Report for the backend test

http://riskanalytics.comule.com/finalapp1.php

I have designed an application and deployed it here ... To highlight the features in a short description I would like to mention the following few points.

- A scalable and robust implementation which can reiterate inputs again and again.
- A complete application whose each value is computed from scratch!!!
- A tabulated form of the results which brings in a better sense of understanding.
- A display of charts and and detailed statistics for each and every sector.
- Interactivity!!!
- A clear separation between a server client model . Every page is a combination in form input(n)+ output(n-1) if n is a time step you iterate values.
- Redundancy . As an example dates are crosschecked with the Unix time system and verified to be matched so as to avoid error.

DESCRIPTION

The backend test has a few important elements. The first was to preprocess the data . I preprocessed the csv input files using Matlab and arranged them as per the dates in a csv file called tabulations.csv. The format of the file is as such that it takes each and every sector and product. and generates a matrix which has rows as dates and columns as prices.

The incomplete dates are filled as per the last dates given and these dates are used up as the filling stock prices for each value. I chose Matlab as a platform due to its ease of use in preprocessing in a cell/array format and it generates the data in an easily readable format.

Despite of Python being my current primary language of use I have used PHP for the requirement of the application. PHP is very easy to implement when it comes on the testing and there are hundreds of libraries and other scripting languages compatible with PHP (I do not say Django doesn't have them but a lot of stuff available and the research done on PHP servers is insane!!!).

Coming back to the project I then took users inputs as per the requirements of the test . All defaults are balanced weights and default period is the entire time. Here is a detailed description on that.

INPUTS

- In the inputs section I have taken 10^{15} as the assumed maximum value for the investment for computational purposes.
- Starting and ending dates can be given.
- All Products/Sectors have an option to be chosen and given weights from 1-100(eventually normalized).
- If a Sector is chosen it gets a weight and if it is not the weight turns out to be zero irrespective of the charts displayed.
- The products are assigned values as per sliders with values 1-100 (e.g) if prod1 and prod2 from a sector both have a value of 100 each of them have equal weight.
- I have also defined a superweight which is the total weight to the sector(if and only if it is chosen). As an example if 3 sectors are chosen then the all the weights of these sectors are given by their ratio with the total sum of the 3 sectors(10/10+10+10 = 0.33 if equal weights are chosen and actual weight is 0.33* normalized superweight).
- The weights are displayed using interactive pie charts for user benefits.
- Rebalance frequency is the number of times the portfolio will be rebalanced. I have restricted the rebalance frequency as a very hhigh value is not recommended. A slider is used for entry purposes.

HANDLING THE DATA.

As of now we have the data in a form of a table 'tabulations.csv' . I used this data and loaded this onto the server . Using a form handle I generate all inputs and put them as PHP variables. The real explanation to why I do this is to keep it simple for the user to submit changes once sure of. (the eurostoxx application in the example is dynamic enough but even if I uncheck a box or slide in a value of change a date, there are too many inputs to take care of. It results in a lot of confusion to keep a straight slide and change application for about 20 inputs!!!. The reason why I kept interactive pie charts is to maintain user friendliness. I tried doing it for the first sector and it works but it becomes difficult for the user to understand anything and the objective is lost!!!) .
Now that I had all the input variables and I needed to compute the desired outputs. Below is a brief detail on the output of the applications.

The following outputs are the features at the server . Each output depends on the simulation dates.

- The data is loaded onto an array. Other inputs are the variables.
- The Net returns and the date at every rebalance.
- The Annualized Returns.
- The sharpe ratio.
- The Standard deviation(the ratio on PnL)
- The mean(the ratio on PnL)
- The cumulative PnL graph
- Above statistics for each individual sector
- Maximum Drawdown percentage for each sector.
- The daily Cumulative PnL values for the simulation dates.

All the values can be restored to defaults and they can be computed once again on the submit button of the browser.

PERFORMANCE OF THE APPLICATION

Here(http://tools.pingdom.com/) is the website I used to compute statistics. The Key performance statistics are as follows:

- The website takes about 1.5-3.0 seconds in order to pull up every request.
- Page size= 549.6 Kb
- 18 requests
- Performance grade = 84/100
- 0 redirect errors, 0 client/ server /connection errors
- Faster than 74% of the tested websites.

I have also attached a report from the website speed test as a pdf Doc.

