle to retrieve money from of this account after your experiment finished, so depending on your specific case you may want to be a bit prudent in the amount of funds you upload at a time.

Step 2: Design your experiment for an online subject pool

Now that you have an account ready and some funds pre-loaded, you can start designing your experiment. Below I've listed several different strategies you can use to collect data from online subjects. However, before I'll get into the details there's an important note to be made about subjects: there are some fundamental differences between online and offline subjects pools. In the last few months I've heard a lot of people worry that most Mturk subjects are bots. I haven't kept up with this debate in the last few months and do not have the expertise to make a truly qualified statement on whether or not this is true. Nonetheless, I have always been somewhat skeptical of these concerns. Personally (again, take this with a bit of salt), I feel that Amazon's bot-detection software will most likely remove all or a large portion of automated responders. From personal experience I feel like a lot of the reported low-quality responses on Mturk result from researchers not properly accounting for the difference in subject pools. Speaking very broadly, there are three main differences between the Mturk sample and a college graduate student sample:

Subjects are extrinsically motivated. In contrast to college grads, Mturk subjects are semi-professional experiment-takers. Mturkers are not participating in experiments because of their innate interest in science, nor because they need the college credit points. A worker completes your task for one reason only: because you pay them. I'll leave it up to you to decide whether that's a feature or a bug (personally, I think it makes experiments much more realistic), but you will have to account for this when designing your experiment. If your experiment is in the form of a questionnaire survey with a fixed payoff, you effectively incentivize subjects to complete the task as quickly as possible (less time spend on your task means more time spend on other tasks, hence a higher hourly payoff). Unsurprisingly, you will end up with low-quality data if you take this approach. Instead, you will have to incentivize your subjects to take the time and pay attention to your experiment.

First, be sure to pay your subjects a sufficient amount (personally I try to ball-park at least around \$6-8/hour average, more for very short experiments). If in any way possible, set up your experiment in such a way that they will earn more for a better task performance. Also be sure to that this incentive is sufficiently powerful: if my options would be to either earn \$4.26 for a sloppy 5-minute rush job or \$4.36 for a 30-minute intense workload, then you may not be happy with your data quality. Finally, make sure that the worker's incentive is at least partially based on their decisions and/or performance, not just on a random draw. Subjects will find out if your payment is randomly allocated because...

Subjects communicate. You can kindly ask grad students not to discuss the contents of your experiment with others. A large fraction of them will even listen. In contrast, it is guaranteed that a large fraction of your subjects will discuss and review your task. If you don't believe me, just check out <https://turkerview.com/requesters/>. If your experiment uses deception or relies on random variables for payment, subjects will review your task badly. Design your experiment with this in mind: assume that all your subjects will be informed about your experiment before you've even started your instructions. Where possible, communicate all the important (i.e. payoff-relevant) elements of your experiment clearly to your subjects. Speaking of which: ...

Subjects will have worse English skills. This should be obvious. College grad students will be much more familiar with academic jargon and will be easily capable of understanding your instructions even if you may not have done the best job of explaining your setup to them. You cannot assume this for an online population. Simplify all instructions to the point that a 3-year old can understand them. Then simplify them some more. Terms which are obvious to you will be completely unfamiliar to a large proportion of your subjects. Not because they're dumb, but because they're not in academia.

How to mitigate these issues

How you deal with these three issues depends entirely on your design. Below are a number of solutions I've found to work for my experiments, feel free to use them as you see fit:

Include performance-incentives and attention checks. As outlined above, you can greatly improve the data quality of your subjects by setting their payment proportionally to their performance, or having it depend on the decisions your subjects make during your tasks. You can do so by setting a low baseline payment with a large variable (“bonus”) payment. See step 4 for details on how to implement this. However, incentives alone are not sufficient. Trust but verify. Be sure to include a few *attention checks*: questions that will help you separate high-quality from low-quality data after you have completed your experiment. Personally, I’ve had some good experiences with using questions that doubled as manipulation checks and attention checks (i.e. prompts which asked the subject about their current within-subject treatment).

Include an introduction screen. Since subjects will communicate with each other, you might as well be up-front with them. Personally, I always include a basic introduction screen at the start of the experiment. This instruction screen includes some basic information about my research institute, a summary of the experiment and details on how payment will be calculated. See below for a template. Including this section may also be a requirement from the ERB, so it’s a safe bet to include it anyway!

In this HIT you will be a participant in an experiment conducted on behalf of [INSTITUTION], an academic institution located in [COUNTRY].

This HIT is expected to last [TIME]. For your participation, you will earn a fixed fee of [BASE PAYMENT] and have the opportunity to earn a considerable bonus (on average around [AVERAGE BONUS])

All the answers that you provide in this HIT are completely anonymous.

[ONE OR TWO SENTENCE OUTLINE OF THE EXPERIMENT]

Throughout the HIT you can earn Tokens based on your decisions.

