0. Contents

1. The Problem

Value at Risk, or VaR, for short, is a useful measure of the riskiness of a stock. The daily VaR answers the question, "What is the maximum a particular stock can lose in one day". For example, we interpret a calculated VaR of 3% with a 95% confidence level as: "We are confident that 95 out of 100 days the stock will not lose more than 3%".

The purpose of this program is to compute the daily Value at Risk for the past 2 years for any single stock listed on Yahoo! Finance, using three common methods: the historical method, the Monte-Carlo method, as well as the variance-covariance method. Next, the program compares this two-year daily VaR to the actual losses of the stock for this two year period to determine whether the Monte-Carlo Simulations and variance-covariance method are within the confidence level.

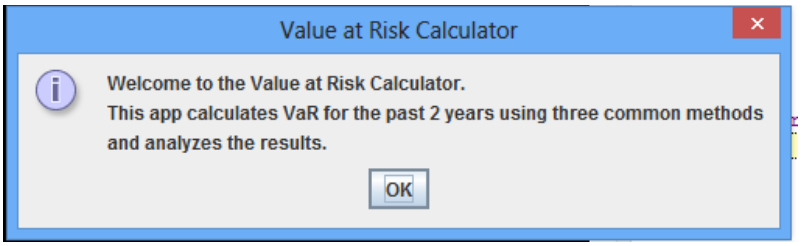
The asset input is limited to stocks only. The program calculates 1-day VaR as a percent. The negative value denotes a maximum loss, while a positive VaR denotes a minimum gain.

2. User's Guide

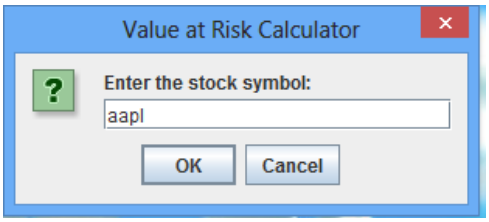
To run the VaRCalculator application, the user should have access to internet, and should have Matlab and java installed on their computer. The provided matlab files, text file, and java files must be saved in the same folder.

Running VaRCalculator:

1) The first screen welcomes the user to the application, and gives a brief description of what tasks the app will perform. Click <OK> to enter.

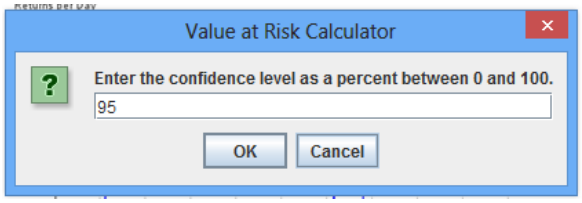


2) The user must input the symbol of the stock for which the VaR will be calculated. The symbol must be for a stock listed on Yahoo! Finance. For a list of valid stock symbols visit: http://investexcel.net/all-yahoo-finance-stock-tickers/, download the excel spreadsheet, and find a list of nearly 15,000 valid symbols, on the 'Stock' worksheet. This input is not case-sensitive. When the desired symbol has been entered, click <OK>. Or, click <Cancel> to exit the app.

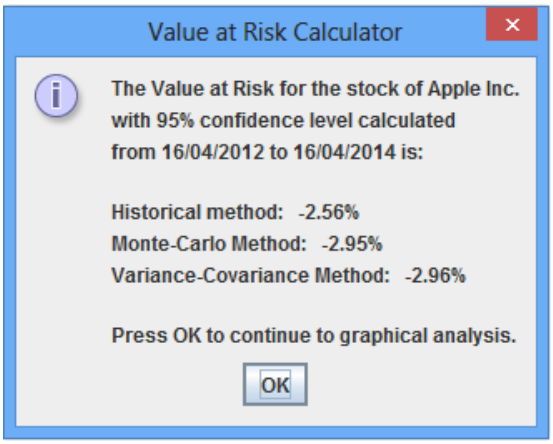


If the entered stock symbol is invalid, a screen will appear indicating the input error. Click <OK>. The previous screen will appear again until a valid symbol is entered.
If the file stockList.txt is not in the class path, a screen will appear asking the user to verify that the file is in the class path.

