Human computation scaling for measuring meaningful latent traits in political texts

Supplementary Information

SI-1 ANALYSIS OF 50 MOVIE REVIEWS

We originally ran a similar movie review analysis as discussed in our first application using 50 reviews. We completed 60 comparisons in batches of 10. Several weeks later we collected another 20 comparisons. The findings are very similar to our application in the main text. Correlations peaked around 20 comparisons, and more comparisons increased precision but only mildly. However, with only 50 reviews every document was compared to every other document, or nearly so, making this analysis somewhat less instructive. As such, we increased the number of reviews to 500 and repeated the exercise in the main text of our paper.

Nevertheless, we present the findings from this smaller study here. Figure SI-1 shows the correlations between stars at each successive iteration. Figure SI-2 shows the correlation between the first 20 and the last 20 comparisons estimated separately in the left panel, and the correlation between the first 20 and an additional round of 20 estimated separately.

Figure SI-1: Correlations after every ten comparisons

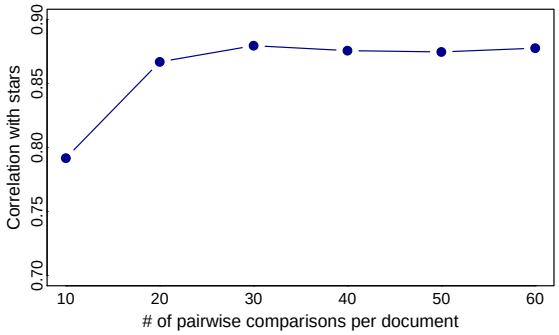
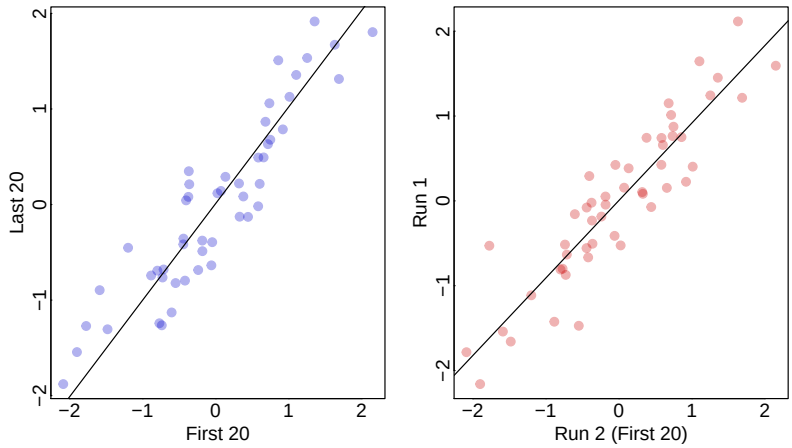


Figure SI-2: Reliability between first and last 20, and first 20 and separate experiment of 20



SI-2 BENCHMARKING AGAINST A MACHINE LEARNING METHOD

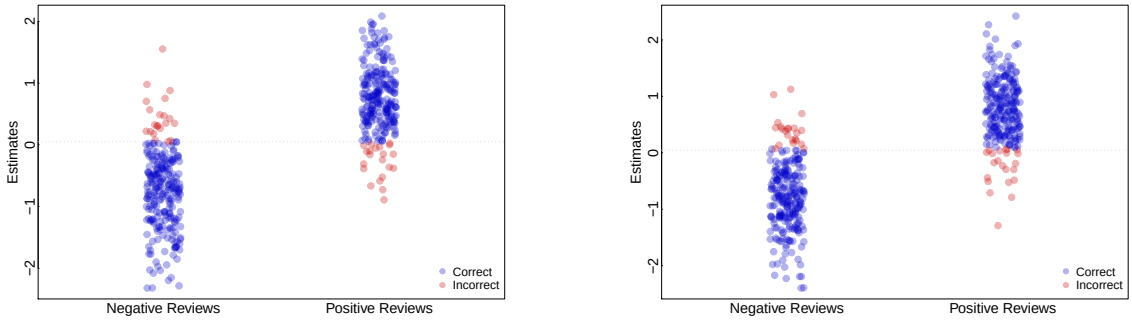
Pang and Lee (2005) analyze the polarity of 10,663 movie review snippets from Rotten Tomatoes. Half of the snippets (extracts from longer reviews) are taken from positive reviews, and half from negative reviews. Pang and Lee (2005) assume that snippets taken from positive reviews (e.g., 5-star reviews) are positive while the opposite is true for snippets taken from negative reviews.

This is a widely used corpus of texts in the computer science literature, and the success of various algorithms are often evaluated based on their success in classifying these statements. Socher et al. (2013), for instance, advertise the success of their supervised model by showing they are able to correctly classify 85.4% of the snippets as either positive or negative relative to the previous baseline of less than 80%. To illustrate the effectiveness of the `SentimentIt` approach we replicate their analysis with a random selection of 500 snippets, half of which are positive and half negative.

We did not choose to present the analysis in the main text because the exercise is aimed at categorization, and the short snippets do not fully leverage the utility of our approach. Further, we discovered that the training sets contained errors (e.g., positive snippets included in the negative grouping).[1]

We ran the experiment with and without a certification. We are able to classify 91.6% of the documents "correctly" without the qualification and 91.1% requiring a qualification. Figure SI-3 show the results of our experiment without and with qualifications. Most of the misclassified documents, upon further investigation, were found to be deceptive. They were largely either positive sentences taken from a negative review, or negative sentences taken from a positive review.

Figure SI-3: Identifying polarity in movie reviews without and with a qualification



Note: The left panel shows `SentimentIt` estimates without requiring a qualification and the right panel shows estimates requiring a qualification. The accuracies are nearly identical, with most misclassifications the same. This is likely due to the simplicity of the task.

[1]For instance, the statement, "The stunt work is top-notch; the dialogue and drama often food-spittingly funny," was included in the negative document set while the statement, "A brilliant gag at the expense of those who paid for it and those who pay to see it," was included in the positive document set.

SI-3 STAN CODE FOR STATISTICAL MODEL

We provide the Stan code for both the random utility model and the hierarchical random utility model in this section. This code was run in `R` using the `rstan` library. This is fully incorporated into our `R` package, `sentimentIt`. Stan utilizes Basic Euclidean Hamiltonian Monte Carlo sampling which involves three tuning parameters. Stan automatically determines these parameters, although allows for manual setting. We do not manually set these parameters, but allow Stan to set them automatically. We do, however, increase the maximum treedepth from the default to 50 for the hierarchical model.

Random utility model

```
data {
  int N; // number of comparisons
  int M; // number of documents
  int P; //Number of coders
  int y[N]; // outcome
  int g[N];    // id  map first item in comparison
  int h[N];    // id map of second itein comparison
  int j[N]; // id map for workers
}
parameters {
  real a[M];
  real<lower=0> b[P];
  real<lower=0> sigma;
}
model {
  sigma~normal(0,3);
  for(p in 1:P){
    b[p] ~ normal(0,1);
  }
  for(m in 1:M){
    a[m] ~ normal(0,sigma);
  }
  for(n in 1:N) {
    y[n] ~ bernoulli(inv_logit(b[j[n]]*(a[g[n]]-a[h[n]])));
  }
}
```

Hierarchical random utility model

```
data {
int N; // number of comparisons
int M; // number of paragraphs
int D; // number of documents (countries)
int P; //Number of coders
int y[N]; // outcome
int g[N];    // id  map first item in comparison
int h[N];    // id map of second item in comparison
int j[N]; // id map for workers
int k[M]; // id map for documents (countries) relating to documents
}
parameters {
```

```
real a[M]; // paragraphs
real t[D]; // documents (countries)
real<lower=0> b[P];
real<lower=0> sigmac[D];
}
model {
for(p in 1:P){
b[p] ~ normal(0,1);
}
for(d in 1:D){
t[d] ~ normal(0,1);
sigmac[d] ~ normal(0,.5);
}
for(m in 1:M){
a[m] ~ normal(t[k[m]],sigmac[k[m]]);
}
for(n in 1:N) {
y[n] ~ bernoulli(inv_logit(b[j[n]]*(a[g[n]]-a[h[n]])));
}
}
```

SI-4 ROBUSTNESS TO MODELING CHOICES

Table SI-1 shows the correlations between the `SentimentIt` estimates and the mean coder selection. Mean coder selection is calculated by averaging how often a document is chosen over the other documents in the comparisons. These correlations are very high, at or above 0.95. This demonstrates that our modeling choice is not driving the results or imposing restrictive assumptions and that our results are robust to the specific modeling and prior choices in the main text.

Table SI-1: Correlations between `SentimentIt` scores and mean coder selection

Application	Correlation
Movie reviews	0.96
Campaign ads	0.96
Immigration survey	0.98
Human rights reports	0.95

Note: The mean coder selection of the documents is highly correlated with the `SentimentIt` estimates. This suggests our modeling choice is not driving any result.