At the end of this experiment, these Tokens will be automatically exchanged to USD and paid out to you as a bonus (expect payment within 48-72 after submitting this HIT).

We will use the following exchange rate: [EXCHANGE RATE]

If you accept the conditions outlined above, please press the button labeled “ACCEPT” below to start the HIT.

You will then be taken to the instructions to the first part of the HIT.

Set the proper selection criteria. Finally, a-priori prevention is better (and cheaper!) than ad-hoc exclusion of subjects. By setting the proper selection-criteria you can prevent a lot of data-quality issues down the line. In previous experiments setting the right worker criteria reduced the percentage of subjects who failed an excessive number of attention checks from about 40% to about 5%. Below I'll discuss the exact steps you'll need to take to implement these criteria into Mturk. But first: what criteria should you want to use? While there are a number of different options available on Mturk, the ones you will definitely want to include are "HIT approval rate > [95-97]%" and "Humber of HITs approved > [400-500]". Why these two? That's because Mturk has a build-in reputation system. If workers submit a particularly low-quality effort, you can reject their work. This has two effects on the worker: 1) they do not receive payment for that HIT and 2) their approval rating drops. This last effect is particularly unfortunate for the worker, as a large number of other requesters (including most of the well-paying ones) require some minimum approval percentage. So by selecting workers who have a high rating AND a large number of approved HITs, you can partially rely on this reputation mechanism to provide an additional incentive for the worker to invest an appropriate amount of attention into your experiment. Note that this reputation incentive alone is not an adequate replacement for performance-based incentives though!

As an additional step you can set a location-based requirement. I've previously used this to ensure that the worker speaks a decent amount of English by restricting the pool to US-only workers. While this appeared to work it did limit both the diversity of the subject pool, so take this into consideration when designing your data collection.

Step 3: setting the parameters of your experiment

Now you can start setting up your experiment within Mturk's environment. To get something out of the way: all the steps in the rest of this guide make a lot more sense if you realize that Mturk was never meant to host online behavioral experiments. The original intent of Mturk was to crowdsource "human intelligence tasks"; i.e. small second-long tasks which cannot feasibly be automated by computers. Most notably to annotate data used for training machine-learning algorithms. This is why the Mturk environment is shaped the way it is: it expects a worker to receive a very short and easy question, fill out a short answer on an input field, submit the work and receive a payment in the line of a few cents. As a result, we have to get a little hacky at times to run much more complicated online experiments via this platform.

To take things one step at a time: first you will have to create a new HIT and define its parameters. On the task-bar, next to the Amazon logo on the left, click on "Create", then select "New Project". This will bring you to a screen containing a wide selection of templates. Exactly which template works best for you depends on our exact setup, which I will discuss in much more detail below. For now it doesn't matter, as the next screen is always the same. Click on "Create Project".

You will next be asked for the general parameters of your experiment: what will the reward be, how many workers you want to recruit, what workers you want to restrict, etc. Most of these boxes are pretty self-explanatory, but there's a few tips/comments I want to point out.

Description: I've noticed faster completion rates if the description contains a line outlining the average bonus amount. After all, workers only see the base payment in their list of available jobs (which will be relatively low, as I usually set most of my payments to be variable based on decisions/performance).

Reward per response: this is the BASE reward. Any bonus amounts will have be paid to subjects individually (discussed in more details below).

Number of respondents: quite straightforward, this is the sample size **per batch**. Note that Amazon charges a 20% fee for HITs with $n \leq 9$ and 40% for HITs with $n > 9$. This only applies to the base reward. Bonuses are always subject to a 20% fee.

Time allotted per Worker: this is the maximum amount of time a worker has to complete your experiment. The timer starts running the moment workers accept the hit. After this timer runs out, workers will not be able to submit their work. It's important to note that workers typically cue a number of experiments at a time and/or may take breaks during or in-between HITs. As a result, you'll want to be **very** generous with the time allotted to your HIT. Overall there's very little cost to increasing the wait time (your entire data collection may take a bit longer, but it will still be incredibly fast compared to lab experiments – what's a couple hours extra wait time anyway?) and you will get very angry emails if subjects do time out. As a result you may want to allot a time several times longer than the expected duration of your experiment.

Survey Expires in: this only applies if subjects are unwilling or unable to complete your experiment in sufficient numbers. As before, err on the generous side here.

Auto-approve and pay Workers in: this is the amount of time you have to approve subject's work and assign bonuses (more on this below) before the workers are automatically accepted. For simple logistical reasons, you'll probably want to give yourself a few days to do so. However: long wait times will scare workers off, as they don't like waiting for their earnings (would you?)

Require that Workers be Masters to do your task: If selected, only highly qualified workers will see your experiment. This adds an additional increase in the quality of your data, but Mturk charges an additional 5% overhead fee for this service. Whether this is worthwhile depends on your own preferences. Personally I haven't bothered with it much thus far, as I find my other requirements to be sufficient.

