A User Manual

Taxi-sim is a piece software for evaluation of variable fare pricing in unregulated taxi markets. It simulates the performance of taxi agents in some environment and establishes a fixed fare taxi pricing benchmark for reference. Usage of this software requires basic familiarity with command line terminal.

This User Manual starts with installation instructions in Section A.1. Instructions for operating the software are in Section A.2.

A.1 Installation

If you do not have the source code of the software available, it can be downloaded as a *zip* archive from `https://github.com/vencha90/taxi-sim/archive/master.zip`.

For running the software it is recommended that you use virtualisation as described in Section A.1.2. If you wish to install the software stack to your operating system natively, please see Maintenance Manual in Appendix B.1.1.

A.1.1 Requirements

The software is recommended to run on a computer with at least 1 GB of RAM. All of the installation steps are compulsory unless stated otherwise. A fast internet connection could be required for installation of some supporting software, and as it relies on external services a 100% availability cannot be guaranteed.

Operating systems that are supported are: Mac OS X, Windows, Debian / Ubuntu, CentOs / RedHat / Fedora.

A.1.2 Virtualisation

Virtualisation is recommended for most Unix-like operating systems (including Mac OS X and Linux derivatives), but absolutely required for Windows. At the time of writing, Windows does not support Ruby 2.1.1 which is required for this software.

Vagrant's configuration files `Vagrantfile` and `vagrant_manifest.pp` used here are adapted from the work of Noria (2014).

- *(web)* Install VirtualBox version 4.3+ (Corporation 2014) from `https://www.virtualbox.org/` using the instructions on the website.
- *(web)* Install Vagrant version 1.5+ (HashiCorp 2013) from `http://www.vagrantup.com/` following instructions on the website.
- *(cmd)* In the taxi-sim directory, run `vagrant up`. This will provision and start the virtual machine. It will take slightly longer the first time as the operating system image needs to be downloaded.
- *(cmd)* `vagrant ssh` will connect to the virtual machine using a secure shell connection, giving you control of the virtual machine's command line terminal.

- *(cmd)* `cd /vagrant/` will change the working directory to the software. Vagrant synchronises this directory with the software directory you cloned to your machine a few steps ago.

- *(cmd)* The final step before you can use Taxi-sim is installing the *gem* (software library) dependencies by running `bundle install --local`. When this step is finished you are ready to use Taxi-sim!

If you can successfully `ssh` on the virtual machine but further steps are not working, please try the following steps:

- log out of the virtual machine by entering `exit`,

- delete the virtual machine by entering `vagrant destroy`,

- recreate the virtual machine by entering `vagrant up`,

- continue as normally.

A.2 Using Taxi-sim

Taxi-sim needs to be used from command line. If you are using Vagrant as recommended in Section A.1.2, in terminal from the main software directory run `vagrant up && vagrant ssh && cd /vagrant/ && bundle install --local && bundle exec rspec` to check that everything is in order.

You might notice that `bundle exec` is used at the start of most commands: this ensures that the commands are executed by bundler (dependency manager software) using the correct gem (software library) versions as a system could have multiple gem versions installed.

This section covers usage of Taxi-sim. To find out how to operate the software from command line terminal, see Section A.2.1. For configuring the simulation inputs see Section A.2.2. You can find out about the structure of output files in Section A.2.3. An easy way to analyse the output data is suggested in Section A.2.4.

A.2.1 Interface

Taxi-sim supports a simple command line interface. You can see the supported tasks and a short description by running `bundle exec rake -T`.

`bundle exec rake spec` runs the automated test suite and is useful for verifying the integrity of the software, especially if you have made any changes.

