Observations

Elaine Wah
ewah@umich.edu

Updated: January 8, 2015

1 Introduction

Observations are created each time a simulation is run. The format of the observation files are dictated by the requirements for our EGTAOnline testbed. They are created as JSON files, and there are two types of observation files (one for each use case).

Environment configuration parameters are saved in the observation file as a nested feature "config" unless `outputConfig` is set to false in `env.properties` (in the `config` directory). Note that in the *EGTA* use case, `outputConfig` should always be false as nested features cannot be used. However, when running simulations for multiple market models locally and then aggregating into a single csv, outputting the configuration may be helpful for differentiating between different environments. Below is the example `observation0.json` file found in the `docs` folder:

```
{
    "players": [],
    "features": {
        "profit_sum_background": -956497.0,
        "trans_num": 1392.0,
        "vol_freq_250_stddev_log_return_market_1":
           0.14263809361637586,
        "vol_freq_250_stddev_log_return_market_2":
           0.11314219073613756,
        "trans_mean_price": 98080.408045977,
        "vol_freq_250_mean_log_price": 9.615571950882519,
        "spreads_median_market_2": 31000.0,
        "vol_freq_250_stddev_price_market_1":
           15269.193111951843,
        "spreads_median_market_1": 33432.0,
        "vol_stddev_log_return_market_2": 0.009396587991555228,
        "vol_stddev_log_return_market_1": 0.011789157834184416,
        "vol_stddev_price_market_2": 14852.020501348485,
        "profit_sum_marketmaker": 0.0,
        "vol_freq_250_mean_stddev_price": 14998.932534557262,
        "control_mean_fund": 99811.48257559388,
        "surplus_sum_disc_6.0E-4": 1826176.693056684,
"surplus_sum_no_disc": -1482624.0,
"trans_rmsd": 34885.77841807928,
"trans_laagent_num": 158.0,
"config": {
    "randomSeed": "271828",
    "arrivalRate": "0.075",
    "privateValueVar": "1E8",
    "nbboLatency": "100",
    "modelName": "example",
    "shockVar": "1E8",
    "LA": "num_1",
    "simLength": "60000",
    "MAMM": "num_0",
    "tickSize": "1",
    "AA": "num_0",
    "ZIR": "num_61_maxqty_10",
    "ZI": "num_0_bidRangeMin_0_bidRangeMax_5000",
    "ZIP": "num_0",
    "presets": "NONE",
    "meanValue": "100000",
    "kappa": "0.05",
    "WMAMM": "num_0",
    "mktLatency": "-1",
    "reentryRate": "0.0005",
    "CDA": "num_2",
    "numSims": "1",
    "CALL": "num_0",
    "BASICMM": "num_0_numRungs_10_rungSize_1000"
},
"profit_sum_hft": 956497.0,
"spreads_mean_markets": 32216.0,
"control_var_fund": 1044590239.5866868,
"vol_mean_log_price": 9.606234227976671,
"exectime_mean": 287.85251215559157,
"vol_mean_stddev_price": 14857.116972219945,
"trans_freq_250_rmsd": 33372.12651671857,
"vol_freq_250_mean_log_return": 0.12789014217625672,
"surplus_ziragent_sum_no_disc": -1482624.0,
"vol_mean_log_return": 0.010592872912869821,
"vol_freq_250_stddev_price_market_2":
   14728.671957162678,
"spreads_median_nbbo": 17008.0,
"control_mean_private": 394.5245901639344,
"vol_stddev_price_market_1": 14862.213443091407,
"trans_ziragent_num": 1234.0,
        "surplus_ziragent_sum_disc_6.0E-4": 343552.6930566839,
        "trans_stddev_price": 18390.46369581062,
        "profit_sum_total": 0.0
    }
}
```

2 Reading an observation file

Observations are saved in the format `observation#.json`, where # is the observation number (from 0 to the one less the max number of samples gathered, if multiple simulations are run via the available scripts). Note also that this number is the second input argument to the simulator. As such, be sure to save different environment configurations in different directories; otherwise you run the risk of overwriting previous observations.

Depending on the use case (*EGTA* vs. *Market Model*), the observation file will differ. Parsing observation files (and converting to CSV) is currently only supported for the *Market Model* use case.

To format the observation file for easier viewing, use the following command:

```
./jpath.py < [observation file]
```

Descriptions for the various types of features (statistics) are listed in Table 1. Note that features must be numeric; *NaN*s are not permitted.

