SAS® GLOBAL FORUM 2016

IMAGINE. CREATE. INNOVATE.

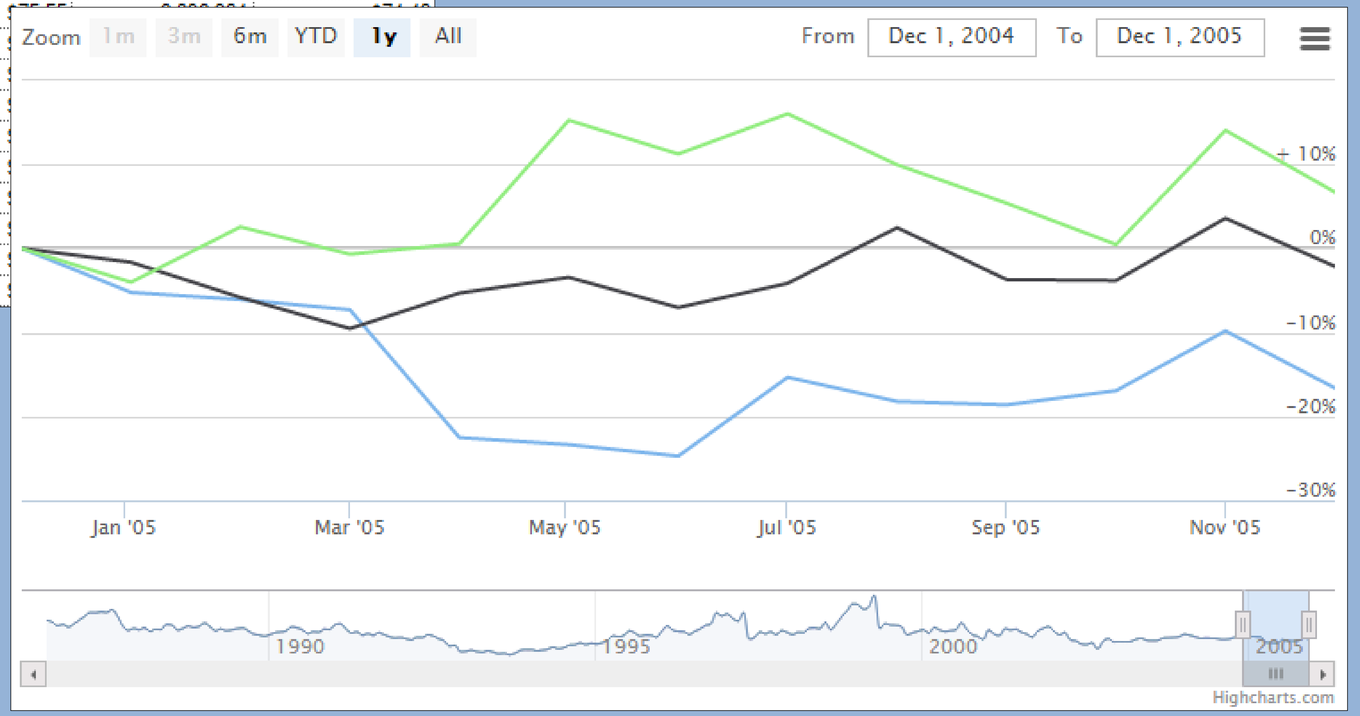
How to Visualize SAS® Data with JavaScript Libraries like HighCharts and D3

Presented by Vasilij Nevlev

#SASGF

How to transform sashelp.stocks in two data steps?