3) The user is prompted for a confidence level, as a percent. For example, 95% should be input as 95. Commonly used confidence levels are 95%, and 99%, but any value between 0%-100%, exclusive, is valid. Click <OK>. If the entered confidence level is out of this range, the previous screen will appear again until a valid confidence level is entered. Or, click <Cancel> to exit the app.

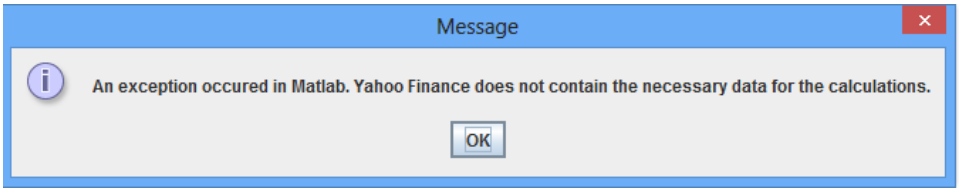


4) The user will notice that the program automatically opens MATLAB and will see a progress bar as the stock prices for the specified stock are downloaded. The VaR is output on the next application screen (as well as the Matlab command window) including the company name and the date interval used to calculate the VaR. The VaR is output as a positive/negative percent, rounded to two decimals, as calculated using the historical method, the Monte-Carlo method, and the variance-covariance method. . Click <OK> to continue to the analysis portion of the app.