2.1 EGTA

In the *EGTA* use case, there is a section in the json file called “players” containing the player’s role, strategy, payoff (i.e., surplus), and any player-specific features, such as those for control variates. See the following for an example which represents the format that would arise from specifying player-strategy assignments as in the example provided in `simulation_spec.pdf`, where there are two background traders and a single market maker employing the BasicMM strategy:

```
"players": [
    {
        "payoff": 13354.12,
        "role": "BACKGROUND",
        "features": {
            "control_var": 0.12323
        },
        "strategy": "ZIR:bidRangeMin_0_bidRangeMax_1000"
    },
    {
        "payoff": 5642.776,
        "role": "BACKGROUND",
        "features": {
            "control_var": 0.56184
        },
        "strategy": "ZIR:bidRangeMin_0_bidRangeMax_5000"
    },
    {
        "payoff": 1212.089,
        "role": "MARKETMAKER",
        "features": {},
        "strategy": "BASICMM:rungSize_100"
    }
],
```

The other section in the json file will be "features" which will include information on the simulation configuration and aggregate statistics computed at the end of the simulation. This is discussed in depth in the following section.

2.2 Market Model

In the *Market Model* use case, the "player" section will be empty (as there are no players). The features section will be the same as in the EGTA use case.

3 Merging observation files

To merge observation files (from simulation runs performed off the testbed), the data from each individual run will be merged into one JSON file or into a merged directory. Merging is different for each use case, as outlined below. For more details, the optional argument `-h` will output the script's help message.

3.1 EGTA

To merge, which means finding the mean for each feature across all observations and the mean payoff for each unique role-strategy setting, use the following command:

```
./merge-obs-egta.py [list of observation files] -o [merged observation file]
```

For example,

```
./merge-obs-egta.py simulations/test/obs*.json -o simulations/test/merged.json
```

will determine the mean values for all observations within the `~/simulations/test` directory and save these values in the specified output file. This merge script will also output the sample standard deviation.

3.2 Market Model

In the *Market Model* use case, the following script will merge observation files from multiple directories (assuming comparable environment settings but varying agent populations or market models) into a single output directory. The `merge-sim-obs.sh` script can also be used for observations generated with *EGTA*-style spec files, although it is not recommended

NAME	DESCRIPTION
`surplus_<agent>_sum_no_disc`	total raw (undiscounted) surplus for agents of the specified type
`surplus_sum_no_disc`	total raw surplus for all agents
`surplus_<agent>_sum_discY`	total surplus (discounted by Y) for agents of the specified type
`surplus_sum_discY`	total surplus (discounted by Y) for all agents
`profit_sum_<role>`	total profit for agents in given role (e.g., background, MM, HFT)
`profit_sum_total`	total profit for all agents
`spreads_median_market_#`	median spread in market #
`spreads_mean_markets`	average median spread over all markets
`spreads_median_nbbo`	median spread of NBBO
`vol_freq_X_mean_stddev_price`	average volatility over all markets, measured by standard deviation of midquote prices sampled every X time steps (if `freq_X` not present, metric is computed for all time steps)
`vol_freq_X_mean_log_price`	average volatility over all markets, measured by log of standard deviation of midquote prices sampled every X
`vol_freq_X_mean_log_return`	average volatility over all markets, measured by standard deviation of log returns sampled every X
`vol_freq_X_stddev_price_market_#`	volatility in market #, measured by log of standard deviation of midquote prices sampled every X
`vol_freq_X_stddev_log_return_market_#`	volatility in market #, measured by standard deviation of log returns sampled every X
`trans_mean_price`	average transaction price
`trans_rmsd`	root mean square deviation (RMSD) between transaction prices and the value of the fundamental at time of execution (ignores NaNs)
`trans_freq_X_rmsd`	RMSD based on prices sampled every X timesteps (ignores NaNs)
`trans_stddev_price`	standard deviation of transaction prices
`trans_num`	total number of transactions
`trans_<agent>_num`	number of transactions by agents of specified type (e.g., ZI, ZIR, LA)
`exectime_mean`	average time between when bid is submitted to when it transacts

Table 1: List of observation features.

due to the size of the output files—each model's list of players is also merged into each merged observation file. To merge, use the following command:

```
./merge-sim-obs.sh [merged directory] [# of observations] [input directories]
```

For example,

```
./merge-sim-obs.sh simulations/merged 100 simulations/{model_1,model_2}
```

will merge the first 100 observations from folders `simulations/model_1` and `simulations/model_2` into the directory `simulations/merged`. The resulting output directory `simulations/merged` will have 100 observation files, with each one containing the corresponding observation outputs from the input directories.

The model name specified in `simulation_spec.json` is used to differentiate between models. In the example above, there will be two entries for each statistic. If the model name is the same as the folder name, the output JSON file will have both `model_1_surplus_sum_no_disc` and `model_2_surplus_sum_no_disc`.

4 Parsing observation files

When parsing observation files, the JSON headers are flattened (in the JSON file itself. This aspect is only relevant for configuration parameters, as nested features are no longer permitted in the testbed.

To merge a directory of observations into a CSV, use the following command:

```
./obs2csv.py [list of observation files] -o [csv-file]
```

For example,

```
./obs2csv.py simulations/test/obs*.json -o merged.csv
```

will parse all observations in the `simulations/test` directory.

Note that this script will only generate column headers if the output CSV does not yet exist; if you are not seeing header files, delete the CSV and re-parse.