# Laboration: Decision Theory

Simon Sigurdhsson

September 14, 2010

**Abstract**

Analyzing stock data and optimizing a portfolio using the statistical tools available in MATLAB and Mathematica, given historical data and a utility function.

## 1 Investigating the model

In order to use the model set up in later parts of this laboration (i.e. the portfolio optimization), we need to test our stochastic data to make sure that it *a*) is independent and *b*) has a Normal distribution.

To do this, I have chosen to employ the Kolmogorov-Smirnov goodness-of-fit test to assert that the data has a Normal distribution, and the autocorrelation function to check for (linear) independence. However, we first need to load our data into MATLAB; the data is loaded from `stockdata.tsv` and logreturns are computed like this, taking advantage of MATLABs linear algebra capabilities:

```
load stockdata.tsv
logs = log(stockdata(:,2:8));
logreturns = logs(2:end,:) - logs(1:end-1,:);
```

This gives us a large matrix `logreturns`, containing the log returns of all seven stocks column-wise. Now, to check for independence using the acf, we estimate it with the given formula:

$$\hat{r}_X(h) = \frac{(t+1)\sum_{i=0}^{t-h}(X(i)-\bar{X})(X(i+h)-\bar{X})}{(t-h+1)\sum_{i=0}^{t}(X(i)-\bar{X})^2}$$

$$\bar{X} = \frac{1}{t+1}\sum_{i=0}^{t} X(i)$$

This estimation looks computationally intensive (since it contains three summations), but using MATLABs linear algebra capabilities one can reduce it to a fairly simple form. Setting up a function `acfhat` that preforms this task on the entire `logreturns` matrix at once (yielding a row vector with the value of $\hat{r}(h)$ for each stock and a given value $h$) makes it possible to use this simple line of code to test independence, according to a 95% confidence interval giving the comparison $|\hat{r}_X(h)| \leq 1.96/\sqrt{n}$, where $n$ is the number of stocks (i.e. 7), and $h = 1$:

```
lindep = (acfhat(logreturns, 1) > 1.96/sqrt(size(logreturns,2)));
```

The function `acfhat` takes the log return matrix (of arbitrary size) as its first argument, and the value of $h$ as its second. The function itself is fairly straight-forward, and looks like this:

```
function r = acfhat(X, h)
  t = size(X, 1);
  Xbar = sum(X)/(1+t);
  ps1 = X(1:(t-h),:) - repmat(Xbar,t-h,1);
  ps2 = X((h+1):t,:) - repmat(Xbar,t-h,1);
  ps3 = (X - repmat(Xbar,t,1)).^2;
  r = (t+1)*sum(ps1.*ps2)./((t-h+1)*sum(ps3));
```

Note that the MATLAB economy toolbox contains its own `autocorr` function doing the same thing, but this toolbox is unavailable at Chalmers. Since the acf estimate is easily implemented, this doesn't pose much of a problem, though.

Running this on our data yields a positive result (i.e. the appropriate element in `lindep` is `false`) for all seven stocks; hence we can assume that the log returns are independent as expected. Doing the same for squared log returns yields the same result, further cementing our theory that the log returns are independent.

We continue by doing a Kolmogorov-Smirnov goodness-of-fit test to assert that our data is distributed the way we think it is (i.e. has a Normal distribution). This is done using the `kstest` function from MATLABs statistics toolbox, like so:

```
for i=1:7
  isnormal(1,i) = kstest(logreturns(:,i));
end
```

The `kstest` function, when used like this, assumes that it should test against the *standard* normal distribution, i.e. $N(0,1)$. Since the above test rejects the null hypothesis for all stocks, it is likely that we have to estimate the parameters first. Performing an ML estimate (using the MATLAB `mle` function) on the stocks and using the mean of these parameters suggests that the data has a distribution $N(0, 0.0225)$. To take this into consideration, the test is modified as follows:

```
for i=1:7
  isnormal(1,i) = kstest(logreturns(:,i), ...
    ProbDistUnivParam('normal', [0 0.0225]));
end
```

However, even with these modifications, the Kolmogorov-Smirnov test rejects the null hypothesis for all stocks. Hence, one must reject the hypothesis that the stock data is Normally distributed. Continuing the investigation using the `kstest2` function to compare the distributions of pairs of stocks gives further information. This tells us that stocks 1, 2, 4 and 6 are of a "similar" distribution, with a 95% confidence interval.

Note however that the fourth (Gambio) stock is not rejected when performing a $\chi^2$ goodness-of-fit test with a 95% confidence interval; along with the `kstest2` results, this indicates that at least stocks 1, 2, 4 and 6 might be Normally distributed after all. All in all, the tests are inconclusive.

## 2 Portfolio optimization

In order to understand how our utility function $U(x) = 1 - e^{-kx}$ behaves for different values of $k$, we plot it for a few values of $k$ as can be seen in Figure 1. Notice how the graph of $U(x)$ "moves" toward the y axis as $k$ grows; this means we can interpret the variable $k$ as a measure of risk aversiveness — a larger $k$ means we are less likely to take risks when speculating. This is due to the fact that $U(x)$ becomes "more" concave at $k$ increases.