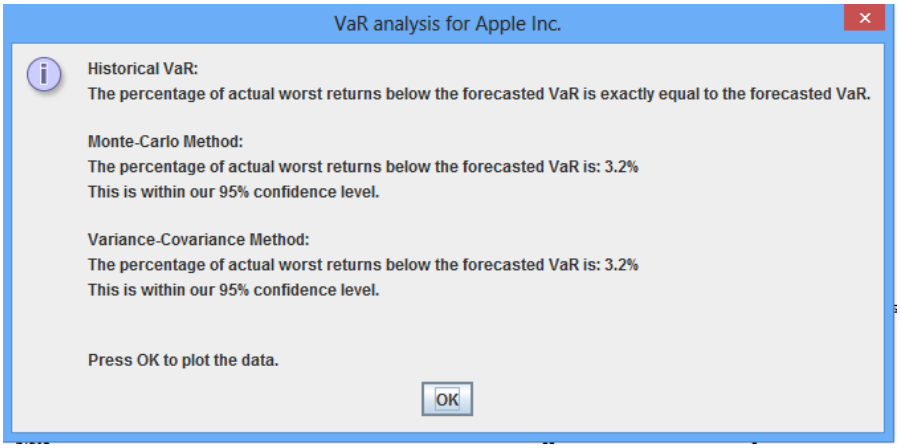


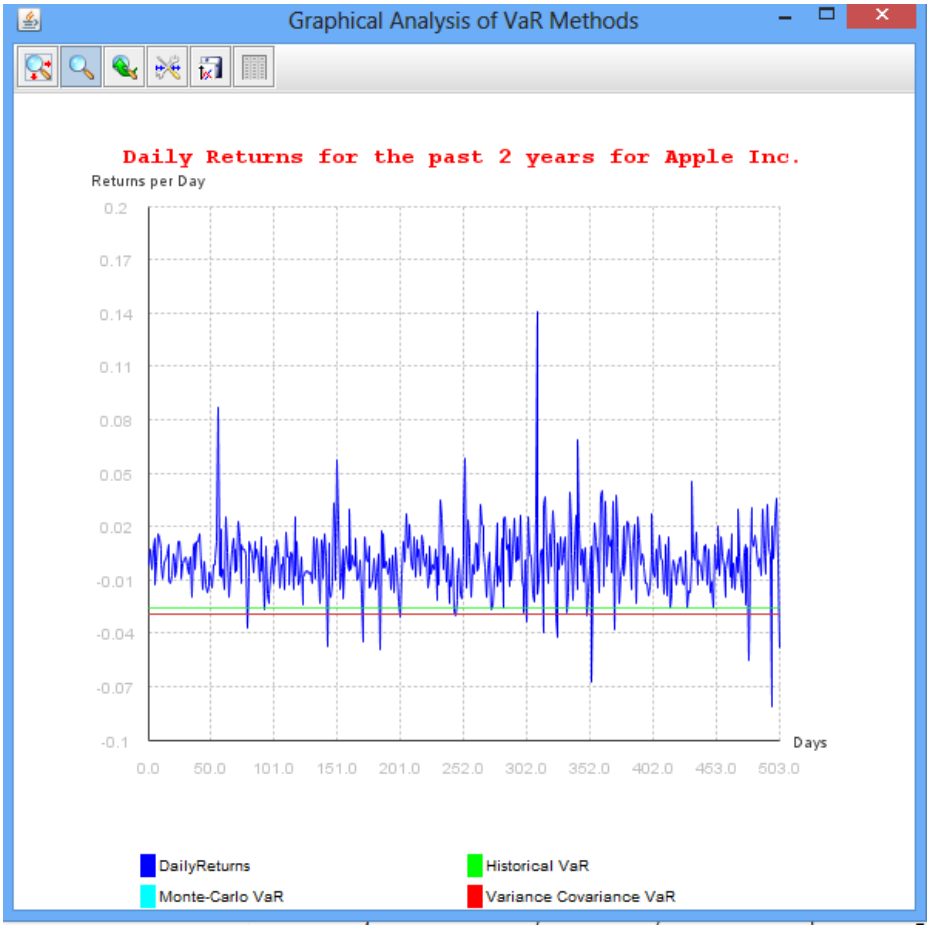
If MATLAB is not properly installed on the computer, a screen will appear asking the user to ensure that MATLAB is installed properly.

If there is no available data for any of the dates within 2 years before the current date, a screen will appear indicating that the necessary data could not be obtained from Yahoo! Finance. The user should terminate the application and restart if they wish to calculate the VaR for a different stock.

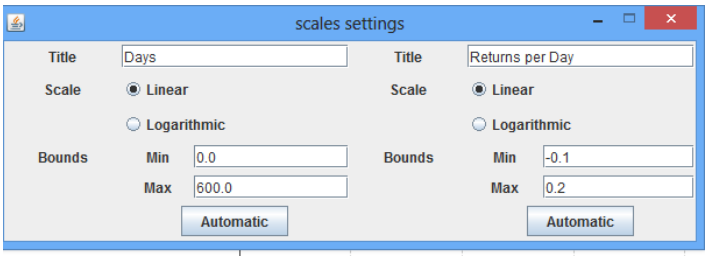


5) A screen for VaR analysis will appear. The screen provides the back-testing results. The historical VaR is always consistent with the forecasted VaR from historical data. For both the Monte-Carlo and variance-covariance methods the actual percentage of worst returns below the forecasted VaR are calculated and the program determines whether or not this falls within the necessary confidence level. Click <OK>.

