workflow. If you are trying to fit a model to a large amount of data and hitting memory limits, it is likely easier to use the high memory EC2 instance with 68 GB of memory.[11] On the other hand, if your data subsetting script is taking an hour or more to run, a cluster or multicore solution might be useful. In addition, some problems are not easily parallelized while others are not parallelizable at all. The type of algorithms that are easily parallelized, however, could serve as the subject of an entirely different article.[12]

If multicore or cluster computing is the best way forward for a given problem, there has been a copious amount of (digital) ink spilled outlining the various options available for parallel computation in both `R` and Python. In `R` there are the `foreach`, `snow`, and `snowfall` packages discussed in this article, in addition to the various implementations of `apply`.[13] There are also explicit implementations of MPI in `R` such as Rmpi, a good example of which, along with a less trivial usage of parallel processing than presented in this article, can be seen here. In Python, MPI is also available, as is the `multithread` package. The easiest and most straightforward approach, however, is to make use of IPython and `joblib`. These two should cover almost any imaginable scenario. With this in mind, the aim of this article was not to provide an exhaustive tutorial on parallel computation; in reality this would devolve into a repetition of the documentation for the various implementations mentioned above. Rather, the hope is that this article has provided the reader with a working understanding of, and a quick-start guide for, 1) initiating and running an AWS EC2 instance and 2) utilizing an EC2 instance for the purposes of parallel computing in `R` and Python.

References

Perez, Fernando and Brian E. Granger. 2007. "IPython: a System for Interactive Scientific Computing." *Computer Science Engineering* 9(3):21–29.`http://ipython.org`.

Rowstron, Antony, Dushyanth Narayanan, Austin Donnelly, Greg O'Shea and Andrew Douglas. 2012. "Nobody ever got fired for using Hadoop on a cluster". In *HotCDP 2012 -1st International Workshop on Hot Topics in Cloud Data Processing.* Bern, Switzerland: `https://research.microsoft.com/pubs/163083/hotcbp12%20final.pdf.`

Revolution Analytics. 2012*a*. *doParallel: Foreach parallel adaptor for the parallel package.* R package version 1.0.1.`http://CRAN.R-project.org/package=doParallel`.

Revolution Analytics. 2012*b*. *doSNOW: Foreach parallel adaptor for the snow package.* R package version 1.0.6. `http://CRAN.R-project.org/package=doSNOW`.

Revolution Analytics. 2012*c*. *foreach: Foreach looping construct for R.* R package version 1.4.0. `http://CRAN.R-project.org/package=foreach`.

Tierney, Luke, A. J. Rossini, Na Li and H. Sevcikova. 2012. *snow: Simple Network of Workstations.* R package version 0.3-10. `http://CRAN.R-project.org/package=snow`.

Fielding Complex Online Surveys using rApache and Qualtrics

Taylor C. Boas and F. Daniel Hidalgo
Boston University and Massachusetts Institute of Technology
tboas@bu.edu and dhidalgo@mit.edu

Introduction

Online surveys are increasingly popular in political science as an easy and inexpensive way to gather data and conduct experiments. Typically, scholars use an online survey engine, such as Qualtrics, SurveyGizmo, or SurveyMonkey, to administer the questionnaire and handle any experimental manipulation.[1] Compared to programming a survey from scratch and hosting it on one's own server, commercial survey packages present many advantages. They are user-friendly and reliable, offer an attractive graphical user interface, and may be available at no cost for scholars whose universities purchase site licenses. Their high levels of data security and built-in anonymity features should also make them more attractive to Institutional Review Boards than

[11]In fact, as I was writing this I received an email from Amazon announcing new types of instances that have 244 GiB of RAM and two Intel Xeon processors, which each have 8 cores for 16 total physical cores and 32 total threads. In reality this instance should be more than enough firepower for nearly any application that could arise in political science research.

[12]In short, if an algorithm contains the summation of results it is probably possible to run it in parallel.

[13]A good resource for a high level overviewfor some of these commands is Ryan Rosario's presentation on parallelizing R, available here.

[1]Recruitment of subjects or respondents—a separate and prior step—might rely on Amazon.com's Mechanical Turk, Facebook advertisements, or a convenience sample drawn from one's university.

self-programmed and locally-hosted surveys.

