Code for Galichon-Salanie's "Estimating Separable Matching Models"

Usage

Create a virtual environment, e.g. with `python3 -m venv env`. Activate it with `source env/bin/activate`.

Install the requirements with `pip install -r requirements.txt`. Among the packages it downloads are two created by Bernard Salanié: `bs_python_utils` and `cupid_matching`. The former is just a set of utility programs. The latter contains code to solve for the stable matching and estimate the parameters of separable matching models with MDE and Poisson-GLM. The code in this folder relies heavily on these two packages, which are documented on Salanie's website: bs_python_utils and cupid_matching.

Choose the parameters in `config.py` and run the code with `python main.py`. The program will create a folder `Results` and save a plot and a pickled file with the estimates for the sample `sample_size` defined in `config.py`.

Each simulation sample (that is, `n_sim=1`) takes a few seconds (4 seconds on a Mac M2 Max 2023) to estimate the Choo and Siow model by the two methods in the paper --- minimum distance and Poisson GLM. The code is parallelized over samples, unless you choose `use_multiprocessing=False`. By default, it uses all except 2 of your CPUs.

Structure of the code

The master program `main.py` reads the parameters in `config.py`.

1. If `do_create_samples` is `True` it uses `create_samples.py` to read the Choo and Siow datasets in the `data_dir` directory and to create two samples in `samples_dir` (both directories are specified in `config,.py`). The two samples correspond to the small and large samples described in Section 6 of the paper. The files created have the marriage patterns by age (`*muxy.txt`), the margins (`*nx.txt` and `*my.txt`), and the variance-covariance matrix of these estimates (`*varmus.pkl`).
2. It calls `read_data.py`, which reads the sample defined by `sample_size` in `config.py` and prepares it for the simulation. `read_data.py` also has code to add a small positive number (see `zero_guard` in `config.py`) for zero cells; this is used in the MDE simulation.
3. `specification.py` creates the basis functions according to the specification given by `degrees` in `config.py`.
4. Then `main.py` runs the simulation via `simulate.py` as defined by `config.py`.

Configuration

All parameters of the simulation are in `config.py`:

- `do_create_samples, do_simuls, plot_simuls, do_simuls_mde, do_simuls_poisson` define what the program does;
- `n_sim` is the number of simulated samples;
- `use_multiprocessing` and `nb_cpus` define the parallelization;

- `zero_guard` is the small positive number added to zero cells in the sample for MDE estimation;
- `degrees` is a list of tuples that define the degrees of the polynomials for the basis functions; e.g. an `(a,b)` tuple means that the basis function is $L_a(x)L_b(y)$, where x and y are the ages of the partners and L_a is the Legendre polynomial of degree a. In addition to these terms, the basis functions also include the constant term; $\mathbf{1}(x>y)$, and a term proportional to $\max(x-y,0)$. The function `generate_bases` in `specification.py` creates the basis functions.

Questions

Please direct all questions to Bernard Salanie.