Economic and Financial Modeling Club, University of Chicago

Value-at-Risk Practice

Zachry Wang, Allan Zhang, Jason Phang

November 12, 2014

Today, we'll be exploring Value at Risk (VaR), a number that seeks to quantify downside risk. In particular, this is a model that connects (1) a time period, (2) a probability, and (3) a quantity of loss. For this exercise, our time period will be 1-month and our probabilities will be 90%, 95%, and 99%.

VaR seeks to answer the question: if I invest in an asset over time period T, what is the threshold for returns R such that $P\%$ of the time, I will get returns higher than R.

1 Set-up and Visualization

1. Grab, in any way you would like, monthly stock data for the S&P 500 Index from Jan 1, 2000 to Jan 1, 2014. You could do this by downloading a CSV file and reading the data using CSV package (see week 2 presentation) or get the data directly using the Quandl Python API.
2. Create a list of monthly returns for S&P 500 Index.
3. Create a histogram of the monthly returns using the Matplotlib package (see end of Week 4 Lecture). Be sure to give a title and label axis.

2 Calculating VaR from the empirical distribution

The idea here is simply to create a histogram that approximates a probability distribution of the returns. The big assumption here is that returns in the future will behave like they have in the past.

1. Take your list of monthly returns and sort them from smallest to largest. Then, find the thresholds $R_{90\%}, R_{95\%}, R_{99\%}$ that answer our Value at Risk question.

2. Finally, plot the thresholds onto your graph.

3 Calculating VaR assuming returns are normally distributed

The idea here is similar, except we approximate the probability distribution of returns using a normal distribution (see week 3 lecture on probability).
Before you begin here, we want to import a function that will help us calculate the inverse cdf of a normal distribution. So at the top of your code, include

```
from scipy.stats import norm
```

We will use the function `norm.ppf()` to calculate the inverse cdf of the normal probability distribution. Feel free to ask us if any of the statistics concepts do not make sense conceptually.

1. Convert your arrays to numpy arrays. Then, calculate mean and standard deviation of returns.

2. Use the function `norm.ppf()` to find thresholds $R_{90\%}, R_{95\%}, R_{99\%}$.