SI-5 EVALUATING WORKER QUALITY AND RELIABILITY

In this section, we argue that the Amazon Mechanical Turk Workers provide high-quality and reliable responses to the binary evaluations administered by the `SentimentIt` system. As has been reported in multiple previous studies, our evidence demonstrates that there is little reason to question the quality of the workers when appropriate steps are taken to correctly monitor workers and analyze the resulting responses.

SI-5.1. *Previous studies*

The availability of online workforces has only recently been utilized for social scientific research, with its primary focus being on inexpensive survey experimentation (e.g., Bohannon 2011). However, the scientific application of online workforces is increasingly directed towards data analysis. Amazon's Mechanical Turk (AMT) service, in particular, has been leveraged to code data related to natural language processing; speech and vision; sentiment, polarity, and bias; information retrieval; and information extraction (Callison-Burch and Dredze 2010). In the realm of natural language processing, tasks such as affect recognition, word similarity, word sense disambiguation, etc., AMT-worker codings have a high degree of similarity with codings provided by expert coders (Snow et al. 2008). Because the online workforces are fast and inexpensive, deficiencies in the reliability of non-expert coders can be ameliorated with multiple labelings of the data (Ipeirotis et al. 2014) and through independent assessment of the observations (Vempaty, Varshney, and Varshney 2014).

For example, researchers of relevance assessment (determining if documents are relevant to a pre-specified request) have had significant success on the AMT platform. As with labeling, relevance assessments from AMT workers are reliable at levels comparable with gold-standard measures, particularly when coders are presented with binary choices (Alonso and Mizzaro 2012). Moreover, the reliability of coding relevance assessments improves when splitting the document into smaller tasks, allowing for shorter individual completion times as well as overall completion time due to parallelizing tasks (Alonso and Baeza-Yates 2011).

In the area of sentiment analysis, Hsueh, Melville, and Sindhwani (2009) compare AMT workers against expert coders to find that the aggregate accuracy of the online workforce approached the gold-standard set by the expert coders. This effect increased when removing the noisiest coders from analysis, suggesting that a screening process ought to improve accuracy.

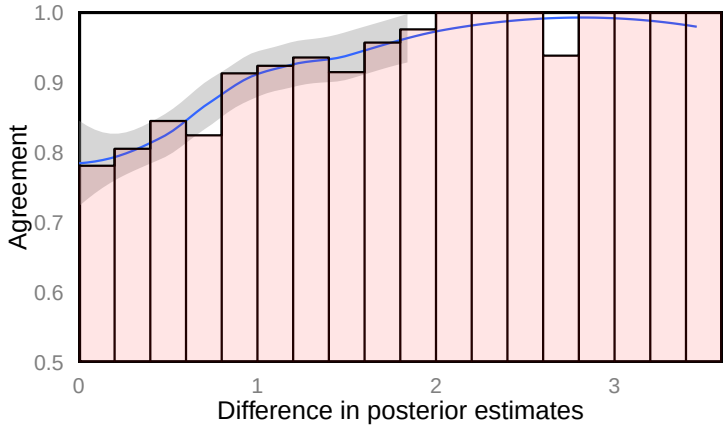
SI-5.2. *Binary codings are reliable*

One way to check worker reliability is to assess the degree of agreement between repeated comparisons. To do this, we analyze the data shown in Appendix SI-1, where 50 movie reviews were evaluated in 60 different pairwise comparisons. This allows us to have a large number of instances where the exact same comparisons were made multiple times. In the application, 1049 HITs were duplicates, with 423 unique comparisons done more than once. (The same comparison was presented to coders from two to six times.) At the comparison level, 73.8% of duplicated comparisons had total agreement, meaning that all coders made the exact same decision. Thus, only 111 of the unique comparisons had some amount of disagreement, with 67 of those being 50-50 splits. In all, the average agreement on coding of these comparisons is 88.84%, meaning that in almost nine out of ten coding decisions of the same comparison were evaluated identically.

Of course, the fact that there is some disagreement is not surprising. Specifically, we would expect that documents that are very similar in terms of their level of positivity would be more difficult

to distinguish leading to more disagreement. Figure SI-4 shows the proportion of agreement as a function of the absolute distance of their posterior estimates. In general, as the distance between the estimates increases, the proportion of agreement increases. Thus, when documents are very similar in their level of positivity, the average level of agreement is roughly 80%. However, once the documents become more distinguishable the agreement rate shoots up to over 90% or better for absolute distances of one or more.

Figure SI-4: Worker agreement on the difference of posterior estimates



Note: The plot shows the proportion of agreement of repeated comparisons on the difference of the posterior estimates. The blue line is a loess smoothed plot of the data points, and the red bars are binned average agreement with differences of 0.2. In general, as the difference in the posteriors increases, the proportion of agreement also increases.

SI-5.3. *Pairwise comparisons are transitive*

A further check to ensure workers are reliably completing the tasks is to check for transitivity in the pairwise comparison. This is also a useful check to ensure uni-dimensionality in the task. We analyze the application of 50 movie reviews described in Appendix SI-1 to increase the number of possible intransitive relationships since more complete triads appear in this dataset.

To assess the transitivity of the comparisons, we begin by generating a data frame of all available comparison triads. That is, we identified all cases where three documents had all been evaluated against each other (e.g., A-C, B-C, A-B). For each row, we then created columns, "codeA, codeB, and codeC", that take on the following interpretations of their values:

$$\begin{aligned}
\text{codeA} &= 0 : A < B; \quad 1 : A > B \\
\text{codeB} &= 0 : B < C; \quad 1 : B > C \\
\text{codeC} &= 0 : C < A; \quad 1 : C > A
\end{aligned}$$

With this setup, it is clear that the only violations of transitivity occur when all three codes take on a value of 0 ($A < B < C < A$) or 1 ($A > B > C > A$). Therefore, to calculate the percent of cases that maintain transitivity, we simply calculate the percent of rows in the data frame of triads where the coded values do not sum to 0 or 3. Of 3,649 available triadic comparisons, 92.79% are transitive. We find these results both in line with uni-dimensionality and a competent workforce.

SI-5.4. *No evidence of systematic worker biases*

One potential problem for our approach is there exist systematic biases among coders. Due to our procedure, idiosyncratic biases by subsets of coders are largely irrelevant. If one coder, for instance, tends to understand all ads as more negative than other coders, this is not relevant due to the pairwise comparison framework. So long as we are asking for only *relative* evaluations of ads, a general bias towards perceiving higher levels of negativity are irrelevant so long as it is applied equally to all documents. Somewhat more problematic is if a coder has a bias against subsets of documents. A liberal coder might, for instance, be more sympathetic to Democrats and view Republican ads as more negative in tone on average. However, in this case the statistical model in the main text will automatically down-weight her choices as they do not align with how other coders are evaluating the same documents.

However, a more serious problem is the possibility that, because these workers are not representative of the population, they may have biases in the aggregate. Imagine, as an example, that a large proportion of the workers are biased against Republicans, something that is not impossible given the general liberal leanings of AMT workers (Berinsky, Huber, and Lenz 2012). In this hypothetical case, biases may persist both the pairwise comparisons framework and the statistical post-processing.

While we cannot entirely rule out this possibility in all instances, we have so far not found any patterns in our data supporting this claim. For instance, when we look at the most extreme estimates in the congressional ad application, there appears no relationship between the partisan affiliation of the advertisement and the direction or extremity of our estimates. The means of the Democratic ad estimates is 0.02 while the mean for Republicans is -0.01, both very close to zero and close to each other. Further, the correlation between the estimates and whether or not the ad is supporting a Democrat is 0.02 while the correlation of the estimates and the ad supporting a Republican candidate is -0.02. Of course, the parties may engage in attack ads with different intensities or frequencies, but we find no evidence that our estimates are at all related to the party of the advertisement as would be the case if the liberal biases of AMT workers were affecting evaluations.

SI-5.5. *Completion times*

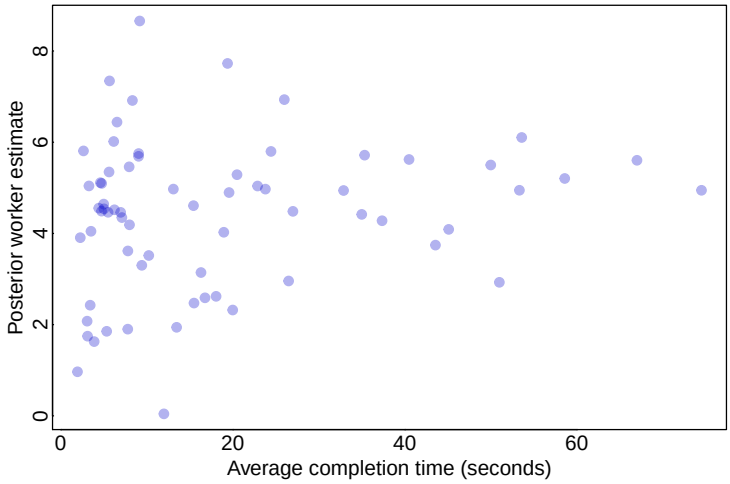
Despite the evidence that AMT workers are high quality workers, particularly when requiring training and a certification, it is possible that workers, once certified, are opportunistic and simply attempting to click through as many HITs as possible to accumulate the most money before (and if) we remove them from our jobs. To assess this possibility, we analyze the posterior worker estimates as a function of estimated HIT completion time. Figures SI-5 and SI-6 shows the posterior estimates of worker reliability plotted against the estimated average completion time for two of our applications (recall that lower estimates indicate lower-quality coders).[2] As can be seen, absolutely no pattern emerges, indicating that speed of completion may be more related to the abilities and pacing of the worker than the quality of their evaluations.[3] Likewise, Figure SI-7 shows the variance of the posterior estimates for the workers on their average completion time in the movie review application. No strong relationship seems present, suggesting further that completion time

[2] Plots for the remaining applications look essentially identical.

