statacpp: a simple Stata / C++ interface

Robert Grant
Kingston & St George's
robertgrantstats.co.uk


WELCOME
F IENDS
RESTAURANT

Stata

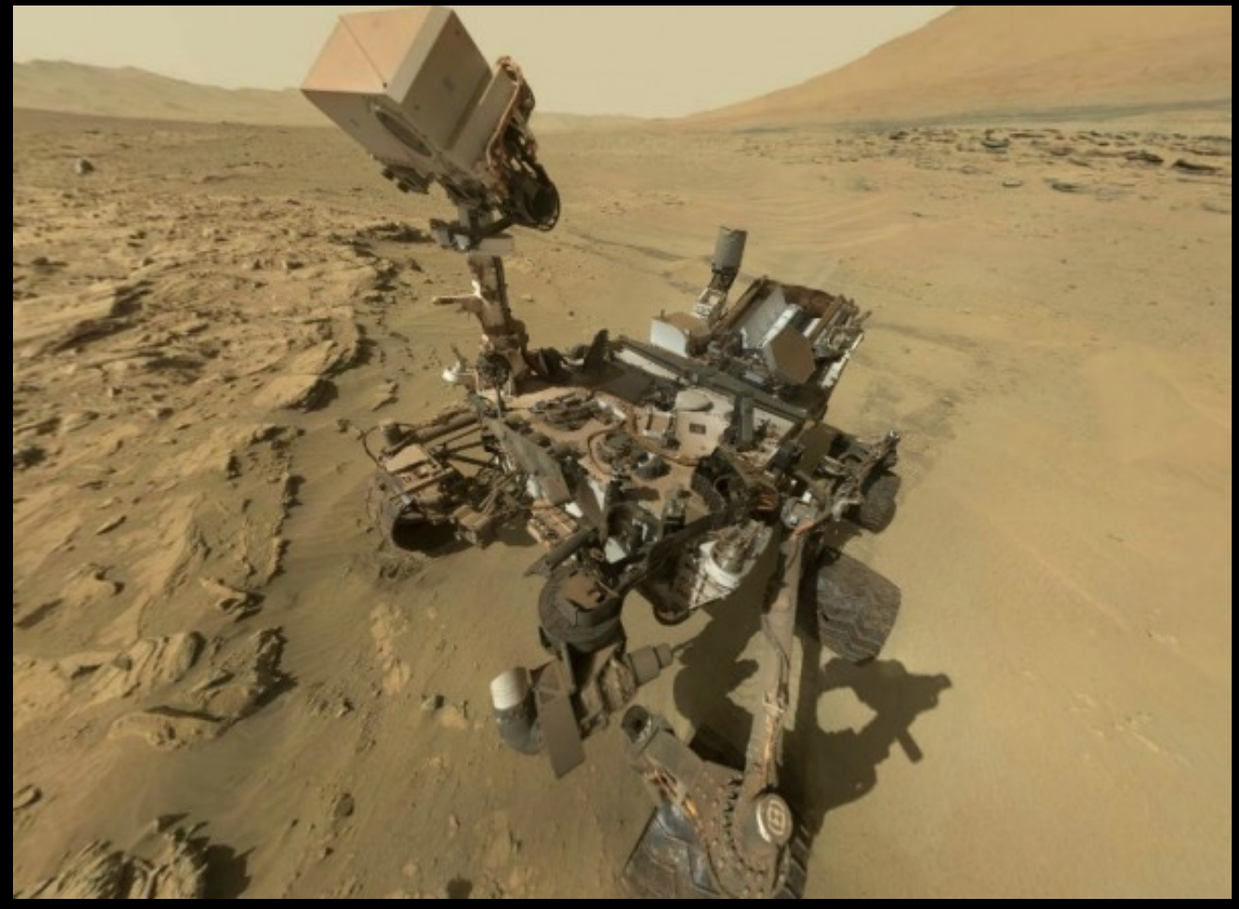
Mata

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



Jay Kreps
@jaykreps

Follow

Startup advice overheard: you have too many hipsters, you won't scale like that. Hire some fat guys who know c++.

