

CMS

C OMMODITY M ARKETS S IMULATOR

USERS MANUAL

Gianfranco Giulioni

February 22, 2017

Contents

List of variables	ii
List of Aggregate Parameters	ii
List of individual Parameters	ii
1 Overview	1
1.1 Market organization	1
1.2 Market Participants	2
1.3 Dynamics	2
2 The loop	3
2.1 Producers decide if allow/forbid export	3
2.2 Buyers with an associated producer decide if allow/forbid import	3
2.3 Buyers update buying strategy	4
2.4 Perform market sessions	8
2.5 Buyers account consumption	9
2.6 Producer Produce	10
3 Installation	10
3.1 Java Development Kit (JDK)	10
3.2 Repast Symphony (RS)	11
3.3 CMS	11
3.3.1 Using git	12
3.3.2 Using a zip archive	13
4 Testing the Installation	13
5 Customizing the simulation	15
5.1 Creation and configuration of agents	15
5.1.1 Producers	15
5.1.2 Buyers	16
5.1.3 Markets	17
5.2 Output	18
5.2.1 Adding a GIS display	18
5.2.2 Adding charts and data sinks	23
6 Parameter setting	27
7 Running in BATCH mode	28

List of variables

- $B_{b,t}$ Quantity of resources bought by buyer b in period t . 8
- C_b^d Desired Consumption of buyer b . 7, 13
- $C_{b,t}$ Consumption of resources by buyer b in period t . 8
- I_b^d Desired Inventories of buyer b . 8, 13
- $I_{b,t}$ Inventories of buyer b at beginning of period t . 8
- $Y_{b,0}$ producer's p average production. 11, 12
- Y_p production realized by producer p . 11
- $\#ms_{b,t}$ number of market session buyer b can participate in time step t . 8
- $\bar{D}_{b,ms,t}$ Intercept of the ms market session demand curve of buyer b in period t . 5

List of Aggregate Parameters

- P_0 Global production at initialization. 7, 11, 12
- ι Share of desired consumption to compute desired inventories. 8
- τ Production cycle length. 7, 12
- τ_{exp} buyers interval in deciding export policy. 4
- τ_{imp} buyers interval in deciding import policy. 5
- pr_{exp} probability that a producer allow export when the export policy is updated. 5
- pr_{imp} probability that a buyer allow import when the import policy is updated. 4
- y boundary of the production percentage change. 11

List of individual Parameters

- $\bar{D}_{b,0}$ Intercept of buyer b the demand curves in the first simulation time step. [5](#), [13](#)
- d_b Buyer's b slope of the demand curves. [5](#), [13](#)
- $s_{b,0}$ Buyer's b market share at initialization. [7](#), [13](#)
- $s_{p,0}$ Producer's p share of global production at initialization. [11](#), [12](#)

1 Overview

The CMS has three types of agents: producers, buyers and markets. These agents' common feature is that each of them has a geographic location given by a latitude and a longitude. In the most straightforward interpretation, Producers can be thought of as sovereign countries, but it is possible to setup the model thinking at different geographical scales such as continents, macro areas, or regions of a country. For convenience of exposition we will identify hereafter producers with countries.

The model is fully customizable, however, we start describing a common setup for exposition convenience. The setup is as follows:

- a producer has at least one associated buyer; in other words if a country produces it also uses the resource;
- a buyer is not necessarily associated to a producer; in other words the resource can be used by countries that does not produce it;
- the number of markets and their geographic location is independent from the number and location of producers and buyers.

Because we think the first two listed features straightforward, we will now focus on the third one.

1.1 Market organization

Considering nowadays information and communication technologies, we model markets as (virtual) places where producers and buyers send information. More trivially, resources are not physically moved to the market by the producer and, once sold, moved again from the market to the buyer. As it commonly happens, buyers and sellers send their will to the market. The market uses this information to reach an agreement. Once an agreement is reached, the resources are directly moved from the seller to the buyer place.

In this context, markets geographic location does not affect producers and sellers behavior. It has a role only when the model has more than one market: opening order is set according to their latitude.

Markets are organized in sessions. Each market session is associated to a producer. A producer can have only one session in a market, and must participate in at least one market. This organization allows buyers who bid in a given session to know who is the producer. The producer geographic location has an important role here because it informs buyers on where the resource is stored. Because we assume buyers bear the transport costs, the proposed markets organization allows buyers to compute such costs and account for them when submitting their bids.

1.2 Market Participants

A producer can always decide to sell exclusively to its associated buyer. In the real world this happens when a country forbids export. Similarly, buyers who have an associated producer can decide to buy exclusively from their producer (a producer country can forbid import). The latter, is not possible if the considered buyer has not an associated producer (a non producer country does not forbid import). Summing up, in each market session participants

- the producers associated with the session;
- buyers associated with the producer;
- buyers not associated with the producer if the two following conditions are both satisfied:
 - the producer allows export;
 - the buyer allows import

1.3 Dynamics

The “cornerstone” of the dynamics is the simulation time step. In each simulation time step, several actions can happen, however, what mostly characterizes it is that all the market sessions are performed. This provides a link between real and simulated time: if we want to simulate a real world situation where markets operate once a day (week, month and so on), a simulation time step represents a day (a week, a month and so on). Starting from this observation we can comment on the other simulation events. Consider for example the dynamics of the resource inventories. Straightforwardly, at each time step, each buyer’s inventories are increased by the quantity bought in all the market sessions it participates, while each producer’s inventories are decreased by the amount sold in the market sessions it is associated. Knowing the time scale is important to model the opposite flow. For buyers, the opposite flow to purchases is consumption. Therefore, if a simulation step represents a day, we have to take into account the daily consumption. Modeling the opposite flow, is more tricky for agriculture producers. The opposite flow to sales is the production flow. Agriculture products have not a continuous production, so we cannot compute a daily, weekly or monthly produced quantity as we can do for crude oil for example. Agriculture production is in general harvested once a year. Our simulator accounts for this: it gives the possibility to adapt the frequency of the production flow to the model time scale. Consider for example a situation where a time step represents a month and the production cycle is a year. In this case, during the setup phase the researcher can (and must) choose to increase inventories by the obtained production every 12 simulated time steps.

We report hereafter the sequence of events that can happen in a simulation time step. It integrates and organizes the elements given above:

1. producers decide if allow/forbid export

2. buyers with an associated producer decide if allow/forbid import
3. buyers update buying strategy
4. perform market sessions
 - market 1
 - session 1
 - * producer sends supply curve
 - * buyers send demand curves
 - * demand curves are aggregated
 - * market price and quantity are determined
 - * buyers increase inventories by the quantity bought in this session
 - * producer decrease inventories by the quantity sold in this session
 - all the other sessions (if exist, perform same actions listed for session 1)
 - all other markets (if exist, perform same actions listed for market 1)
5. buyers account consumption
6. producers produce

Following this list, we will now detail the simulation loop.

2 The loop

2.1 Producers decide if allow/forbid export

Each producer has a boolean variable named `exportAllowed`.

This event consists in updating this variable with a true/false value.

At the present state, this is updated randomly. The probability that import is allowed (pr_{imp}) is a parameter of the model. The researcher has to modify the `stepExportAllowedFlag()` method of the `Producer` class to model producers' export behavior.

The update frequency (τ_{exp}) can be changed setting the `exportPolicyDecisionInterval` parameter as explained in section 6.

2.2 Buyers with an associated producer decide if allow/forbid import

Each buyer has a boolean variable named `importAllowed`.

This event consists in updating this variable with a true/false value.

At the present state, this is updated randomly. The probability that import is allowed (pr_{exp}) is a parameter of the model. The researcher has to modify the

`stepImportAllowedFlag()` method of the `Buyer` class to model buyers' export behavior.

The update frequency (τ_{imp}) can be changed setting the `exportPolicyDecisionInterval` parameter as explained in section 6.

2.3 Buyers update buying strategy

The buying strategy is updated each simulation time step.

Updating the buying strategy is an elaborate action. It is especially because buyers have to fulfill their needs looking for cheapest opportunities in a changing environment. The most relevant change in buyers' environment is represented by the possible switch in producers export policy. In fact, buyers cannot continue buying in sessions associated to those producers who change their policy forbidding export. In these cases, buyers attempt to gather the quantities bought in these sessions in other sessions. On the other hand, new opportunities open when producers switch their policy allowing export.

