Consumers' model

This model is taken from the paper: Smallwood and Conlisk, 1979, *Product Quality in Markets where Consumers are Imperfectly Informed*, Quarterly Journal of Economics, 93-1.

Consumers are, in the most relevant markets, no expert about the quality of the products they buy. However, people try to find some clues about which is the best product. A typical rule is to trust more products that are more common among other consumers.

The model we are going to implement is a model where products have different qualities, unknown to the consumers. Consumers retain their currently owned product if they are satisfied, or buy a new one if not. The rule to choose a product, *when they need to buy a new one*, is random with probabilities proportional to the current market shares.

Consumers' model

Let's see the main equations. Consumers keep the current product if it keeps on working, and choose a new product if the product is broken:

$$\text{Current Product} = \begin{cases} \text{Current product} & \text{, if it is not broken} \\ \text{New product} & \text{, if breaks down} \end{cases}$$

We consider that each product when used has a probability of "breaking down", meaning it does not satisfy any longer the consumer.

$$\text{Broken} = \begin{cases} Yes & \text{with probability} = BD \\ No & \text{with probability} = 1 - BD \end{cases}$$

Consumers' model

When a consumer needs to buy a product, she does not know the probabilities, otherwise the lowest probability product will be chosen. But the consumer can observe the other consumers' products. We assume that the consumer can choose any of the products currently available, with probabilities proportional to the *shares* of consumers.

$$Prob(i) = \frac{ms_i^\alpha}{\sum_{j=1}^n ms_j^\alpha}$$

For the time being, we ignore the parameter α, assuming it to be equal to 1.

Consumers' model

Let's start a new model as usual. Use **LMM / Browse Models** to create a new model called **Consumer Model** in directly **SC**. Firstly, we consider the model structure, without implementing it, just to "place" the equations in the different objects. Then we look at all the equations, understanding what they do. After the implementation of the equations we will compile and create the model configuration.

Consumers' model

The model consider a group of firms and a group of consumers. At each time step consumers have a currently used product. Firm register the number of consumers using their own products.
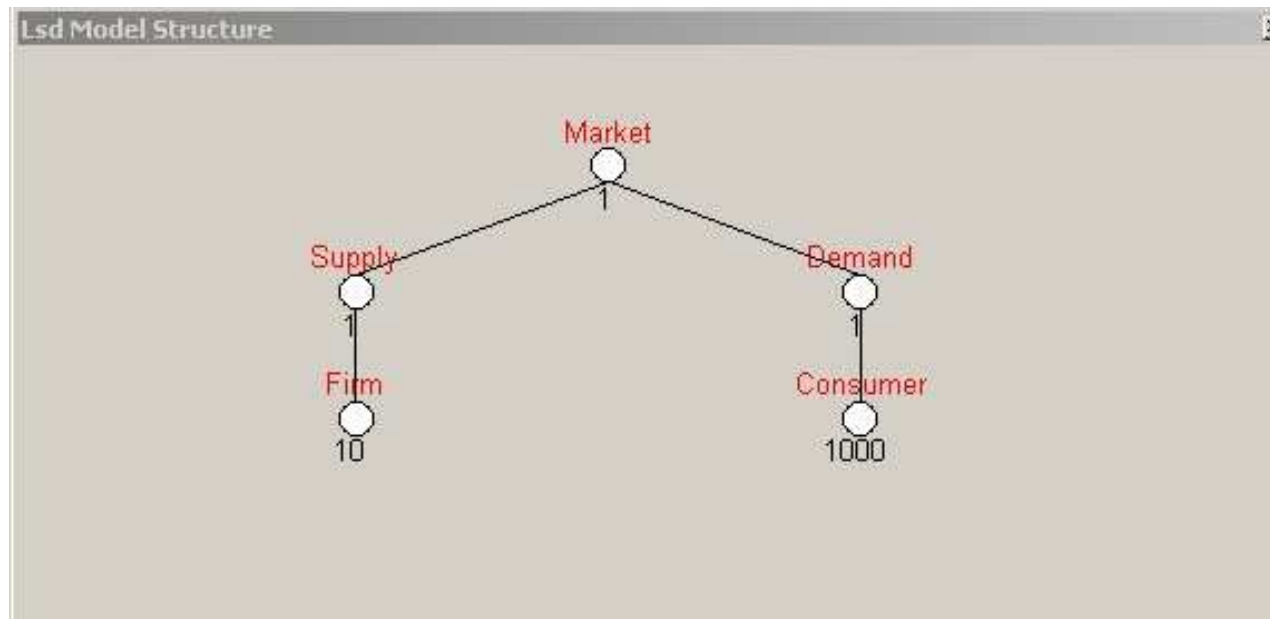
Consumers' model