RETWEETS	LIKES
1,987	510

6:10 AM - 12 Aug 2011

2K 510

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)
```

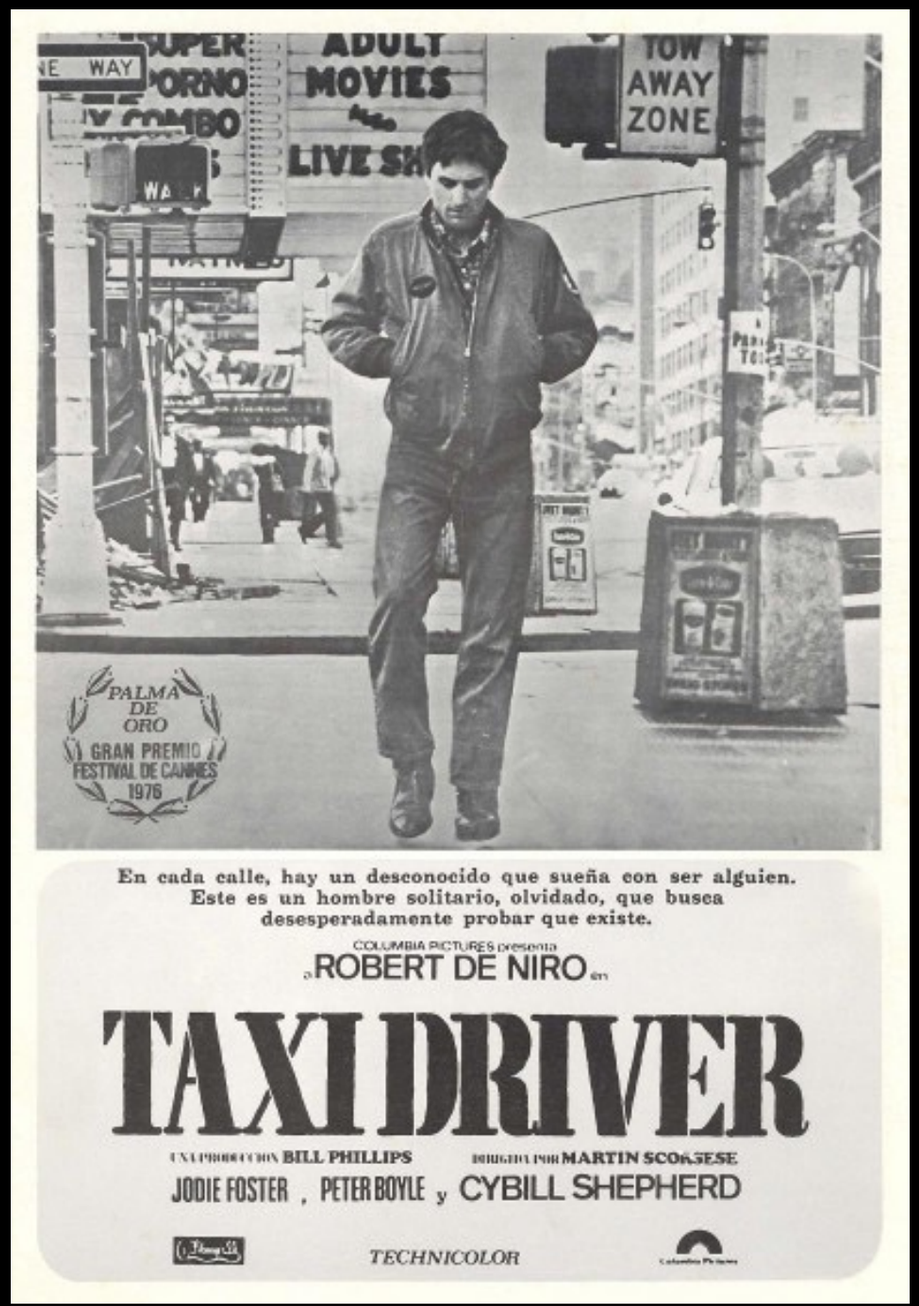
```
1  #include <iostream>
2  #include<array>
3  #include<vector>
4  #include <fstream>
5  #include <sstream>
6  using std::cout;
7  using std::endl;
8  using std::array;
9  using std::vector;
10 using std::ifstream;
11 using std::ofstream;
12 int main () {
13 std::vector <int> mpg = {22, 17, 22, 20, 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};
14 double mymat[5][2] = { { 2930,186 },  { 3350,173 },  { 2640,168 },  { 3250,196 },  {
   4080,222 } };
15 double myglob = 2;
16 cout << "Now running the Fuel Efficiency Boosterizer"  << endl;
17 cout << "We will multiply mpg by: " << myglob << endl;
```

