Exercise: Solving for Market Equilibria

Summary

This exercise analyzes the impact of a sales tax on a market with 1000 buyers. Each buyer has a demand equation of the form below, where Qi is household i's quantity demanded, Ai and Bi are demand parameters for household i, Mi is household i's income, and prd is the buyer price of the good. In case you're interested, this is a demand equation derived from a Stone-Geary utility function.

```
Qi = (Ai*Mi)/prd - Bi
```

The market supply of the good is given by the equation below, where Qs is total market supply and prs is the seller price of the good:

```
Qs = 5000*prs
```

Finally, the buyer and seller prices are related by a tax as shown below. Note that the tax may be zero.

```
prs = prd - tax
```

The exercise focuses on analyzing the impact of a policy that raises the tax from its base case value of 0 to a proposed policy case value of \$5.

Input Data

All of the input data is contained in the file **households.csv**. It includes information about 1000 households, each on one line of the file. Using Python's zero-based subscript convention, the fields are as follows: [0] an identification number for the household, [1] the household's demographic type, [2] the household's income, Mi, [3] parameter Ai in the household's demand equation, and [4] parameter Bi in the demand equation. The demographic type is reserved for use in a subsequent assignment and is not used in this exercise.

Deliverables

A script called **market.py** that computes the market equilibrium in both the base case (equilibrium 1) and policy case where the tax is set to \$5 (equilibrium 2), and then evaluates a couple of aspects of the policy.

Instructions

Please prepare a script called market.py that does each of the following steps:

- 1. Imports csv
- 2. Imports scipy.optimize as opt.
- 3. Defines a function called read_households() that takes filename as an argument. Use str as the type hint for filename and list as the type hint for the function's return value. The function should:
 - 1. Create an empty list called households.
 - 2. Open filename for reading using fh as the file handle.
 - 3. Create an object called reader to read the file by calling csv.DictReader() with fh as its argument.
 - 4. Use a for loop to iterate over reader using hh (short for household) as the loop variable. The loop should use float() calls to make the values stored under the following three keys numeric: inc, a, and b. The loop should then append hh to households.

- 5. After all lines in the file have been read the function should print a message saying "Lines read:" and the value of len(households). If all has gone well, the length should be 1000. After the print statement the function should return households.
- 4. Defines a function called <code>ind_demand()</code> that takes two arguments: <code>prd</code> (the buyer price, use hint <code>float</code>) and <code>hhlist</code> (a list of households, hint <code>list</code>). It should return a <code>list</code> (remember to include the hint) consisting of the quantities demanded by each of the households.
 - Build the list in the same way you've built several previous lists. Start with an empty list and then use variable <a href="https://ht
- 5. Defines a function called mkt_demand() that takes the same two arguments as ind_demand(): prd
 (hint float) and hhlist (hint list). The hint for the return value should be float. The first line should compute a list of individual quantities, qlist, by using ind_demand(). The next line should use the sum() function to add up the demand values in qlist. The function should then return the sum.
- 6. Defines a function called mkt_supply() that takes one argument, prs, the seller price (hint float), and returns the market supply (hint float) computed using the equation above.
- 7. Defines a function called <code>excess_d()</code> that takes three arguments: <code>prd</code> (hint <code>float</code>), <code>tax</code> (hint <code>float</code>), and <code>hhlist</code> (hint <code>list</code>) and returns the excess demand at that price (hint <code>float</code>). The excess demand is the difference between the total demand and total supply in the market. The first line of the function should compute <code>prs</code> using the accounting rule above. The second and third lines should compute the market demand, <code>qd</code>, and maket supply, <code>qs</code>, using <code>mkt_demand()</code> and <code>mkt_supply()</code>. The function should then return the difference: <code>qd-qs</code>.
- 8. After all the functions have been defined, use read_households() to read the input file into a variable called hhlist.
- 9. Create a variable called guess for the initial guess of the price prd and set it to 20.
- 10. Solve for the base case (equilibrium 1) by doing the following:
 - 1. Set variable tax to 0.
 - 2. Calculate the initial value of the buyer price, prd1, by calling opt.newton() using excess_d as the first argument, guess as the second argument, maxiter=20 as the third argument, and args=[tax,hhlist] as the fourth argument.
 - 3. Calculate the initial market quantity, qd1, by calling mkt_demand() using prd1 and hhlist as the arguments.
 - 4. Print a message giving prd1 and qd1. Include some text to indicate which equilibrium is being printed.
- 11. Then solve for the policy case (equilibrium 2) by doing the following:
 - 1. Set variable tax to 5.
 - 2. Calculate the policy-case value of prd, prd2, using opt.newton() again. The call will be the same as before except the tax will now be 5 instead of 0.
 - 3. Calculate the new market equilibrium by calling mkt_demand().
 - 4. Print a message giving prd2 and qd2. As before, include some descriptive text.

- 12. Calculate and print total tax revenue under the policy case.
- 13. Calculate and report the percentages of the tax burden that fall on the buyers as a group and the seller. Round the percentages to integers so that, for example, 12.345% is printed as 12%.

Submitting

Once you're happy with everything and have committed all of the changes to your local repository, please push the changes to GitHub. At that point, you're done: you have submitted your answer.

Tips

- This is a type of analysis known as "microsimulation". It allows very fine-grained analysis of the impacts of policies. In this case, the analysis could pick up differences across demographic groups, or across income deciles, or both. We'll do that in a subsequent exercise.
- You may want to check each equilibrium by computing mkt_supply() to make sure it's equal to the market demand.