[3] Based on email conversations with workers, many begin to increase speed with practice as they become more accustomed to the task structure.

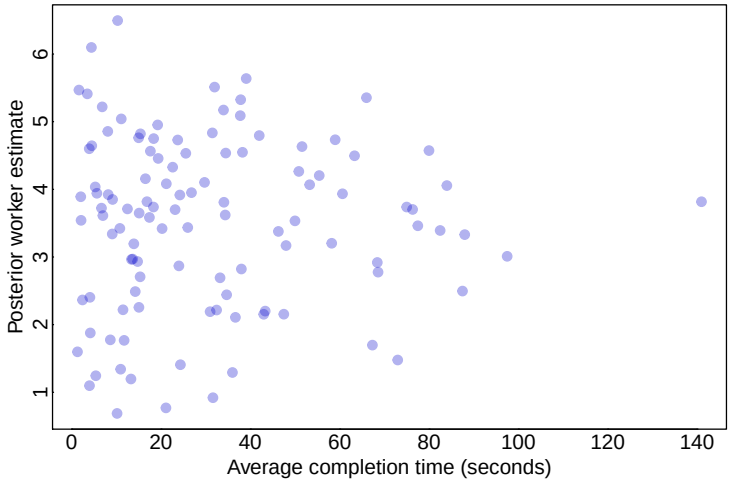
is not a significant determinant in worker quality or our uncertainty of the estimates. Thus, although we acknowledge that a small number of workers may act opportunistically if unmonitored, there is no evidence that a significant number of "bad" workers are simply clicking through HITs.

Figure SI-5: Posterior worker estimates on average completion time: Movie review application



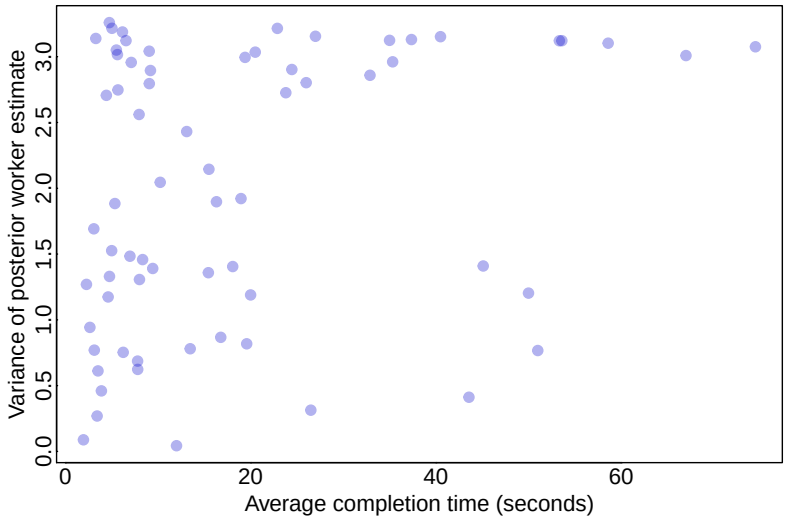
Note: There appears no relationship between average completion time and worker quality.

Figure SI-6: Posterior worker estimates on average completion time: Congressional advertisement application



Note: There appears no relationship between average completion time and worker quality.

Figure SI-7: Variance of posterior worker estimates on average completion time: Movie reviews application



Note: There appears no relationship between average completion time and the variance of the worker posterior estimates.

SI-6 TRAINING MODULE FOR HUMAN RIGHTS APPLICATION

Below is an example of a training module used in the `SentimentIt` platform. This is the training we utilized for the human rights application regarding torture. The module is meant to train and certify workers to answer the following question:

> Which of the two statements show more significant levels of torture? Torture is more significant if it there is evidence it is more frequent, more severe, unpunished, or systematic. Legal punishments and maltreatment of prisoners that is not used to intimidate or extract confessions are not considered to be torture.

The workers were presented with this question and two paragraphs from the human rights reports. Their task was to select the paragraph indicating greater degrees of torture. They were required to successfully complete this module in order to perform the task. If they failed to pass the qualification, they could not participate and could not retake the qualification.

SI-6.1. *Explanation of coding task*

If you finish this training module with a passing score, you will be qualified to complete HITs posted by the requester **SentimentIt** with the title **Compare Human Rights Report Extracts**.

This task involves comparing two text extracts from human rights reports on different countries detailing torture and prison conditions in that country. The texts are drawn from United States Department of State Country Reports on Human Rights from a particular year. Your job is to determine which extract indicates more **significant levels of torture**. What is meant by "more significant levels of torture" is explained below.

Each text extract is one complete paragraph with information on the state of torture, abuse, and prison conditions in a particular country. The country in question, as well as political groups, ethnic groups, and individuals may be named in the text extract, but please do not use your own knowledge to inform your decision in comparing the two texts. Instead, use only the information in the two texts displayed, and judge which text indicates more significant levels of torture by the following standards.

An extract indicates more significant levels of torture if:

- There is evidence of **more frequent** beatings, abuse, torture, and extrajudicial killings (killings that are not carried out in accordance with the legal procedures of the country); There is evidence of **more severe** instances of beatings, abuse, torture, and extrajudicial killings;
- There is evidence that when acts of torture are committed, they are **ignored, unpunished, or encouraged**, or that the use of torture is **routine, widespread, or systematic**;
- There is evidence that maltreatment or abuse in prisons is **specifically aimed at intimidating, penalizing, or obtaining a confession from detainees**;
- The **evidence given is well substantiated**, or better supported by reports from independent agencies (Nongovernment Organizations, The US State Department, etc.).

An extract **does not** indicate more significant levels of torture if:

- The punishment is **pursuant to the legal system** or standard legal practices of the country, even if some people might consider such practices as torture;
- **Prison conditions are poor or inadequate**, for example if there is overcrowding, inadequate food, or lengthy detention without trial;

- Allegations are specifically noted by the report to be **unsubstantiated or unlikely to be true**.

For each HIT, you will see text from **two** text extracts. Your task is to read both and select which of the two extracts indicates more significant levels of torture. That is, **based only on the two text extracts, which country do you think has more severe torture practices**?

It is important that you read each text extract carefully, and that you judge each by the standards listed above, and based only on the information in the text. **Do not** make your judgments on your own knowledge of the countries in question, on text extracts from previous HITs in this exercise, or on definitions of torture different to those listed above. Skimming or reading quickly will result in low-quality evaluations, and you may not be invited to participate in our future studies.

This training module has two parts. In Part 1, we will provide three practice HITs followed by instructions about how the text extract should be coded. In Part 2, we will give you six example HITs to complete. To receive the qualification for the Compare Human Rights Report Extracts task, you must complete five out of six of these example HITs correctly.

SI-6.2. *Example practice task*

This is your second practice Compare Human Rights Report Extracts HIT. Your answer will not be scored. Please read the two statements below and click on the button that corresponds with the statement which demonstrates more evidence of torture.

Which of the two statements show more significant levels of torture? Torture is more significant if it there is evidence it is more frequent, more severe, unpunished, or systematic. Legal punishments and maltreatment of prisoners that is not used to intimidate or extract confessions are not considered to be torture.

Statement A: According to defense attorneys and former prisoners, prison conditions ranged from Spartan to poor and, in some cases, did not meet minimum international standards. Credible sources reported that overcrowding continued to be a serious problem, with 40 to 50 prisoners typically confined to a single 194-square-foot cell and up to 140 prisoners held in a 323 square-foot-cell. A defense attorney reported that his client was imprisoned in a cell that contained 140 prisoners who were forced to sleep 3 to a cot. Defense attorneys reported that prisoners in the Ninth of April prison in Tunis were forced to share a single water and toilet facility and a single razor with their cellmates, creating serious sanitation problems.

