Snaplab

The Ultimatum Game

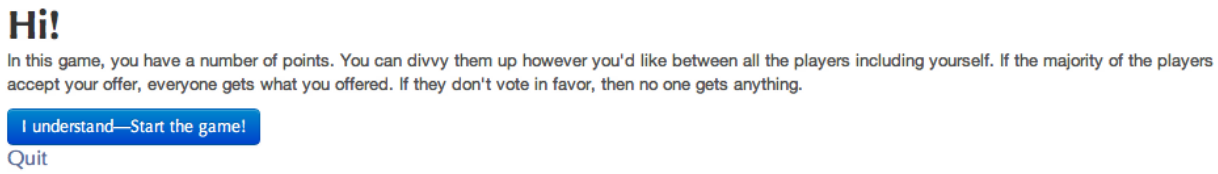
In the classic Ultimatum Game, one player is given a certain amount of money that they can split between them and one other player as they'd like. If the other player accepts this offer, then both players get the amount of money promised. If the other player rejects this offer, then neither player gets any money. If both players were rational actors, then the giving player would give all of the money to themselves save for one unit to be given to the other player. The other player would accept this offer because even 1 unit is better than none. However, the results of this game, as you can guess, are much different in practice, with many players from many countries dividing the money differently and rejecting differently. This result has puzzled economists, sociologists, and psychologists for years.

With the introduction of the Snaplab platform, experimenters can now get much more and better data about subjects and their moves in experiments. It is the aim of this project to extend the Ultimatum game to this framework, both to replicate the results of the classic Ultimatum game in an online setting, and to increase the number of players of the game to see how this influences both giving and receiving players' behaviors.

The Game

Quite simply, the Ultimatum game is a many-player extension to the classic Ultimatum game. What follows is a description of how the game works.

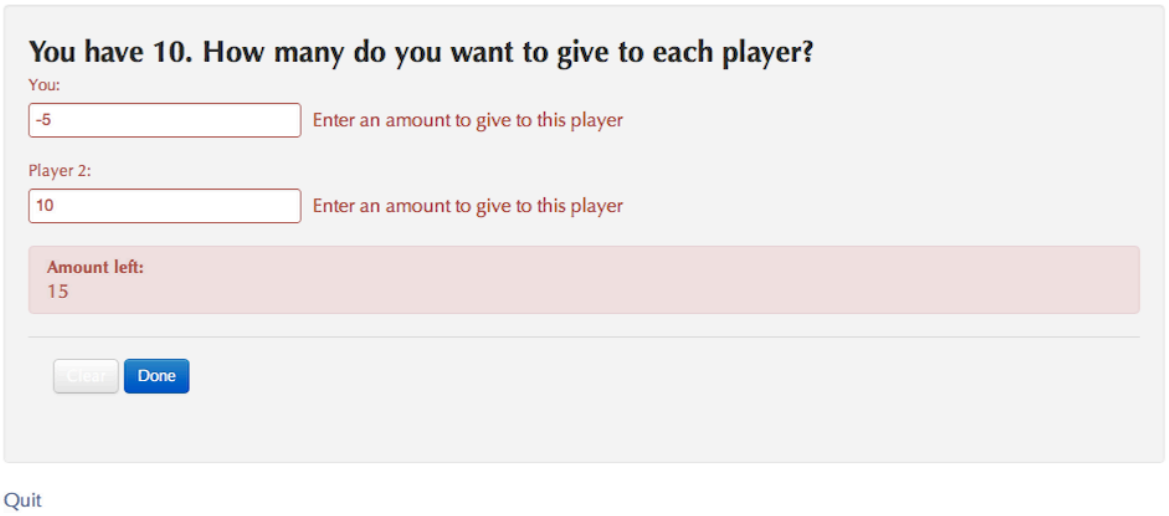
One player is chosen randomly to be the giving player ("the giver") and all other players are marked as receiving players ("the receivers"). Both the giver and the receivers are given a screen of instructions explaining how the game works. The giver has a button they must press in order to advance to the next screen while the receivers have a message saying that the giver is currently choosing divvying points.



The instruction round for the giver

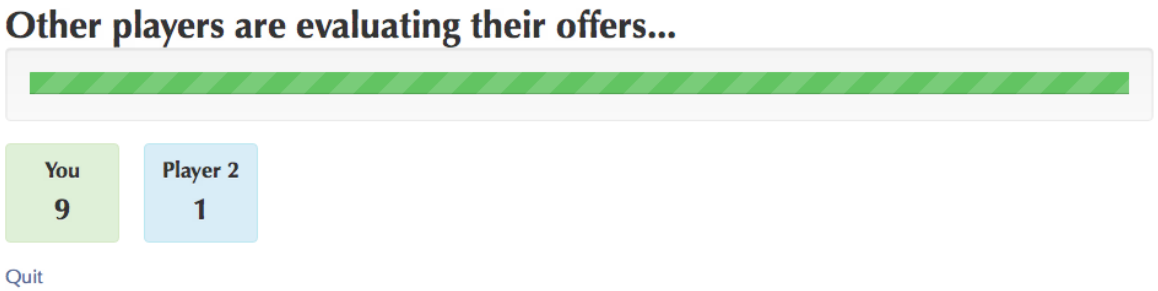
The instruction round for the receiver

Once the giver hits the button, the giver receives a form with the total number of points available to them, text boxes for each player's amount to be received, a clear button, and a submit button.



