

D3 and Ufora

Dr Jonathan Roberts

When do you use D3?

- During analysis?
 - exploratory plots are probably made in matplotlib or matlab or R
- D3 comes in when we want to make it pretty.
- What if we could use D3 as part of the data exploration?

What's a Ufora?

- Fora is a language that's perfectly suited for parallel computing
- Ufora (the company) develops Fora and the runtime that underpins the parallelism

But I don't code in parallel!

- Fora parallelises for you
- Fora runs on clusters (or across the cores of a single machine) and distributes the work, and the data, without you having to think about it.

It can be a cloud computer - so there's a browser based IDE

The screenshot displays the UFORA browser-based IDE interface. The top navigation bar includes 'UFORA', 'PROJECTS', 'DATASETS', 'HELP', and 'CORES' (8 ACTIVE · 32 DESIRED). Two performance graphs are visible in the top right corner. The left pane shows R code for data processing and visualization. The right pane displays a table of transaction data with columns: #, LINE, HOUSEHOLD_ID, TRANSACTION_NBR, TRANSACTION_TOTAL, and TRANSACTION_DATE. Below the table is a pagination control and a 'Vector info' link. The bottom right pane shows a list of execution results for specific lines of code, each indicating '6 cores' usage.

```
1 let firstPass = parsing.parseCSV(ISMSdata_csv);
2 let transactions = firstPass.data;
3 transactions.filter({_.QUANTITY > 1})
4
5 let groupsByAge = sorting.reduce(
6   transactions,
7   fun(row) {row.AGE_H_HEAD},
8   fun(group) { [] },
9   fun(group, el) { group :: el },|
10  fun(group1, group2) { group1 + group2 }
11 );
12 groupsByAge = groupsByAge.filter({_.group != "NA"})
13 groupsByAge
14
15
16 let spendingByAge = fun(group){
17   (Float64(group[0]), group.reduction.sum({ max(_.EXTENDED_PRICE, 0) })))
18 };
19 let spendingGroups = groupsByAge ~ spendingByAge;
20 let spendingByAgePlot = barPlot(spendingGroups);
21
22 let underForty = spendingGroups.filter({_[0] < 42});
23 let overForty = spendingGroups.filter({_[0] >= 42});
24
25 let (predictors, responses) = Regressors.VectorToMatrices(underForty).convert
26 let regressor = Regressors.Simple(predictors, responses);
27 let underFortyPlot = regressor.regressionPlot(barPlot)
28
29 let (predictors, responses) = Regressors.VectorToMatrices(overForty).convert
30 let regressor = Regressors.Simple(predictors, responses);
31 let overFortyPlot = regressor.regressionPlot(barPlot);
32
33 let regressionPlot = spendingByAgePlot + underFortyPlot + overFortyPlot;
34 regressionPlot
35
36
37 let gatherByHousehold = fun(rows) {
38   sorting.reduce(
39     rows,
40     fun(row){ row.HOUSEHOLD_ID }
```

#	LINE	HOUSEHOLD_ID	TRANSACTION_NBR	TRANSACTION_TOTAL	TRANSACTION_DATE
0	402	100102766	9	12	"29JUN2004
1	938	100262709	14	14	"14JUN2003
2	947	100266348	1	6	"17DEC1995
3	964	100275292	5	6	"12JAN2001
4	1001	100281706	6	37	"27DEC2002
5	1076	100285755	9	34	"20MAY2002
6	1081	100285755	14	34	"25JUN2002
7	1101	100285755	34	34	"11DEC2002
8	1179	100316953	14	19	"11MAR2001
9	1180	100316953	15	19	"11MAR2001

« 1 2 3 4 5 6 » Vector info ...