	Stock	Date	Open	High	Low	Close	Volume	AdjClose
1	IBM	01DEC05	$89.15	$89.92	$81.56	$82.20	5,976,252	$81.37
2	IBM	01NOV05	$81.85	$89.94	$80.64	$88.90	5,556,471	$88.01
3	IBM	03OCT05	$80.22	$84.60	$78.70	$81.88	7,019,666	$80.86
4	IBM	01SEP05	$80.16	$82.11	$76.93	$80.22	5,772,280	$79.22
5	IBM	01AUG05	$83.00	$84.20	$79.87	$80.62	4,801,386	$79.62
6	IBM	01JUL05	$74.30	$85.11	$74.16	$83.46	8,056,590	$82.23
7	IBM	01JUN05	$75.57	$77.73	$73.45	$74.20	6,439,536	$73.10
8	IBM	02MAY05	$76.88	$78.11	$72.50			
9	IBM	01APR05	$91.49	$91.76	$71.85			
10	IBM	01MAR05	$92.64	$93.73	$89.09			
11	IBM	01FEB05	$93.67	$94.97	$91.55			
12	IBM	03JAN05	$98.97	$99.10	$91.44			
13	IBM	01DEC04	$94.50	$99.00	$94.47			
14	IBM	01NOV04	$89.33	$96.63	$89.23			
15	IBM	01OCT04	$85.95	$90.27	$84.29			
16	IBM	01SEP04	$84.05	$87.28	$83.24			
17	IBM	02AUG04	$86.87	$87.39	$81.90			



How to Visualize SAS® Data with JavaScript Libraries like HighCharts and D3

Presented by: Vasilij Nevlev, Analytium Ltd

Introduction

The visualisation is done by a JavaScript library called HighCharts. JavaScript library is a collection of code that is executed to modify HTML page in some way.

To achieve the visualisation shown on previous slide, two stored processes must be defined with two data steps as their code content. We are using two stored processes in order to implement Model-View framework, where one stored process is responsible for providing data (Model) and the other stored process is providing visualisation (View). The elements are highly reusable because of separation of concerns, as the same model could be linked to a number of visualisations and same visualisation can be linked to a number of models.

1st Stored Process (Model)

First stored process (Model) provides the data in JSON format. JSON stands for JavaScript Object Notation . It is a lightweight data-interchange format. It is easy for humans to read and write and it is easy for machines to parse and generate. JSON is a subset of JavaScript.

There are two main methods to generate JSON. If you are on SAS 9.4, then the simplest way is to use PROC JSON. Another way is to use put statements to direct SAS to output the data in JSON format.

Please see the next slide for numbered SAS code.

Lines 1-3 are filtering and sorting the data. You might have noticed a parameter or a macro variable called “stock”. This parameter is passed by the second stored process. We will come back to it later.

Line 5 instructs SAS that we are not actually creating a data set, so we are using _null_ for the dataset name.

Line 6 instructs SAS to read in the sorted data while setting a marker for end of file in a form of EOF Boolean which will get used later. Line 7 instructs SAS to output to a SAS reserved _webout fileref that then gets send to the web browser.

Line 8 produces date in JavaScript format. JavaScript counts number of milliseconds from 1970. INTCK function is used to count the number of days between 01/01/1970 and stock trade date, then it gets multiplied by the number of milliseconds in one day.

Lines 9 – 12 actually produce the JSON data where line 9 outputs the first name pair that is used by second stored process to identify which stock is described. Line 9 also opens an array of arrays which will get populated with name pairs described on line 10. These name pairs contain the date in milliseconds and the stock close price. Lines 11-12 either separate name pairs with comma or close the arrays if the end of file has been reached. End of file is marked by Boolean EOF which was defined earlier.

The actual code and the results are shown on the next slide.

2nd Stored Process (View)

Second stored process (View) visualises the data. In essence, it is sends an html file to the user’s browser that then downloads necessary JavaScript libraries and calls the first stored process using AJAX (Asynchronous JavaScript and XML) method. For the full code, please scroll two slides down.

Lines 1-2 instruct SAS, that just like in first stored process, we do not intend to create a data table, but we want to output to a reserved fileref _webout. Put statement on Line 3 instructs SAS to output anything that follows into the _webout.

The rest of the lines are pure HTML and JavaScript that make up the functionality of the webpage and therefore of the visualisation. Line 4 opens the body and the head of the HTML document. Lines 5-7 load JavaScript libraries that provide the functionality for visualisation. The libraries are jQuery, HighCharts and HighChartsExport. jQuery simplifies JavaScript syntax by introducing shorthands. HIghCharts introduces visualisation functionality and HichChartsExport introduces file export functionality to the graphs generated by HighCharts. Lines 8-23 defines optional configuration, such as line colours, array names, comparison type.