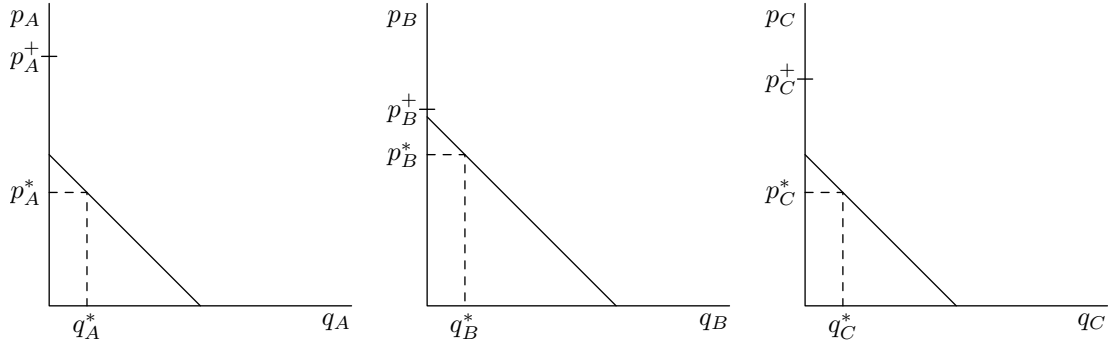
In the present version of the model, the following factors are managed during the buying strategy update:

1. reducing the unit cost;
2. allocating quantity bought in market sessions that are now not available;
3. formulating demand in market sessions that are now available;
4. obtaining the desired quantity.

To understand how these tasks are implemented we specify that at each time step, each buyer send to the available sessions a demand curve. In the current version of the model demand curves are assumed to be linear. The slope (d_b) can be different for each buyer (see section 5), but it is the same for the same buyer in the various market sessions. The buyer updates the buying strategy by managing the intercepts of the demand curves s/he will send to the various market sessions ($\bar{D}_{b,ms,t}$). The initial level of the demand intercepts is a parameter ($\bar{D}_{b,0}$) (see section 5). Summarizing, the demand in market session ms formulated by buyer b at time t is:

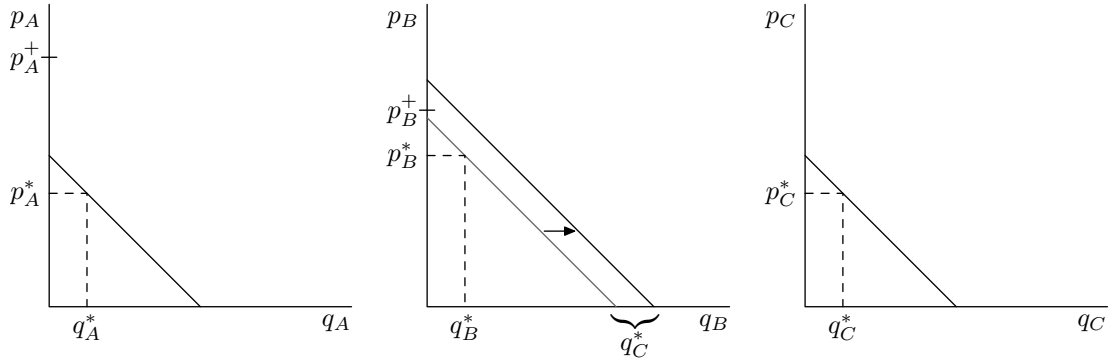
$$D_{b,ms,t} = \bar{D}_{b,ms,t} - d_b p_{mc}$$

Consider the following example where a buyer participated in three market sessions (labeled A , B and C) in the previous period.

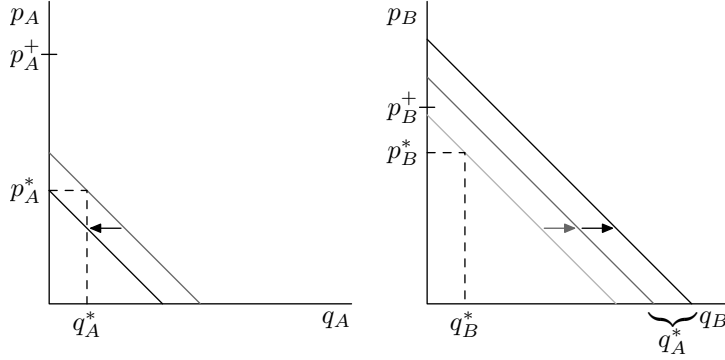


The charts show that the market prices were $p_B^* > p_A^* = p_C^*$ and that the buyer bought the quantities q_A^* , q_B^* , q_C^* and the total quantity.

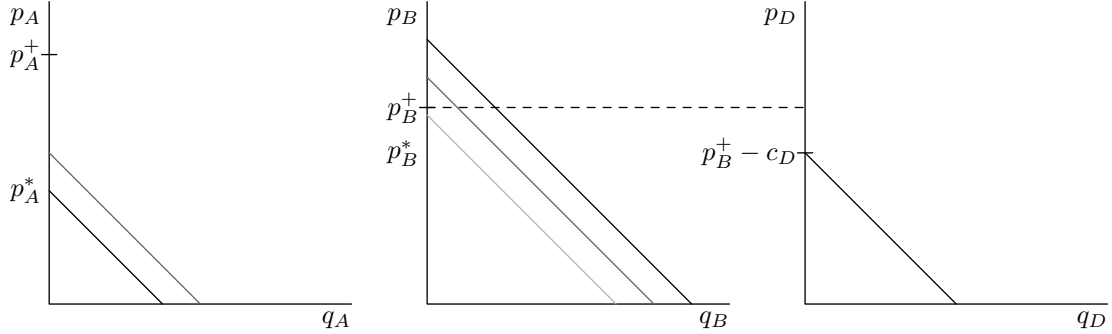
Consider however, that the transport cost (c) must be added to the market price (p^*) to obtain the unit cost of the commodity. It is this cost, that we denote with p^+ , that is used by the buyer to rank market sessions. Suppose now that the buyer we are considering and the producer selling in market session C are from different countries and the latter forbids exports. So, the buyer have to gather the quantity q_C^* in other available market sessions. The assumption is that the buyer performs this attempt in the market session with the lowest p^+ . Looking at the chart above, we see that $p_B^+ < p_A^+$. So, the demand curve in market session B is shifted to the left by q_C^* as happens in the following charts.



This account for factors 1 and 2 listed above. However, there will be no movements in the demand curves if the buyer can continue participating in all markets sessions. The attempt to reduce the unit cost (factor 1 in the list) is lost in this case. To allow for factor 1 even when there are no export/import policy changes, we introduce the following device. The buyer moves demand from the most expensive market session (A in our example) to the cheapest one (B). The two charts below shows the leftward shift of the demand curve in market session A and the rightward shift of the curve in market session B .



Next, consider the case in which the buyer can participate in a new market session named D . The problem now is how to formulate the demand curve for this new session (item 3 in the list of factors a buyer account for). The idea is that the buyer is willing to buy in this new market only if the unit cost will be lower than the lowest unit cost observed in the previous period. This allows to set the demand curve in the new market session as in the rightmost chart of the following figure.



The last movement of the demand curves before sending them to the market sessions is done considering the gap between the desired level of inventories and the level of inventories under the assumption that the desired consumption could be achieved (item 4 in the list of factors a buyer account for). The mechanism is described hereafter.

The reasoning starts from the definition of Buyers' desired consumption (C_b^d). It is set at the beginning of the simulation as follows

$$C_b^d = \frac{s_{b,0} P_0}{\tau}$$

where $s_{b,0}$ is the buyer market share, P_0 is the global production at initialization, and τ is the production cycle length. The division by τ deserves a comment. While the buying strategy is updated each simulation time step, the global

production relates to the production cycle length, this motivates the division by τ . An example could help clarify. Consider a situation where a simulation time step represents a month and the production is realized yearly. P_0 is the yearly global production and $s_{b,0}P_0$ is the amount bought yearly by the buyer. Since this event manages the monthly demand, we have to divide by 12 (which is the τ in this example).