Statement B: There are credible reports that torture occurred in prisons under the control of both the Taliban and the Northern Alliance. Local authorities maintain prisons in territories under their control and reportedly established torture cells in some of them. The Taliban operate prisons in Kandahar, Herat, Kabul, Jalalabad, Mazar-i-Sharif, Pul-i-Khumri, Shibarghan, Qala-e-Zaini, and Maimana. The Northern Alliance maintains prisons in Panjshir and Taloqan, and there also is a prison in the north at Faizabad, in Badakhshan province. According to Amnesty International, there have been reports that the Taliban forced prisoners to work on the construction of a new story on the Kandahar prison, and that some Taliban prisoners held by Masood were forced to labor in life-threatening conditions, such as digging trenches in mined areas.

After coders make a choice, the following text is shown. Relevant text in the statements are highlighted to make the decision criteria clear.

[Correct/incorrect]. Statement B shows more evidence of torture. While both statements describe harsh prison conditions, Statement B specifically mentions torture. Statement A describes

severe overcrowding and maltreatment, but does not indicate that this is specifically designed to intimidate or extract confessions.

SI-6.3. *Example scored task*

Your answer to this example HIT will be scored. To receive the Compare Human Rights Report Extracts qualification, you must assess at least five of the six comparisons correctly. Please read the two statements below and click on the button that corresponds with the statement which demonstrates more evidence of torture.

Which of the two statements show more significant levels of torture? Torture is more significant if it there is evidence it is more frequent, more severe, unpunished, or systematic. Legal punishments and maltreatment of prisoners that is not used to intimidate or extract confessions are not considered to be torture.

Statement A: Methods of torture included electric shock, beatings (especially on the soles of the feet), suspension by the wrists or feet in contorted positions, burning, and near drownings. In other cases, victims are forced to remain in unnatural positions for extended periods, or have bags laced with insecticide, chili powder or gasoline placed over their heads. Detainees have reported broken bones and other serious injuries as a result of their mistreatment. There were no reports of rape in detention.

Statement B: Togo Security forces reportedly tortured a human rights monitor (see Section 4).

After coders make a choice, the following text is shown. Relevant text in the statements are highlighted to make the decision criteria clear.

[Correct/Incorrect]. Statement A contains more evidence of torture. While both statements indicate torture, Statement B reports an individual case of unknown severity, Statement A suggests widespread practice of very severe and methodical torture.

SI-7 SETTING UP CERTIFICATIONS

This section details the process of setting up a certification once the survey is completed in Qualtrics. The first step is to create a new qualification on Amazon. Then a new certification needs to be created through the `SentimentIt` API which will link to the Amazon qualification. Finally, the Qualtrics module needs to interact with the `SentimentIt` platform to grant the received certification, add workers who did not successfully complete the module to a different certification (the banned certification), and ensure that workers taking the survey have not received the banned certification.

SI-7.1. *Create a qualification on Amazon*

The first step is for the researcher to log in to Amazon Mechanical Turk as a requester. Once logged in, navigate to the "Manage" tab. Here there should be three links, the last of which is "Qualification Types." Click on this link and you will see a button "Create New Qualification Type." Create a new qualification by clicking this button. Two fields will appear, "Friendly Name" and "Description." Fill out the first with a meaningful name. This name will not be used by the `SentimentIt` platform but the workers will see this name under their received qualifications and when they see the posted HITs requiring this qualification. The second field should include a description of the certification and include a link to the qualification test. Again, workers will see this description when viewing the HITs. As an example, our congressional ad certification is titled "Senate Story Boards" and the description is:

> This task involves reading the text of two television advertisements aired during the 2008 U.S. Senate elections. Each advertisement consists of about one paragraph of text. Researchers will use your responses to better understand the "tone" of each political ad. To qualify for these HITs, you must complete a short training module located at: `https://wuslpolysci.co1.qualtrics.com/SE/?SID=SV_aWcYT6FeQbr8ZyB`

The link takes the worker directly to the Qualtrics survey.

A second qualification should also be created, one that is granted to workers who either fail the initial training or are banned by the researcher from participating in the tasks associated with the above qualification. For the same congressional certification we titled this second qualification "Banned Senate Story Boards" with the following description:

> This certification is granted if the Senate Story Board certification was not achieved, or the responses once certified were invalid. This is only applicable to tasks involving the Senate Story Board certification.

Once both qualifications are created, they are assigned identification codes by Amazon. These IDs are necessary for linking the MTurk qualification to the `SentimentIt` certifications.

SI-7.2. *Link the qualification to `SentimentIt`*

At this stage we can link the Amazon qualification to our certifications used in the `SentimentIt` platform with the unique ID retrieved from the previous step. On the `SentimentIt` API, navigate to Data / Worker Certification. Here there will be a link "New." Once the button is clicked, a prompt for the name and Amazon ID will appear. The name should be something meaningful and short and contain no spaces. This is the name used by the `SentimentIt` platform and the

`R` package. For the congressional ads, we used the name "congressads" and "bannedcongress" for the certification and banned list, respectively. The Amazon ID should be the exact ID given by Amazon for the certifications. Once these certifications are in the `SentimentIt` platform, they can be used in the HIT settings for comparisons.

SI-7.3. *Setting up the Qualtrics survey*

When the certifications are created, the Qualtrics survey can be altered to check if the worker is on the banned list as the training begins in order to disallow continuing, then grant certifications to those successfully completing the training, and add workers who do not successfully complete the survey to the banned list. Following the prompt at the beginning of the survey requesting the worker ID, a page break should be inserted followed by a page with just a Next button. Our page reads: "Click the next button. It will be disabled if you have already taken the survey or have been banned." On this page JavaScript code is embedded that checks if the worker is on the banned list (i.e., received the banned certification). The following code checks if the worker is banned from a task:

```
Qualtrics.SurveyEngine.addOnload(function()
{
this.disableNextButton();
var that = this;
var worker_id = "${q://QID29/ChoiceTextEntryValue/1}";
    //this is the worker ID inputted by the respondent at
    //the beginning of the survey (question ID 29, text field 1)
var post_url = "https://www.sentimentit.com/api/sessions.json";
$j.ajax({ //get auth token
type: 'POST',
url: post_url,
data: {email: 'name@school.edu', password: 'UniquePassword'},
    //replace with SentimentIt email and password
success: function (data) {
if(data.success == true){
auth_token = data.auth_token; //get the authorization token
}
},
});
function waitForElement () {
       //wait for the auth_token variable to be defined before executing next call
if(typeof auth_token !== "undefined"){
var get_url ="https://www.sentimentit.com/api/certifications/certname/turk_workers/" +
    worker_id + ".json?email=name@school.edu&auth_token="+auth_token;
        //replace with SentimentIt email and password, and change certname to the
        //banned certification associated with the certification, e.g. bannedcongress
$j.ajax({ //if the worker is banned, the next button will stay disabled
type: 'GET',
url: get_url,
success: function (data) {
if(data.allowed == true){
alert('You have already taken this survey and cannot continue.');
}else{
   that.enableNextButton();
}
```

```
},
error: function (data) { //function will error if worker is not in system - means
        //the worker has not attempted any certifications so should
        //be allowed to continue
that.enableNextButton();
}
});
} else {
setTimeout(function(){
waitForElement();
},250);
}
}
waitForElement()
});
```

This code disables the next button and checks if the worker is on the banned list, here named "bannedcongress," through a `GET` curl command. If the worker is not on the list, the next button is enabled. This name needs to be altered to match the banned certification of interest. The `worker_id` variable may need to be altered to match the specifics of the survey in question, but in our formatting this is simply the first text field in the survey asking for the MTurk worker ID. This code could easily be extended to check multiple certifications, such as a master banned list that tracks all workers the researcher would never want participating.

Once the training is completed, there should be a pass block and a fail block in the Qualtrics survey. In the failed block, we grant the worker the banned certification with the following JavaScript:

```
Qualtrics.SurveyEngine.addOnload(function()
{
var worker_id = "${q://QID29/ChoiceTextEntryValue/1}";
    //this is the worker ID inputted by the respondent at
    //the beginning of the survey (question ID 29, text field 1)
var post_url = "https://www.sentimentit.com/api/sessions.json";
$j.ajax({
type: 'POST',
url: post_url,
data: {email: 'name@school.edu', password: 'UniquePassword'},
    //replace with SentimentIt email and password
success: function (data) {
if(data.success == true){
auth_token = data.auth_token; //get the authorization token
}
},
});
function waitForElement () {
        //wait for the auth_token variable to be defined before executing next call
if(typeof auth_token !== "undefined"){
var input_data = "{\"email\":\"name@school.edu\",\"auth_token\":\"" + auth_token +
    "\",\"certification\": \"certname\",\"workers\":[\""+worker_id+"\"]}";
    //replace with SentimentIt email and password, and certname should
    //be the name of the passed certification in the block if the worker
    //passed the qualification, and should be the name
    //of the banned certification if the worker did not pass
```