Lines 24-30 set out a do loop in Java script that instructs the browser to open the first stored procedure and request the data for each of the stock names provided. This allows for a large amounts of data loaded quickly. The requested stock name is the parameter “stock” used in the very beginning by the first stored process.

Line 31 closes the custom script and the html head section. Line 32 sets out a div html element that will be modified by JavaSctipt to contain our visualisation. We passed the name of the element to JavaScript on line 13. Line 33 closes HTML elements and instructs SAS to run the code. Now if you open the location of the second stored process in your browser, you should see visualisation. The results could be seen on slide 6 that contains a recording of the final output.

There are hundreds, if not thousands open source libraries out there. Many of them offer countless versions of visualisations, tools and utilities. The same method and technique can be applied to any of them. It opens the door for a very cost effective and flexible data exploration, dashboards and tools that previously were either very laborious to build with SAS or outright impossible.

Contact Information:

Any questions in regards to this presentation can be send to author: Vasilij Nevlev
Email: vasilij.nevlev@analytium.co.uk
Tel: +44 20 03826 8827

Analytium Ltd
4th Floor, 86-90 Paul Street
London, EC2A 4NE
United Kingdom
Tel. +44 20 03826 8827
Email: enquiries@analytium.co.uk

1st Stored Process Code (Model): Getting data into JSON.

```
1.    proc sort data=sashelp.stocks(where=(Stock="&Stock")) out=stocks_sorted;
2.       by date;
3.     run;
4.
5.     data _null_;
6.        set stocks_sorted end = eof;
7.        file _webout lrecl=20480;
8.        Date = INTCK("days","01JAN70"d,Date)*86400000 ;
9.        if _N_ eq 1 then put '{ "name":'"""&Stock"""',"data":[';
10.       put '[' Date ',' Close']' ;
11.       if eof then put ']}';
12.       else put ',';
13.   run;
```

Output: {"name":"IBM","data":[[523238400000,138.75],[526003200000,134.50],[528508800000
,123.62],[531360000000,127.12],[533779200000,120.00],[536544000000,128.75]...]}