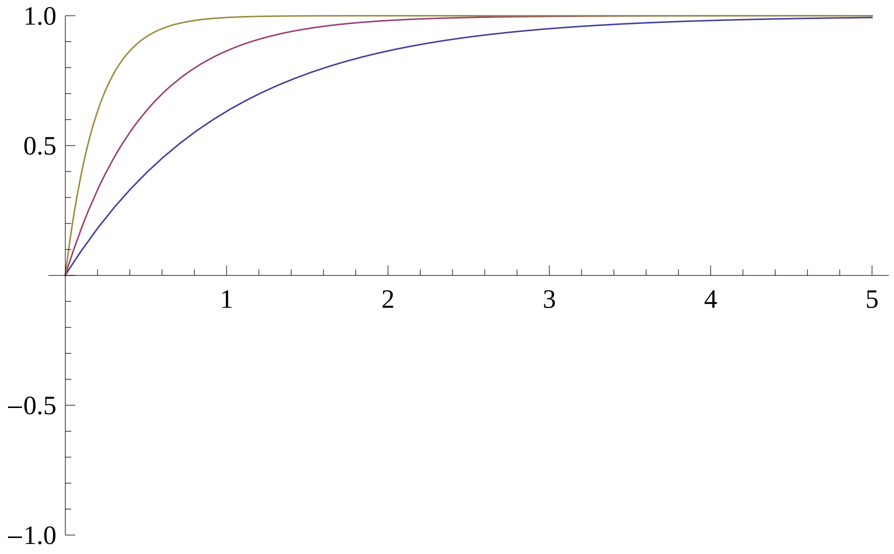

Figure 1: The utility function for $k = \frac{1}{2}$, 1 and 2.

With this basic understanding of $U(x)$ handled, we can continue dealing with the task at hand: optimizing the portfolio. To begin with, we import the stock data; after this we estimate the covariance matrix $Q$ and the expected value vector $\mu$.

```
In[1]:= data = Import["stockdata.tsv"];
In[2]:= data = data[[All, {2, 3, 4, 5, 6, 7, 8}]];
In[3]:= Q = Covariance[data];
In[4]:= mu = Mean[data];
```

Additionally, we set up a function `EU` according to the given formula for $E\{U(w)\}$, as well as the utility function $U(x)$ itself, as `U`:

```
In[5]:= U[x_] = 1-Exp[-k x]
In[6]:= EU[z_, sigma_] = Integrate[U[x] 1/(Sqrt[2 Pi sigma^2])
        Exp[-(x-z)^2/(2 sigma^2)], {x, -Infinity, Infinity}]
```

Using these functions, we can calculate $E\{U(w)\}$ for some different values of $\mu$ and $Q$, before we use the actual data.

Assuming that $w$ contains seven elements, all equal to 1/7, that $\mu$ is a zero vector, and that $Q$ is defined so that $w^T Q w = 1$, we have that $E\{U(w)\} = 1 - e^{\frac{k^2}{2}}$. Further testing shows that the results tend to have the form $E\{U(w)\} = 1 - e^{\frac{ak+bk^2}{2}}$.

Moving on to test this with the actual data, we obtain the following result (again, assuming that $w$ only contains elements equal to 1/7):

$$E\{U(w)\} \approx 1 - e^{\frac{134k+137k^2}{2}})$$

Clearly we cannot assume that $w$ will look like this; in fact, we have to optimize the portfolio with *respect* to $w$. The only constraint we have is that the 1-norm of $w$ has to be 1, and that it only contains positive elements.

Now, assuming $\Re\left(\sigma^2\right) > 0$ (which most likely will be the case, since both $w$ and $Q$ are real-valued), Mathematica gives us a more sensible form for $E\{U(w)\}$:

$$E\{U(w)\} = (1 - e^{\frac{1}{2}(k^2\sigma^2 - 2kz)}), \quad \sigma^2 = w^T Q w, \quad z = \mu^T w$$

A couple of simple arithmetic operations can rearrange the exponent:

$$\frac{1}{2}(k^2\sigma^2 - 2zk) = -k\left(z - \frac{k}{2}\sigma^2\right)$$

Since we want to *maximize* the expected utility, we want the exponent to be as large as possible, since this will bring the value toward 1. Hence, the problem is equivalent to maximizing

$$z - \frac{k}{2}\sigma^2 = \mu^T w - \frac{k}{2} w^T Q w,$$

which of course is a much easier problem, computationally speaking.

Doing this in Mathematica is not very difficult; the `NMaximize` function performs a numeric optimization. We use it like this:

```
In[7]:= QuadProb[x_] = mu.x - (k/2 x.Q.x);
In[8]:= k = 1/2;
In[9]:= NMaximize[{QuadProb[{a,b,c,d,e,f,g}],
        Norm[{a,b,c,d,e,f,g}, 1]==1,
        a>=0, b>=0, c>=0, d>=0, e>=0, f>=0, g>=0},
        {a,b,c,d,e,f,g}]
Out[9]= {88.2773, {a -> 0.330831, b -> 0., c -> 0., d -> 0.119876,
        e -> 0.330015, f -> 0.219277, g -> 1.30998 10^-7}}
```

To see what different values of $k$ do to this optimization, refer to Table 1. From the results one can see that some of the “safer” stocks to invest in seem to be Ericsson and Nokia, while some of the more risky stocks are AstraZeneca, Swedish Match and Svenska Handelsbaken. These results may seem counter-intuitive, especially considering the history of Ericsson stock, but as seen in Figure 2, Ericsson and Nokia are actually quite stable.