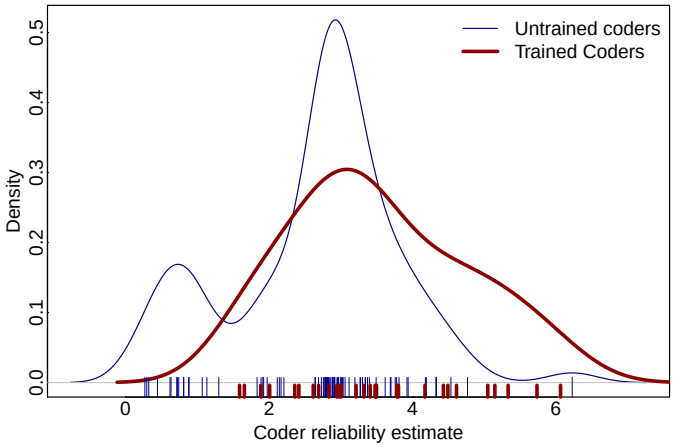
```
$j.ajax({
type: 'POST',
url: "https://www.sentimentit.com/api/certifications/create.json",
contentType: 'application/json',

data: input_data,
success: function (data) {
console.log(data);
},
error: function (data) {
console.log(data);
},
processData: false,
});
} else {
setTimeout(function(){
waitForElement();
},250);
}
}
waitForElement ()
});
```

Again, this code needs to be altered in order to match both the `worker_id` variable and the banned certification name. It grants the worker the certification through a `POST` curl command. In the passed block, the code is identical to this except the name of the certification should be the name of the certification required to complete the task (e.g., "congressads").

SI-8 EVIDENCE ON THE ADVANTAGE OF TRAINING MODULES

In Appendix SI-2, we briefly discussed an experiment where we coded snippets from movie reviews using untrained and trained workers. Figure SI-3 shows that the classification rate was not noticeably affected by the training (possibly due to the simplicity of the task). However, athough requiring a qualification test did not improve our accuracy given the simplicity of the task, we found that the workers in the task requiring qualifications were of higher quality. (We did not ban any workers throughout the process.) Figure SI-8 shows the worker-level estimates. Clearly, requiring a simple qualification test disincentivised some low-quality workers from participating.

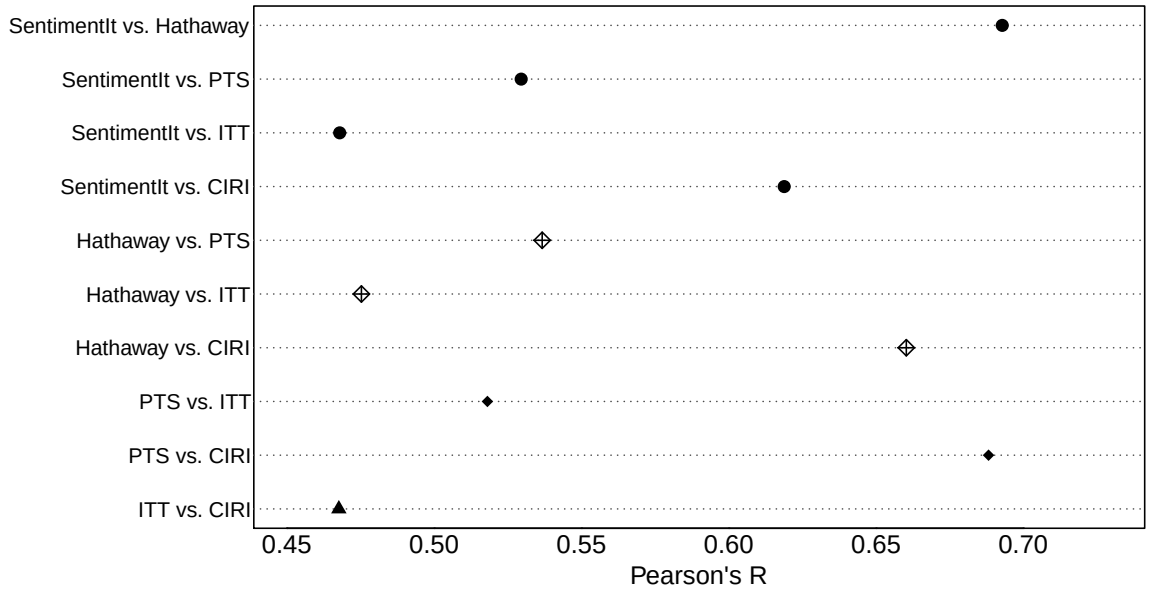
Figure SI-8: Worker-level estimates of untrained and trained workers



Note: Worker estimates in the experiment requiring training and a qualification shows that workers are performing more reliably than when no training is required.

SI-9 ALTERNATIVE MEASURE OF TORTURE

Figure SI-9: Correlations of various measures of torture



Note: CIRI correlations are the negative correlations because this measure is reverse coded. The highest correlation is between `SentimentIt` and Hathaway.

In the main text, we compare our measure with three other well-known measures of torture to assess the validity of these estimates as capturing the true degree of torture within the countries. First, we analyze the State Department variable of the Political Terror Scale (PTS) data on human rights violations (Gibney et al. 2015). This variable is constructed by a team of experienced coders, accounting for both the Human Rights Reports documents and other relelvant information. Mark Gibney and Reed Wood each code every country, and several other coders also read certain countries. When there is disagreement they discuss their decisions to reach a consensus. The second measure is the the Ill-Treatment and Torture (ITT) data (Conrad and Moore 2012). This measure is created from content analysis of Amnesty International's Annual Reports, press releases, and Action Reports. Multiple expert coders follow a very strict coding procedure. Finally, we use the torture variable from The CIRI Human Rights Dataset (Cingranelli, Richards, and Clay 2014). These data are coded by at least two expert coders, with any disagreements discussed with senior staff. Human Rights Reports and Amnesty International's Annual Reports are used for the coding. Figure SI-9 shows how all of these measures are correlated.

SI-10 R CODE USING OUR PACKAGE SENTIMENTIT

We provide as an example our first application to movie reviews. This data is also example data in the R package. We first show how interacting with online workforces can be done in stages to give an intuition behind the process and introduce the base functions used in our package. We then show how the entire process can be done using a wrapper function that will automate the entire process as a single command.

SI-10.1. *Basic functionality*

To begin, we first read in the text data, send it to the server, retrieve the document identification numbers, append these to the data, and write the data to a new file. The call is:

```
readText(email = 'researcher@school.edu',
    password = 'uniquePassword',
    read_documents_from = 'Reviews.csv',
    write_documents_to = 'ReviewsWitdIds.csv',
    sep = ',', index = 'Review', header = TRUE, quote = '"')
```

The read_documents_from argument is the file and path in which the text data is stored. This could optionally be a dataframe rather than a file path. Since this data is included as an example in the package, the following would load the data and perform the same operation:

```
data(reviews)
readText(email = 'researcher@school.edu',
    password = 'uniquePassword',
    read_documents_from = reviews,
    write_documents_to = "ReviewsWithIds.csv",
    index = 'Review')
```

The write_documents_to argument is the path and file name to store the data with the IDs appended. If set to the default, NULL, the function will return a data frame with the information rather than write to a file. The what, sep, and quiet arguments are used in the scan() function to read in the data when only text is provided, and all have defaults that would be standard for reading in datasets that contain only text. The index argument is the index indicating which column the text can be found, either the column name or column number. If an index is provided, read.table() is used instead, and the argument sep is used in this function. Not shown, the which_source argument is an indicator specifying from where the data came (e.g., "Rotten Tomatoes"). Optional arguments are all passed to the scan() function. In the example, we specify quote='"' to indicate the quoting characters. Since the movie reviews contain quotation marks the specification is necessary. All of the functions that interact with the server require the researcher's email and password used to register with SentimentIt.

Now that the data is on the server and we have the document IDs, we can create the batches of pairwise comparisons. We start with 10 comparisons per document, leading to 2,500 comparisons. We wish to send them out in batches of 1,000, so we need three batches. We run the following function:

```
batch_ids <- createBatches(email = 'researcher@school.edu',
    password = 'uniquePassword',
    task_setting_id = 8,
    num_batches = 3)
```

This assigns `batch_ids` to the batch identification numbers returned from the server. The first argument following authentication, `task_setting_id` refers to the HIT setting we wish to use. These can be set on our server using a simple form to indicate what certification (if any) is required, how long workers have to complete the task once they have started, how much money they should be paid, and how long HITs should be posted for completion before expiration. In this case, the HIT setting requires a training certification, pays the workers $0.04 per HIT, allows the worker to take up to an hour answering the question, and leaves the HIT active on MTurk for 10 hours. This also dictates what the workers will see before selecting the HIT. In this case, they see:

> You will be asked to read some text from two movie reviews and decide which seems like a more positive review. To be qualified to complete this HIT, you must complete the training module at `https://wuslpolysci.co1.qualtrics.com/SE/?SID=SV_blLMfxcmlVOtOOF`

The URL is where our training certification is located. If the workers follow the link and successfully complete the training, their worker ID will be automatically added to the list of approved workers and complete the associated HITs. If they fail the certification, they are banned from retaking the training or answering HITs associated with this setting.

