User Manual

Ioana Iulia Lucaci

2021

This appendix explains how to use the system, either through the Jupyter notebook or through the command line, as well as what is required for the system to run.

1 Requirements

This program was designed to be cross-platform, and can be run on any device that can install the following tools:

- Python version >= 3.9 (with pip version >= 21.0)
- JupyterLab or Jupyter notebook
- Requirements found in `requirements.txt`

Python can be installed from the Python website[1]. This will also automatically install pip as well. Instructions on how to install JupyterLab or Jupyter notebook are found on Jupyter's website[2]. The extra requirements can be installed using pip with the command `pip install -r requirements.txt` once you are in the program's folder.

2 Configuring a simulation

The file `auction.txt` is the only file that needs to be changed. Listing 1 shows an example of the `auction.txt` file.

```
# Simulation
Number of Rounds = 100
Data Type = 'Efficiency','Speed','Revenue'
Agent Type = 'Auction Types'

# Auction
Number of Bidders = 100
Auction Types = ED,DE,E,D
Reserve Price = 2000
Auctioneer Type = A,B,C,D

# Bidders' Type Percentages
A = 25
B = 25
C = 25
D = 25
```

Listing 1: Example of the input file `auction.txt`

For the data to be examined, two things must be completed: the agent type and data type. The agent type refers to what will be the focus of the analysis, whether it's auction types, auctioneer

[1]`https://www.python.org/downloads/`
[2]`https://jupyter.org/`

type or winner type. Data type refers to what the analysis will be done on, and this can be multiple elements. In the example below, all three metrics presented in this project have been selected.

The `Auction Types` variable must contain either one of the four auction types that have been configured (E for English auction, D for Dutch auction, F for First-price sealed-bid auction and V for Vickrey auction) or a combination of two letters from those four (for example, ED for English-Dutch auction). The auction types have to be comma-separated with no space between them.

The auctioneer type must also be comma-separated with no space between the letters. Every single auction with every single auctioneer type will be run for the number of rounds specified above. So, for this example, every single auction will be run 400 times.

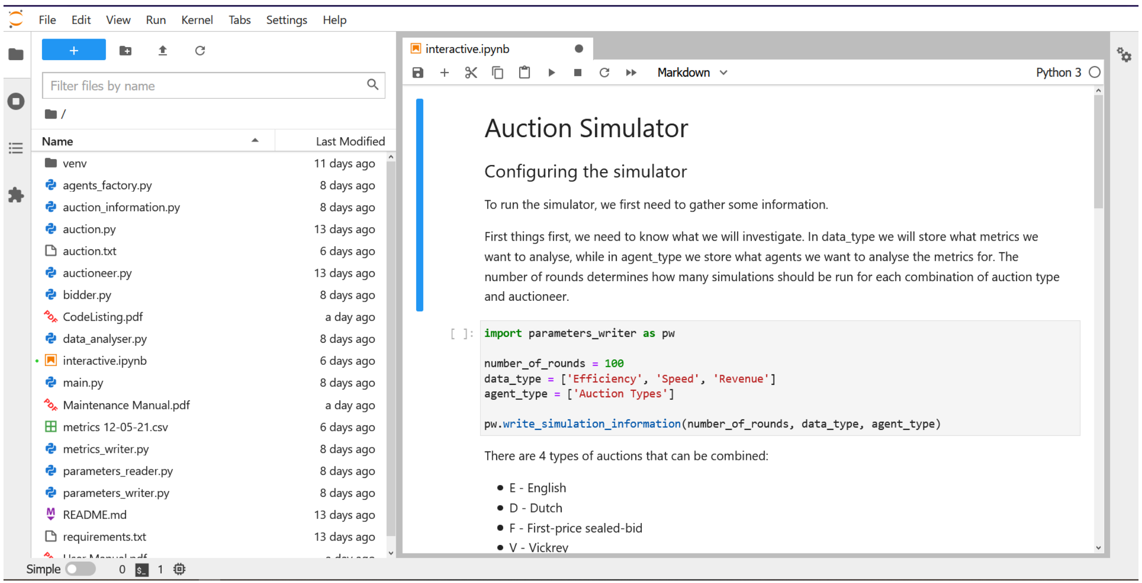
3 Running the simulations

This simulation is configured to either run solely on the command line or to be used within a JupyterLab/Jupyter Notebook. Below are instructions on how to run both versions.