```
18  std::vector <int> mpg2 = mpg;
19  for(int i=0;i<mpg.size();i++) {
20  mpg2[i] = mpg[i]*myglob;
21  }
22  double mymat2[1][2]= {{mymat[0][0], mymat[0][1]}};
23  // send var mpg2
24  // send matrix mymat2
25  ofstream wfile;
26  wfile.open("output.do",ofstream::out);
27  wfile << "input mpg2" << endl;
28  for(int i=0; i<=(mpg2.size()-1); i++) {
29  wfile << mpg2[i] << endl;
30  }
31  int ncells; int ncols; int nrows;
32  ncells = sizeof(mymat2)/sizeof(double);
33  ncols = sizeof(mymat2[0])/sizeof(double);
34  nrows = ncells/ncols;
35  wfile << "matrix mymat2 = [";
36  for(int i=0; i<nrows; i++) {
37  for(int j=0; j<ncols; j++) {
38  wfile << mymat2[i][j];
39  if(j<(ncols-1)) { wfile << ", "; }}
40  if(i<(nrows-1)) { wfile << " \\ "; }}
41  wfile << "]" << endl;
42  wfile.close();
43  return 0;
44  }
```

```
input mpg2
44
34
44
40
30
36
52
40
32
38
28
28
42
58
32
70
36
62
36
46
82
50
50
34
matrix mymat2 = [2930, 186]
```

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