HOW I TESTED/DEBUGGED THE APPLICATION for ROBUSTNESS?

- **Replaced every inputs with Null values, add one, subtract one**
- **Avoided negative values(Try it!!)**
- **Restricted the inputs to ranges.**
- **Replaced every value by maximum and minimums , plus one, minus one**
- **Insert Invalid dates, one more than max, one less than max etc.**
- **Add strings in place of numbers and numbers in place of strings(Wont work either!!)**

APPENDIX

1.)How I computed these values?

i.)Simulation dates: Converted all dates to numbers and the first date was day 1. Calculated from the timestamp of the corresponding dates.

ii.)Net returns: I compute the values on every rebalance frequencieth day. using the PnL percentage for every sector. I find the cumulative value and then I rebalance the distribution. I keep iterating till the final date is achieved.

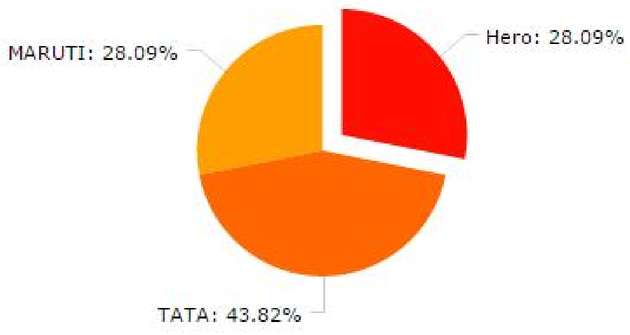
iii.)Annualized returns: (exp((tradable days in a year)*mean(daily_log_return_series))-1)*100 %

iv.) Cumulative PnL: I get the Pnl % across every sector. I then find the total price difference on an everyday basis.

v.) Sharpe Ratio: I use the PnL % across every sector . I then find its mean and standard deviation. I assume a risk free return of 1.78 % annual as per the US treasury stable returns(investopedia). I take the difference of these annualized Pnl % and divide by the standard deviation. I also compute the overall sharpe ratio of the portfolio.

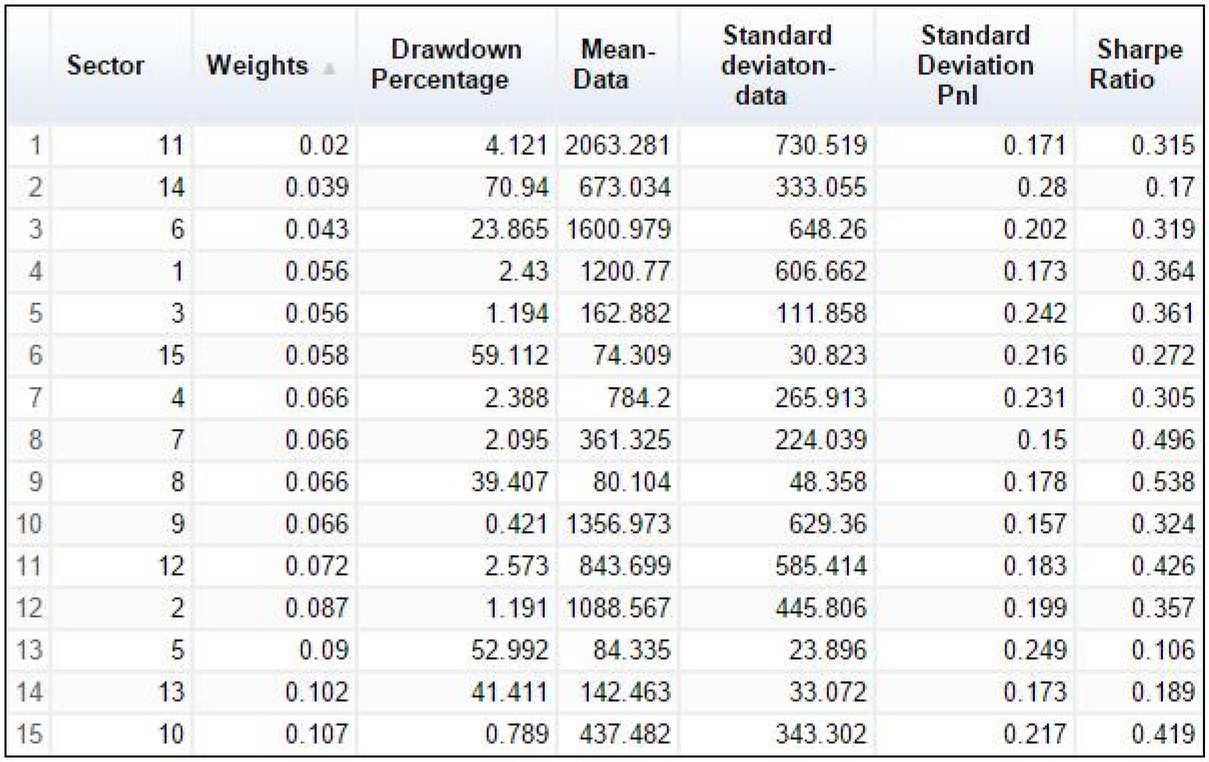
2.) Interactivity and application features.