3.1 Running in JupyterLab

JupyterLab and Jupyter Notebook have similar configurations in terms of how it will be run. Because of that, only the steps to run JupyterLab will be described here.

First, a command prompt has to be open in the folder containing the project's code. Inside the command prompt, type `jupyter-lab`. This will open the JupyterLab interface in a new tab of your default browser. By clicking on `interactive.ipynb`, the Jupyter interactive notebook is opened. Figure 1 shows how the JupyterLab is expected to look like.



Name	Last Modified
venv	11 days ago
agents_factory.py	8 days ago
auction_information.py	8 days ago
auction.py	13 days ago
auction.txt	6 days ago
auctioneer.py	13 days ago
bidder.py	8 days ago
CodeListing.pdf	a day ago
data_analyser.py	8 days ago
interactive.ipynb	6 days ago
main.py	8 days ago
Maintenance Manual.pdf	a day ago
metrics 12-05-21.csv	6 days ago
metrics_writer.py	8 days ago
parameters_reader.py	8 days ago
parameters_writer.py	8 days ago
README.md	13 days ago
requirements.txt	13 days ago

Auction Simulator

Configuring the simulator

To run the simulator, we first need to gather some information.

First things first, we need to know what we will investigate. In data_type we will store what metrics we want to analyse, while in agent_type we store what agents we want to analyse the metrics for. The number of rounds determines how many simulations should be run for each combination of auction type and auctioneer.

```
[ ]: import parameters_writer as pw

     number_of_rounds = 100
     data_type = ['Efficiency', 'Speed', 'Revenue']
     agent_type = ['Auction Types']

     pw.write_simulation_information(number_of_rounds, data_type, agent_type)
```

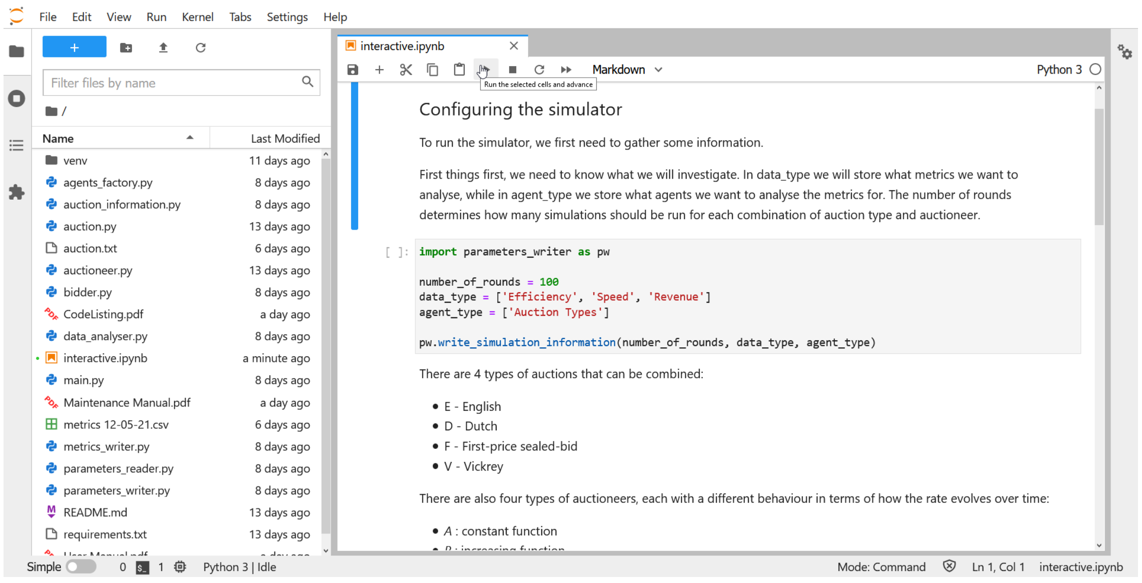
There are 4 types of auctions that can be combined:

- E - English
- D - Dutch
- F - First-price sealed-bid
- V - Vickrey

Figure 1: JupyterLab interface

The notebook is separated in two parts, mainly configuring the simulator and running the simulator. The first part is concerned with populating the `auction.txt` file. The `auction.txt` file (a sample file for it can be seen in Listing 1) is separated in three parts: Simulation, Auction and Biders' Type Percentages. Because of that, the notebook also has three cells where information can be changed, each cell corresponding with a part in the input file, as can be seen in Figure 2. All three cells must be filled in **and** the code executed for the `auction.txt` file to be properly populated. This can be always checked by double-clicking on the `auction.txt` file and inspecting it manually. To execute a cell, as seen in Figure 2, you must click on the cell and click on the run button at the top.

Once the simulator is configured, all that is left to do is to run it. The first cell in the next part of the notebook will run the simulator, saving the metrics in a .csv file whose name will be printed out, as can be seen in Figure 3. There is also the chance to visualise the information directly, with



Name	Last Modified
venv	11 days ago
agents_factory.py	8 days ago
auction_information.py	8 days ago
auction.py	13 days ago
auction.txt	6 days ago
auctioneer.py	13 days ago
bidder.py	8 days ago
CodeListing.pdf	a day ago
data_analyser.py	8 days ago
interactive.ipynb	a minute ago
main.py	8 days ago
Maintenance Manual.pdf	a day ago
metrics 12-05-21.csv	6 days ago
metrics_writer.py	8 days ago
parameters_reader.py	8 days ago
parameters_writer.py	8 days ago
README.md	13 days ago
requirements.txt	13 days ago

Configuring the simulator

To run the simulator, we first need to gather some information.

First things first, we need to know what we will investigate. In data_type we will store what metrics we want to analyse, while in agent_type we store what agents we want to analyse the metrics for. The number of rounds determines how many simulations should be run for each combination of auction type and auctioneer.

```
[ ]: import parameters_writer as pw

     number_of_rounds = 100
     data_type = ['Efficiency', 'Speed', 'Revenue']
     agent_type = ['Auction Types']

     pw.write_simulation_information(number_of_rounds, data_type, agent_type)
```

There are 4 types of auctions that can be combined:

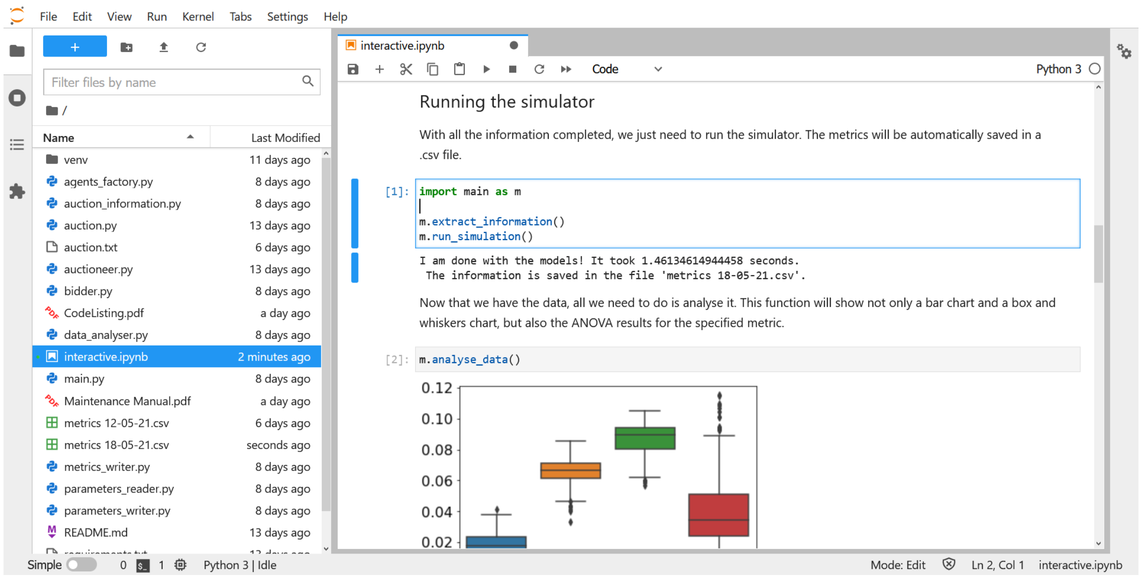
- E - English
- D - Dutch
- F - First-price sealed-bid
- V - Vickrey