Specify any additional qualifications Workers must meet to work on your tasks: this is where you can set the worker requirements. Press the gray button to add additional restrictions. Note that some qualifications are subject to an additional fee imposed by Amazon. All the way at the bottom of the list you will find any custom qualifications you have created. These custom qualifications will be crucial if you plan to run multiple batches of the same experiment, or to exclude subjects who participated in one of your previous experiments. Since these qualifications are so important, I'll take a quick detour here to explain how you can create them. You may want to re-read this section again later.

How to create your own qualifications (i.e. excluding subjects from previous experiments)

You can assign your own custom qualification to workers, but only to those who have previously completed at least one of your HITs. You can find a step-by-step description walking you through the process on [this Github page](#) (credits to a guy called Arnoud Plantinga. Thanks Arnoud!). Once you have created a new qualification, it'll automatically show up as an option when creating a new HIT.

TIP: if your experiment consists of multiple batches, be sure to create a qualification denoting whether or not the subject has already completed the experiment. Exclude all workers who had this qualification assigned to them. Then update this qualification between batches! This is crucial to prevent workers from participating in your experiment multiple times. A single worker can only complete a single HIT within a batch, but can do multiple HITs across different batches.

Step 4: adding the content of your experiment

After you have finished setting up the basic parameters, it's time to add the actual meat of your experiment. There are two different ways to go about doing this. You can directly insert the HTML/CSS/JS of your experiment into the Mturk page. Alternatively, you can use an indirect approach by hosting your experiment on some other platform and redirecting the Mturk subjects to that platform. Exactly which approach is best for you depends on a number of factors including your programming experience (indirect approaches typically require less programming), the need for flexibility (the direct approach allows for more freedom in terms of JavaScript – although most indirect approaches also allow for JS) and ERB requirements.

Insert content directly into Mturk

If you have a bit of programming experience with HTML and JavaScript (see step 0 above), you can opt to directly upload your experiment in HTML form to Mturk. On the same repository where you found this guide, you can find a zip file containing an example experiment I've previously created. This should provide you with some basics.

First, “compile” your code into an Mturk-ready format. First you'll need to make sure that the HTML, CSS and JavaScript are all contained in a single text document. For simplicity, you can use the “Empty Mturk Template.html” document included in the zip file. If you open this webpage with a text editor (I recommend using Notepad++), you see three comments: “Paste HTML here” / “Paste CSS here” and “Paste Javascript here”. If you are using an IDE, you can copy-paste the code directly over these comments.

Next: insert your code into Mturk

Now you have the code needed to run your Mturk experiment, but you still have to include this code into the Mturk environment. You can do so by editing an existing project or when creating a new project. Go to the “(2) Design Layout” tab in the project. In the text editor, click on the “source” button. Select all the code and replace it with the contents of the Mturk template. Click save. That's it, your experiment is now ready.

TIP: tell Mturk which data to store. When programming your experiment in HTML, you'll have to make sure that Mturk knows which data to store. Luckily this is quite straightforward: when the worker submits the HTML form (i.e. the experiment page), Mturk will grab any input field with a name assigned to it. The contents of this input field will then be stored as the variable [NAME].

TIP: using sub-pages. Remember: Mturk is fundamentally made for simple, short tasks. As a result, all the code underlying your experiment must be contained in a single HTML page. But what if you want to show multiple pages of contents to your subjects? In this case, you can use JavaScript to show/hide parts of the HTML at will. See the “Example Experiment.html” file for more details on how to implement this.

TIP: test different browser types. In a perfect world your code will look and perform irrespective of which browser your subject is using. Unfortunately, this is almost never the case. Internet Explorer and Edge are particularly notorious for this, with some of the basic JavaScript functions not working as expected on these browsers. For this reason, you should always test your experiment for all the big browser types!

TIP: use the JSON format when recording large amounts of data

In some experiments you will know at the start of the experiment that you want to store a limited number of variables. In other experiments you may want to record a large amount of data, or you're unsure

exactly how many variable you want to store. One hacky trick here is to set up a JavaScript code such that it stores all relevant data into a single [JSON](#) object. Assuming that your data only contains primitives (and not functions), you can transform this JSON file to a single string with the Javascript command `JSON.stringify(...)`. You can assign this string to a text input object (preferably one that is not visible to the subject) at any point before the HIT is submitted by the worker. If you're using R, you can later on unbox this data using the `fromJSON(string)` function in the JSONlite package. It's super hacky, but it works. See the example experiment for an implementation of this.

TIP: prevent accidental submissions

One issue you will run into is accidental submissions: both basic HTML and Mturk implicitly assume that the worker submits a single answer. As a result, if you're not careful then any button on your page will automatically submit the work. The same happens when the subject hits enter in a text input field. This is bad! Luckily, both issues are relatively easy to prevent. For buttons, make sure that you tell HTML that you do NOT want this button to automatically submit the HTML form. You can do this by specifying "type=button" in the HTML tag. Seems redundant right? Well, the default button type for HTML is "submit". "type=button" basically tells HTML: "no, don't do your default submission thing. This is just a button people can press. Leave me alone".