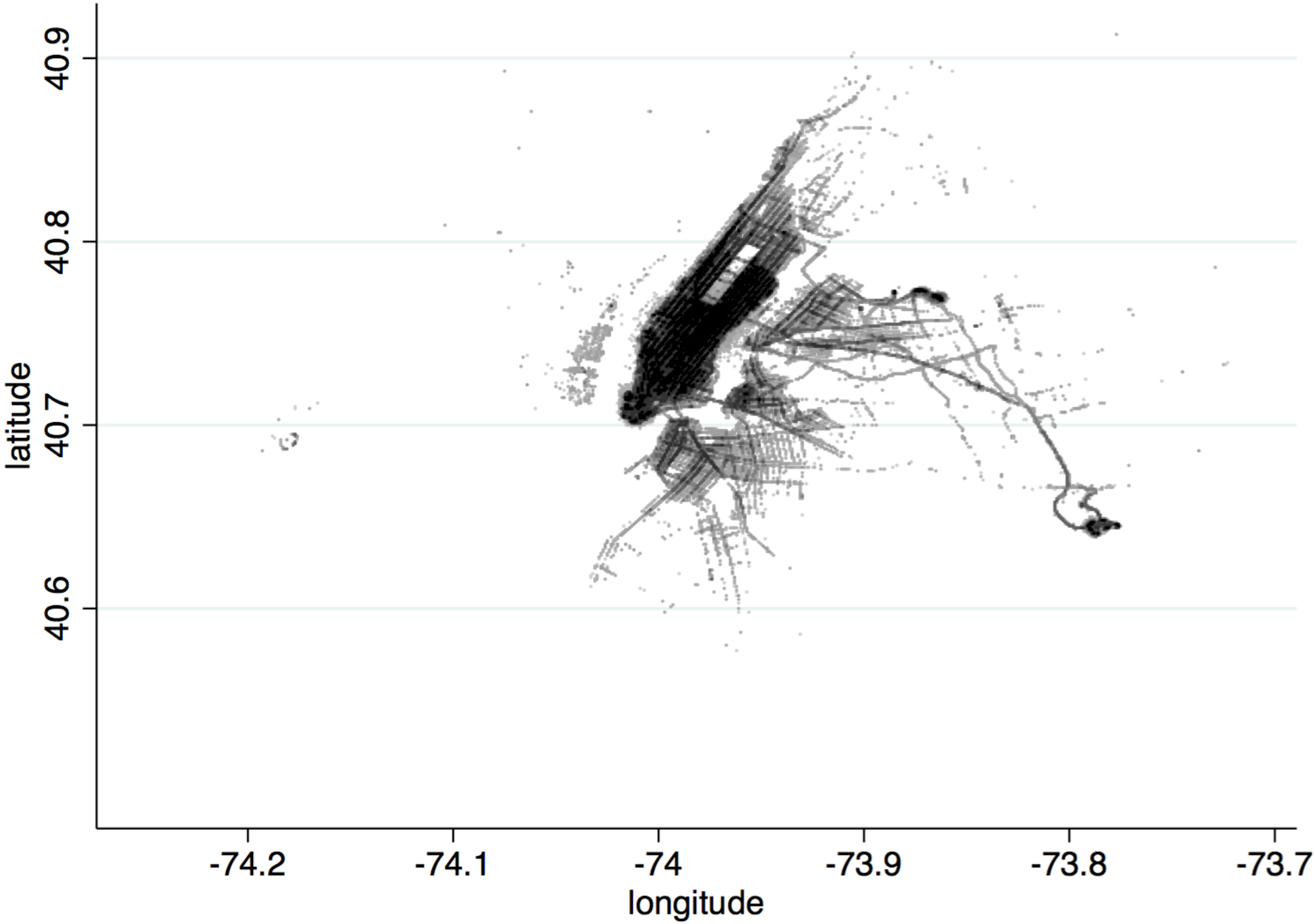


NYC taxi data

- 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

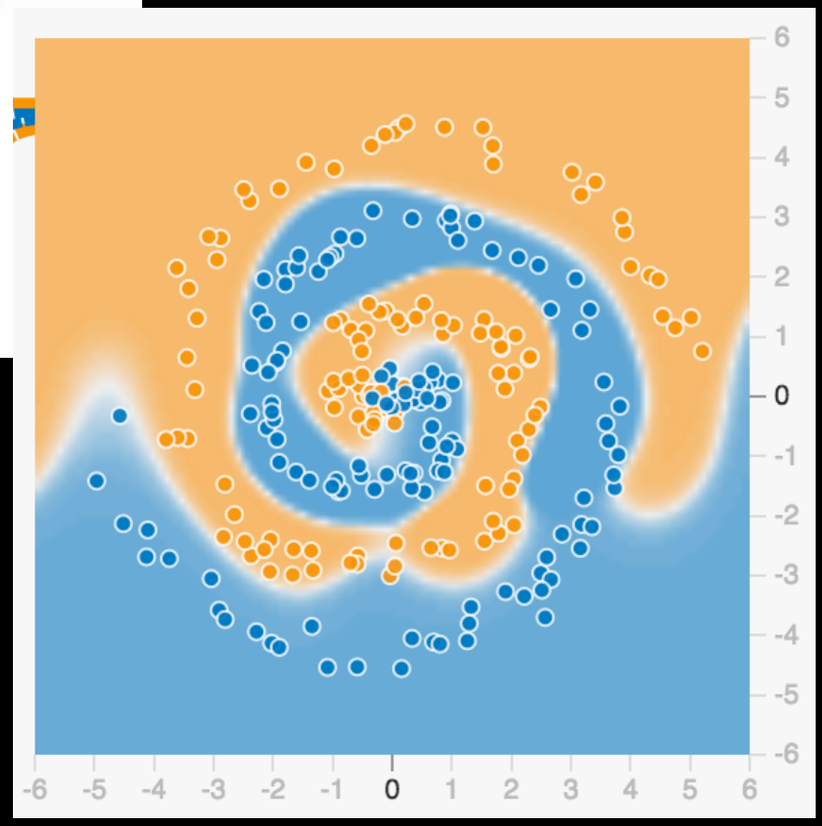

latitude
40.9
40.8
40.7
40.6
-74.2
-74.1
-74
-73.9
-73.8
-73.7
longitude

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)

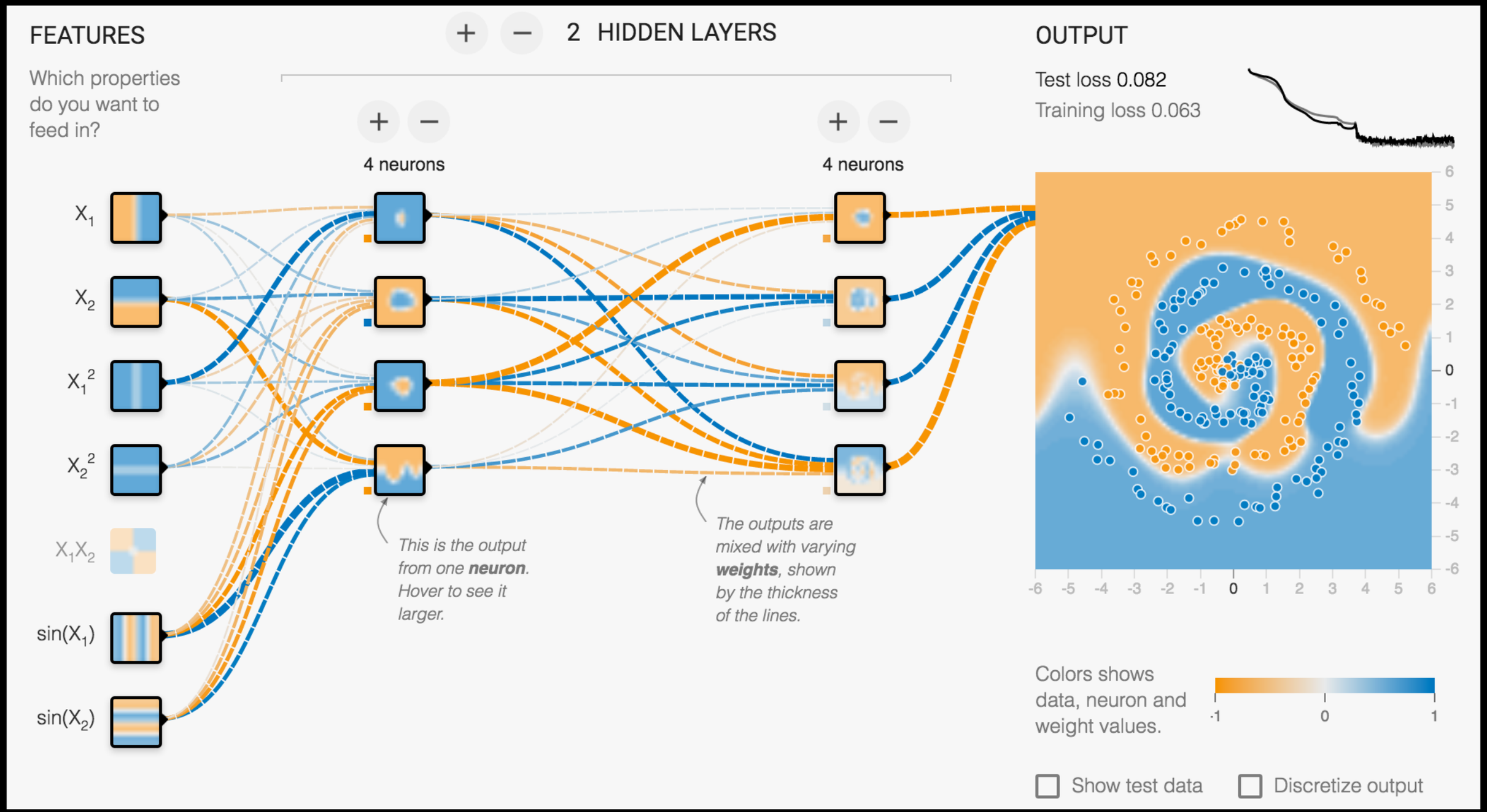


playground.tensorflow.org

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



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 [=‘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