LINE 13 · RESULT · LOG · STATS · STACKTRACES · TIME TO COMPUTE: ...
6 cores X

LINE 34 · RESULT · LOG · STATS · STACKTRACES · TIME TO COMPUTE: ...
6 cores X

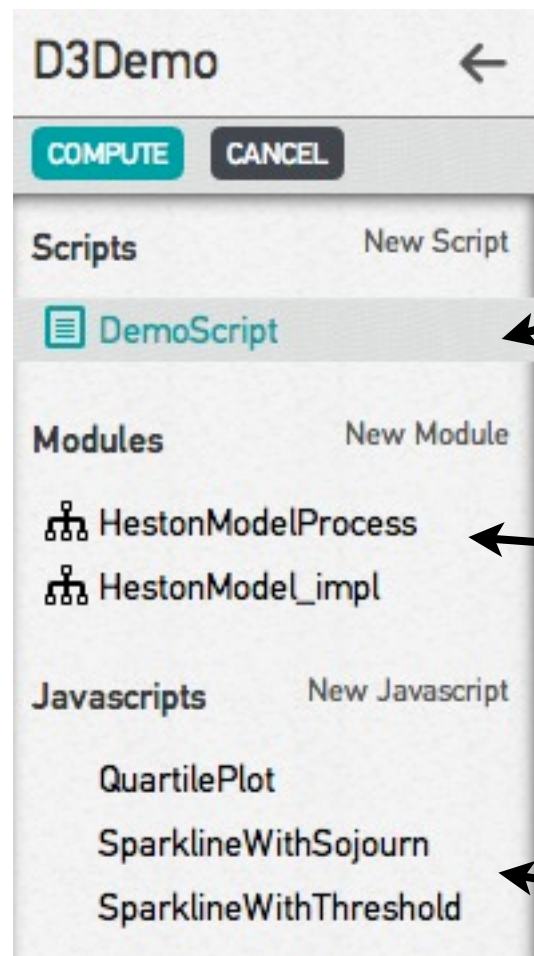
LINE 55 · RESULT · LOG · STATS · STACKTRACES · TIME TO COMPUTE: ...
6 cores X

LINE 76 · RESULT · LOG · STATS · STACKTRACES · TIME TO COMPUTE: ...
6 cores X

Enter D3

- As the data is transferred via json, why not integrate D3 at the heart of the IDE?

Code Javascript as a Module



Fora Scripts

Fora Modules

Javascript Modules

and just call it as a function

The screenshot displays the D3Demo application interface. On the left, a sidebar contains sections for Scripts, Modules, Javascripts, and Datasets. The 'Modules' section is expanded, showing 'HestonModelProcess' and 'HestonModel_impl'. The 'Javascripts' section is also expanded, showing 'QuartilePlot', 'SparklineWithSojourn', and 'SparklineWithThreshold'. The 'Scripts' section shows 'DemoScript' as 'Unsaved'. The main area displays a code editor with the following code:

```
1 let nsteps=100;
2 let texpiry = 2; // years
3 let dt = 2.0/Float64(nsteps); // daily time step
4 let seed = 365244;
5
6 let realization1 = iter.toVector(
7   iter.subseq(
8     HestonModelProcess.underlying_in_case4(seed, texpiry, dt),
9     0, nsteps
10  );
11
12 SparklineWithSojourn([realization1],
13   (
14     xlabel:"time",
15     ylabel:"dollars",
16     title:"Single Sample (click for sojourn time)"
17   ))
```

Arrows point from the labels 'Module', 'Function Call', and 'Output!' to the corresponding parts of the interface. 'Module' points to 'SparklineWithSojourn' in the Javascripts list. 'Function Call' points to the function call in the code editor. 'Output!' points to the line chart on the right, which shows a fluctuating line representing 'dollars' over 'time'.

Module

Function Call

Output!

Just Like That?

- There are a couple of tricks to it:
 - The function gets called once when the line of code executes
 - Therefore the modules are a single function
 - The function takes an ID that specifies the div.


```
function DrawSparkline(elementSelector, lines, options){  
  
  var opts = {  
    "width":550,  
    "height":350,  
    "margin":40,  
    "xlabel":"","  
    "ylabel":"","  
    "title":""  
  };  
  
  for(var k in options){  
    if(k in opts){  
      opts[k]=options[k];  
    }  
  }  
  
  $(elementSelector).empty();
```

Wrapper function
(with div selector)

Use defaults,
or overwrite with inputs

Clear the div
(we have jQuery too!)

Then write D3 like you would normally!
(see demo and github for the live code)

To find out more

- Sign up for updates about Ufora:

www.ufora.com/subscribe

- Check out the GitHub for the D3 code
(coming soon after this meetup)

<https://github.com/ufora/d3plots>