Matlab tic toc timeit Timers and Profiler Save to HTML, Testing to Improve Code Speed

back to Fan's Intro Math for Econ, Matlab Examples, or MEconTools Repositories

Tic and Toc and Timeit

Simple timing example. In the following example, we want to simulate many random normal draws with different means. Calling Method B which uses mean and standard deviation as parameters for normrnd is similar in speed to using the z-score inverse function to transform from standard normal to normal as in method A. Using tic and toc, it is much faster to use method A when we want to draw from many normal distributions with different means.

```
% Two method to generate random normal draws
f_replace_method_a = @() normrnd(0, 1)*10+10;
f_replace_method_b = @() normrnd(10, 10);

% Time replacing one value
fl_speed_method_a = timeit(f_replace_method_a);
fl_speed_method_b = timeit(f_replace_method_b);
fl_speed_a_b_ratio = fl_speed_method_a/fl_speed_method_b;
disp(['Load table cell time, fl_speed_a_b_ratio=' num2str(fl_speed_a_b_ratio)]);
```

```
Load table cell time, fl_speed_a_b_ratio=1.0042
```

```
% Timing assignment with Method A
ar_rand = rand([1,1e4]);
fl_time_start = tic;
ar_z = normrnd(0, 1)*10+ar_rand;
fl_time_end = toc(fl_time_start);
disp(['Method A fl_time_end = ' num2str(fl_time_end)]);
```

```
Method A fl_time_end = 0.0026843
```

```
% Timing assignment with Method A
fl_time_start = tic;
for (it_rand_ctr=1:length(ar_rand))
    ar_z(it_rand_ctr) = normrnd(ar_rand(it_rand_ctr), 10);
end
fl_time_end = toc(fl_time_start);
disp(['Method B fl_time_end = ' num2str(fl_time_end)]);
```

```
Method B fl_time_end = 0.062244
```

Profiling Lines of Code

There is a program that we have written. We want to know which lines of code is taking more or less time, and identify opportunities for speed improvement. Use Matlab Profiler to run through code and generate

performance report line by line. This can be achieved by clicking on the Run and Time button under the Editor Pane. Below, we start the profiler inline, run a function, and save profiling results to a HTML file.

The function below finds the file to the current m file where the profiler is running at, and generates a subfolder which will contain a subfolder that contains all HTML results from a particular profiler timing run.

```
% Define profiling folder and file names
% Find current path to m file and generates profiling subfolder
srn_profile_folder = '_profile';
% Name of the profling output folder (including HTML files)
srn_mprofile_subfolder = 'fs_profiler_tester';

% Turn profiler on
profile on

% OLS Regression
fci_ols_lin = @(y, x) (x'*x)^(-1)*(x'*y);
% Regression inputs
it_obs_n = 10000;
it_k_n = 5;
rng(123);
ar_y = rand([it_obs_n,1]);
mt_x = rand([it_obs_n, it_k_n]);
% Regression
ar_esti = fci_ols_lin(ar_y, mt_x);

% Turn Profiler off
profile off;

% Save profiling results to file
% Find current file folder and generate a profiling subfolder
spn_path2file = matlab.desktop.editor.getActiveFilename;
[spt_path_folder_root, ~, ~] = fileparts(spn_path2file);
spn_profiler = fullfile(spt_path_folder_root, srn_profile_folder);
if ~exist(spn_profiler, 'dir')
    mkdir(spn_profiler);
end

% Store results to file
spn_profiler_results = fullfile(spn_profiler, srn_mprofile_subfolder);
profsave(profile('info'), spn_profiler_results);
```