To prevent submission upon pressing enter is a bit trickier. One solution I've previously used is as effective as it is hacky: workers can't accidentally submit by pressing enter if the browser ignores all enter inputs! To do so, you'll need to include the following code into JS:

```
//Block enter from submitting the form
document.onkeypress = function(e) {
  var key = e.charCode || e.keyCode || 0;
  if (key === 13) {
    e.preventDefault();
  }
};
```

This basically tells the browser: "hey, whenever you detect that "enter" has been pressed (keycode 13), then just ignore whatever you planned to do with it". This approach has one downside though: it effectively disables the enter key for the entire experiment. For me this has never been a big issue, but this is something to keep in mind.

Indirect methods: using a third-party platform to host your experiment

While creating your own HTML page is incredibly powerful and offers a lot of freedom, it does come with a bit of a learning curve. An alternative method would be to host your questionnaire on a third party platform such as Qualtrics or Gorilla, and re-direct your Mturk workers here. This is probably the easiest solution for most researchers, especially those with a limited background in programming.

So how do you integrate these third party platforms into Mturk? The easy answer is: you don't. Instead, you'll have to host your experiment on these platforms separately. You can then recruit workers via Mturk and effectively tell them: "I will pay for you to click on a link to go to Qualtrics/Gorilla and complete all the questions there."

After these workers have completed your experiment, you'll need to have some way to match their Mturk Worker ID to their Qualtrics/Gorilla responses (both to validate their work as well as to assign any bonuses). The most straightforward and low-tech way here is to tell Qualtrics/Gorilla to generate a unique code for each subject. This code should then be saved in both the Qualtrics/Gorilla platform AND be

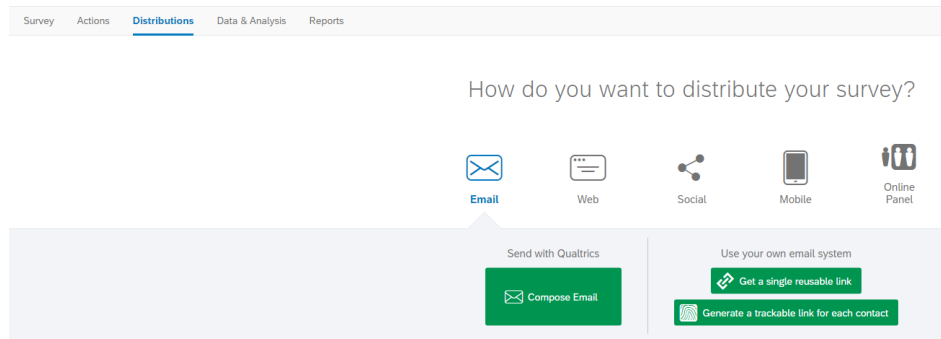
shown to the Mturk worker. The worker can then copy-paste this code into the Mturk environment. Now you can link the Mturk workers with their data!

So let's go over this process step-by-step.

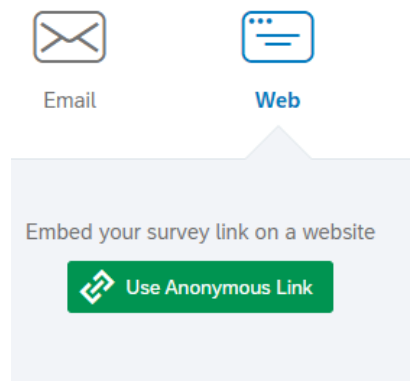
Qualtrics

Create an anonymous link

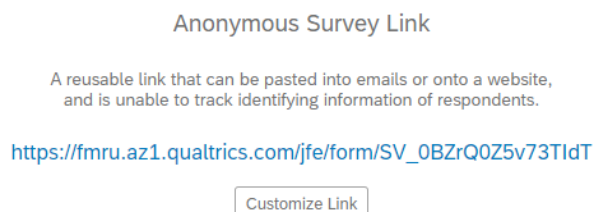
The first step is to make sure that Mturk workers are able to access your Qualtrics experiment. In order to do so, you'll need to generate an anonymous link to your experiment. You can do this by going to the "Distributions" tab:



... Select "Web" ...



... click on the big green button ...



... and store this link. This refers anybody who clicks on the link to your survey. Exactly what you need!