Buyers also has a desired level of inventories (I_b^d), which is a fraction ι of desired consumption

$$I_b^d = C_b^d \iota$$

At a given simulation step, a buyer starts with a level of inventory $I_{b,t}$.

Consider now the buyer buys an amount $B_{b,t}$ in the considered time step.

Now, it is possible to compute buyers effective consumption ($C_{b,t}$).

If $I_{b,t} + B_{b,t} \geq C_b^d$, then $C_{b,t} = C_b^d$; otherwise $C_{b,t} = I_{b,t} + B_{b,t}$.

The buyer enters the new period with an inventory level equal to

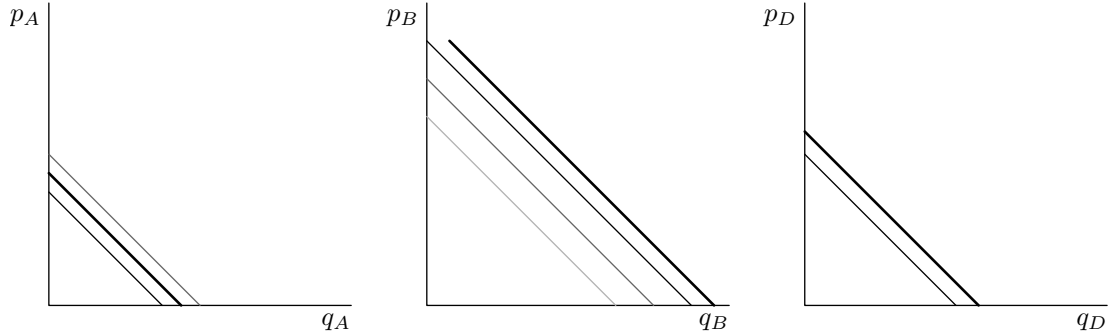
$$I_{b,t+1} = I_{b,t} + B_{b,t} - C_{b,t}$$

Note, that the buyer can also compute the quantity

$$\tilde{I}_{b,t+1} = I_{b,t} + B_{b,t} - C_{b,t}^d$$

All the demand curves are shifted by $\frac{I_b^d - \tilde{I}_{b,t+1}}{\#ms_{b,t+1}}$, where $\#ms_{b,t+1}$ denotes the number of market session the buyer will participate.

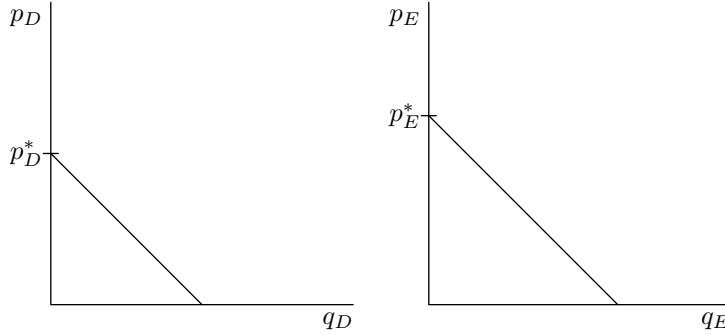
The following charts display the case in which the demanded quantities are increased because the level of observed inventories is lower than those desired. In fact, all the demand curve have a rightward shift.



Demand curves shifts are opposite when inventories exceed the desired level.

Because there are Buyers without an associated producer, it may happen that there are no available session because all producers forbid exports. In these cases, if none of the producers switch the export policy, no demand curve is set, otherwise demand curves for newly available sessions are set. The idea is similar to that presented above, however, the buyer has not the benchmark of the cheapest unit cost at which s/he bought in the latest time step. However,

the latest prices observed in the newly available market sessions are known. The current version of the model first sets the demand curve in each market in such a way that the demanded quantity is zero at the latest observed market price as displayed in the following charts.



Then, the demand functions are moved as explained above according to the $\frac{I_b^d - \tilde{I}_{b,t+1}}{\#ms_{b,t+1}}$ quantity.

Changes to the basic behaviors presented in this paragraph can be done modifying the method `stepBuyingStrategy` of the `Buyer` class.

2.4 Perform market sessions

We will now describe the functioning of a market session. The program will perform the following actions for all the market sessions.

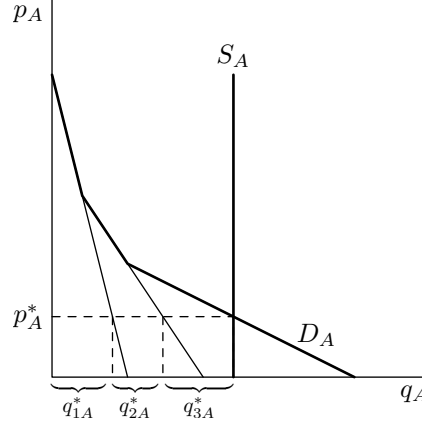
First of all, we recall that in a market session, the exchanged items come from a given producer. This is basically because using the producer's geographic location, buyers can compute transport costs that are used to update the buying strategy as explained above.

We have already discussed how buyers set their demand curves, now we need to specify how the producer sets the supply curve. In the present version of the model, the easiest option of a vertical supply curve is adopted. In other words, the supplied quantity is independent of price. Despite this simplification, managing the supply policy is tricky when accounting for production that are not realized at every simulation time step and/or for producers who participates in more than one market session in each time step. Even in these cases, the simplest solution is adopted. At the beginning of each market session, the producer checks the level of the resource stock and divides it equally among the market sessions to come before the production is realized.

Because in a market session there is one seller (producer), its supply curve represents the whole session supply curve. Differently, we can have more than one buyer attending a session (this is usually the case except when the seller forbids exports). When we have two or more buyers, the demand curves they send to the market are aggregated by summing them horizontally to obtain the

session demand curve.

Now, using the session demand and supply curves, the market price and exchanged quantity are computed. The quantity bought by each buyer is obtained using the market price and the individual demand. The following chart, which focuses on market session A , can help understand.



Bold lines are the market session supply (S_A) and demand (D_A) curves. For exposition convenience it is assumed that there are three buyers in this market. The thin black lines keep track of the horizontal sum of the individual demand curves. The intersection point between the session demand and supply curves (bold lines) determine the market price (p_A^*) and the total exchanged quantity (q_A^*). The quantity bought by each buyer (q_{1A}^* , q_{2A}^* , q_{3A}^*) are also reported. Obviously $q_A^* = q_{1A}^* + q_{2A}^* + q_{3A}^*$.

At the end of the session agents update their inventories level. For buyers we have:

$$I_b = I_b + q_{bA}$$

while for the producer

$$I_p = I_p - q_A^*$$

This updating is performed at the end of each market session, so, in a time step, the level of inventories is updated by each agent as many time as the number of session it participates.

It can be useful to highlight the link with the notation used in section 2.3: $B_{b,t}$ is the sum of all the quantity bought by buyer b in all the market sessions it participates in a given time step.

2.5 Buyers account consumption

This action is performed to account for the consumption of resources occurred during the time step. Buyers' inventories are newly updated:

$$I_b = I_b - C_b$$

Being consumption a continuous phenomenon, this update is performed every time step.

2.6 Producer Produce

This action is performed to account for the production of resources. Producers' inventories are newly updated:

$$I_p = I_p + Y_p$$

where Y_p is the production realized in a period. In the present version of the model it is modeled as a white noise:

$$Y_p = Y_{b,0}(1 + u)$$

where $Y_{b,0}$ is seller's b average production computed as $s_{p,0}P_0$, and u is the realization of a uniform random variable: $u \sim U(-y, y)$.

Differently from buyers, this update is not necessarily performed at each time step. As mentioned above, there are commodities whose production is not continuous. For those commodities, if the time step represents a shorter time interval than the production one, this update is performed at regular interval.

3 Installation

In this section we describe the standard installation process needed to prepare the system to run simulations. After taking the steps described below, the user should be able to run the model regardless of the operative system s/he is using.

The model needs Repast Symphony (RS), who in turn needs the Java Development Kit (JDK). Therefore we need first to check if the JDK is installed in the system and install it if needed. Once JDK is properly running, we have to install RS. Finally the model can be installed and run in RS.