Of course, one would expect Swedish Match to be a safe investment as well, along with Gambio (whose sudden decline near the end probably relates to some kind of financial transaction); but the analysis is not perfect. Part of the reason this doesn’t work is probably because we assume the stock data is Normally distributed, even though initial analysis suggested otherwise.

Also seen in Table 1 is the value of the utility function based on the naive investment method of investing equally in all stocks. When using a highly risk-aversive utility

Table 1: Optimal values of $w$ for some different values of $k$.

| Value of $k$ | Optimal value of $w$ (two decimal places) | $E\{U(w)\}$ | $E\{U(w_{\text{naive}})\}$ |
|---|---|---|---|
| 10 | (0.00, 0.00, 0.84, 0.00, 0.16, 0.00, 0.00) | $\approx -10^{992}$ | $\approx -3 \cdot 10^{5390}$ |
| 2 | (0.02, 0.00, 0.74, 0.00, 0.24, 0.00, 0.00) | $\approx -10^{11}$ | $\approx -10^{-122}$ |
| 1 | (0.12, 0.00, 0.38, 0.00, 0.28, 0.22, 0.00) | 1 | -16.35 |
| 0.5 | (0.33, 0.00, 0.00, 0.12, 0.33, 0.22, 0.00) | 1 | 1 |
| 0.25 | (0.72, 0.00, 0.00, 0.00, 0.11, 0.00, 0.17) | 1 | 1 |
| 0.1 | (1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00) | 1 | 1 |

function there is a large difference, implying that the naive way of investing is much too risky for very risk-averse investors.

However, as $k$ decreases and the utility function becomes more risk-neutral, the difference disappears almost completely. This indicates that the nave investment method is fairly risk-neutral, and that one can expect it to be almost as good as investing more thoughtfully.

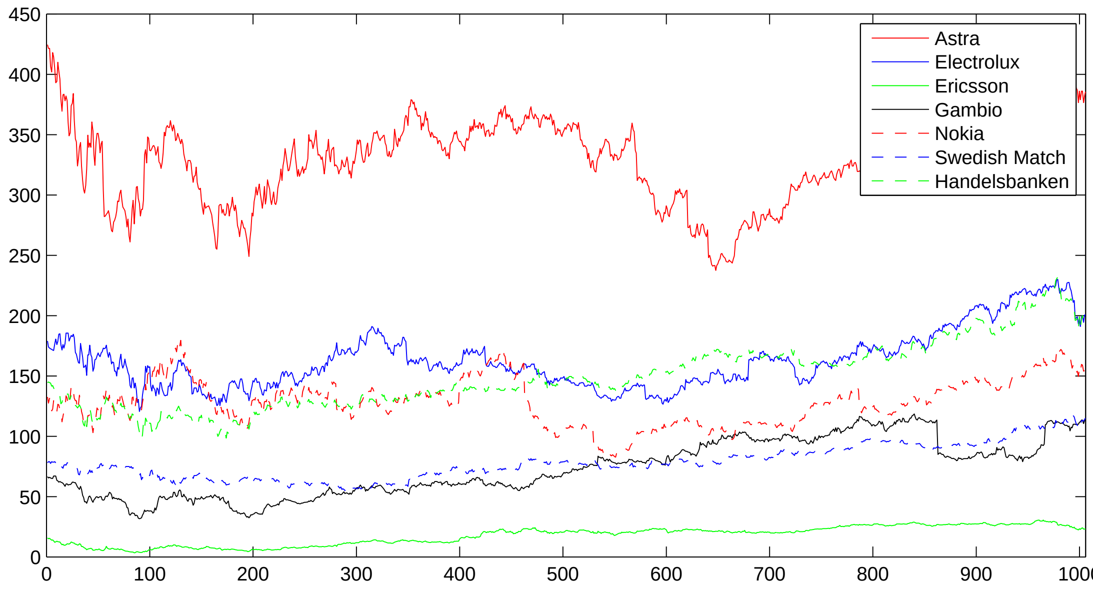

Figure 2: The seven stocks plotted for comparison and graphical analysis.

Of course, since the data was shown not to be Normally distributed, this optimization is not really valid. One could do many things to “fix” this; dropping seemingly incorrect data (such as the drop in Gambio stock) or finding a better distribution (although this would invalidate the simplification done toward the end of the optimization) would be two relevant solutions in this case.

The utility function can of course not be guaranteed to be perfect, but in this case it simplifies the problem *and* works the way we want it to, so there’s not really any reason to reject it. One should be careful with the value of $k$, however — large values make the optimum solution invest a lot in fairly “constant” (i.e. slowly increasing) stock. While this is good from one point of view, it does take *very* long before any significant profit is made from the investment.