Despite their many advantages, even the most powerful online survey engines may present limitations in terms of question wording customization, complex randomization, or other goals that a researcher may wish to accomplish. For example, in our research on Brazil, one task we were unable to accomplish within the survey engine Qualtrics was to pull up a list of candidates for city council from the respondent's municipality and randomly choose one of these names to be inserted into a survey question.

In this article, we describe a strategy for augmenting the capacity of online survey engines using rApache, a version of R that runs on the Apache web server. The approach involves routing respondents to a server running an R script that, via a web interface, administers an initial set of survey questions. Based on the answers to these questions, the R script conducts any necessary randomization and database lookups and then passes the results to the online survey engine via the redirect URL. No respondent data are retained by the server that is used for this preprocessing step.

We focus on our experience integrating rApache with Qualtrics, a powerful online survey engine that is oriented toward academic research and is commonly available through university site licenses.[2] Qualtrics has recently been used for a number of online surveys in political science, many of which involve experimental manipulation (Kamal et al., 2012; Kriner and Shen, 2012; Morey, Evaland and Hutchens, 2012; Nyhan and Reifler, 2011; Popescu and Toka, 2012; Sances, 2012; Shineman, 2012; Young and Hoffman, 2009; Zahedzadeh and Merolla, 2012). As we discuss below, some of these studies might have benefited from using the method described here.

The Survey: Religion, Race, and Class in Brazilian Municipal Elections

In late September–early October 2012, we used Qualtrics to administer an online survey, including several experimental treatments, to 1820 registered voters in Brazil.[3] The survey was designed to explore issues of religion, race, and class in the country's October 7 municipal elections. Following the approach of Samuels and Zucco (2012*a*,*b*), we recruited respondents using Facebook advertisements and raffled off an iPad as an incentive for participating. The research design and results of the survey are described in greater detail elsewhere (Boas, 2013; Smith, 2013). Here, we focus on features that we wished to include in the online survey but were unable to implement natively (or could do so only awkwardly) in Qualtrics. We then describe how we were able to accomplish these tasks using rApache on a virtual server.

A first set of survey questions sought to gauge the effect on vote intention of candidates using professional titles, such as "Pastor" or "Doctor," in their ballot names (a practice this is permitted under Brazilian electoral law, and quite common). Each question described a candidate for city council, including his or her party, age, marital status, and education level. Respondents in the treatment condition were given a version of the candidate's name that included the professional title, such as "Pastor Paulo" or "Dr. Carlos"; those in the control condition were given full legal name, without the title.

To increase the external validity of the survey experiment, we wanted to ask a subset of respondents about real candidates from their municipality that were running in the upcoming election. Doing so required two steps that were impossible in Qualtrics: gathering the respondent's state and municipality in such a manner that the data could be used elsewhere in the survey, and looking up a list of candidates from that municipality. Qualtrics offers a question type, Drill Down, that involves selecting items from nested drop-down menus, and we could have used this option to ask for the respondent's state and then municipality. However, if one wants to use the answer to a Drill Down question in display logic or as piped text elsewhere in the survey (e.g., showing certain questions only to residents of state capitals, or inserting municipality name into the question "How long have you lived in _______"), there is a limit on the number of categories that can be included. We are unaware of the exact limit (and Qualtrics technical support was unable to specify it), but Brazil's 5568 municipalities and the 3141 counties or county-equivalents in the United States both lie above it. We might have been able to circumvent this problem by asking respondents to input their postal code in a text box. However, we still would have run up against a second limitation: Qualtrics has no table-lookup feature that we could have used to match municipality name or postal code to a list of candidates.

A second part of the survey sought to have respondents characterize a large database of candidate photos in terms of race and social class. We started with 1000 photos and intended to show a randomly chosen photo to each of 3000 respondents, such that each photo was rated by three respondents.[4] Randomizing among 1000 photos would have been possible in Qualtrics, but exceedingly awkward; we would have had to create 1000 versions of the same question, each with a different image URL.

In a third part of the survey, we wanted to ask a random sample of 10% of respondents if they would be opposed to

[2]Given Qualtrics's reputation as one of the most capable online survey engines available (e.g., Leland, 2011), we assume that the limitations we have encountered while using it apply to most other packages as well.

[3]The survey was conducted jointly with Amy Erica Smith of Iowa State University and was approved by the Institutional Review Boards of each of our institutions.