3.1 Java Development Kit (JDK)

Check the list of installed software to know if JDK is installed in your system. If yes, note the JDK version. Alternatively, you can open the command line interface of your system, type `javac -version` and hit the return key.

Once verified if JDK is installed and, if yes, its version, visit the following URL:

<http://www.oracle.com/technetwork/java/javase/downloads/index.html> to know which is latest released version of JDK.

If JDK is not installed in your system or if it is not at the latest release, follow the instruction found in the JDK download page to install or upgrade it. You can also search the internet for alternative ways to install or upgrade the JDK on your system.

This installation phase is complete when the `javac -version` command returns what you expect. If not, you should first check if the folder containing

the JDK executables are in your execution path and add it manually if needed. Furthermore, some Linux distribution must be informed on which JDK to use using the `update-alternatives` command.

3.2 Repast Symphony (RS)

The Repast suite website: <http://repast.sourceforge.net> has information needed to download and install RS.

Note that RS is provided as a plugin of the eclipse Integrated Development Environment. The Repast development team provides a customized version of eclipse, so you could encounter problems with already installed versions of eclipse.

3.3 CMS

CMS has to be installed as an eclipse RS project. We will give hereafter the instructions to achieve this goal.

First of all, open the eclipse downloaded and installed as described in the previous section.

Suppose your workspace has the following path:

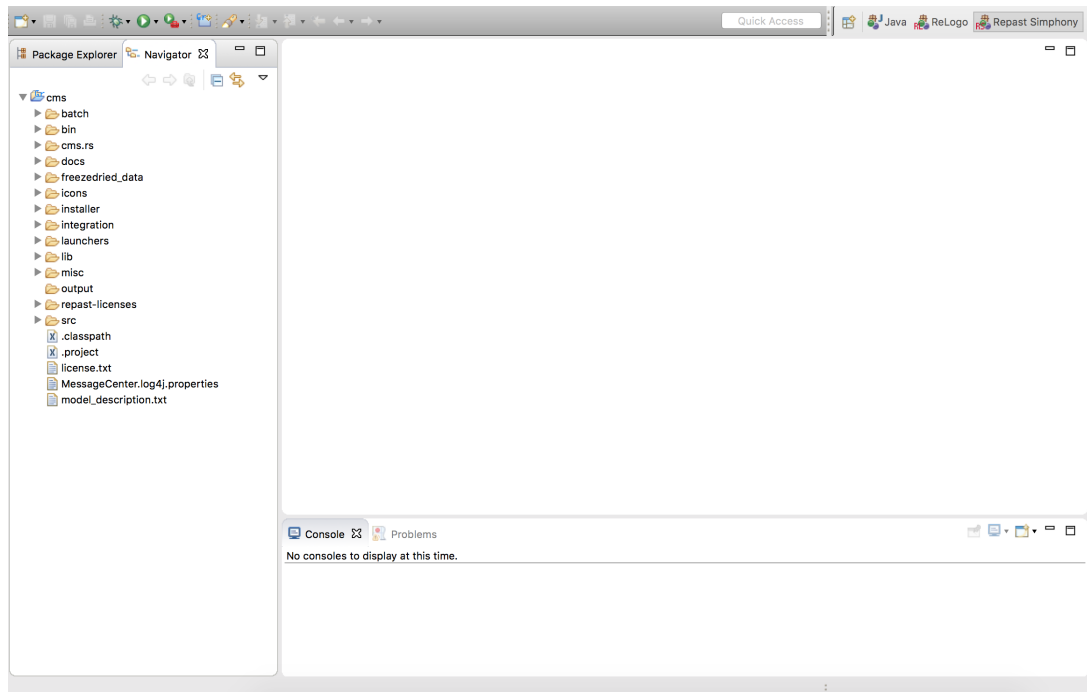
`/Users/coolcoder/Documents/workspace`

Open the RS perspective. To do that click Window → Perspective → open perspective → other and choose Repast Symphony in the window that appears.

Create a new RS project called `cms` (file → new → Repast Symphony project)

This creates the `cms` folder and a series of sub folders inside the workspace:

The following figure show the `cms` rs project folders tree.



Now, the `cms` files have to be added to the just created RS project folders tree.

We give here two alternatives: via git and using a zipped archive.

3.3.1 Using git

As you probably know, this is a largely used way to share code developments.

So, let us show how to fetch the `csm` files via git. To do that, you must have a git client installed in your system. Many system comes with a git client already installed, otherwise you have to install it. Mac and windows users can consider to install the GitHub Desktop software.

You can verify if git is installed in your system by checking if your command line interface recognize the `git` command. If your check is successful, change directory to the `cms` project folder:

```
cd /Users/coolcoder/Documents/workspace/cms
```

and type the following commands:

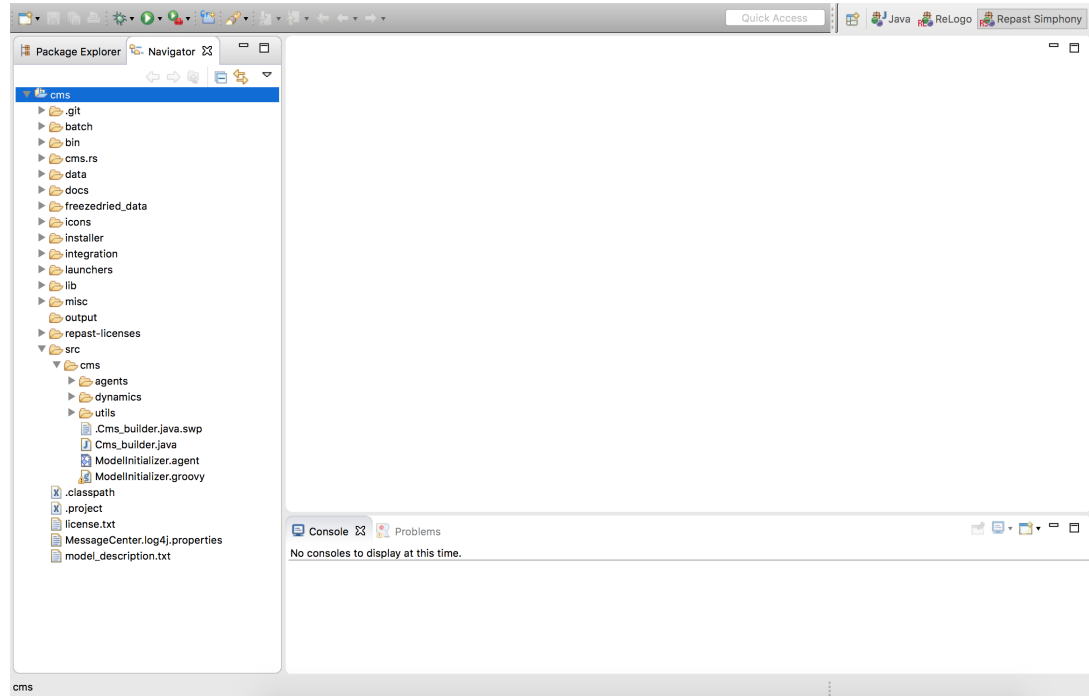
```
git init
git remote add origin https://github.com/ggiulion/cms.git
git fetch origin master
git reset --hard FETCH_HEAD
```

Then, if you plan to pull or push updates, add the command:

```
git push --set-upstream origin master
```


Now, the `cms` files should show up in the RS project folders. Refresh the `cms` RS project with the navigation tab selected in the side bar (file → refresh) to make them visible in eclipse.

The following figure show how the `src` sub folders should look like.



3.3.2 Using a zip archive

Point your browser to

<https://github.com/ggiulion/cms>

Click on the “clone or download” button and choose “download zip”.

This will download the `cms-master.zip` file in your system.

Unpacking it creates the `cms-master` folder. Move the whole content of this folder in the `cms` RS project folder:

`/Users/coolcoder/Documents/workspace/cms/`

Choose to overwrite existing files and folders if you will be asked (this will merge folders).

Now refresh eclipse (file → refresh).

4 Testing the Installation

There are two ways of running RS models: The GUI and the BATCH mode.