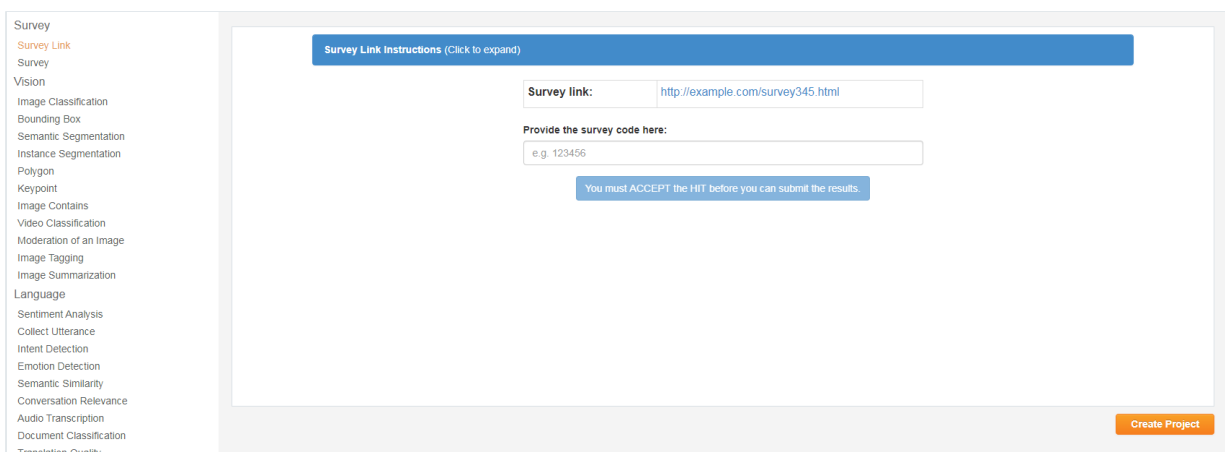
Generate a random code

Now you'll need to tell Qualtrics to generate a unique code for each subject. This code should be shown to subjects after they have submitted all their data to Qualtrics, i.e. as a post-survey message. I could tell you how to do this step-by-step, with the inclusion of a lot of screenshots to show you exactly which buttons to press. However, why reinvent the wheel if Qualtrics already created an [excellent guide](#) on their webpage.

Distribute this link to your Mturk workers and collect their unique IDs

Now you'll have all the puzzle pieces ready. The last step is to put it all together: distributing the link to your Mturk workers and collecting their unique codes after they've finished. The Mturk environment has you covered here. Remember those templates that Amazon offers when creating a new HIT? One of these is called "survey link" (currently this is even the default option – I guess Amazon realizes that this has become a very popular usage of their platform).

Select a customizable template to start a new project

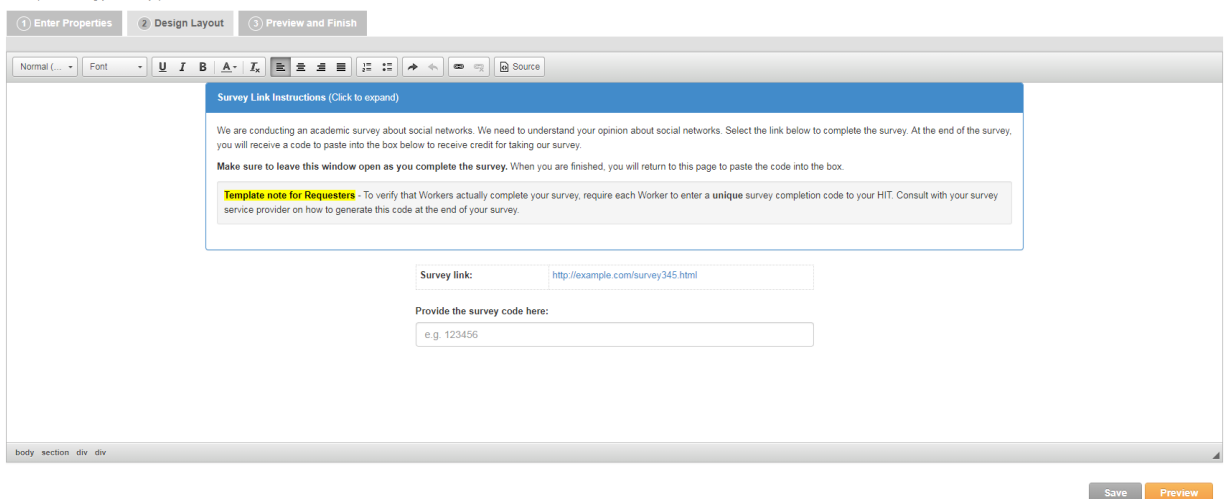


The screenshot shows the Qualtrics interface for selecting a survey template. On the left is a sidebar with a list of templates: Survey, Survey Link, Vision, Image Classification, Bounding Box, Semantic Segmentation, Instance Segmentation, Polygon, Keypoint, Image Contains, Video Classification, Moderation of an Image, Image Tagging, Image Summarization, Language, Sentiment Analysis, Collect Utterance, Intent Detection, Emotion Detection, Semantic Similarity, Conversation Relevance, Audio Transcription, Document Classification, and Translation Quality. The 'Survey Link' template is selected. The main area displays the 'Survey Link Instructions (Click to expand)' template. It includes a 'Survey link:' field with the value 'http://example.com/survey345.html', a 'Provide the survey code here:' field with the value 'e.g. 123456', and a button that says 'You must ACCEPT the HIT before you can submit the results.' A 'Create Project' button is located at the bottom right.