[4]We ultimately recruited fewer respondents than initially expected, and thus had to reduce both the number of photos and the number of times that each was shown.

their son or daughter marrying a black person. To do this within Qualtrics, we would have had to randomize among this question and 9 copies of a blank question—not difficult, but rather cumbersome.[5]

A final challenge was specific to our compensation mechanism—a raffle prize for one respondent who completed the survey. In contrast to Mechanical Turk, which allows for directly compensating participants without obtaining their personal data, our approach required us to collect the names and email addresses of respondents who completed the survey and wished to be entered into the drawing for an iPad. However, to obtain exempt status from the IRB, names and emails could not be linked to survey responses. The solution was to collect personal data through a second Qualtrics survey to which interested respondents would be redirected after completing the first one. However, we were concerned that savvy users might copy the URL of the second survey and distribute it to friends, or use it themselves to enter the raffle multiple times with different email addresses.[6] Hence, we needed a unique identifier for each respondent that would be passed from the first survey to the second, but, for purposes of anonymity, not generated by or recorded in the first survey.

The Solution: Preprocessing with rApache on a Virtual Server

We employed rApache as our server side solution. rApache is a module for Apache developed at Vanderbilt University that embeds the R interpreter inside a web server. rApache allows the Apache web server to interact with R, which is useful when one wants to use R's advanced statistical and data management tools to manipulate data received through a web interface. rApache requires Apache 2.2.x or above with an installation of R. In our particular case, we used an Ubuntu virtual server hosted by the firm Linode (http://www.linode.com), but any standard Linux-based web server should suffice. Instalation instructions are documented in the rApache manual (http://rapache.net/manual.html), but some familiarity with Apache server administration is required.

Our questionnaire, a standard web form, initiates a GET request that passes values of the items entered into the form to R. The HTML web form would look something like the following:

```
<form method="GET" action="/r-scripts/SendToQualtrics.R">
Choose the state where you vote
<select name="state" id="state">
<option value="">Select a state</option>
<option value="AC" >Acre</option>
<option value="AL" >Alagoas</option>
<option value="AP" >Amapa</option>
<option value="AM" >Amazonas</option>
...
</select>
<input type="submit" value="Advance">
</form>
```

In this example, the user is presented with a drop-down menu and asked to choose the state where they vote. One could then conduct a survey experiment in which the content of the question prompt would depend on the respondent's state. For example, the researcher could ask respondents for an evaluation of their state governor's job performance, randomly assigning them to be presented with the governor's party affiliation. Much more complicated randomization schemes and data lookups could be accommodated—including the one we used, which depended upon the respondent's state and municipality—but this running example illustrates the basic method. While this particular example could have been implemented in Qualtrics using branching or display logic, it would have been cumbersome to do so for Brazil's 27 states.

The only necessary change to standard HTML is the need to specify the R file to be called once the form is submitted. Notice that in the `action` variable in the form header, we specified an R file called `SendToQualtrics.R`. This is the R file that will handle the data entered by the user. Upon clicking on the "Advance" button, the data entered into the form is sent to an R instance. The data arrives in R in the form of the of a list object called `GET`, with an element for each variable in the form.

R receives the data as a list object that can then be manipulated. In this particular case, R would launch and run the script `SendToQualtrics.R` which could operate on the data contained in the `GET` list object, where the name of each element is given by the name attribute of the form. The element in the list object would contain the "value" specified for that particular choice in the HTML (i.e., the state abbreviations). In the case of a respondent selecting Acre in the menu, R would receive the following web object:

```
> GET
$state
[1] "AC"
```

In this hypothetical case, one could have the server-side R script load a dataset containing the names and party affiliations of all state governors. The R script could then contain functions that operate on the `GET` list, such as:

[5]More generally, one cannot directly specify unequal probabilities of selection when randomizing among questions. Hence, assigning one-tenth of a sample to a treatment condition and the rest to control would require making 9 copies of the control question.

[6]Qualtrics has an HTTP Referrer Verification feature that could be used to block users not being redirected from the first survey. However, it would also block all those who had HTTP referral disabled on their browser for security or privacy reasons, and we were reluctant to do this.

```
getGovernorData <- function(state){
  treat <- sample(0:1, 1)
  gov_name <- governors$name[governors$state == state]
  gov_party <- governors$party[governors$state == state]
  question_name <- ifelse(treat == 1,
                          paste(gov_name, "␣(", gov_party, ")",
                                sep = ""),
                          gov_name)
  URLencode(question_name)}
```

This function accepts the state abbreviation passed by the web server, randomly assigns the respondent to a treatment or control condition, and then produces the name that will be presented in the survey prompt (with or without the party label, depending on treatment status). Because this customized text will be passed to Qualtrics in the form of "embedded" text in a URL, special characters must be encoded. This list includes accented characters such as "á", which needs to be transformed into "%c3%a1", as well as a variety of more common characters, such as a space ("%20"). The R function `URLencode()` performs this transformation as necessary; we recommend using it regardless of whether one's data include accented characters.