`bundle exec rake run[path/to/file]` is the main command for you as a user. You need to specify a path to an input file in the square brackets relative to the working directory, e.g. `rake run[sample_input.yml]`. The accepted file format is YML. Open `sample_input.yml` in a text editor to see a sample input file, a detailed explanation of the accepted inputs is in Section A.2.2.

```
time_limit: 1000000
graph:
  - "0, 1, 2, 4, 2, 1"
  - "1, 0, 2, 2, 3, 0"
  - "2, 2, 0, 0, 1, 3"
  - "4, 2, 0, 0, 1, 0"
  - "2, 3, 1, 1, 0, 5"
  - "1, 0, 3, 0, 5, 0"
passenger:
  price: 20
taxi:
  prices: '5..100'
  benchmark_price: 20
```

Figure 8: Sample Input File

A.2.2 Inputs

All types of inputs supported by the simulation are used in `sample_input.yml` file, their order does not matter. They are explained in the following list in alphabetical order:

- *graph*: an adjacency matrix representation of a road network. It has to be a connected undirected graph, meaning that the matrix is symmetrical! Self-loops are ignored. The accepted format for each row is - *"x, .., z"* where x and z are integer number edge weights.
- *passenger* is a top-level entity for passenger details and has this parameter:
  - *price* is an integer number of the price that passengers expect to pay for a single unit of distance, used to calculate probability of accepting a fare as described in Section 3.1.2.
- *taxi* is a top-level entity for taxi details and has the following parameters:
  - *benchmark_price*: an integer number of the fixed price that the simulation will be benchmarked against as described in Section 3.1.5.
  - *prices*: a range of prices that the taxi can choose from. An integer number, an array of integer numbers in the format *[x, .., z]*, or a range of integer numbers in the Ruby range format *"x..z"* (includes z) or *"x...z"* (excludes z).
- *time_limit*: an integer number for the units of time the simulation will run for

A sample input file is shown in Figure 8.

A.2.3 Output files

Simulation output is written to `logs/` directory in the application's home directory. These files can be opened using most text editing software, although some mainstream software might crash due to

```
["time", "0"]["time_limit", "1000000"]
["fc", "1"]["vc", "1"]["prices", "5..100"]
["time", "1000000"]["profit", "839289"]
["fc", "1"]["vc", "1"]["prices", "[20]"]
["time", "1000000"]["profit", "-1677056"]
```

Figure 9: Sample Simulation Summary File

```
["time", "155"]["reward", "-4"]["location", "5"]["busy_for", "3"]["action", "drive"]
["time", "156"]
["time", "157"]
["time", "158"]["passenger", "accepted"]["fare", "15"]["location", "5"]
        ["destination", "6"]["reward", "9"]["location", "6"]
        ["busy_for", "3"]["action", "offer"]
```

Figure 10: Excerpt from a Sample Simulation Log File

the large file size of `simulation.log`. In this case software that supports huge file sizes can be easily found, for example, notepad++ (Ho 2011). Parameters are logged in this format: `[‘‘name", ‘‘value"]`.

`summary.log` is a summary of the simulation results with six rows. The first row is written when the simulation starts stating the time limit, the next two rows state taxi details and the total profit for the taxi using a range of prices. The last two rows state the same data for the benchmark run with a fixed price. Contents of a sample summary file are shown in Figure 9.

`simulation.log` is a full log of the simulation as it progressed. It has two parts, the first being the run with a range of prices and the second run with a fixed taxi price. It has a single row for each time unit in both runs, therefore this file is likely to only be useful for data analysis. An excerpt from a simulation log is shown in Figure 10.

If you require customised output, the data written to log files can be changed by modifying just a few lines of the code. This is explained in the Maintenance Manual in Appendix B.2.3.

A.2.4 Data Analysis

The R Project provides an open-source statistical programming software package (Foundation 2014). An R language script file `analysis.R` for Taxi-sim data analysis is included in the project source code.

The first step before the script can be used is separation of benchmark data and variable pricing data from the main data source. Open `simulation.log` in a text editor and make two new files by splitting the original file in half by copy-pasting each run's data. Name the file with variable prices `variable.log` and the file with fixed price `benchmark.log`, and store them in the same directory. For the script to work, you have to delete the first and last line of `variable.log` where no actions are selected.

Open the script file in an R work environment. If you followed the step above, you only have to modify line 4. Change the path of your working directory in `setwd("your/path")`, for example, to be `setwd("c:`
`User/Documents/Taxi-sim/Data/")`

Run the script – it will take some time depending on the amount of data. This is a memory expensive operation and depends on the available RAM. When the script is finished, it will print a summary pdf file with graphs and descriptive statistics. This script uses external libraries available online, therefore an internet connection is required. It is expected that you will see warnings about removal of empty rows – the script measures reward that was empty when taxi had been busy with some actions.