Looking for Correlations from Mongo DB

How it make statistics simpler

Shulin Kang

Inspiration

Think about the life of stock statisticians without support of DBMS in statistics software (Excel, SPSS) if they want to do similar work:

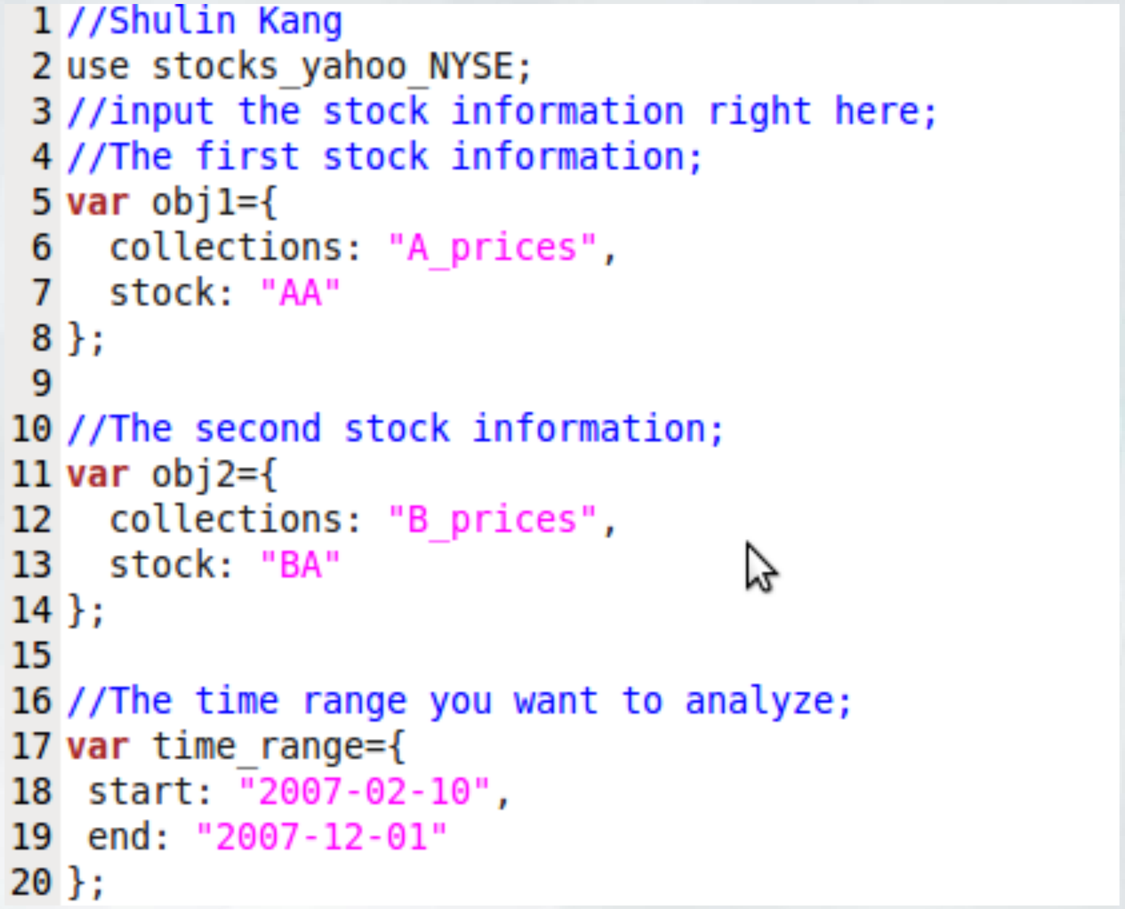
1. Open huge data file
2. Copy two target stock into a same file
3. Mange the file naming if need to save
4. Select the data with in the targeted range, and then analyze the data
5. If the output is not what I want, re-do Step 4
6. ...

What I have done?

Input:

