Export PDF (http://account.quantjudge.com/lib/htmltopdf.php?id=18&sar=c4kg&qm=cqcm_8724&name=_John Lewis)

QuantJudge

Evaluation report: Quant consultant

John Lewis

Candidate
ID : 56
Mail : hh820731@gmail.com
Phone :
Degree : Msc Quantitative Finance
School/University : Imperial college
Experience : 1
Your comment :
Candidate comment: .

Session
ID session : c4kg
Started : 2017-05-02 21:42:09
Finished : 2017-05-02 22:28:41

Test	Test type	Absolute score	Relative score
1. Quantitative Finance	MCQ	3 / 6	50%
2. [C++] VAR ,Expected shortfall	Coding	5 / 5	79%
3. [C++] Implied volatility	Coding	5 / 5	85%

Global score

13 / 16

1. Quantitative Finance

1-1. Details MCQ

Question	Candidate answer	Correct answer	Marks
Question 1 "Consider 3 Call options of the same maturity T and with different strikes K1=10, K2=20 and K3=30.If we buy today one call with strike K3 and one with strike K1 and sell two call options with the strike K2, should this cost us money or should we paid for that ? ":	Yes, this must cost us money.	Yes, this must cost us money.	1
Question 2 "Compute $E[c^{\sigma W_t}]$ where W is a Brownian motion and ? is the volatility. Select the from the following: (a) 1 (b) e^{σ} (c) $e^{\frac{1}{2}\sigma^2 t}$ ":	c	c	1
Question 3 "Compute the following covariance $Cov(\int_0^t W_r dr, \int_0^s W_r dr)$ for 0 <s <t, select the right answer from the following: (a) 0 (b) $\frac{1}{2}s^2$ (c) $\frac{1}{2}t^2$ (d) 1 ":	b	a	0
Question 4 "Compute the following covariance $Cov(\int_0^t rdW_r, \int_0^s rdW_r)$ for 0 < s < t: Select the	c	c	1

right answer from the following: (a) 0 (b) $\frac{1}{2}s^2$ (c) $\frac{1}{3}s^3$ (d) 1 ":			
Question 5 "Compute the following integral for a given x : $\int_0^x (t-x)e^{-t}dt$ Select the right answer from the following: (a) $1-e^{-x}$ (b) xe^{-x} (c) $1-x-e^{-x}$ (d) None of this ":	a	c	0
Question 6 "Suppose X is a random variable with : $E[X^2] < \infty$ What is the constant C that minimizes: $E[(X-C)^2] < \infty$ Select the right answer from the following: a) E[X] b) $E[X^2]$ c) Var(X) d) None of this ":	d	a	0

2. [C++] VAR ,Expected shortfall

2-1. Subject coding

VaR computation test
1- Description:
This test consists on computing the VaR of a portfolio with the historical method. Given the historical daily PnL of the given portfolio, the objective is to compute the Value at Risk for a given level of confidence(95% for example). A skeleton of the code is provided. .

2- Objective:
The objective is to implement the method **var(...)** that returns the VaR given the history of daily PnLs at a given confidence level(95% for example).

3- Provided code:
You only need to implement the var method. **4- Important remarks:**
- In its final version, your code should not print anything to the console.You can print values only for debugging purpose. Please make sure that you remove any printing instructions

2-2. Unit test cases

Test name	Candidate Output Value	Correct output	Execution time	Memory consumption	Result
Unit test1	-3.794	-3.794	0.00 sec	0 KB	Passed
Unit test2	-2.35674	-2.35674	0.00 sec	0 KB	Passed
Unit test3	-1.60828	-1.60828	0.00 sec	0 KB	Passed
Unit test4	-1.24225	-1.24225	0.00 sec	0 KB	Passed
Unit test5	-0.811638	-0.811638	0.00 sec	0 KB	Passed

2-3. Candidate Source code

Main File: Risk.h

before submitting your final version.

- Press the button Execute to launch a unit test of your solution.

Language : cpp
Compilation:Successfully
Marks Scored: 5/5

```
// Histirical VAR



#include <iostream>
#include <iomanip>
#include <vector>
#include <math.h>
#include <stdlib.h>
#include <time.h>
#include <algorithm>

using namespace std;

//VAR compute the historical VAR for one day for the portfolio with Pnl as dai
ly pnl vector
    // pnl is the daily pnl vector
    // threshold is the var thresold ex 0.95

double var(vector<double> pnl,double threshold) {

    std::sort(pnl.begin(),pnl.end());
    size_t SizeSample = pnl.size();
    double rank_min = floor((1-threshold)*SizeSample);
    return pnl.at(rank_min);


    return 0 ;
}
```

3. [C++] Implied volatility

3-1. Subject coding

Black Scholes Implied vol [C++]
1- Description:
This test consists of creating a European option implied volatility, under the Black-Scholes model (extended for dividends by Merton). A skeleton of this implied volatility tool is provided.

2- Objective:
The objective of this test is to Implement the following function BSImpliedVol (...) to return the implied volatility of the option (a function BS (...) giving the price by the closed formula is provided to help you).

3- The skeleton code provided:
NormalFunctions.h // Math / utility function
Volatility.h // the file BSImpliedVol based implied vol

3-2. Unit test cases

Test name	Candidate Output Value	Correct output	Execution time	Memory consumption	Result
Unit test1	0.25	0.25	0.00 sec	0 KB	Passed
Unit test2	0.25	0.25	0.00 sec	0 KB	Passed
Unit test3	0.25	0.25	0.00 sec	0 KB	Passed
Unit test4	0.250154	0.25	0.00 sec	0 KB	Passed
Unit test5	0.449998	0.45	0.00 sec	0 KB	Passed

3-3. Candidate Source code

Main File: Volatility.h

```
// Implied volatility  Black-Scholes



#include <iostream>
#include <iomanip>
#include <vector>
#include <math.h>
```

4- Note:
1- the only work to do is to complete the implementation of the BSImpliedVol function (...) File Volatility.h

2-Your code does not print the screen unless you want to test / debuger in this case make sure you remove these impressions screen before submitting your solution

3- The **run** button allows you to run a unit test on your solution

Language : cpp
Compilation:Successfully
Marks Scored: 5/5

```
#include <stdlib.h>
#include <time.h>
#include <algorithm>
#include "NormalFunctions.h"
using namespace std;

//BSImpliedVol compute the implied volatility of an european option using
 Black Scholes model
    // S Spot Price
    // optPrice market option Price
    // v Volatility
    // T Maturity in Years
    // rf Interest Rate
    // q Dividend  yeild
    // PutCall 'P' for put and 'C' for call
    // Nsims Number of simulations

double BSImpliedVol(double S,double K,double optPrice,double T,double
rf,double q,char PutCall) {

size_t N = 40;
size_t i(0);
double x = 0.;
double y = 1.;
double z = 0.;
double sigma = 0.;

if(PutCall == 'C')
{
for(; i < N; ++i)
{
    z = 0.5*(x+y);
    double temp_v = z/(1-z);
    if(optPrice > BS(S,K,temp_v,T,rf,q,'C'))
    {
        x = z;
    }
    else
    {
        y = z;
    }
}

z = 0.5*(x+y);
sigma = z/(1-z);
}
else if(PutCall == 'P')
{
for(; i < N; ++i)
{
    z = 0.5*(x+y);
    double temp_v = z/(1-z);
    if(optPrice > BS(S,K,temp_v,T,rf,q,'P'))
    {
        x = z;
    }
    else
    {
        y = z;
    }
}

z = 0.5*(x+y);
sigma = z/(1-z);
```

```
    }

    return sigma;
}
```