We consider that each firm has two parameters **Sales** and **NumLost** that during a period contain the number of new consumers choosing the firm's product and the number of lost consumers who switched to other products.

Sales and **NumLost** cannot be variables, because their values depend on the choices of consumers, while their must be placed in objects **Firm**. We need to ensure that during each time step other variables correctly update **Sales** and **NumLost** as required.

Consumers' model

We will need to ensure that both parameters are set to 0 when a step of a simulation begins, particularly before the first consumer has the chance of making a decision. Then, we need that any time a consumer drops a product and buys a new one, the firms involved have their parameters correctly updated. Finally, we will need to ensure that all consumers complete their decisions, so that **Sales** and **NumLost** can be considered as definitive for the time step.

We will use one variable **InitTrade** whose only goal is to ensure that **Sales** and **NumLost** are set to 0 before consumers start to make their purchases. And another variable **EndTrade** that ensures firms that they can safely use **Sales** and **NumLost** after all consumers did make a decision.

Consumers' model

After this variable has been computed we are sure that each firm has parameters **Sales** and **NumLost** set to 0.

```
EQUATION("InitTrade")
/*
Initialize the trading period
*/
CYCLE(cur, "Firm")
 { //for all firms set to 0 Sales and NumLost
   WRITES(cur,"Sales",0);
   WRITES(cur,"NumLost",0);
 }

RESULT(1 )
```

Consumers' model

Ensure that all consumers did their purchases, if this is the case.

```
EQUATION("EndTrade")
/*
Sum all the consumers, providing their number.

It actually serves to guarantee that all consumers
have made up their mind, making their purchasing decisions.
*/
CYCLE(cur, "Consumer")
 VS(cur,"ProdUsed");

RESULT( 1)
```

Consumers' model

Variable **ProdUsed** indicates the product used by the consumer, which is a parameter called **IdFirm** different for each firm.

```
EQUATION("ProdUsed")
/*
Determine the product used
*/
v[0]=V("IsBroken"); //breaks ?
if(v[0]==1)
  v[1]=V("Purchase"); //yes, buy a new produc
else
  v[1]=VL("ProdUsed",1); //no, keep on using the previous one
RESULT(v[1])
```

Consumers' model

We will use a **function** to compute the compute whether a product fails or not, **IsBroken**.

In Lsd a function is identical to an equation associated to a variable. The difference is that variables associated to an equation are computed always one and only once at each time step. Instead, functions are computed only and any time they are requested. Therefore, a function can never be computed during a time step, or can be computed many time.

Consumers' model

The variable **IsBroken** *knows* the particular consumer that requested its value, which in the code is expressed with the pointer `c->`. The function returns 1 if the product breaks down and 0 otherwise.

This function serves also other two purposes. Firstly, it fix the problem of the very beginning of the simulation. At time 0 no consumer is using any product, so everybody needs to make a purchase. The code must therefore return the order to make a purchase in this case.

Secondly, we need to register the number of consumers who switched product. Any firm must know the number of lost consumers. If the product is broken, then this function tells the firm that one of its customer switched.