The giving round

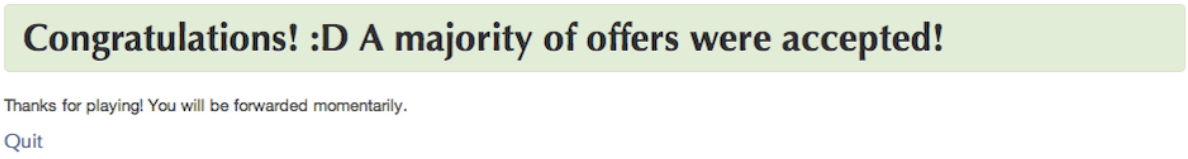
The giving player then must enter a positive integer for each player that sums up to the total amount for the game. If the player enters something that is not a number or is a negative number, then that player's input controls will appear in red. If the giver enters in points that sum to something that is not equal to the number of total points given to the player, then the "Amount left" area at the bottom appears in red. Once the giver enters valid amounts, they click the "Done" button to take them and the receiving players to the receiving round.



The giver view of the receiving round

The receiver view of the receiving round

The giver view changes to a read-only view of other user activity with a message that other players are deciding on their offers. The receiver view now changes for the first time. They have a form with an accept and reject button. Once they hit one of these buttons, their screen is no longer editable. Once all receivers have accepted or rejected, the game ends.



A winning results screen

A losing results screen

For the game to work, the experimenter must create and set two variables, *total_amount* and *percent_needed*. *total_amount* is the number of points initially given to the giver, while *percent_needed* is the percentage calculated by (number of accepting players / total number of players) over which a game must be for the players to win. For example, in a two-player game, if the percentage was .5 and the receiver rejected, then the calculated percentage would be 1 accepting player divided by 2 total players. Since this is not over the .5 percent needed (it is equal), no players receive any points.

To see the code for this game and the revisions in the development process, please visit https://github.com/Glench/Snaplab-Ultimatum.

Future Work

There are several features to this game that I could not finish due to time constraints. In the

interests of documentation for possible future work, I will list them here in no particular order:

- Add the ability for players to play against an experimenter-configured robot. If an experimenter wanted to see how players would react in a certain situation, the experimenter could program a robot player to contrive that situation in each game. Having robotic players would also be useful if there were not enough players available at a given time to make a full game.
- Add the ability to handle when another player drops out of the game. When this happens, either from voluntary quitting or from something like a player's internet connection dropping in the middle of a game, it would be nice if both this game and the Snaplab framework could add functions for gracefully handling these cases. The game could exit, or perhaps a robotic player could take over.
- Add the ability for players to receive real money for this experiment to up the stakes of the game.
- Procure data from many different cultures. One of the most interesting aspects of the Ultimatum game is the differing results across different cultures. With a large base of Snaplab users from across the world, it would be much easier for experimenters to gather cultural data on a large scale.
- Add configurable graphical parameters. A well-known phenomenon in web interface design is the ability for the design of the interface to influence a user's behavior. This game is no different. In order to control for this phenomenon, it might be useful for experimenters to vary graphical elements on the experiment, as well as add certain suggestions, such as a "Split Evenly" button to automatically divvy up the points evenly. Would the mere presence of that button change user behavior?
- Add a configurable variable for players to see other players' offers. Being able to see what other players received as offers might instill a sense of jealousy, betrayal, or favoritism in players, and thus change their decision.
- Add a configurable variable for players to see other players' decisions before they make their decision. Being able to see what other people in the game are deciding might change the player's decision. For example, if I see that everyone else had already accepted their offers, will I still accept mine?
- One challenge exists in trying to abstract this game in order for many different types of games to use this as a template. It is unclear at this juncture what other types of games would fit well within this paradigm of one user entering values and other players reacting to those values. This remains an area of exploration.
- Add time limit to rounds as a configurable option. Perhaps seeing a time limit will change when people make decisions and how they make decisions.
- Add the ability to repeat the game between two players to see how it works round after round.
- Change the colors of the boxes to indicate in real-time what decisions other players are making.
- Add a possible default amount to the giving player to see how that changes their behavior (if they will actually change it or tend to leave it where it is).

Making The Game

While making the Ultimatum game for Snaplab, I had the chance to make improvements to the framework in order to allow me to finish the project. I will document those things briefly here.

To begin, I documented the existing Snaplab experiment JavaScript API. As it stood before I

came to the project, no documentation save for a few small examples showed how experiment developers were to integrate their games with the Snaplab framework. Learning the API through making the Ultimatum game, I contributed back to the Snaplab framework by adding a page under the documentation section called *API Reference*. This page is a listing of all the JavaScript functions, variables, and dependencies that Snaplab provides experiment developers. This page alone should explain what the developer must provide at minimum in order to work within the provided framework, as well as other features to facilitate their development. This will significantly reduce time to familiarity with Snaplab and make it easy for developers to make experiments.

Next, I added to the Snaplab API in a very minor way. Specifically, the *submit* function provided in the Snaplab experiment page used to take as a parameter just the value to be submitted. I added the ability for the developer to add an optional second parameter, which is a function to be called when submitting a player's move has finished. This is useful to developers that need must submit moves in a certain order, and so must be notified when a player's move is finished completing in order to call another one. This is what I did in the Ultimatum game.

Finally, I added a very important feature and security enhancement to Snaplab. In short, I escaped user-defined templates so they would not be interpreted by the Django template renderer. This is useful to developers so that they can define their code however they want using any characters they want and their template will still work. This is also a security enhancement because developers can no longer run arbitrary Django template code in their experiments. This will be a very important feature as the framework is exposed to more and more people.