To pass the data to Qualtrics, we need to embed it in the URL that sends the user to the rest of the online questionnaire. Qualtrics provides a unique URL for each survey, of the form `http://www.qualtrics.com//SE/?SID=SV_545454`, where the `SID=` field specifies the survey number. Additional variables can be appended to this URL in the form `&var=value`. To construct the URL in R, the following code would suffice:

```
gov_name <- getGovernorData(GET$state)
forwardUrl <- paste("http://www.qualtrics.com//SE/?SID=
    SV_545454&gov=, gov_name, sep = "")
```

The last step is to forward the respondent to the Qualtrics website with the data embedded in the redirect URL. To do so, at the end of our R script we used the following code:

```
setContentType("text/html")
cat("<head>")
cat("<code>")
cat("</code>")
cat(paste("<meta HTTP-EQUIV=\"REFRESH\"
    content=\"0; url=", forwardUrl, "\">", sep =""))
cat("</head>")
cat("<html>")
cat("<body>")
cat("</body>")
cat("</html>")
```

This code asks R to generate a blank HTML website with the single purpose of immediately forwarding the respondent to the Qualtrics website and passing along the embedded data.

In our survey, we used preprocessing in rApache to handle each of the four randomization and wording customization tasks described in the previous section. Through a similar process to the "governor" example above, we used the respondent's state and municipality to look up a list of city council candidates with "Pastor" and "Doctor" titles, randomly choose one candidate from each list, assign the respondent to the treatment or control condition (ballot name with professional title or full legal name), and pass the resulting name and candidate biographical data to Qualtrics. This included full phrases, such as

`&marital=she%20is%20married_`

that were inserted directly into the survey question as piped text.

Our additional randomization tasks were straightforward. To choose a candidate photo for the respondent to evaluate, we had R sample from a vector of 1000 filenames, passing along the result in the form `&photo=photo1.jpg`. In order to ensure that each photo was shown no more than three times, we had our photo-choosing function check the server log; if the limit had been reached, another file was chosen. To display the photo in Qualtrics, we created a Text/Graphic item, and then, in HTML view, inserted the embedded data field into the image URL, e.g., `<img src="http://www.mysurveyphotos.net/${e://Field/photo}">`. To select the 10% of respondents who would be asked about their son or daughter marrying a black person, we did `sample(c(1,rep(0,9)),1)` in R and passed along the result to Qualtrics as `&marry=`. We then used display logic to show the question only when `marry=1`.

Finally, in order to generate a unique identifier for each respondent, we used R's system time function. The command `as.character(as.numeric(Sys.time()))` returns the number of seconds since 1970, including fractional seconds, with the level of precision determined by the operating system (5 decimal places for our server). Hence, we could generate a unique 15-digit code for each respondent with `as.character(as.numeric(Sys.time())*100000)`. This code was passed to the main survey, and, upon completion, to the raffle survey, where it was recorded.[7] As a result, we could eliminate anyone who entered the raffle via a "shortcut" process that did not begin with their visiting the Linode server and obtaining a unique code.

The opportunities for complex randomization and question wording customization go well beyond the specific examples outlined here. Qualtrics places no limits on the number of embedded data fields that can be passed in via the URL, and virtually all browsers can handle URLs of up to 2000 characters. One practical limitation concerns possible delays in executing the R script. Computationally-intensive tasks might generate long pauses between when the user hits the "submit" button on the web form and when she is redirected to the survey. The same is true when working

[7]To pass embedded data *out* of Qualtrics, one simply creates a customized End of Survey element in Survey Flow and includes the field in the redirect URL.

with large data files, since a new instance of R is started for each user and the data have to be loaded from scratch. That said, we found that rApache was remarkably fast, and the delay from executing our preprocessing tasks was virtually undetectable.