Consumers' model

```
EQUATION("IsBroken") //beware, it is a function
/*
Look whether the product breaks down or not.
The object c-> is the consumer using the product
*/
v[0]=VLS(c,"ProdUsed",1);
if(v[0]==0) // at time 1 no consumer has a product
  v[1]=1;   //therefore choose to buy
else
 {  cur=SEARCH_CND("IdFirm",v[0]);
    v[2]=VS(cur,"BD");
    if(RND<v[2])
     {v[1]=1; //product broken
      INCRS(cur,"NumLost",1);
      }
    else
     v[1]=0; //product not broken
 }
RESULT(v[1] )
```

Consumers' model

Also the variable **Purchase** is a function, computed any time (and only) it is requested by an unsatisfied consumer.

```
EQUATION("Purchase")
/*
Make a purchase for the calling object (supposedly a consumer).
RNDDRAW("Obj", "VarOrPar") is a Lsd function choosing randomly an
object called "Obj" probability equal to "VarOrPar".
*/
V("InitTrade"); //ensure that firms are ready to sell
cur=RNDDRAW("Firm","Visibility");
INCRS(cur,"Sales",1); //increase the Sales of the chosen firm
v[0]=VS(cur,"IdFirm");

RESULT(v[0] )
```

Consumers' model

Variable **Visibility** is simply the ms^α. Note that it used the past market shares, because they cannot be updated before consumers make up their mind.

```
EQUATION("Visibility")
/*
The apparent quality of a firm is computed as the past market
share raised to the power of alpha.
Shares are lagged because the new ones are not yet ready
to be computed before consumers finish to buy.
*/
v[0]=V("alpha");
v[1]=VL("ms_user",1);


RESULT( pow(v[1],v[0]))
```

Consumers' model

The market shares **ms_user** is the share of users using the firm's product

```
EQUATION("ms_user")
/*
Market shares of users, computed as the ratio of this users over the sum of
*/
v[0]=V("TotalUsers");
v[1]=V("NumUsers");

RESULT(v[1]/v[0] )
```

Consumers' model

TotalUsers is the sum of all users.

```
EQUATION("TotalUsers")
/*
Total number of users
*/
v[0]=0;
CYCLE(cur, "Firm")
  v[0]+=VS(cur,"NumUsers");


RESULT(v[0] )
```

Consumers' model

NumUsers is the current number of users owning the product of the firm. We compute this variable as the update of previous number of users plus the new users (**Sales**) minus the discontent users computed in **NumLost**.

```
EQUATION("NumUsers")
/*
Number of users, computed, after the end of the trading period,
by summing to the previous users the new sales and
removing the lost users
*/
V("EndTrade"); //ensure that buyers finished to update their product
v[0]=VL("NumUsers",1);
v[1]=V("Sales");
v[2]=V("NumLost");
RESULT(v[0]+v[1]-v[2] )
```

Consumers' model

Now we have the equations. Implement them and fix the possible compilation errors.

When the Lsd model program runs, let's create the model structure, composed by:

- Object **Market**.

- Items **InitTrade(0)** and **EndTrade(0)** in object **Market**.

- Object **Supply** in object **Market**.

- Items **Purchase(0)**, **TotalUsers(0)**, **IsBroken(0)** and **alpha(P)** in object **Supply**.

- Object **Firm** in object **Supply**.

- (continue ...)

Consumers' model

- (... continue)

- Items **IdFirm(P)**, **BD(P)**, **NumUsers(1)**, **ms_users(1)**, **Visibility(0)**, **Sales(P)** and **NumLost(P)** in object **Firm**.

- Object **Demand** in object **Market**.

- Object **Consumer** in object **Demand**.

- Item **ProdUsed(1)** in object **Consumer**.

Consumers' model

Let's initialize the model. Consider 10 **Firm**'s and 1000 **Consumers**. Set the following values:

- **alpha** $= 1$;

- **IdFirm** $=$ increasing values from 1 step 1

- **BD** $=$ increasing values from 0.1 step 0.01

- **ms_user** $= 0.1$

- **ProdUsed** $= 0$

Consumers' model

Use the following settings:

- Num. Simulations = 1;

- Initial Seed = 1

- Simulation Step = 500

- Insert debugger at = 0

- Mark the **Save** option for **ms_user**, **NumUsers**, **Sales**, **Lost**:

Consumers' model

Do you have any expectation on what will happen? And why?

Run the simulation and investigate the model's behaviour. Test different seeds and different values for **alpha**.