i.)



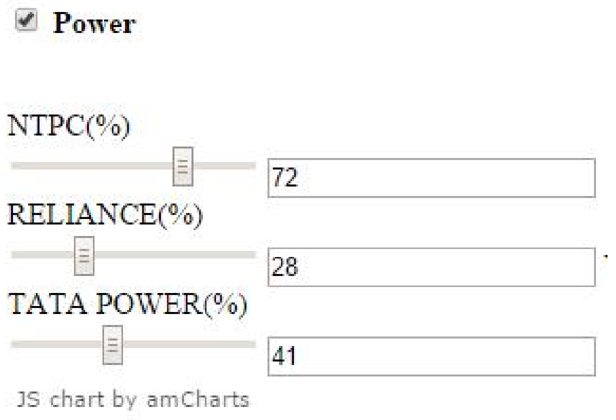
You can click on any of the feature on a sector and find out the weight just by clicking on the sector.

ii.)Sorting. I just click on the weights section and can sort them according to the weights every sector gets. Note that the sum achieved is 1. This can be done for each aspect in the table.



	Sector	Weights ▲	Drawdown Percentage	Mean-Data	Standard deviaton-data	Standard Deviation Pnl	Sharpe Ratio
1	11	0.02	4.121	2063.281	730.519	0.171	0.315
2	14	0.039	70.94	673.034	333.055	0.28	0.17
3	6	0.043	23.865	1600.979	648.26	0.202	0.319
4	1	0.056	2.43	1200.77	606.662	0.173	0.364
5	3	0.056	1.194	162.882	111.858	0.242	0.361
6	15	0.058	59.112	74.309	30.823	0.216	0.272
7	4	0.066	2.388	784.2	265.913	0.231	0.305
8	7	0.066	2.095	361.325	224.039	0.15	0.496
9	8	0.066	39.407	80.104	48.358	0.178	0.538
10	9	0.066	0.421	1356.973	629.36	0.157	0.324
11	12	0.072	2.573	843.699	585.414	0.183	0.426
12	2	0.087	1.191	1088.567	445.806	0.199	0.357
13	5	0.09	52.992	84.335	23.896	0.249	0.106
14	13	0.102	41.411	142.463	33.072	0.173	0.189
15	10	0.107	0.789	437.482	343.302	0.217	0.419

iii.) A complete built from scratch weight balancer. This was simply built to show anyitem can be built from the scratch itself.



NTPC(%)	72
RELIANCE(%)	28
TATA POWER(%)	41

]

3.)Guidelines for Running the code

MATLAB

You need to install a version of matlab and run the main.m file . It contains a function call to arrange.m to arrange all the given data in cell formats . The job done is collecting the data. Rearranging it, modifying the dates to readable days, filling gaps and concatenating the data to a single matrix and eventually a csv file.

PHP

There are three relevant files which are webapp.php, webappresponse.php and tabulations.csv which run on the server . The job done in these files is giving the inputs and adjusting the data matrix into an array of variables. Adjusting all these variables and importing them into php readable formats . After these datasets were available in arrays , I computed net returns with equal rebalance dates. All dates were converted into readable formats. The same table was used for computing the annualized returns , the drawdown percentage and the daily cumulative values.

4.) References for computations

http://www.investopedia.com

http://en.wikipedia.org

http://www.eurexchange.com

http://tools.pingdom.com/

http://www2.informatik.huberlin.de/~wolter/teaching/seminar07/023_assessingRobustnessOfWebServices.pdf