The major drawback to preprocessing survey respondents in rApache is that it significantly increases the number of moving parts in one's survey. We found that extensive testing was necessary in order to work out the kinks prior to launching, and we cannot rule out the possibility that some unknown problem arose for some respondents. Our ad click-to-completed survey ratio of 22:1 was much higher than the 7:1 figure obtained by Samuels and Zucco (2012*b*) for a similar survey in Brazil the previous year. We think the difference is largely attributable to our longer questionnaire and to people having less patience for surveys during election season, but it is possible that unknown bugs caused us to lose some respondents.

A final drawback is that it is necessary to "activate" (i.e., launch) the survey in Qualtrics in order to do testing on data read in via the URL. While responses generated during this testing period are easily discarded or separated from real data, they will count against any quota on one's account, in contrast to testing done via the "Preview" function.

Alternative Solutions

While combining rApache with Qualtrics worked well for our project, there are other solutions for implementing complex randomization schemes and database lookups in an online survey. A powerful and flexible solution would be to build a survey engine using a web framework such as Django, built using Python, or Ruby on Rails, built using Ruby.[8] These web frameworks employ full-fledged programming languages, which would allow for the customization required by many social science online surveys. Furthermore, they are designed to work with database backends, thus allowing for rapid data lookups and data collection. Packages specifically designed for constructing online surveys are available, such as Django-crowdsourcing (`https://pypi.python.org/pypi/django-crowdsourcing`) and Surveyor for Rails (`https://github.com/NUBIC/surveyor`). Furthermore, these popular frameworks have been used for websites with millions of users and thus have proven robust and scalable for a variety of tasks. The chief drawback to this approach is that web frameworks require a good deal of expertise to use effectively and thus can be "overkill" for the kinds of surveys political scientists wish to field.

Another approach would be a pure R-based solution that relies on rApache to administer the survey and collect response data. This approach is facilitated by the use of the "Brew" package (`http://cran.r-project.org/web/packages/brew/index.html`), which is a templating framework for mixing R code and text. Using Brew, one could write an HTML template with embedded R code. When users visit the web page, R dynamically generates the necessary customized HTML within the template and serves it to visitors. Of course, this solution would forgo the ease of use and reliability of commercial tools like Qualtrics. Other approaches to using R to dynamically generate web pages are covered in Verzani (2012).

Finally, a promising recent development is the release by the makers of RStudio of "Shiny" (`http://www.rstudio.com/shiny/`), an R package that allows for the fast and easy creation of web applications which can take advantage of R's statistical capabilities. The chief advantage of Shiny is that it does not require knowledge of HTML or JavaScript and is very easy to use. At present, it is primarily designed to serve HTML locally, but Shiny's developers have announced plans for a hosting service that would allow for the deployment of applications online.

Conclusion

Using rApache to augment the capacities of online survey engines such as Qualtrics takes advantage of R coding skills that many quantitative researchers already possess in abundance, and it requires only a minimal amount of additional skills, such as HTML coding or server administration, that are less common. Even for those with the programming ability to design and host their own surveys, integrating with an existing survey engine might be attractive due to their proven reliability and data security, as well as the fact that they are often available for free through university site licenses.

Our article has described applications of the rApache-plus-Qualtrics method that were specific to our own survey, but potential uses are much broader. Kriner and Shen (2012), for instance, use an online survey experiment conducted in Qualtrics to test the hypothesis that public support for war is more greatly affected by news of casualties that are local. In the treatment condition, respondents read a news story about the death in Afghanistan of a U.S. soldier from their home state, which they had specified earlier in the survey. In the control condition, a randomly chosen different state is specified. State, of course, only imperfectly operationalizes the concept of "local." An ideal strategy might be to ask for the respondent's ZIP code early in the survey and then describe the casualty as being from the corresponding county—a data lookup task that would be impossible within Qualtrics but is easily accomplished using our method. The effects of a "home county" treatment might be even larger than those (already sizable and significant) that Kriner and Shen (2012) found for causalities

[8] A promising set of tools for the creation and administration of online surveys is "Shanks," developed by Mark Fredrickson (`https://github.com/markmfredrickson/shanks`). These tools are written in Clojure and run on top of Google App Engine.

from one's home state.