When you create a new project with this template, the following will appear in the “(2) Design Layout” tab:

Edit Project

For help customizing your survey, please refer to this article.



The screenshot shows the 'Design Layout' tab in the Qualtrics editor. At the top are three tabs: '1 Enter Properties', '2 Design Layout' (which is active), and '3 Preview and Finish'. Below the tabs is a rich text editor toolbar with options for font, bold, italic, underline, link, and source. The main content area shows the 'Survey Link Instructions (Click to expand)' template. The text inside the template reads: 'We are conducting an academic survey about social networks. We need to understand your opinion about social networks. Select the link below to complete the survey. At the end of the survey, you will receive a code to paste into the box below to receive credit for taking our survey. Make sure to leave this window open as you complete the survey. When you are finished, you will return to this page to paste the code into the box.' Below this text is a 'Survey link:' field with the value 'http://example.com/survey345.html' and a 'Provide the survey code here:' field with the value 'e.g. 123456'. At the bottom right are 'Save' and 'Preview' buttons.

You can simply edit this template in whatever way you see fit. If you want to get a bit more technical, click on the “Source” button to see and edit the underlying HTML

Make sure to change the default text accordingly, remove the gray box and change the survey link to the anonymous link you created above. Save the results. Now you're good to go!

Redirect subjects to Gorilla

[SECTION COMING SOON]

Interactive experiments

All the different methods outlined above have implicitly been focused on experiments in which subject's decisions are either individual (i.e. their decisions do not impact other workers) or where any inter-subject interactions can be determined offline – after the data has been submitted. But what if you want an interactive experiment where multiple subjects interact with each other during multiple rounds? Here, I have some bad news and some good news. The bad news is that you will want to catch up on a bit of Python. The good news is that 99% of the work (especially the very complicated programming stuff) is already taken care of by [oTree](#) (online toolbox for ready-made economic experiments). This software is designed for experimental economists who want to host online interactive experiments but who do not have the time or programming resources to deep-dive into server management. While it still requires some basic knowledge of Python, it comes with a set of very detailed instructions and examples. Although a detailed discussion of oTree is far beyond this guide, if you do decide to use this software then I recommend checking out [oTree studio](#) to decrease the steepness of the learning curve quite substantially.

Step 5: Test, test, test.

At this point your experiment is all ready to go. However, before recruiting workers there's one more crucial step: testing your experiment. For obvious reasons, Mturk workers do not appreciate bugs in your survey (how would you feel if HR lost your paycheck for last month?) and they will send you very strongly worded emails if any do occur. That's from experience. So before you launch your experiment, test the living hell out of it. Try every possible combination of actions and inputs that the worker could possibly do (especially the entirely illogical ones). Actively try to break your own experiment before you launch it.

One great way to facilitate testing is by using Amazon's sandbox environment. This is a testing environment where you can simulate being a worker. In order to use this environment, you will need to create two new accounts: a [sandbox worker account](#) and a [sandbox requester account](#). Note that the sandbox requester account is not linked to your normal requester account. As such, you will have to re-create your experiment in the sandbox environment (alternatively, you can create your experiment in the sandbox account and then copy-paste it to the live environment after testing).

Using the sandbox environment

After you have created your experiment in the sandbox environment, click on "Create". You will see the following table:

Start a New Batch with an Existing Project				
Project Name	Title	Created ▼	Last Edited	
Investment Game	Does this image contain the object of interest?	April 26, 2019	June 2, 2019	Publish Batch Edit Copy Delete
Keypoint	Place points on the requested item(s)	April 25, 2019	April 25, 2019	Publish Batch Edit Copy Delete

On the row containing your experiment, click on the “Publish Batch” button. You will now see a preview of your HIT as it will be shown to the subjects. Click on “next”. Amazon will now provide you with a summary of the batch and the associated costs:

Investment Game

Batch Summary

Batch Name: Investment Game 12

Description: Does this image contain the object of interest?

Batch Properties

Title: Does this image contain the object of interest?

Description: Does this image contain the object of interest?

Batch expires in: 7 Days

Results are auto-approved and Workers are paid after: 3 Days

Tasks

Number of tasks in this batch: 1

Number of assignments per task: x 3

Total number of assignments in this batch: 3

Cost Summary

Reward per Assignment: \$0.01

x 3 (total number of assignments in this batch)

Estimated Total Reward: \$0.03

Estimated Fees to Mechanical Turk: + \$0.03 (see details)

Estimated Cost: \$0.06

Applied Prepaid HITs Balance: - \$0.06

Remaining Balance Due: \$0.00

Back Publish

Check if all these numbers add up. Note that the fee here only includes the BASE payment, not the bonus. **At this point, double check the link on the webpage to make sure that you are actually in the sandbox environment (requestersandbox.mturk, not requester.mturk).** Otherwise you will accidentally be publishing a live experiment. If you do accidentally publish a live experiment and catch your error quickly enough, you can cancel the HIT before any workers start. Realistically however, you won’t catch it in time and will have to pay some workers for useless data. And reply to a bunch of angry emails if your experiment still contained any bugs. I’ve made that mistake before, and now I obsessively check the link first. After double-checking, press the “Publish” button. You will now see a screen with a status-bar. I’ll come back to that later.

