statacpp: a simple Stata / C++ interface

Robert Grant
Kingston & St George’s
robertgrantstats.co.uk
WELCOME
FIENDS
RESTAURANT
Stata
Mata
Greata
Ada
Why?

- RCpp has been very popular
- Interface from a data analysis-specific high-level language to a compiled fast low(er)-level language
- C++ is widely used and trusted
- There are many powerful libraries
- You can run on multiple cores without Stata/MP
Startup advice overheard: you have too many hipsters, you won't scale like that. Hire some fat guys who know C++.
How?

- Built by smashing StataStan & sticking it back together
- Write code out to a .cpp text file
- Add in variables, globals, matrices from Stata
- Add in code to write results back into a new do-file
- Shell command to compile it; shell command to run the new executable file
- Do the new do-file to get the results into Stata; carry on where you left off
“they say no thing is wrote now-a-days, but low nonsense and mere bagatelle”

–Alain René le Sage, 1759
Silly example

• Grant’s Patented Fuel Efficiency Boosterizer

• We pass the mpg variable from the auto dataset, and a global, to C++

• There, mpg values are multiplied by the global, and passed back as mpg2

• Trebles all round
sysuse auto
global myglob=2
mkmat weight length in 1/5, mat(mymat)

/* C++
   int main () {
   cout << "Now running the Fuel Efficiency Boosterizer"  << endl;
   cout << "We will multiply mpg by: " << myglob << endl;
   std::vector <int> mpg2 = mpg;
   for(int i=0;i<mpg.size();i++) {
       mpg2[i] = mpg[i]*myglob;
   }
   double mymat2[1][2]= {{mymat[0][0], mymat[0][1]});
   // send var mpg2
   // send matrix mymat2
   return 0;
}
*/

statacpp mpg, codefile("myprog.cpp") inline globals("myglob") matrices("mymat")
tabstat mpg mpg2, stat(min q max)
#include <iostream>
#include <array>
#include <vector>
#include <fstream>
#include <sstream>
using std::cout;
using std::endl;
using std::array;
using std::vector;
using std::ifstream;
using std::ofstream;
int main () {
    std::vector<int> mpg = {22, 17, 22, 20, 15, 18, 26, 20, 16, 19, 14, 14, 21, 29, 16, 22, 22, 24, 19, 30, 18, 16, 17, 28, 21, 12, 12, 14, 22, 14, 15, 18, 14, 20, 21, 19, 19, 18, 19, 24, 16, 28, 34, 25, 26, 18, 18, 18, 19, 19, 19, 24, 17, 23, 25, 23, 35, 24, 21, 21, 25, 28, 30, 14, 26, 35, 18, 31, 18, 23, 41, 25, 25, 17};
double mymat[5][2] = { { 2930, 186 }, { 3350, 173 }, { 2640, 168 }, { 3250, 196 }, { 4080, 222 } };
double myglob = 2;
cout << "Now running the Fuel Efficiency Boosterizer" << endl;
cout << "We will multiply mpg by: " << myglob << endl;
std::vector<int> mpg2 = mpg;
for(int i=0; i<mpg.size(); i++) {
  mpg2[i] = mpg[i]*myglob;
}

double mymat2[1][2] = {{mymat[0][0], mymat[0][1]});

// send var mpg2
// send matrix mymat2
ofstream wfile;
wfile.open("output.do", ofstream::out);
wfile << "input mpg2" << endl;
for(int i=0; i<= (mpg2.size()-1); i++) {
  wfile << mpg2[i] << endl;
}

int ncells; int ncols; int nrows;
ncells = sizeof(mymat2)/sizeof(double);
ncols = sizeof(mymat2[0])/sizeof(double);
nrows = ncells/ncols;

wfile << "matrix mymat2 = [";
for(int i=0; i<nrows; i++) {
  for(int j=0; j<ncols; j++) {
    wfile << mymat2[i][j];
    if(j<(ncols-1)) { wfile << ", ";} // Use ',' instead of space
    if(i<(nrows-1)) { wfile << " \ ";} // Use backslash for \n
  }
  wfile << "]" << endl;
}
wfile.close();
return 0;
```plaintext
<table>
<thead>
<tr>
<th></th>
<th>input mpg2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>44</td>
</tr>
<tr>
<td>2</td>
<td>34</td>
</tr>
<tr>
<td>3</td>
<td>44</td>
</tr>
<tr>
<td>4</td>
<td>40</td>
</tr>
<tr>
<td>5</td>
<td>30</td>
</tr>
<tr>
<td>6</td>
<td>36</td>
</tr>
<tr>
<td>7</td>
<td>52</td>
</tr>
<tr>
<td>8</td>
<td>40</td>
</tr>
<tr>
<td>9</td>
<td>32</td>
</tr>
<tr>
<td>10</td>
<td>38</td>
</tr>
<tr>
<td>11</td>
<td>28</td>
</tr>
<tr>
<td>12</td>
<td>28</td>
</tr>
<tr>
<td>13</td>
<td>42</td>
</tr>
<tr>
<td>14</td>
<td>58</td>
</tr>
<tr>
<td>15</td>
<td>32</td>
</tr>
</tbody>
</table>

```

```plaintext
<table>
<thead>
<tr>
<th></th>
<th>matrix mymat2 = [2930, 186]</th>
</tr>
</thead>
</table>
```
Application 1

• Big(-ish) data

• Let’s draw a heatmap of pickup locations for every taxi journey in New York city in 2013.

• MTA dataset obtained by Chris Whong, ~50GB
NYC taxi data

• Loop through each of 24 text files

• No need to load to RAM; process one line at a time

• Binning on rectangular grids: latitude, longitude

• Simplest form of MapReduce concept

• You could also extract a random sample, and don’t forget the value of sufficient statistics…
NYC taxi data

- Get the latitude & longitude from line 1
- Add each line (1 taxi journey) to the relevant bin
- Move to the next line
- Return the binned counts to Stata as data
- Draw some plots, do some analysis
NYC taxi data

• But Robert, you could do that with Stata file commands

• Sure, but
  • this can be parallelised without Stata/MP and
  • there are many other input streams in C++, e.g. from sensors on serial ports
Application 2

- Deep(-ish) learning
- Let’s send our data through a C++ library that offers analyses we don’t have inside Stata
- Fisher’s irises
- Interlocked spirals (artificial data)

playground.tensorflow.org
Fisher’s irises

- An example from the OpenNN library
- A simple neural network for classification
- 4 input neurons, 6 hidden neurons in 1 layer, 3 output neurons
- This is an easy problem
Interlocked spirals

FEATUEs
Which properties do you want to feed in?

+ - 2 HIDDEN LAYERS

+ - 4 neurons

4 neurons

OUTPUT
Test loss 0.082
Training loss 0.063

This is the output from one neuron. Hover to see it larger.

The outputs are mixed with varying weights, shown by the thickness of the lines.

Colors shows data, neuron and weight values.

playground.tensorflow.org
Interlocked spirals

• An artificial ‘hard’ problem
• Classical statistical tools will not help
• 6 input neurons \((x, y, x^2, y^2, \sin x, \sin y)\)
• 4:4 hidden neurons (2 layers \(=\text{‘deep’}\))
• 1 output neuron
• Very hard without knowing the structure
Limitations & grumpiness

- One .cpp file, limited linking capability
- g++ (& makefile) only
- Not even tested in W*****s
- But wouldn’t it be nice to have:
  - StataCUDA
  - the reverse interface to call Stata for analysis
- Don’t ask for stuff, go to github.com/robertgrant/statacpp and make it