Now that the batch settings are specified, we can create 10 random pairwise comparisons. We first need to retrieve the document IDs created earlier, written to the file specified, then create the comparisons using these IDs. The following code will set up the comparisons:

```
docInfo <- read.table("ReviewsWithIds", header=TRUE)
makeCompsSep(email = 'researcher@school.edu',
    password = 'uniquePassword',
    ids = docInfo[,'ids'],
    number_per = 10,
    batch_id = batch_ids,
    question = 'Below is text taken from two movie reviews.
            Please choose the text that you think comes
            from the most positive review',
    pairwise_path = 'Comparisons/first10.Rdata')
```

The first argument, `ids`, indicates the numerical IDs for the documents. The argument `number_per` is the number of comparisons desired (in this case 10). The `batch_id` argument indicates the batch IDs to be used for the HITs. The `question` argument specifies the question the worker will see once the worker selects the HIT. There is an argument `per_batch` indicating the number of comparisons per batch desired, defaulted to 1,000. If the number of comparisons is not a multiple of this number, the final batch will have fewer comparisons. We have 2,500 comparisons, so the final batch only has 500 comparisons. The number of comparisons per batch could have been automatically determined, but forcing this number to be provided ensures no mistakes are made,

such as providing too few batches. The `pairwise_path` argument is used to save the comparisons that were created to a specified path and file name where the comparisons should be stored. The function returns the batch IDs as returned by the server (which we already have stored). This serves as an assurance that the function has been correctly called.

Now that the comparisons are set up and on the server, we can post them as HITs to AMT. If we wish to send our first batch, we run:

```
createTasks(email = 'researcher@school.edu',
    password = 'uniquePassword',
    batch_id = batch_ids[1])
```

The argument `batch_id` is the identification number for the batch that we want to send, which we retrieved from the call to `createBatches()`.

At this point, we want to occasionally check the status of a batch. That is, how many of the comparisons have been completed. To do this we run the function:

```
batchStatus(email = 'researcher@school.edu',
    password = 'uniquePassword',
    batch_id = batch_ids[1])
```

The only argument to this function, other than authentication, is `batch_id`, the ID of the batch you wish to check. This could be a vector of batch IDs. This returns a dataframe with the batch ID and the number of comparisons total, submitted, completed, and expired.

Once the batch is completed (or near completed), we can check the workers to find any that are deviant. First, we need to read in the data from the server. We accomplish this by running:

```
output <- readInData(email = 'researcher@school.edu',
    password = 'uniquePassword',
    batch_id = batch_ids[1])
```

The argument to this function is `batch_id`, which could be a scalar or a vector of batch ID numbers. The returned output is a data frame with the following columns: batch_id, comparison_id, document_id, result, hit_id, worker_id, and completed_at. The data is organized by document-comparison, and the result is an indicator if the document was chosen over the other document in the given comparison. (There are, therefore, two rows for every comparison conveniently grouped by comparison so every odd row is immediately followed by an entry for the other document in the same comparison.) The worker_id is the AMT identification number of the worker, important for keeping track of the performance of workers to determine deviant (or highly reliable) workers. Finally, completed_at is a time stamp for the HIT completion time. This can be used to determine how quickly workers are completing tasks. If a worker is finishing HITs in a very fast amount of time this may suggest the worker is simply clicking through as fast as possible. In our experience only a very small proportion of workers do this, and it is quite obvious if workers are simply providing insincere evaluations.

We now need to fit the Stan model to estimate worker reliability. For the non-hierarchical model, which we used in this example, we run:

```
fit <- fitStan(data = output)
```

This function optionally has the following arguments with defaults:

```
fitStan(email = NULL, password = NULL,
    data, chains = 3, iter = 2500, seed = 1234, n.cores = 3)
```

The first two arguments are only necessary if the data provided are batch numbers rather than actual data. The following argument is the output from `readInData`, but can alternatively be a (vector of) batch ID number(s), allowing the researcher to skip the earlier step. The latter arguments are all used in the call to Stan through `rstan`. The first, `chains`, is the number of chains to run in the sampling process. The second, `iter`, is the number of iterations to run in the sampler. The default, 2500, is a conservative number that should ensure convergence even in larger datasets. However, the researcher may want to increase this number if computing time is not an issue so convergence does not need to be checked. The argument `seed` is the random seed used in the model fitting, allowing reproducibility. Finally, `n.cores` is the number of cores to run in parallel. If the researcher wants this entire process to be running in the background, changing the number of cores to use should be considered. If the process is running on a four core machine, for example, these defaults would only leave one core free for other processes.

There is a related function that fits the hierarchical Stan model. It takes the following arguments:

```
fitStanHier(email = NULL, password = NULL,
    data, hierarchy_data, hierarchy_var,
    chains=3, iter=2500, seed=1234)
```

The arguments are the same as `fitStan()`, but with two additional arguments, `hierarchy_data` and `hierarchy_var`. In order to fit the hierarchical model, the data used to set up the documents and comparisons needs to be provided in order to map the document IDs to their respective higher-level grouping. In the case of the human rights reports, this would be data that consisted of the paragraphs, the document IDs for the paragraphs, and a variable for the country of the reports from which the paragraphs came. The column name or index number for the country would be the argument for `hierarchy_var`, while the data itself would be `hierarchy_data`.

Once the model is fitted, we can check for outlier workers. To do so, we run:

```
ban_workers <- checkWorkers(stan_fit = fit, data = output)
```

The first argument is the `rstan` object obtained from the previous step, and the data is the output from the batch(es). The function has optional arguments with defaults. The full list of arguments is:

```
checkWorkers(stan_fit, data, cut_point=1, cut_proportion=0.9,
                n.questions=50, plot_hist=FALSE,
                hist_path=NULL)
```

The argument `cut_point` is the estimate cut-off for a worker to be considered an outlier. If the proportion of posterior draws falling below the cut-off point is greater than `cut_proportion`, and the worker has answered at least `n.questions`, the worker is considered an outlier. The function returns a character vector of the MTurk IDs of workers estimated as outliers. The argument `plot_hist` is an indicator to plot the histogram of workers with a rug plot so the researcher

can visually inspect the distribution. The argument `hist_path` is an argument of the file name where the plot will be saved. If left as the default, NULL, the plot will only appear through the `R` session and will not be saved.

We want to revoke the certification for these workers and add them to the list of banned workers for this task. We keep track of banned workers by granting them a different certification that indicates their certifications have been revoked. This is also how we keep track of workers that fail the qualification. First, to revoke the certification, we run:

```
revokeCert(email = 'researcher@school.edu',
    password = 'uniquePassword',
    cert = 'snippets', workers = ban_workers)
```

The `cert` argument is the name of the certification as used on the server, which in this case is titled snippets for movie reviews. The next argument is a vector of worker IDs obtained from the previous step. We now add them to banned list:

```
createCert(email = 'researcher@school.edu',
    password = 'uniquePassword',
    cert= 'bannedmovie_reviews', workers = ban_workers)
```

This will grant the workers the certification bannedmovie_reviews which indicates they can no longer participate in HITs requiring the snippets qualification.

We can now proceed with posting the other batches, checking workers whenever we choose. Once all the batches are completed, we find it common that a few HITs remain incomplete with a few HITs per batch overlooked. We can then run:

```
repostExpired(email = 'researcher@school.edu',
    password = 'uniquePassword',
    batch_id = batch_ids)
```

This will repost all of the expired HITs from the vector of batch IDs. Finally, we can run `readInData()` and `fitStan()` to retrieve final estimates of all the data.

SI-10.2. *Higher-level functions*

The functionality discussed to this point is the lowest level of functionality, where the researcher has the most control. However, we also provide "wrapper" functions that make the process easier. The most comprehensive is the `sentimentIt` function, which automates every step of the process. Complete documentation of the software is beyond the scope of this paper, but below we provide an example of a call to this function.

```
data(reviews)
movies <- sentimentIt(email = 'researcher@school.edu',
    password = 'uniquePassword',
    read_documents_from = reviews,
    write_documents_to = 'ReviewsWithIds',
    index = 'Review', task_setting_id = 8,
    number_per = 10,
```

```
question = 'Below is text taken from
    two movie reviews. Please
    choose the text that you
    think comes from the most
    positive review',
pairwise_path = 'Comparisons.Rdata',
certone = 'snippets', certtwo = 'bannedmovie_reviews',
timed = FALSE, check_workers_at = c(1,2),
rest_time = 60, rate = 1/3, threshold = 5,
return_stan = TRUE, return_data = TRUE)
```

This command will perform the following function:

- All documents in the `reviews` dataframe will be read in and passed to our servers.
- HIT settings for the task will be assigned (question wording, compensation, duration, etc.) and the `snippet` certification will be required.
- Ten random pairwise comparisons per document will be created.
- All unique identifiers for documents will be stored at `ReviewsWithIds`
- Comparisons will be posted to AMT in batches of 1,000.
- New batches will be posted once the current batch is completed up until the `threshold` of five incomplete comparisons. Completion status will be checked every `rate=1/3` of an hour.
- A Stan model with three chains and 2,500 iterations will be fit when the workers are checked and when the final data is analyzed.
- Workers will be evaluated after the first 1,000 and second 1,000 comparisons are complete, and workers with 0.9 of the posterior draws falling below the default cut point of $b_k = 1$, will be banned from completing future tasks.
- After posting comparisons, the function will wait `rest_time = 60` seconds before posting HITs to Mechanical Turk to ensure all of the comparisons are ready to be posted.
- All incomplete tasks will be re-posted.
- After all tasks are completed, the data and Stan estimates of all model parameters will be returned.