The GUI mode can be of great visual impact because several monitoring devices continuously updating during runtime can be added to the RS GUI

window. The flip side of the coin is that these devices slow down simulation execution. Second, the simulation runs exclusively in a machine running an X server. Notwithstanding the GUI mode can be a valid tool during the model development, however when massive simulations are performed, the BATCH mode should be used instead. BATCH mode runs are faster because all the graphics elements are turned off. Furthermore, the absence of graphics makes it possible to run the model in parallel on several machines. It is worth saying that RS has very useful facilities for running a model in parallel.



Let us start this section from the very beginning i.e. with the eclipse software just opened and show how to run the model in GUI mode. We discuss the BATCH mode after the discussion on how to customize the model.

First of all, click on the down black arrow highlighted by the red circle in the following figure



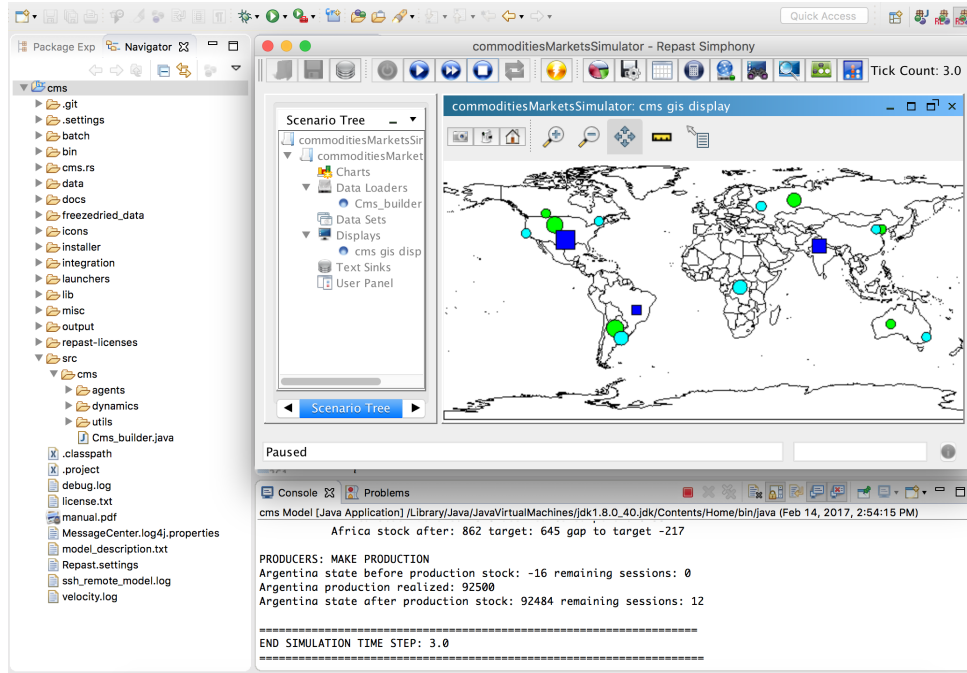
After clicking, a menu opens as shown by the following figure.



The GUI execution is activated by choosing the `cms Model` item and after few seconds the model window will show up. Clicking the  button, the model is initialized and the first simulation time step is performed. The initialization phase also builds the graphical elements. Click the  button to center the map in the gis display and you should see something similar to the next figure in your screen.

Use the RS GUI window intuitive buttons to interact with the simulation. A more detailed description on how to control the simulation is given at page 24 of the “Repast Java Getting Started” document available in RS web site.

Now the model has no dynamic monitoring graphical elements, so one can check the progress of the simulation watching at the tick count number in upper right corner of the RS GUI window or looking at the text output in the console panel of Eclipse window as shown in the following figure.



5 Customizing the simulation

5.1 Creation and configuration of agents

As explained above, the model is populated by three types of agents: producers, buyers and markets.

The agents creation and configuration process is guided by three configuration files located in the folder **data** folder. They are named after three agent types and have a comma separated values format. They are: **producers.csv**, **buyers.csv** and **markets.csv**.

Each line of these files (except the first one which is for headings) contains the information to setup an agent. The researcher can thus configure the simulation by writing these files.

We report hereafter examples of these files to show the fields needed in each of them.

5.1.1 Producers

An example of the **producers.csv** file is the following

```
name,latitude,longitude,production share,markets,products,first production time
China,39.9390731,120.1172706,0.05,New Delhi,Product A,2
United States of America,43.01,-104.08,0.2,Dallas|Brasilia,Product A,1
Argentina,-27.9878842,-62.6300825,0.3,Brasilia,Product A,3
```

```
Russian Federation,60.02,60.04,0.15,New Delhi,Product A,4
Canada,50.8725518,-110.1561254,0.05,Dallas,Product A,6
Australia,-24.9872587,126.1857131,0.05,New Delhi,Product A,8
```

After loading this file, the simulator creates six producers. Note that the last line of the file must be empty. As one can understand from headings (first line), each line contains the following information:

- the producer's name,
- the producer's latitude,
- the producer's longitude,
- the producer's production share $s_{p,0}$. Together with the global production P_0 , is used to compute the producer's average production $Y_{b,0}$,
- the markets where the producer sells. The | character is used when more than one market is specified (see for example the United States configuration line). The program creates a session for the producer in each specified market,
- the products obtained by the producer. Even in this case, the | can be used to signal that the producer obtains different products. However, in case the researcher is interested in setting up a model with more than one product, s/he has to modify the code to model agents' choice concerning the supply and demand of these different products,
- the first production time variable. It is introduced to account for production that are obtained in different seasons across producers. Together with the τ parameter, this enable the simulator to handle non-continuous production processes such as those of agriculture products. As an example we can have a yearly production cycle for each producer, but some producer obtain the product in June while other in December.

5.1.2 Buyers

An example of the `buyers.csv` file is the following

```
name,latitude,longitude,demand share,demand curve intercept,demand curve slope
China,39.9390731,116.1172706,0.1,5000,500
United States of America,37.1957928,-123.7650303,0.10,5000,500
Argentina,-34.6155729,-58.5033604,0.2,5000,500
Russian Federation,55.7498598,37.3523163,0.15,5000,500
Canada,45.5581968,-73.8516467,0.1,5000,500
Australia,-33.8474027,150.6517756,0.1,5000,500
Africa,0.1659234,22.938695,0.10,5000,500
```

After loading this file, the simulator creates seven buyers. Note that the last line of the file must be empty. As one can understand from headings (first line), each line contains the following information:

- the buyer's name. This is matched against the first column of the `producers.csv` file to find out if the buyer is associated to a producer or not. According to our assumption, all the producers have an associated buyer, but the reverse is not true. Therefore, the number of lines in the `buyers.csv` file equals or is higher than those of the `producers.csv` file. In the case of the configuration file reported above, the first six lines specify buyers associated to producers. The seventh line (`Africa`) has not an associated producer. This buyer will never forbid import of the commodity during simulation runs.
- the buyer's latitude,
- the buyer's longitude,
- the buyer's demand share $s_{b,0}$. This is used to setup the desired consumption (C_b^d) and the desired level of inventories (I_b^d) as explained above. The level of inventories at the beginning ($I_{b,0}$) is set at the desired level, then they evolve as explained above.
- the intercept of the demand curves ($\bar{D}_{b,0}$). This is used to setup the initial level of the intercept in each market session. As explained above, the intercepts evolve during the simulation.
- the slope of the demand curves (d_b). This remains the same in all the market sessions during simulation runs.

5.1.3 Markets

An example of the `markets.csv` file is the following

```
City,latitude,longitude,exchanges share
Brasilia,-15.6308631,-47.9994128,0.1
Dallas,32.8209296,-97.0115313,0.2
New Delhi,28.5275198,77.0688989,0.15
```

After loading this file, the simulator creates three markets. Note that the last line of the file must be empty. As one can understand from headings (first line), each line contains the following information:

- the market's name. This is matched against the fifth column of the `producers.csv` file when market sessions are created.
- the market's latitude,

- the market's longitude. This is used to determine the order of market opening in case the model has two or more markets.
- the market's exchange share. This is used for graphical uses.