- Time range
- Two stocks symbol

format: .js file into MongoDB console

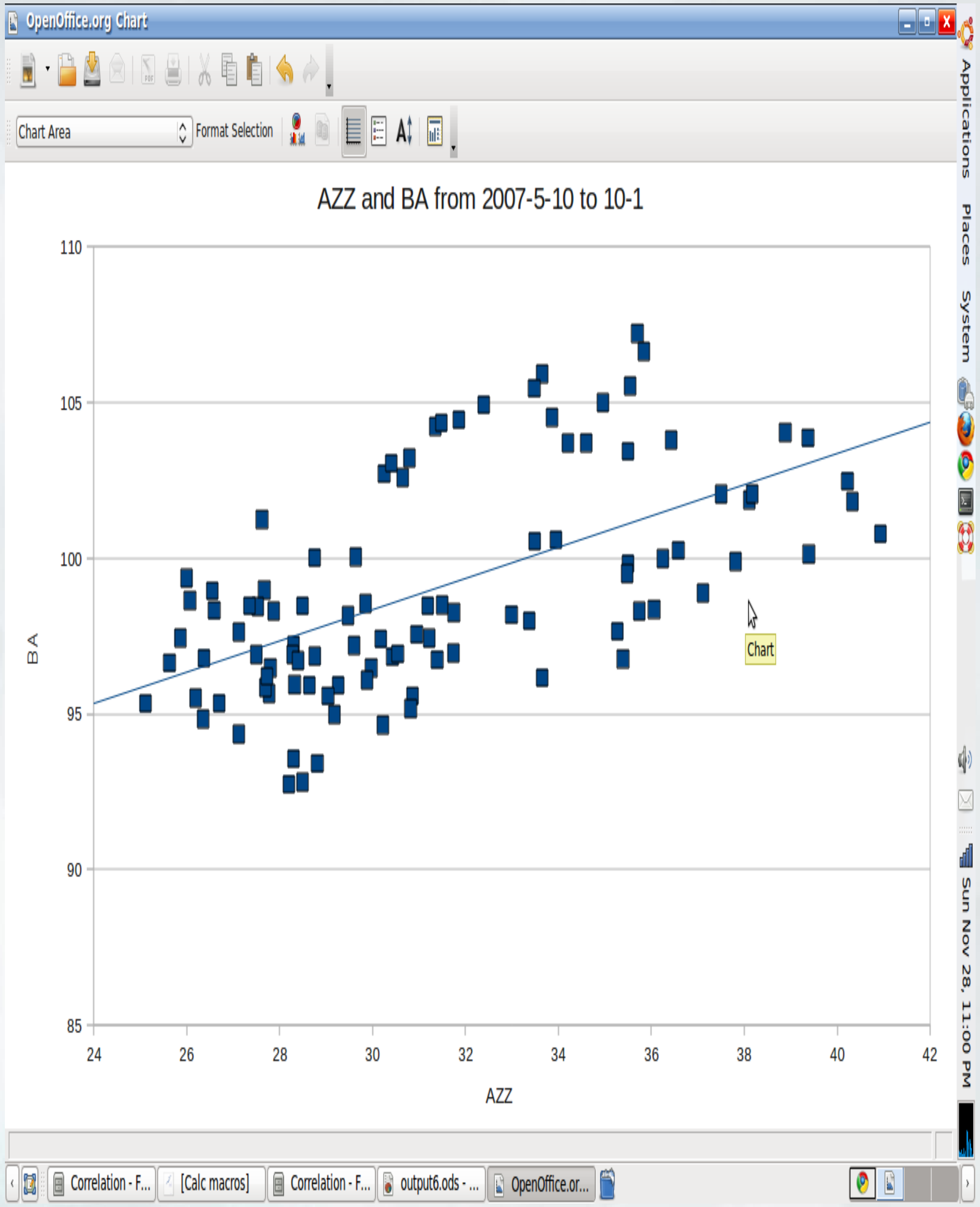


```
//Shulin Kang
use stocks_yahoo_NYSE;
//input the stock information right here;
//The first stock information;
var obj1={
  collections: "A_prices",
  stock: "AA"
};

//The second stock information;
var obj2={
  collections: "B_prices",
  stock: "BA"
};

//The time range you want to analyze;
var time_range={
 start: "2007-02-10",
 end: "2007-12-01"
};
```

What I have done?

Output:

- linear correlation coefficient of the two stocks according to the closing price (from -1 to 1)
- Output the queried data as .xls/.ods so we could graph it as scatter plot picture



Phase 1: A Javascript Correlation Method

A simple web page to test Correlation computation code(.js)

$$r = \frac{N\Sigma xy - (\Sigma x)(\Sigma y)}{\sqrt{[N\Sigma x^2 - (\Sigma x)^2][N\Sigma y^2 - (\Sigma y)^2]}}$$

Where:

N	=	number of pairs of scores
Σxy	=	sum of the products of paired scores
Σx	=	sum of x scores
Σy	=	sum of y scores
Σx^2	=	sum of squared x scores
Σy^2	=	sum of squared y scores

Phase 2: Reuse phase 1 method in the final query file

- Option1: Array Mode in Shell (Use excess memory)

```
db.collections.find({..}).sort({date:1}).toArray();
```

- Option2: Cursors (MongoDB iterator)

```
while (query1.hasNext() && query2.hasNext())
  {
  var x = query1.next().close;
  var y = query2.next().close;
  sumOfProducts += x*y;
  sumOfX += x;
  ...

}
```

Phase 2: Reuse phase 1 method in the final query file Cont...

- Encapsulate query data in Javascript Object:

```
var obj1={
  collections: "A_prices",
  stock: "AA"
};

db[obj1.collections].findOne()  = db.A_prices.findOne()
```

Phase 2: Reuse phase 1 method in the final query file Cont...

```
var time_range={
 start: "2006-02-10",
 end: "2006-12-01"
};

db.collections.find({stock_symbol: obj1.stock, date: {$gte: time_range.start, $lte:
time_range.end}}).sort({date:1})

=db.collections.find({stock_symbol: obj1.stock, date: {$gte: "2006-02-10", $lte: 2006-
12-01}}).sort({date:1})
```

Phase 3 Output the file

- mongo < inputquery.js > output.xls
- Print Table Header

```
print("Date "+ query1[1].stock_symbol + " "+ query2[1].stock_symbol +" ");
```

- Print Data

```
print(query1[i].date + " "+ query1[i].close + " "+ query2[i].close)
```

Demo

Run the query and open the output file

Compare with SQL implementation

+ Rich functions: make use all the power of JavaScript(
Array,   Math, Object)
+ Could be test it in a web browser
+ Not vendor-specific: easy to pick up from a JavaScript lover

- Hard to reuse in a "real" application right now