There are several advantages to this higher-level approach. First and foremost, this functionality makes the process of interacting with online workers simple and efficient from the perspective of a researcher. Once the qualifications and HIT settings have been created, a single command can supervise the collection of worker evaluations and the creation of a meaningful measure – even if this process requires several days. However, a further advantage of this approach is that it makes the process of turning text into data highly replicable. Researchers wishing to evaluate the reliability of any measure can simply re-run the task using the original call above to create a full replication. Thus, the `SentimentIt` platform has the potential to bring about a higher degree of transparency to the task of turning natural language into meaningful data.

SI-11 UNCERTAINTY IN ESTIMATES

One concern readers may have is that documents that score in the middle of the spectrum may simply be more difficult to code rather than being located in the middle of the distribution. To examine this possibility, we examined the measures of uncertainty (posterior standard deviation) associated with each document in our datasets. In short, we find no evidence to support this contention. Rather, as is typical for random utility models (and other models closely related to item-response theoretic models) we find that uncertainty estimates are actually highest for the most extreme documents.[4]

As an example, figure SI-10 shows our estimates of the movie reviews, grouped by the star rating provided by the reviewer, with 95% posterior density bars. Note that the estimates at the extreme have larger density intervals, with longer tails towards the extremes. There is overlap in the intervals between star ratings, which we discuss in the text, but overall the majority of estimates do not have overlapping intervals with estimates as proximate as a two star difference. Also of note is the intervals do not seem to vary considerably between estimates except as a function of extremity. This again suggests that there are no discernible patterns in our posteriors that are not imposed by the modeling choices, coder unreliability, and the like.

Figures SI-11 and SI-12 show the degree of uncertainty in our posterior estimates plotted on the document estimates for the movie review and congressional advertisement applications respectively. There is a very clear pattern that as the estimates become more extreme in either direction, the uncertainty of those estimates increases as is typical in measurement models in this family.

Figure SI-13 show the standard deviations of the paragraph-level estimates for the human rights applications on the estimates. Again, we see that more extreme estimates have larger levels of uncertainty. Figure SI-14 shows the same relationship for the document-level estimates. Here there is no clear pattern in the highest density region. However, there is a clear decrease in uncertainty as the document estimates become much lower. This is sensible, because the documents estimated at the low extreme are short, containing only a few paragraphs that convey only positive information. The estimate of the document with the lowest uncertainty is, however, Cuba. In this document, all paragraphs are estimated very close to the document estimate, because each paragraph is an account of some torturous act. The document with the greatest amount of uncertainty is Nepal, with a document-level estimate of 0.36 with a standard deviation of 1.18. The paragraphs for this document have estimates ranging from -2.06 to 1.87, with some clearly positive statements such as stating that the government allows human rights visits and some clearly negative statements regarding widespread, brutal torture methods used to extract confessions.

[4] Speaking loosely, the posterior variance is most significantly reduced by comparing documents with others that are nearby in the latent space on both sides. By their nature, extreme documents are less likely to be compared to many nearby documents in the latent space leading to higher levels of posterior uncertainty.

Figure SI-10: Movie review stars on estimates with 95% posterior density

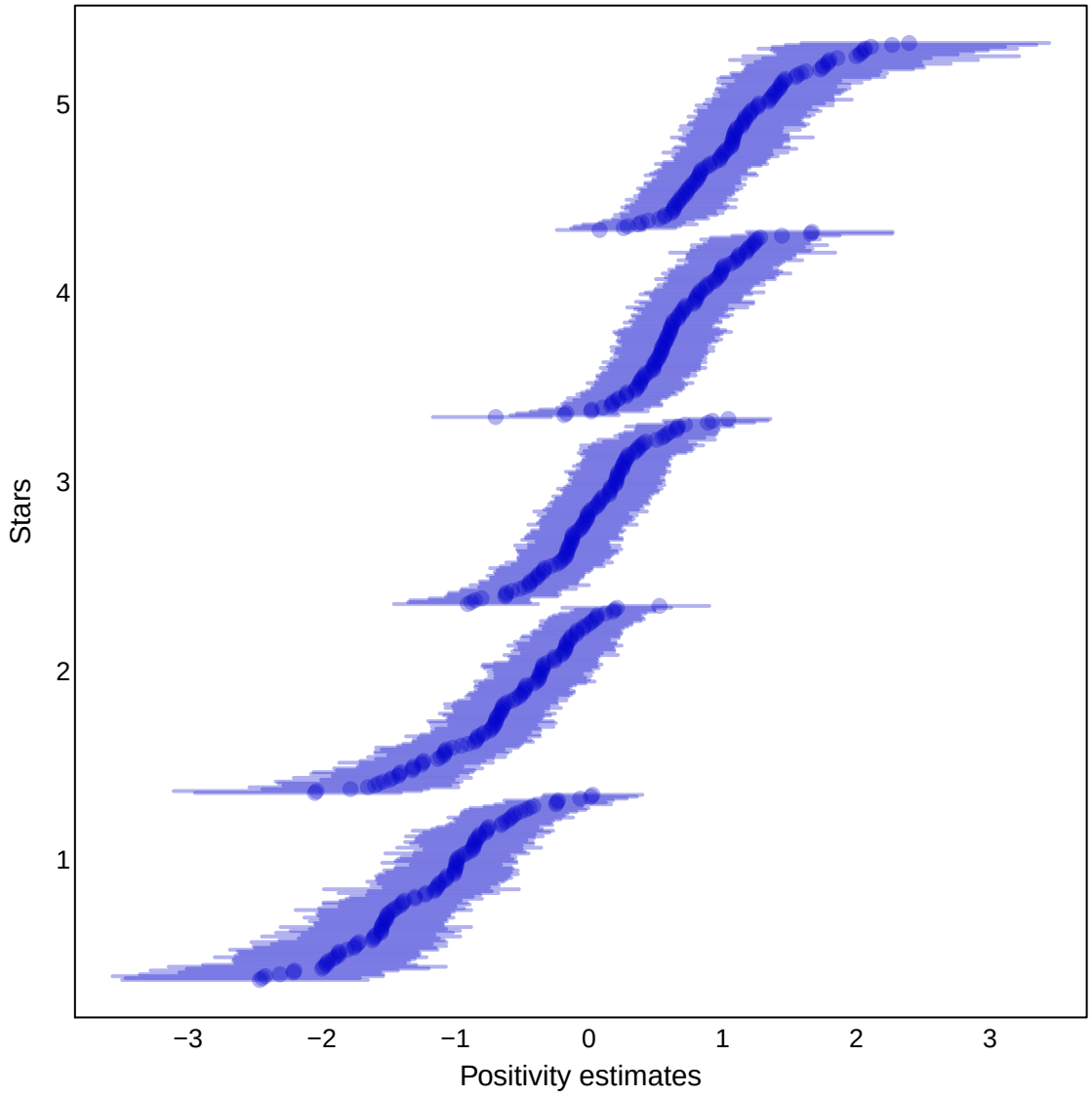


Figure SI-11: Movie review uncertainty: Standard deviations on positivity estimates

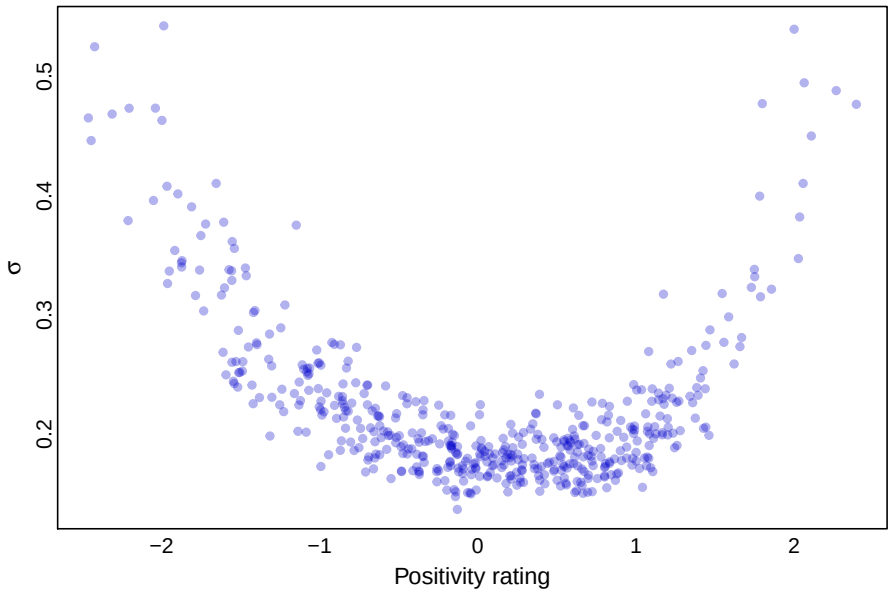


Figure SI-12: Congressional advertisements uncertainty: Standard deviations on negativity estimates