5.2 Output

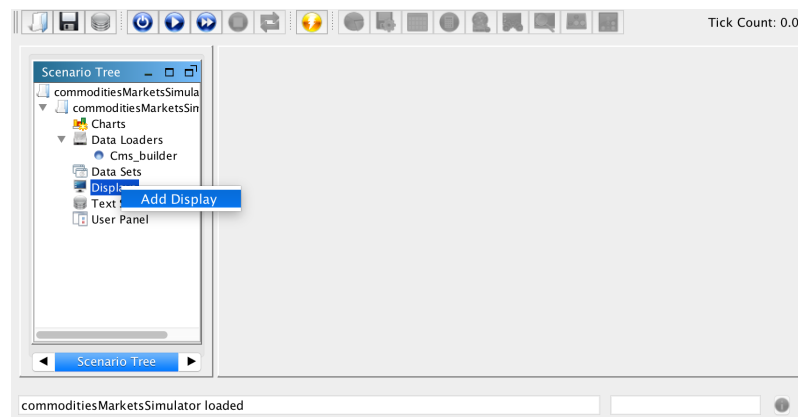
This subsection shows how to add graphical elements to the model interface or to make the model able to save data.

5.2.1 Adding a GIS display

The models comes with a gis display showing a world map. We will first delete it and make it again in order to show how the user can configure her/his own display.

Right click the `cms gis display` in the scenario tree and choose delete to remove the display. Now we will build again the display.

Right click the Display item in the Scenario tree and click on Add display as shown in the following figure



a wizard opens. Edit the display name, chose the GIS option and move the Geography item from the left to the right

Display Details
Please enter the name and type of the display as well the projections the display should visualize

General
Name: cms gis display
Type: GIS

Projections and Value Layers

Geography

Previous Next Finish Cancel

Click the next button and move the agents to be displayed from the left to the right

Agent Selection
Please select the agent types to display and the order in which the layers (2D) will appear

Foreground

Cms_builder
Cms_scheduler
Contract
ContractComparator
DemandFunctionParameters
ElementOfSupplyOrDemandCurve
ElementOfSupplyOrDemandCurveComparator
MarketLongitudeComparator
MarketSession

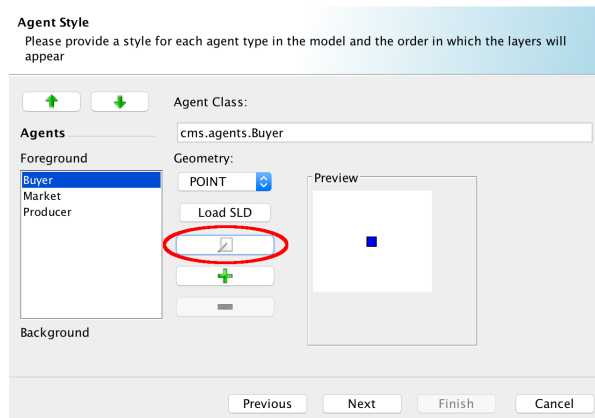
Buyer
Market
Producer

Background

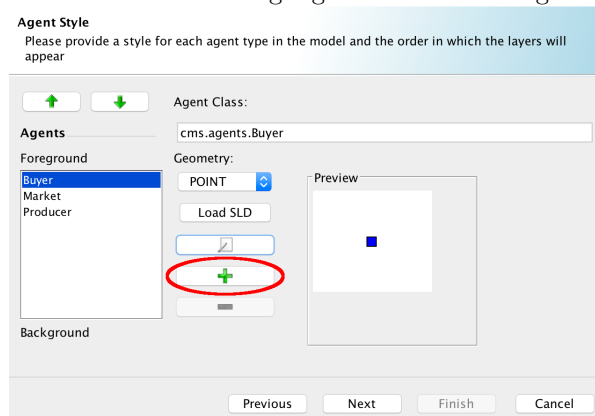
Previous Next Finish Cancel

By clicking the next button a new window opens. It gives the possibility to customize the appearance of each type of agents on the display and to add additional elements.

To edit the appearance of an agent, select an agent type and click on the edit icon which is highlighted in the next figure (in the case of the figure the appearance of Buyers will be configured)



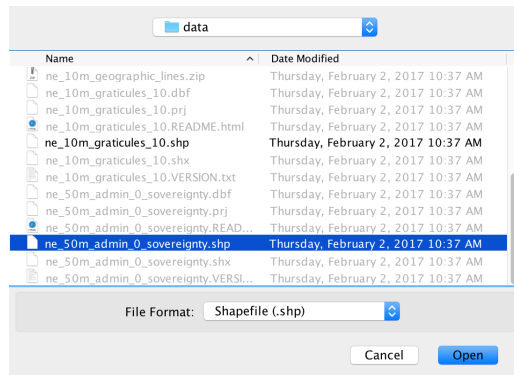
Click on the button highlighted in the next figure to add elements



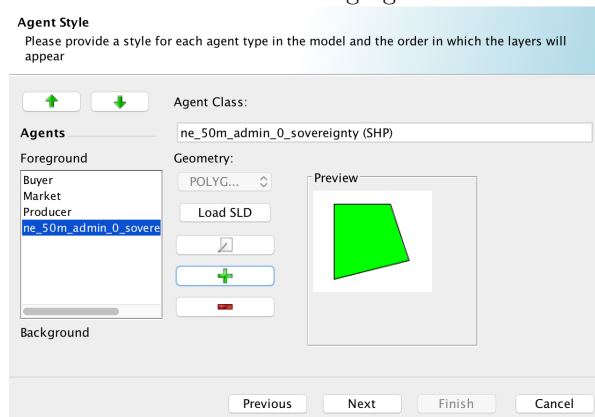
We will show hereafter how to add a background showing a geographic zone. First of all, a shapefile of the zone is needed. Shapefiles are often available for download in Internet. Some shapefiles downloaded from <http://www.naturalearthdata.com/>¹ are already included in the `data` folder.

By clicking on the button highlighted in the figure above, a file browser opens. Manage to select your shapefile. In the following figure, a shapefile in the `data` folder is selected

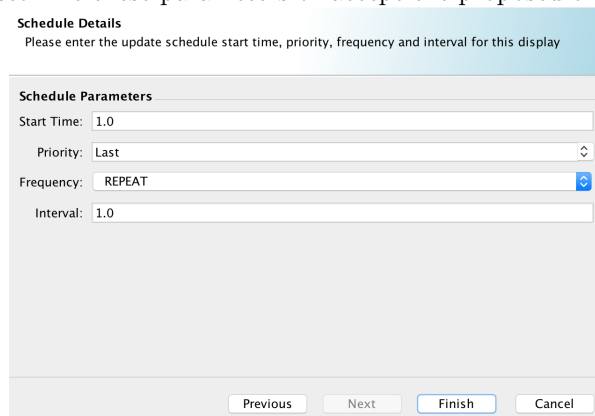
¹ in particular they relate to world maps available at <http://www.naturalearthdata.com/downloads/50m-cultural-vectors/> or <http://www.naturalearthdata.com/downloads/10m-physical-vectors/>.



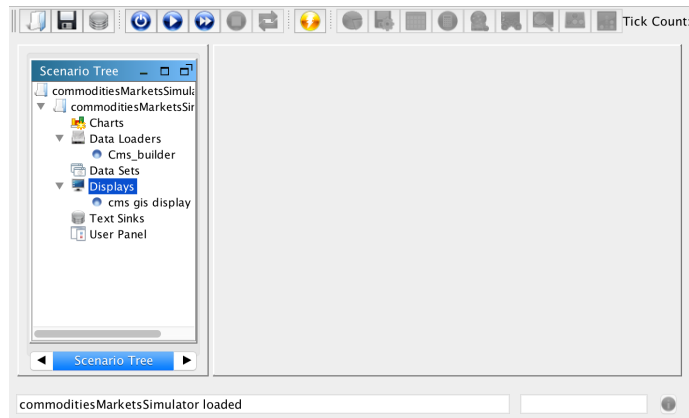
Clicking the open button, an additional element appears in the agent style window as shown in the following figure.



Edit this additional element if you want to customize it, and then click the next button. The following window opens. Refer to the RS documentation to customize these parameters or accept the proposed ones.