Now, in a different tab open your worker sandbox account. You will see a long list of other people testing their HITs as well. Sort by “created”. At some point, your experiment will appear in this list. If you don’t see it right away, don’t panic. The servers usually take a while to update, sometimes up to a few minutes. Just refresh the page every so often until your experiment pops up.

After your experiment shows up in the list, click on “Preview” or “Accept & Work”. You will now see your experiment exactly the way your subjects will see it. You can run through the experiment however you like. Be sure to complete at least one full run of the experiment so that all subject data will be submitted.

After you have submitted at least one response, you can check if your data is stored properly. To do so, first go back to your sandbox requester account. Go to “Manage”. You will see something like this:

Investment Game 12

Review Results Delete

Created: June 02, 2019

Assignments Completed: 1 / 3

Time Elapsed: 7 days

Estimated Completion Time: Not yet available

Batch Progress:

33% submitted

100% published

Click on “Review Results” (right-hand side). Then, on the right-hand side you will see a button “Download CSV”. Press this one. The download will start automatically afterwards. Note: as before, the system is usually a bit slow to update. If nothing appears yet, then give it a minute first. Now double check to make sure that this CSV file indeed contains all the data you want to record.

TIP: when testing your experiment, be sure to not set any worker requirements! If you do, you will not be able to accept your own HIT as your sandbox account probably does not qualify!

Step 6: Publish experiment

Finally, at this point you’re ready to publish the live version of your experiment. The procedure for this is very similar to that of the sandbox experiment. First, double check the following:

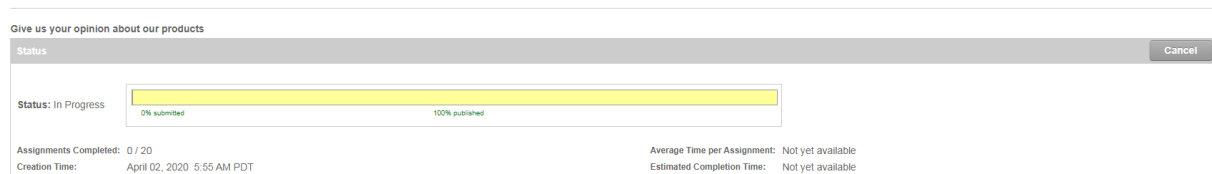
1. Do you have the latest and final version of the experiment on your requester account? Remember: your requester account is not connected to your sandbox requester account! You can preview the current state of your HIT via Create -> Edit. Then go to “Preview and Finish”.
2. Make sure that the batch size is set correctly.
3. Make sure that the rewards are set correctly
4. Make sure that the exclusions are set correctly.

If everything checks out, you can launch the HIT. Go to “Create”. As before, click on “Publish Batch” for the row listing your experiment. Amazon will now show you a preview. If this checks out (it really can’t hurt to excessively check your experiment at this point), click “Next”. Amazon will now give you an overview of all the parameters, including the estimated costs. Double check all the elements here. Once you hit “Publish”, the experiment will be published to all available workers (if they meet the requirements that is).

You will now see this:

Survey Link 2

View the latest status of this batch, make changes, or get results.



If you’re anything like me, this status-bar will become the focus of your attention and obsessive refreshing for the immediate foreseeable future. Over time this bar will fill with a green color as subjects complete the HIT. Do not panic if it takes forever before the first subjects complete your task! There are many reasons why it always takes a bit before the first workers submit their work: the system may be slow to actually publish your HIT, workers may have a large number of tasks cue up, etc. If this is your first experiment (or the first time you run a new experiment), you may want to keep the initial batch size small and keep your eye on the email address coupled to your requester account. If any issues pop up, workers will be sure to let you know right away. Sometimes in a friendly tone.

At this point, the waiting game starts.

Step 7: Assign rewards and validate work

At some point the data will start pouring in. At any point in time, including during the time that your HIT is live (remember: you can set the maximum duration of this window in the parameters of the HIT) you can check in on your subject responses by going to “Manage” and clicking on the “Review Results” link of your current batch.