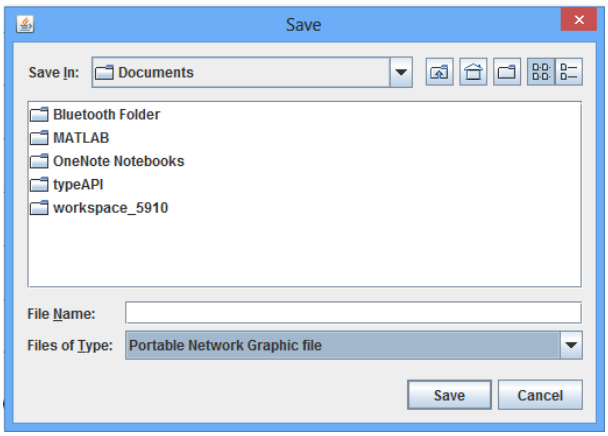




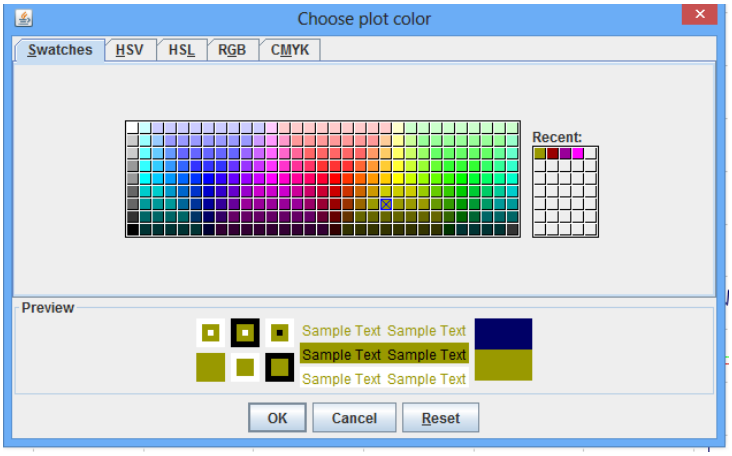
6) A plot will appear. It includes the daily returns for the past two years for the input company, as well as the computed VaRs; the historical VaR in green, the Monte-Carlo VaR in cyan, and the variance-covariance VaR in Red. The user can resize the plot by dragging the edges. Along the top is toolbar with icons. The first can be used to center the axes. The second can be used to zoom in and out while the third will reset the plot. The fourth icon has scale settings.



The fifth icon can be used to save the plot in a specified folder in PNG format.



The last icon contains the data for each plot within the graph. The user can turn a plot on or off by clicking the visible checkbox and can change the colour of the plots by clicking the colour bar beside it. The user can copy and save the data elsewhere. Close the plot to end the app.



3. Programmer's Guide

See documentation within the source code.

4. Design and Implementation

The intention was to create a user-friendly application to calculate the Value at Risk of a specified stock and compare the accuracy of the three common methods for computing VaR with back-testing and graphical analysis. To make the app user-friendly and minimize the possibility of errors the user input was limited to only the stock symbol and the confidence level. Using this information the Java application was developed, in tandem with MATLAB functions, to obtain stock data from Yahoo! Finance, calculate the VaR using all three methods, and output the numerical results as well as a graphical analysis in a series of easy-to-navigate dialog boxes. The step-by-step structure of the application can be found in Appendix A.

4.1. VaR Calculation

The program was developed in the following sequence. First, the three MATLAB functions were created. Hist_stock_data.m is an open source code created by Josiah Renfree to retrieve stock data for a user specified interval and stock. Next, VaR_Calculator.m was designed to take as input the confidence level, start date, end date, and stock symbol and output the historical VaR, Monte-Carlo VaR, and variance-covariance VaR. VaR_Analysis was developed to prepare the data that would be necessary in our back-testing analysis and the data to be plotted in java. It takes as input the start date, end date, stock symbol, as well as the three VaR values output by VaR_Calculator.m and outputs a count of the actual number of worst returns below the calculated VaR for each method, as well as a vector of returns, x-axis vector, and vector of each VaR value for the plot. More details about each of these functions can be found within their m-files. Each of these files was adjusted and updated during the java coding process so that MATLAB could be utilized optimally by the Java application.

The Java code was then developed while integrating these MATLAB codes. First, the necessary input data is obtained. The stock symbol is obtained and verified. A text file containing the symbol followed by the company name on each line was prepared from the abovementioned excel spreadsheet, and saved as stockList.txt. The symbol is verified with a string tokenizer by looping through the first token on each line. When the matching string is found, the remaining tokens on the line, if there are any, are stored as the company name for later use. If the matching string is not found, the user is informed that the stock symbol is not valid and is given an opportunity to re-enter a symbol.