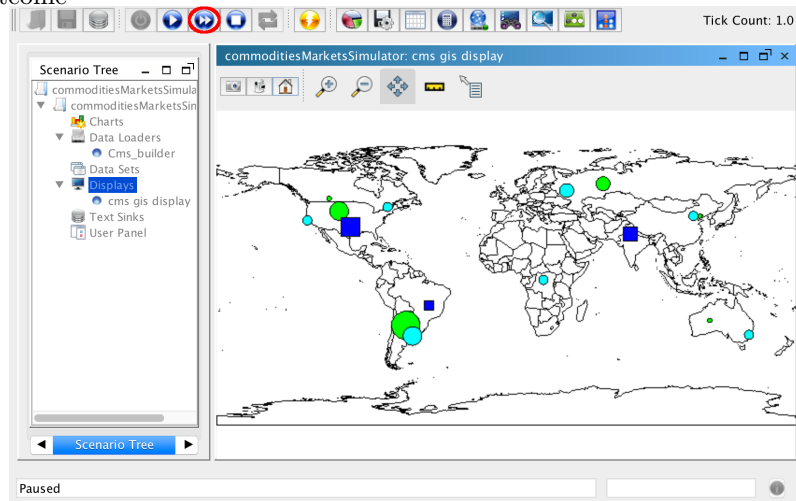


By clicking the finish button the display item appears in the scenario tree as shown in the figure below



Save the changes to the scenario tree by clicking the floppy disk image.

Now, clicking the Step Run button (highlighted in red in the following figure), the display will show up after few seconds. If the map is not centered in the windows, click button to frame it. The following figure shows the final outcome



In the figure above, the following customization are used:

- producers are represented by green circles whose size is proportional to the production share specified in the **producers.csv** file;
- buyers are represented by light blue circles whose size is proportional to the demand share specified in the **buyers.csv** file;
- markets are represented by blue squares whose size is proportional to the exchanges share specified in the **markets.csv** file.

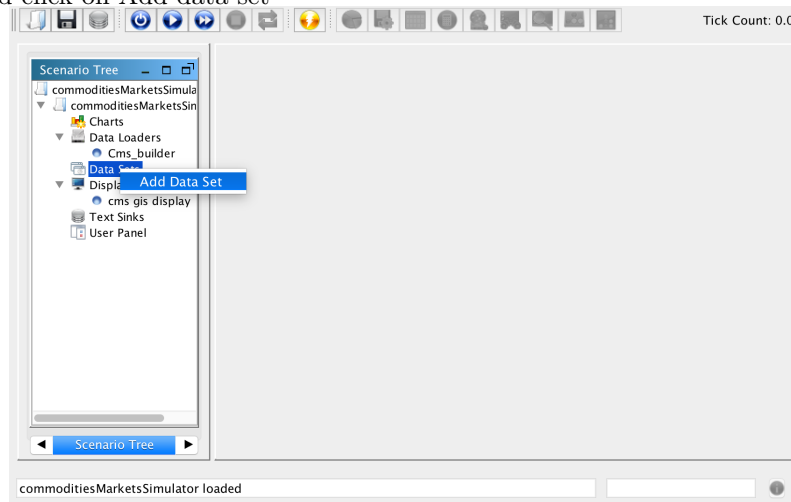
5.2.2 Adding charts and data sinks

These features can be added by managing the scenario tree displayed in the model GUI. The process is similar to the one described above for the display.

We will briefly describe this process pointing the reader to the RS documentation² for a more detailed description.

Charts and Data Sinks take Data Sets as an input. Therefore, Data Sets have to be created first.

To create a data set, right click the corresponding item in the scenario tree and click on Add data set



The wizard for creating the data set opens. Suppose now we want to gather data from all the market sessions. In the first wizard window displayed below we name the dataset `market sessions dataset` and inform the program that it is a Non-aggregate data set

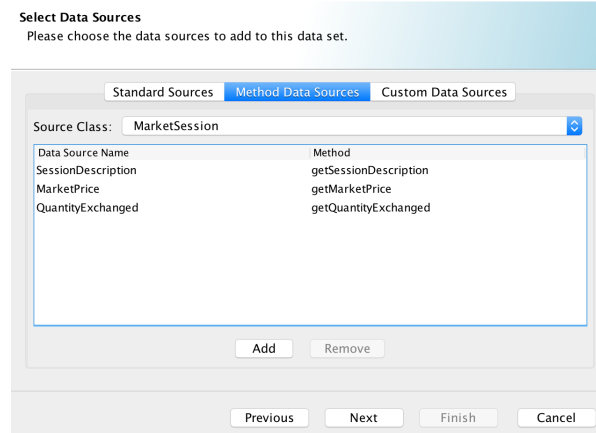
General Settings
Please enter a unique id for that data set and select the data set type.

Data Set Id:

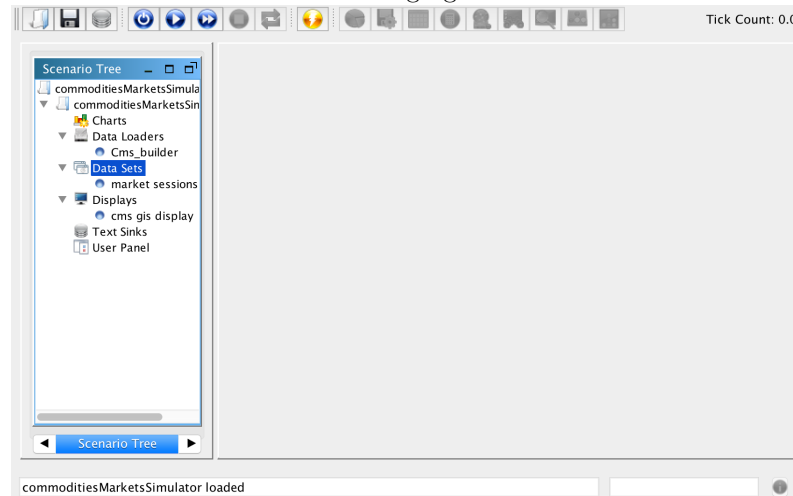
Data Set Type:

²In particular, the document describing data collection available at <https://repost.github.io/docs/DataCollection.pdf> is recommended.

The following window allows to select the variables to be included in the data set. There are three tabs in this window. The first one (**Standard Sources**) lists the variable made available by RS. The second one (**Method Data Source**) allows to add new variable obtained using get methods of the chosen class. While the third one is for more advanced uses. The following figure shows the second tab, where the market description, the market price and the exchanged quantity are added to the data set



Once all the wizard steps are performed, the new dataset appears in the scenario tree as shown in the following figure



Now that we have a dataset, we can use its variables to chart data or to record them in a file (sink them).

To create a new time series chart, start the corresponding wizard by right clicking the charts item in the scenario tree and select the time series option. The chart name can be edited and the data set must be chosen among those created with the previously described procedure. The data set `market sessions dataset` previously created is selected in the following figure:

General Settings
Please enter a chart name and the data set to provide the chart's data

Name:

Data Set:

A new window appears. The variables in the dataset are displayed. Because we use a Non-Aggregate data set, a legend will be added to identify the lines. The **Series ID** choice serves to select which variable will be used to this aim. In the following part, the variable to be plotted against time is selected (in this case it is the MarketPrice)

Chart Data
Please select the data to be displayed by the chart.

Series ID:

Data To Display:

QuantityExchanged	<input checked="" type="checkbox"/>
MarketPrice	<input type="checkbox"/>

The chart titles can be edited in the next wizard window as shown in the figure below

Chart Properties
Please configure the chart's display properties.