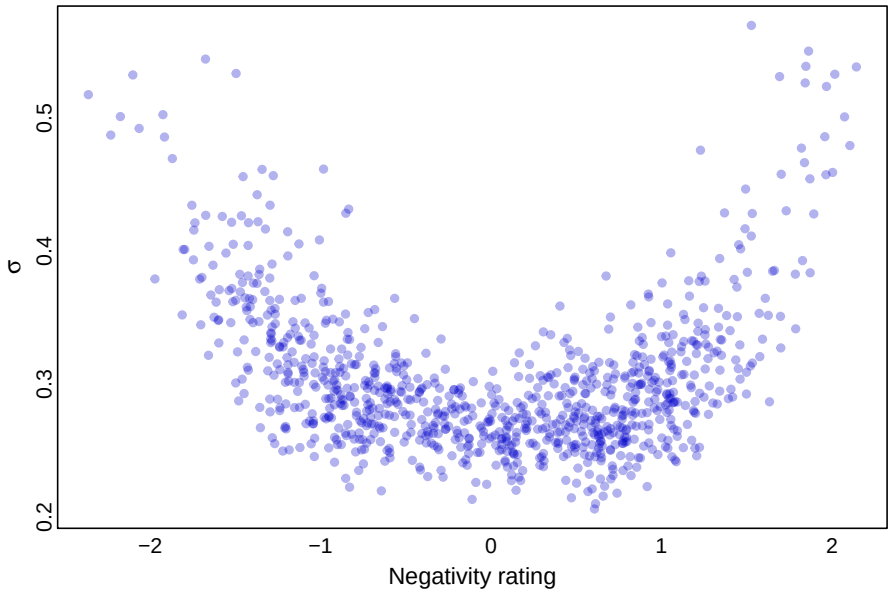


Figure SI-13: Human rights paragraph uncertainty: Standard deviations on torture estimates

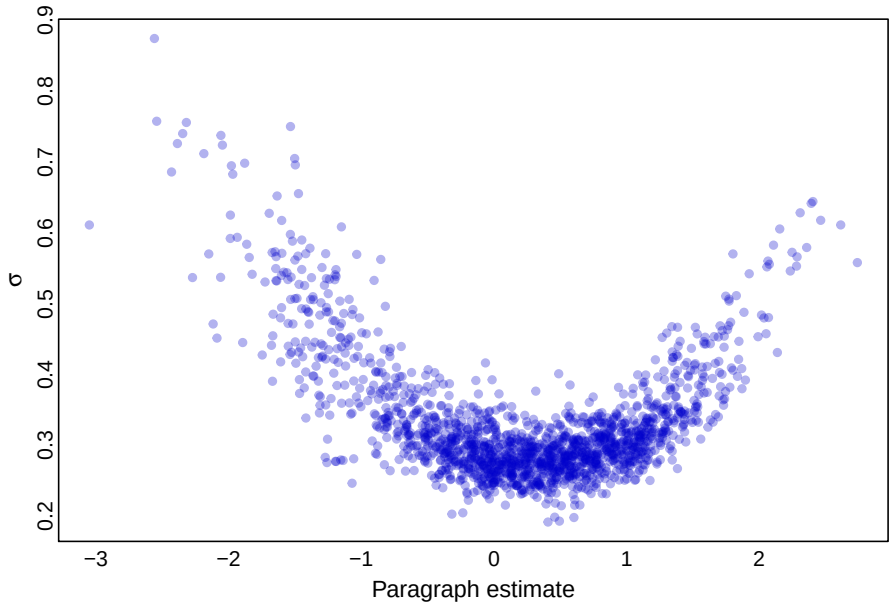
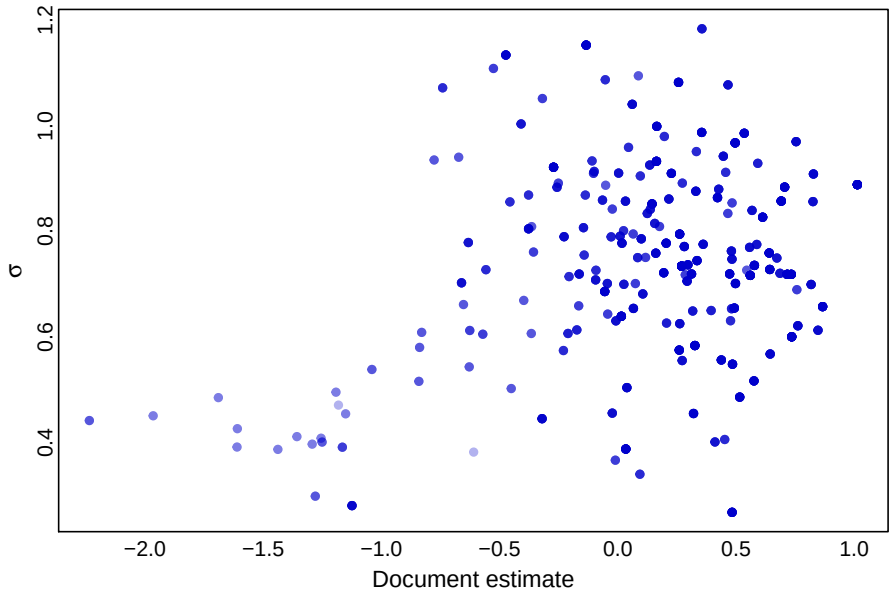


Figure SI-14: Human rights document uncertainty: Standard deviations on torture estimates



SI-11 References

Alonso, Omar, and Ricardo Baeza-Yates. 2011. "Design and implementation of relevance assessments using crowdsourcing." In *Advances in information retrieval*, ed. Paul Clough, Colum Foley, Cathal Gurrin, Gareth J.F. Jones, Wessel Kraaij, Hyowon Lee, and Vanessa Mudoch. Springer.

Alonso, Omar, and Stefano Mizzaro. 2012. "Using crowdsourcing for TREC relevance assessment." *Information Processing and Management: An International Journal* 48(6): 1053–1066.

Berinsky, Adam J., Gergory A. Huber, and Gabriel S. Lenz. 2012. "Evaluating online labor markets for experimental research: Amazon.com's Mechanical Turk." *Political Analysis* 20(3): 329–50.

Bohannon, John. 2011. "Social science for pennies." *Science* 334(6054): 307–307.

Callison-Burch, Chris, and Mark Dredze. 2010. Creating speech and language data with Amazon's Mechanical Turk. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*. Association for Computational Linguistics pp. 1–12.

Cingranelli, David L, David L Richards, and K Chad Clay. 2014. "The CIRI human rights dataset." *CIRI Human Rights Data Project* 6.

Conrad, Courtenay R., and Will H. Moore. 2012. "Ill-Treatment and Torture (ITT) Dataset." `http://www.politicalscience.uncc.edu/cconra16/UNCC/ITT_Data_Collection.html`.

Gibney, Mark, Linda Cornett, Reed Wood, Peter Haschke, and Daniel Arnon. 2015. "The Political Terror Scale 1976-2015.".

Hsueh, Pei-Yun, Prem Melville, and Vikas Sindhwani. 2009. Data quality from crowdsourcing: a study of annotation selection criteria. In *Proceedings of the NAACL HLT 2009 workshop on active learning for natural language processing*. Association for Computational Linguistics pp. 27–35.

Ipeirotis, Panagiotis G, Foster Provost, Victor S Sheng, and Jing Wang. 2014. "Repeated labeling using multiple noisy labelers." *Data Mining and Knowledge Discovery* 28(2): 402–441.

Pang, Bo, and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics pp. 115–124.

Snow, Rion, Brendan O'Connor, Daniel Jurafsky, and Andrew Y Ng. 2008. Cheap and fast—but is it good? Evaluating non-expert annotations for natural language tasks. In *Proceedings of the conference on empirical methods in natural language processing*. Association for Computational Linguistics pp. 254–263.

Socher, Richard, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*. Vol. 1631 Citeseer p. 1642.

Vempaty, Aditya, Lav R Varshney, and Pramod K Varshney. 2014. "Reliable crowdsourcing for multi-class labeling using coding theory." *IEEE Journal of Selected Topics in Signal Processing* 8(4): 667–679.