Volkswirtschaftliches Seminar
Tobias Werner, MSc.

Heinrich Heine Universität Düsseldorf

Winter term 2020/2021

MW86 Seminar

Programming Experiments in oTree: Problem set 1

1 Problem set

1.1 Another survey

1.1.1 Basic version

Consider a new survey app. We want to ask the participants for their age, their gender, and their last math grade. Additionally, we want to elicit their time preferences. The app should consist of two pages. On the first page, you ask the participants the survey questions. On the next page, you show them their answers.
The survey questions should have the following properties:

- Age (Minimum 18 years, Maximum 99 years)
- Gender (Choices between Male, Female, Other and "Don't want to say")
- Math Grade (German Scale 1,0 - 6,0; It should be optional to answer this question)
- To measure the time preferences use the question by Falk et al. (2018) "How willing are you to give up something that is beneficial for you today in order to benefit more from that in the future?"
  - Use a 11-point Likert scale to elicit this question
  - Hint: Check the RadioSelect widget in oTree
- Add labels to each question!

To program the app, you may want to follow those steps:

1. Create a build-plan for the app, in which you list all things you need to include in models.py, pages.py, and which template you will need. Follow the build-plan template provided in the lecture.

2. Implement your app following the same procedure as discussed in the lecture using your build plan. Afterward, register the app in settings.py and run it.

1.1.2 Revise and resubmit

We want to give our participants the option to revise their answers. For this purpose, we want to modify the current version of the app. After the participants answered the questions, we still want to show the participants the answers on the following page. However, on this results page, we want to add an additional question asking if the answers to the previous questions are correct. The default answer should be "Yes", but the participant may change it. If the participant answers with yes, the experiment ends. If they answer with no, we want to ask them all the main questions again such that they can change their answers. After they answered the question for the potential second time, the experiment ends and they cannot revise them again.

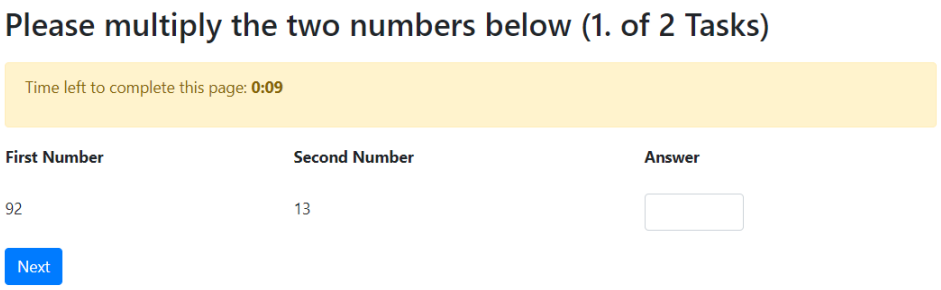
1.2 Real effort task

1.2.1 Getting started

In this task, we want to program a Multiplication Task. Variations of this real effort task have been used in various versions and contexts. Versions range from adding sets of five two-digit numbers (Niederle and Vesterlund, 2007; Riener and Wiederhold, 2016), to solving simple mathematical equations (Sutter and Weck-Hannemann, 2003), to multiplying a one-digit and a two-digit number (e.g. Dohmen and Falk, 2011).
In the version we consider, participants are asked to multiply two two-digit numbers without the help of a calculator or smartphone. For every correct calculation, the participant gets one point. One point is worth *0.40 EU*. The task will consist of multiple multiplication tasks. It should be possible to change the exact number of tasks (the number of rounds) easily by changing a constant in models.py. The two numbers which each participant has to multiply are supposed to be randomly generated.

Some further notes/hints:

- Use the create_session()-method in the Subsession class to draw the random integer numbers for each round and each player. Here you can also already calculate the correct answer for each multiplication problem. Also read up on using random numbers in oTree Here
- You will need a method in the Player class to check if the answer the participant entered is correct.
- We want to keep track of the number of correct multiplication problems by saving the points to the build-in payoff field.
- Use multiple rounds and restrict yourself to two pages (Multiplication and Results). The Results page should only be shown at the very end of the real effort task.
- You can use the HTML table-tag in the template to organize the information on the Multiplication page. You can read up on it Here.
- The number of correctly answered multiplication tasks should be shown on the final page (Hint: Use may have to use the player method, in_all_rounds() to calculate this value).
- Also show the final payoff on the Results Page.



1.2.2 Adding a timeout

We want to add a timeout of 60 seconds to each multiplication problem. Read up on how to do this Here.
Some things to note:

- We want to record it in the database if the timeout occurred. You will need a new field in the Player class.
- The answer of the participant should be marked as false, if a timeout happened.

1.2.3 Adding treatment variations

We want to add a treatment variation to this real effort task, in which we deduce one point for every wrong answer (COST-Treatment). Both treatments, COST and COSTLESS should be balanced across participants in the session. Include this new treatment to the existing app. You may find this information helpful: Here and Here.
Some things to note:

- For COST treatment add on the results page also the number of incorrectly answered tasks. This information should not be visible for the COSTLESS treatment. (Hint)
- The final payoff for the participant should be set to zero if he would make a loss elsewise.