There are also four types of auctioneers, each with a different behaviour in terms of how the rate evolves over time:

- *A* : constant function

Figure 2: ‘Configuring the Simulator’ in JupyterLab

the last cell. Running it will output a box-and-whiskers graph, a barchart graph and the ANOVA results for all of the data types specified in the first cell.



Name	Last Modified
venv	11 days ago
agents_factory.py	8 days ago
auction_information.py	8 days ago
auction.py	13 days ago
auction.txt	6 days ago
auctioneer.py	13 days ago
bidder.py	8 days ago
CodeListing.pdf	a day ago
data_analyser.py	8 days ago
interactive.ipynb	2 minutes ago
main.py	8 days ago
Maintenance Manual.pdf	a day ago
metrics 12-05-21.csv	6 days ago
metrics 18-05-21.csv	seconds ago
metrics_writer.py	8 days ago
parameters_reader.py	8 days ago
parameters_writer.py	8 days ago
README.md	13 days ago

Running the simulator

With all the information completed, we just need to run the simulator. The metrics will be automatically saved in a .csv file.

```
[1]: import main as m

     m.extract_information()
     m.run_simulation()

     I am done with the models! It took 1.46134614944458 seconds.
      The information is saved in the file 'metrics 18-05-21.csv'.
```

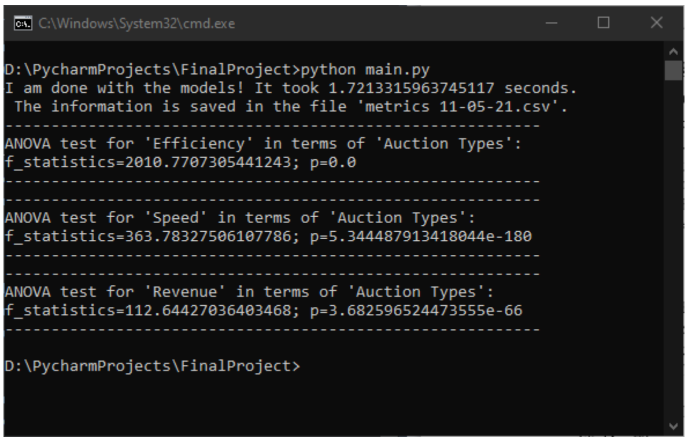
Now that we have the data, all we need to do is analyse it. This function will show not only a bar chart and a box and whiskers chart, but also the ANOVA results for the specified metric.

```
[2]: m.analyse_data()
```

Figure 3: ‘Running the Simulator’ in JupyterLab

3.2 Running from the Command Line

To run this program from the command line, a command prompt must be open in the folder where the files are. After configuring the simulation as instructed in Section 2, the simulator can be run by typing the command `python main.py`. Figure 4 shows an example of a possible outcome, as directed by the sample input file seen in Listing 1.



```
D:\PycharmProjects\FinalProject>python main.py
I am done with the models! It took 1.7213315963745117 seconds.
 The information is saved in the file 'metrics 11-05-21.csv'.
----------------------------------------------------------
ANOVA test for 'Efficiency' in terms of 'Auction Types':
f_statistics=2010.7707305441243; p=0.0
----------------------------------------------------------
----------------------------------------------------------
ANOVA test for 'Speed' in terms of 'Auction Types':
f_statistics=363.78327506107786; p=5.344487913418044e-180
----------------------------------------------------------
----------------------------------------------------------
ANOVA test for 'Revenue' in terms of 'Auction Types':
f_statistics=112.64427036403468; p=3.682596524473555e-66
----------------------------------------------------------

D:\PycharmProjects\FinalProject>
```

Figure 4: Example of a run from the command line

The output will display where the information is saved, as well as the ANOVA results of the experiments. In this case, the results were saved in a file called `metrics 11-05-21.csv`.