Eliciting process variables using scripts

Christian Peters
Mar 10, 2020

```
<script>
    var timeShown = 0;
        var timeDown;
        function show() {
            document.getElementById('matrix').style="display:inline;";
            timeDown = new Date().getTime();
        }
        function hide() {
            document.getElementById('matrix').style="display:none;";
            if (timeDown) {
                timeShown += (new Date().getTime() - timeDown);
                document.getElementById('timer_id').value = String(timeShown);
            }
            timeDown = null;
        }
       document.getElementById('image_button').addEventListener('mousedown', show);
       document.getElementById('image_button').addEventListener('mouseup', hide);
       document.getElementById('image_button').addEventListener('mouseleave', hide);
</script>
```

what happens to something else later. In other words, experiments are used to find out how a *cause* and *effect* are related. The 19th-century philosopher John Stuart Mill stated three necessary conditions for a causal relationship to occur: (i) the cause must preceed the effect, (ii) the cause must be related to the effect, and (iii) there is no plausible alternative explanation for the effect other than the cause. During my youth, I liked to play football and when I scored a goal it was clear to me that the goal (the effect) was caused by my shot (the cause). However, causal relationships are not always that clear. Often theories predict that a cause (X) is mediated or moderated (M) and this moderation or mediation leads to a certain effect (Y). In a recent paper, Asay, Guggenmos, Kadous, Koonce, and Libby (2019) posit that when the process is central to the research question, tests of process theories are indispensable. The authors mention mouse clicks, mouse movements, eye movements, items opened, time to select or read an item as unobtrusive measures that can be used to elicit process data.

Measuring To What Extent Participants Consult Information

So, suppose you want to experimentally examine how long participants consult particular information, such as the financial statements of a firm. One way to measure this would be to ask participants in the post-experimental questionnaire whether and to what extent a participant examined the financial statements. An advantage of this method is that measuring the process does not intervene with your causal relationship, as you measure the process after your causal relationship has been measured. A disadvantage of this method is that participants may have long forgotten this or may overreport or underreport this, bringing noise into your data. Another way to measure this would be to use the time spent on the page with the financial statements. However, this may also lead to noise as there may be other information on the page or the participant may have been distracted by something. In a recent experiment conducted by Bart Dierynck, Martin Jacob, Maximilian Müller, Victor van Pelt, and myself (2020) used a button that participants could hold to access financial information. We used a script to count the number of seconds during which the mouse-button was held by the participants to examine the financial information. In this post, I show how to use such a script to measure process variables. I use coding in JavaScript as most experimental software allows for this.

Programming a Simple Button

First, I will show how you can program a button that can be either hidden or shown. After that, I will gradually introduce extensions such as buttons that need to be held and scripts that count time.

We start by creating a button in HTML:

```
<div class="container">
    <button style="color: rgb(255,255,255); background-color: rgb(255,3,0); border-color: rgb(255,3,0)" type
        <div id="demo" class="collapse">
                <div align="center">
                        <p> Insert financial statements here... </p>
                </div>
    </div>
</div>
```

This code serves to make a button visible to participants on which they can click. In this case, once the button is clicked an example text is shown. Alternatively, you can insert images, widgets, and spreadsheets (see this post).

Click here to view the financial statements.

Now you have a working button, you may want to capture a dummy variable that captures whether participants clicked on the button or not. To do so, you first need to create an input variable where your (dichotomous) variable can be stored in HTML:

```
<input type="hidden" name="participant_clicked" value="" id="id_participant_clicked"/>
```

Next, you can write the following script in JavaScript:

```
<script>
    $('#name_of_button').click(function() {
        $('#id_participant_clicked').val(1);
    });
</script>
```

This function sets the variable *participant_clicked* to '1', once the button is clicked. It is important to make sure that the variable used in *('#name_of_button')* is identical to the 'id' in the HTML code. Also, *('#id_participant_clicked')* should be identical to the input variable you created in HTML. In this way, the created variable and the button are linked to each other by the script.

Measuring the Time Spent Consulting Information

Next, we extend the simple button in the previous section to a more advanced button that times how long participants consult information. To do so, we create a button that can be held and as long as the button is held, the time spent is registered. Holding the button can mimick a cost of effort and makes sure that (online) participants do not open the information and subsequently iron their shirt before continuing with the experiment, which would introduce noise into the process variable. To make such a button, we first need to create another variable *timer_id* to store the number of milliseconds spent holding the button:

```
<input type="hidden" name="timer_id" value="" id="timer_id"/>
```

Next, we program the timer in JavaScript:

```
<script>
    var timeShown = 0;
        var timeDown;
        function show() {
           document.getElementById('matrix').style="display:inline;";
           timeDown = new Date().getTime();
        }
        function hide() {
           document.getElementById('matrix').style="display:none;";
           if (timeDown) {
               timeShown += (new Date().getTime() - timeDown);
               document.getElementById('timer_id').value = String(timeShown);
           }
           timeDown = null;
        }
       document.getElementById('part2_button').addEventListener('mousedown', show);
       document.getElementById('part2_button').addEventListener('mouseup', hide);
       document.getElementById('part2_button').addEventListener('mouseleave', hide);
</script>
```

Here, *timeShown* starts with '0', after which the *timeDown* is captured by the *.getTime* function whenever the content is shown. When the participant stops holding the button, the amount of milliseconds which was stored in *timeDown* is stored in *timeShown*. If the participant subsequently presses the button again *timeDown* starts to count from zero again and the amount is cumulated in *timeShown*. Afterwards, the cumulative amount in *timeShown* is transferred to the variable you created: *timer_id*. The last three lines of the script are important as they define when the show and hide function should run, which depend on whether the mouse is down or up.

As you can see in the JavaScript code, the code refers to an element named "matrix". Hence, in order to determine what needs to be shown when the button is held, you need to add the following to the information in the button environment:

```
<p id="matrix" style="display:none;"> Financial statements here... </p>"
```

The identification *matrix* is important to link the script to the information in the button and the *display:none;* is important as you do not want to show the information by default. Below, you can see an example of a button that reveals information on holding the button and counts the number of milliseconds that you hold the button.

Keep pressing to show financial report

Applications

With this code you are able to elicit process variables related to the time spent examining certain information. The way in which the process data is elicited is unobtrusive and this allows the experimenter to carefully collect process data without interferring in the causal relationship and bringing in noise into the data by measuring ex-post. I hope this guide helps fellow researchers to enhance tools to elicit process data. For any questions or comments, you can use the comment box below this post or feel free to send me an e-mail.

References

Asay, H. S., Guggenmos, R., Kadous, K., Koonce, L., & Libby, R. (2019). Theory Testing and Process Evidence in Accounting Experiments. *Available at SSRN 3485844*.

Acknowledgements

I thank Victor van Pelt and Farah Arshad for their indispensable help with the programming of this code.

oTree experiments surveys software elicitation process variables dependent variable time spent

Christian Peters
PhD Researcher

Login

Add a comment

MARKDOWN ADD COMMENT

Related

- Embedding excel spreadsheets in your experiment
- Sliders with feedback and without anchoring
- Four amazing oTree apps