Next the confidence level is verified and converted to a decimal. The start and end date are determined. An instance of the Calendar class is used to obtain the current time. A two year period to calculate the VaR has been chosen since this provides enough data for our calculations, but is still limited to avoid extensive calculations. Also, the calculations always go up until yesterday since maybe today's stock prices have not yet been updated. Therefore, the end date is yesterday and the start dat is two years prior to yesterday. The application then extracts the Date instance of both dates and they are formatted as ddMMYYYY as required by our MATLAB function.

Now, a connection with MATLAB is established using an instance of the MatlabProxyFactory class. All java variables are converted to arrays of doubles, since this is required by MATLAB, and are exported to MATLAB using the method setVariable(java.lang.String java.lang.object) of MatlabProxy. Next we obtain the stock data using the hist_stock_data function and evaluate the VaR_Calculator function in MATLAB. The outputs are imported to java using getVariable(java.lang.String). The MATLAB results are in the form of matrices, so they must be cast to arrays of doubles and then the first element is set as the VaR result. The values are formatted to a percent with 2 decimals and the date is formatted as dd/mm/yyyy. The results are displayed to the user in a dialog box.

4.2. Graphical Analysis

The MATLAB function VaR_Analysis is used to obtain the necessary plotting and back-testing data and the results are imported to java in a similar manner as described above. The application disconnects from MATLAB.

The back-testing portion to analyze worst results compares the actual number of returns that are below the historical VaR to the forecasted VaR of the other two methods and determines whether the estimate is good or bad. The daily returns of the stock, historical VaR, Monte-Carlo VaR, and variance-covariance VaR are plotted against time in days in blue, green, cyan and red, respectively using jMathPlot.

5. Error Checking

The application was designed to minimize sources of error in the following ways: limiting user input, using Graphical User Interface with simple to follow instructions, and using reliable data sources such as Yahoo! Finance to obtain stocks. The input data is verified. The stock symbol is verified with a text file to avoid feeding an invalid symbol to Yahoo! Finance. If the text file does not contain a company name, it is set to be the stock symbol. As well, the application verifies that the user input confidence level is within the range of 0-100% to ensure that the calculated VaR yields reasonable and meaningful results. In both of these cases, the app will prompt the user to re-enter the input, as well as provide an opportunity to exit the app with a <Cancel> button.

The application has been designed to handle any errors encountered when connecting to MATLAB, obtaining data from Yahoo! Finance, or verifying the stock symbol with the text file. Error messages are displayed for the user if a connection to MATLAB cannot be obtained, or if data extraction fails. An error message also appears if the application cannot find the file stockList.txt and asks the user to ensure that it is saved properly. The MatlabConnectionException, MatlabInvocationException and FileNotFoundExceptions are handled through try-catch blocks.

6. Testing

The different methods used to calculate VaR yield similar results, which indicates that all three methods are good estimators for daily VaR. We tested our application in various ways. The VaR results are verified using back testing. Details can be found in the Design and Implementation section. Different stock symbols were used to ensure the program runs smoothly without crashing upon encountering some exception. To ensure the app is user friendly we invited other students to use the app, so we could see if the instructions are easy to follow and whether they encounter any difficulties.

7. Conclusions

VaR is simple number that captures the notion of how bad things can get! This application calculates today‘s VaR for any given stock. The results are consistent with what we would expect based on our

financial knowledge. The back testing results indicate that the calculated values are, in general, consistent with historical data of the past two years. However these results may not be reliable for other time periods as we did not include stress testing and two years is a short period of time.

We essentially achieved the goals outlined in our proposal excluding some extensions, while adding a few extra features.

8. Code Listing

The following files have been included with this report and are necessary to ensure the application runs.

- VaRCalculator.java
- VaR_Calculator.m
- hist_stock_data.m obtained from http://www.mathworks.com/matlabcentral/fileexchange/18458-historical-stock-data-downloader/content/hist_stock_data.m

  Note: a Mathworks account may be needed to view the code
- VaR_Analysis.m
- stockList.txt
- jMathPlot.jar obtained from www.jmathtools.sourceforge.net
- MATLAB jar file

Appendix A :
Design Flow Chart