You will now see something like this:

Survey Link 2

Customize View Filter Results Upload CSV Download CSV

1 of 1 assignments (FILTER APPLIED: only show assignments that are in 'Submitted' status)

Approve	Reject	HIT ID ▲	Worker ID	Lifetime Approval Rate	Surveycode
<input type="checkbox"/>		302OLP89D0E41O2B02RQPMNR0VDACF	A3TFZ8ZUXCGLC5	100% (14/14)	Testcode

Approve Reject

1 of 1 assignments (FILTER APPLIED: only show assignments that are in 'Submitted' status)

Each row (only 1 in this case) represents the data submitted by a single worker. If there's a lot of variables being recorded by the Mturk environment, you will only see a subset of the variables here. You can add/remove variables with the “Customize View” button. Same as in the sandbox environment, you can download the data at any time by clicking on the “Download CSV” button.

Assigning a bonus

Remember: in the current format the workers will only receive their base pay. If you have included any incentives in your experiment, you'll have to assign them to your subjects in an additional step. In order to do so, you first need to know what amount to assign to each subject. If you hosted your experiment directly on Mturk, you probably have this data stored in one of the variables (if not, then you really want to consider changing this as it'll save you a lot of effort during this step). If you hosted your experiment on a third party platform like Qualtrics or Gorilla, you will have to retrieve the worker's payment data from these platforms and connect them to the individual workers via their unique ID code.

Once you have the payment data ready, you can start assigning bonuses to workers. If you google around a bit, you'll quickly find out that there are a number of ways to do this automatically. I've previously tried to create such a pipeline, but failed in doing so for a number of reasons. At some point I realized I'd already invested 8 hours into automating a task which would probably only take a handful of hours to do manually, so I cut my losses and decided to stick with a manual approach. It's a boring job, but not the end of the world. Feel free to try and set up an automated system yourself though!

You can assign variable payment to a worker by assigning them a bonus. On the screenshot above, you can see that each Worker ID is a link. For every worker, clicking on this link will bring you to the following page (for reasons of efficiency, you may want to open this link in a new tab):

[Manage Workers](#) > [Manage Individual Worker](#)

Worker Details

Worker ID: A3TFZ8ZUXCGLC5	
Worker Status for your work	Never Blocked
APPROVAL RATE ON YOUR ASSIGNMENTS	
Lifetime	100% (14/14)
Last 30 days	0% (0/0)
Last 7 days	0% (0/0)

Bonus Worker

(Why Bonus?)

Block Worker

Worker's Qualifications For Your Work

Assign Qualification Type

You do not have any Qualification Types to assign to Workers. To create a Qualification Type, go to the [Manage Qualification Types](#) page.

You can assign a bonus by pressing on the “Bonus Worker” button. This will open the following pop-up field:

Bonus Worker

Amount:

\$ 0.00

Assignment ID:

36WLNQG781HR4GPBS92JZ53G79VBE1

Reason:

Optional

Note: If provided, we will share this Reason with the Worker

Cancel

Pay Bonus Now

Here you can assign the amount you want to pay to the subject. Remember: this is the variable, incentive-based payment and should not include the base payment. Be sure to check that you entered the correct amount, as there is no way to rescind a bonus after assigning it! (I’m always somewhat anxious about misplacing a decimal mark for that reason...). You can also add a reason. As a courtesy to your workers, I strongly recommend mentioning the name of your experiment here (as in: the name that was visible to your workers). Since these workers will complete a number of HITs throughout a day, this will help them to keep track of those HITs that have already made good on their promised rewards.

Once you have filled in both the amount and the reason, click “Pay Bonus Now”. Again: there’s no way to undo this assignment after pressing the button, so make sure that everything has been entered properly.

Accepting/Rejecting work

Once you have assigned a bonus, you still have to validate the quality of your worker’s inputs. Workers will only receive their base pay after you have validated their quality OR after the auto-validation time (set in the HIT parameters) runs out.

You can validate a subject’s work by going back to the “Review Results” page. On the left-hand side of the table you will see the following:

Survey Link 2

Customize ViewFilter Results

1 of 1 assignments (FILTER APPLIED: only show assignments t

<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	302OLP89D0E41O2B02RQPMNR0VDACF
<input type="checkbox"/>	

ApproveReject

1 of 1 assignments (FILTER APPLIED: only show assignments t

Check the box for all workers that you want to validate. Then press “Approve”. If the worker supplied obvious nonsense, you can reject their work by pressing “Reject”.

A note on rejections: due to Mturk’s build-in reputation mechanism, rejecting workers’ efforts may have far-reaching consequences for them. They may be unable to participate in future well-paying HITs! Do not take rejections lightly. Workers will typically have to grind through shitty, low-paying HITs for an extensive amount of time before reaching the qualifications set by a lot of experiments and a single rejection can ruin this. For this reason, **ONLY REJECT WORK WHICH IS OBVIOUSLY GARBAGE** (like random inputs, etc). Personally, I tend to strongly err on the side of caution here.

Step 7: Extract Data

After all of your outstanding HITs have been concluded and all subjects have been paid, you can download your final data from Amazon’s servers. The process is the same as before: in the “Review Results” page, click on the “Download CSV” button. Congratulations, your data-collection is now complete!