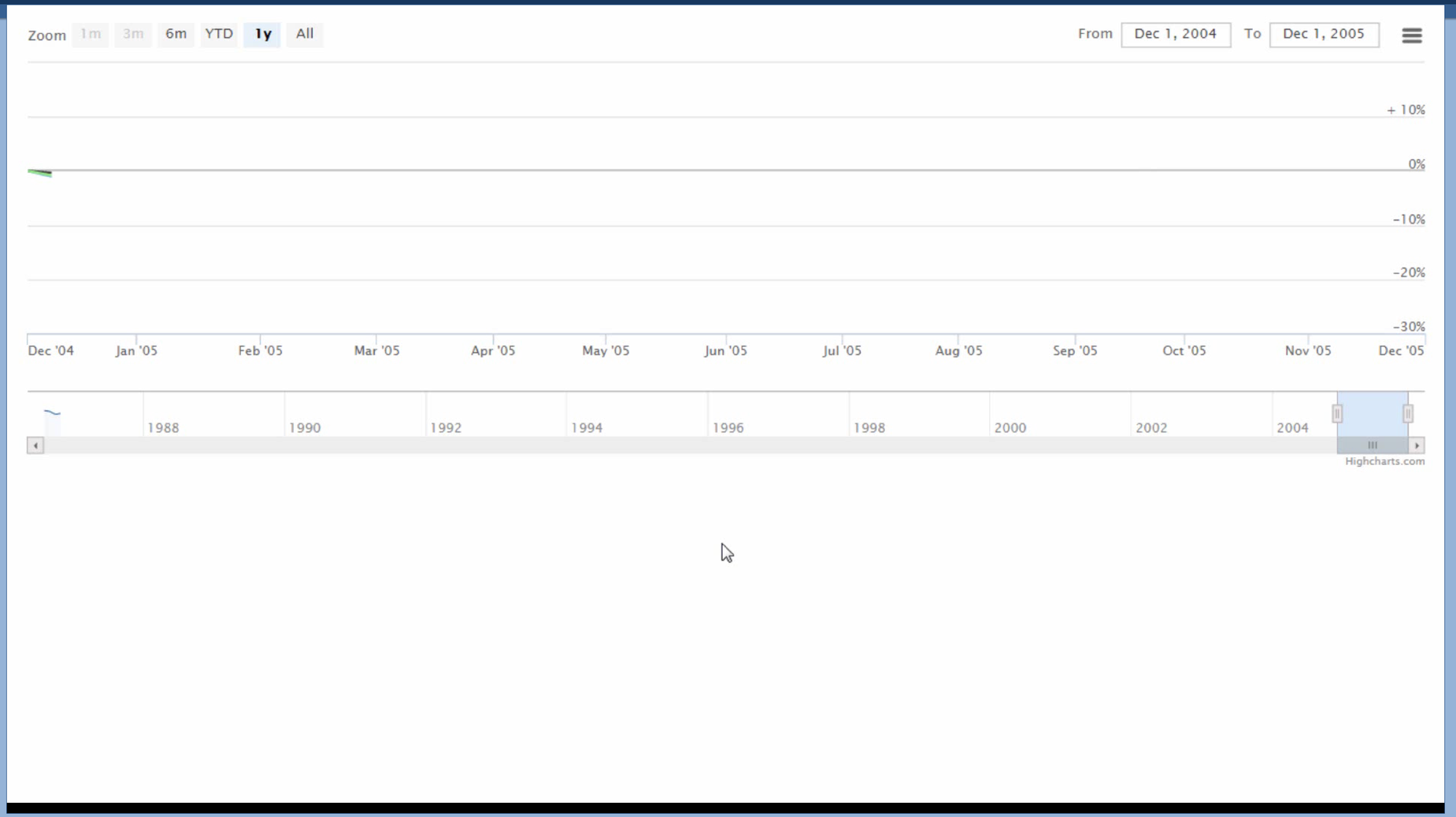
Go back to code description

2nd Stored Process Code (View): Visualising JSON.

```
1.  data _null_;
2.      file _webout;
3.      put
4.      '<html><head>' /
5.      '<script src="http://code.jquery.com/jquery-1.11.3.min.js"></script>' /
6.      '<script src="http://code.highcharts.com/stock/highstock.js"></script>' /
7.      '<script src="http://code.highcharts.com/stock/modules/exporting.js"></script>' /
8.      '<script type="text/javascript">' /
9.         '$(function () {var seriesOptions = [],'/
10.            'seriesCounter = 0,'/
11.            'names = ["IBM", "Microsoft", "Intel"],'/
12.            'createChart = function () {'/
13.                '$("#container").highcharts("StockChart", {'/
14.                   'rangeSelector: {'/
15.                       'selected: 4},'/
16.                          'yAxis: {labels: {
17. formatter: function () {return (this.value > 0 ? " + " : "") + this.value + "%";}},'/
18.                       'plotLines: [{'/
19.                          'value: 0,'/
20.                          'width: 2,'/
21.                          'color: "silver"}]},'/
22.                    'plotOptions: {series: { compare: "percent"}},'/
23.                        'series: seriesOptions});};'/
24.     '$.each(names, function (i, name) {'/
25.         '$.ajax({ type: "GET" , url:"/SASStoredProcess/do?_program=/getStockData&Stock=" + name, dataType: "JSON",
26.             success: function (data) {'/
27.             'seriesOptions[i] = data;' /
28.             'seriesCounter += 1;' /
29.             'if (seriesCounter === names.length) {'/
30.                 'createChart();}}});});});' /
31.     '</script></head><body>
32.     <div id="container" style="height: 400px; min-width: 310px"></div>` /
33.     '</body></html>';run;
```

Go back to code description

Result (Video Demo)



SAS® GLOBAL FORUM 2016

IMAGINE. CREATE. INNOVATE.

LAS VEGAS | APRIL 18-21

#SASGF