Chart Labels

Title: sessions prices

X-Axis: time

Y-Axis: price

Plot Properties

Background Color:

Show Grid Lines: ☒

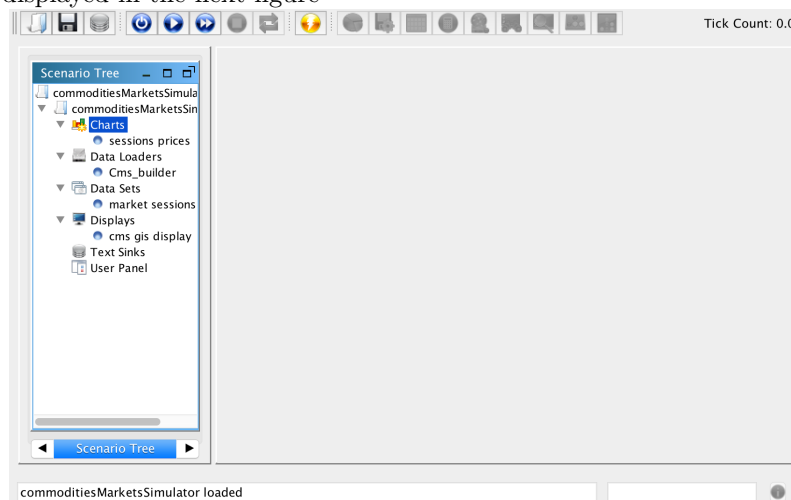
Grid Line Color:

X-Axis Range: -1

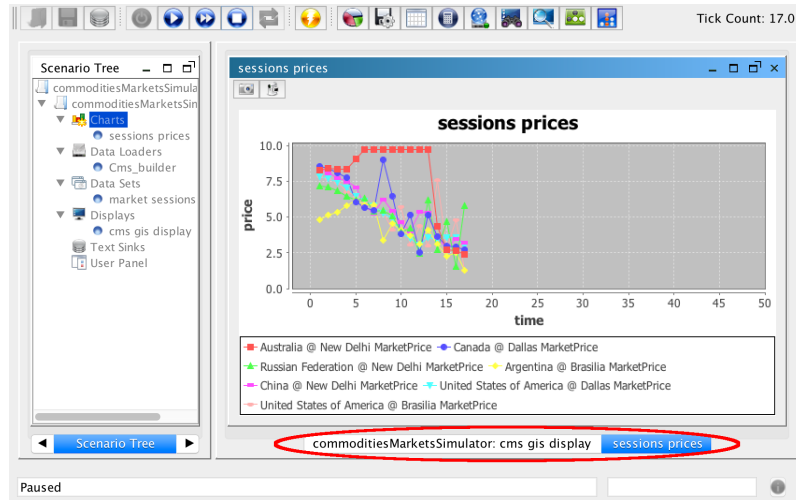
Show Legend: ☒

Previous Next Finish Cancel

Clicking the finish button, the chart item will show up in the Scenario tree as displayed in the next figure



By clicking the Step Run button, RS builds all the created graphic elements and lists them in the bar highlighted in the following figure:



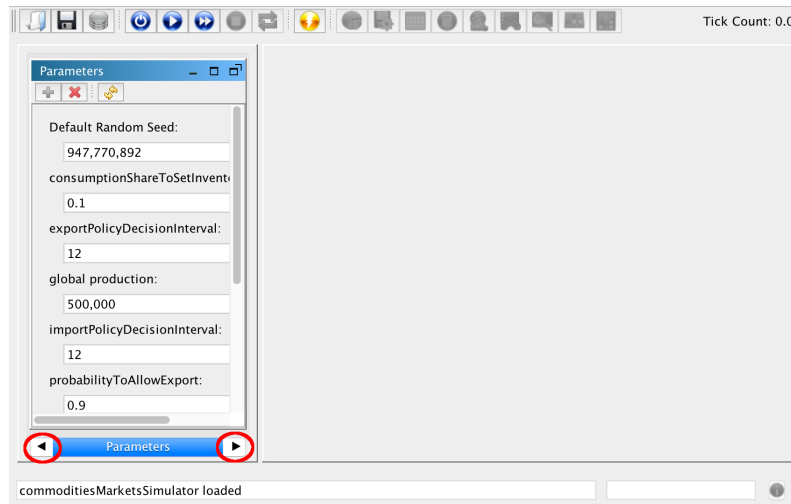
It is possible to configure other graphic elements following the process explained above. As an example, using again the **market sessions dataset** one can build a second chart showing the exchanged quantities. Other charts for producers, buyers or aggregate variables can be created provided that data sets for this purpose are previously created.

Data can be saved by adding **Text Sinks**. The process starts by right clicking the **Text Sinks** and goes on following the step proposed by the wizard. The reader should be now familiar with this way of creating elements, so we overlook the details pointing readers to the [repat data collection documentation](#).

6 Parameter setting

There are two types of parameters in this model: aggregate and individual.

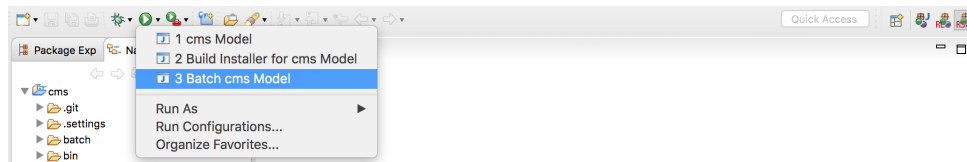
Aggregate parameters relates to the whole economy, therefore they are equal for all the agents. When using the GUI, they can be changed before running the model. To change them, reach the parameters tab in the model GUI by using the arrows highlighted in the following figure.




Change the parameters and then start running the model. Alternatively, the aggregate parameters can be changed by editing directly the `parameters.xml` file in the `cms.rs` folder.

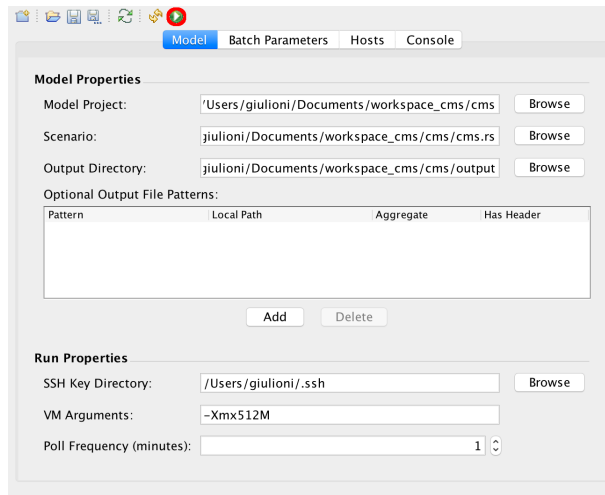
7 Running in BATCH mode

Clicking on the Batch `cms` model as in the following figure



activates the batch run configuration wizard. Alternatively, the wizard can be started in the model GUI by clicking the  button.

In both cases, the window displayed in the following figure will appear

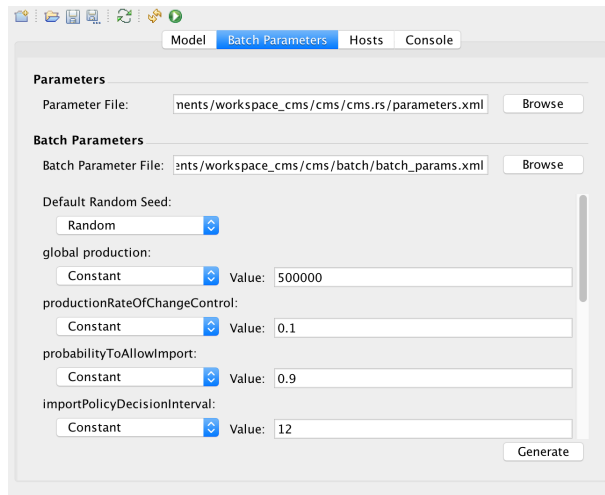


The wizard has four tabs. They serve to configure the run. We point the reader to the “Repast Symphony Batch Runs Getting Started” document available in RS website for a full description of the wizard.

To run the model accepting all the proposed configurations, click the button highlighted by a red circle in the figure above. After clicking this button, the **Console** tab is activated, and several messages flow on the console.

It makes sense to run the model in batch mode if it collects the data of the run. Follow the instructions given in the “Data Collection for Repast Java and ReLogo” document provided in the documentation section of the RS web site to create text sinks. The data files defined in this way will be found in the **output** folder at the end of the run.

It is worth mentioning that the aggregate parameters can be changed before starting the batch run using the second tab of the wizard. It is displayed in the following figure



change the value of the parameters and click the generate button before running the model in batch mode.