Another potential application of our method involves assigning respondents to experimental conditions when one wants to end up with a specific number of completed surveys in each group—something that may be especially important for smaller-N survey experiments. In their $N = 60$ study, for instance, Zahedzadeh and Merolla (2012, 13) ended up with fewer respondents than desired in some experimental conditions because subjects dropped out after assignment but before completing the survey. Qualtrics is capable of ensuring that equal numbers of respondents are *assigned* to different conditions—using the "evenly present elements" option in the Randomizer—but it cannot take into account whether respondents in each condition actually complete the survey. Using our method, Qualtrics could communicate with the server running rApache upon survey completion, e.g., by loading one of two blank image files, `treat.jpg` or `control.jpg`, on the last page of the survey. Probabilities of assignment, which would be done in advance by rApache, could be adjusted each time by consulting the server log, which would record the number of times each image was loaded.

For our specific research on voting behavior and candidate evaluations in Brazil's 2012 local elections, preprocessing via rApache allowed us to substantially boost the external validity of a survey experiment by asking respondents about real candidates from their municipalities. It also permitted forms of randomization, such as choosing 1 out of 1000 candidate photos, that would have been exceedingly awkward to implement in Qualtrics. Hopefully other researchers can use the method we propose to combine the data-processing power of R with the reliability, data security, and graphical user interface of online survey engines.

References

Boas, Taylor C. 2013. "Vote for Pastor Paulo: Religious Ballot Names as Heuristics in Brazil." Paper presented at the Annual Meeting of the Southern Political Science Association, Orlando, FL, January 3–5.

Kamal, Mia, Jason Turcotte, Donyelle Davis and Christy Arrazattee. 2012. "An experiment in tolerance: How religious stereotypes shape attitudes of reciprocity and political engagement." Paper presented at the Annual Meeting of the Southern Political Science Association, New Orleans, January 12.

Kriner, Douglas L. and Francis X. Shen. 2012. "How Citizens Respond to Combat Casualties: The Differential Impact of Local Casualties on Support for the War in Afghanistan." *Public Opinion Quarterly* 76(4):761–770.

Leland, Eric. 2011. "A Few Good Online Survey Tools." `www.idealware.org`, February.

Morey, Alyssa C., William P. Eveland Jr. and Mylah J. Hutchens. 2012. "The 'Who' Matters: Types of Interpersonal Relationships and Avoidance of Political Disagreement." *Political Communication* 29(1):86–103.

Nyhan, Brendan and Jason Reifler. 2011. "Opening the Political Mind? The effects of self-affirmation and graphical information on factual misperceptions." Manuscript, Dartmouth College/Georgia State University.

Popescu, Marina and Gabor Toka. 2012. "How can mass media help citizens make sense of the political world? Media systems and citizens cognitive political engagement in Europe." Paper presented at the workshop "Advancing comparative political communication research: New frameworks, designs and data," ECPR Joint Sessions of Workshops, Antwerp, April.

Samuels, David and Cesar Zucco. 2012*a*. "The Power of Partisanship in Brazil: Evidence from Survey Experiments." Working paper, University of Minnesota/Rutgers University.

Samuels, David and Cesar Zucco. 2012*b*. "Using Facebook as a Subject Recruitment Tool for Survey-Experimental Research." Working paper, University of Minnesota/Rutgers University.

Sances, Michael W. 2012. "Is Money in Politics Harming Trust in Government? Evidence from Two Survey Experiments." Unpublished Manuscript, Massachusetts Institute of Technology.

Shineman, Victoria Anne. 2012. "Incentivizing Participation Increases Political Information: Evidence from a Randomized Field Experiment." Paper prepared for presentation at the Annual Meeting of the American Political Science Association, New Orleans, Aug. 30–Sept. 2.

Smith, Amy Erica. 2013. "ClericalWork: Clergy, Media, and Religious Polarization, Brazil 2008–2012." Paper presented at the Annual Meeting of the Southern Political Science Association, Orlando, FL, January 3–5.

Verzani, John. 2012. "gWidgetsWWW: Creating Interactive Web Pages within R." *Journal of Statistical Software* 49(10):112.

Young, Dannagal Goldthwaite and Lindsay H. Hoffman. 2009. "An experimental exploration of political knowledge acquisition from The Daily Show versus CNN student news." Paper prepared for presentation at the Annual Meeting of the American Political Science Association, Toronto, Aug. 30–Sept. 2.

Zahedzadeh, Giti and Jennifer Merolla. 2012. "How do negative political ads impact public trust in candidates?" Paper presented at the Annual Meeting of the Western Political Science Association, Portland, Oregon, March 22–24.