Homework 8: A Diversified Portfolio

36-402, Advanced Data Analysis

Due at 11:59 pm on Thursday, 24 March 2016

Classical financial theory suggests that the log-returns of corporate stocks should be IID Gaussian random variables, but allows for the possibility that different stocks might be correlated with each other. In fact, theory suggests that the returns to any given stock should be the sum of two components: one which is specific to that firm, and one which is common to all firms. (More specifically, the common component is one which couldn't be eliminated even in a perfectly diversified portfolio.) This in turn implies that stock returns should match a one-factor model.

The `RData` file stockData.RData can be loaded using the `load` function. It contains three objects:

- `close_price`: a data frame containing the daily closing prices in 2015 for the stocks of 28 selected large US corporations. Each row is labeled by the relevant date, which can be extracted by the `rownames` function.
- `stock_info`: a small data frame containing basic information about the component stocks (courtesy of Wikipedia). The rows are in the same order as the columns of `close_price`. This is not quite necessary, but could be interesting.
- `tricky_prices`: This is explained in the last problem on this assignment.

1. Visualizing and transforming the data.
   (a) (5) First, visualize the closing prices. Plot the closing prices for all stocks on the same graph. Use lines, not points. There is no need, in this instance, to label the individual traces uniquely. You just want to see the shape of the data.
   (b) (5) The closing prices are on very different scales, and show clear dependence over time. It is more common to analyze the log returns, rather than the raw prices over time. Create a new data frame with the log daily returns for each stock. The log daily return at time t is defined as $\log\left(\frac{\text{price at time } t}{\text{price at time } t-1}\right)$. This data frame will have the same number of columns, and one fewer row.
   (c) (5) Plot the log returns over time, placing all 28 time series on the same plot. Do the log returns look more comparable than the closing prices? (You may see an even nicer plot if you add a little transparency to the lines.)
2. Exploring the distribution of log returns.
   (a) (5) Focusing on General Electric (Symbol GE, column 11), plot a histogram of the log returns. Use 30 cells instead of the default, so that you can better see the shape (`breaks=30`).

(b) (5) The distribution of log returns is often modeled by a normal distribution. Estimate the best fitting normal distribution for these returns, still focusing on GE. Using the `curve` command, overlay the best fitting normal distribution onto the histogram of the data. To make the curve and the histogram comparable, use the `probability=TRUE` option in the `hist` function. How well does the normal distribution appear to fit?

(c) (5) It can be hard to see the shape and the deviations from normality very well in a histogram. Use a kernel density estimate with a Gaussian kernel to approximate the distribution. Use cross-validation to choose the appropriate kernel bandwidth. Plot the kernel density estimate, along with the best fitting normal density from the previous part. Where is the true density notably higher/lower than the best fit Gaussian? Is the distribution symmetric? How do the tails compare?

(d) (5) Plot kernel density estimates for all 28 stocks on the same plot, separately cross-validating each one. Adjust the axes so that the curves are visible. Do the other curves look similar to the GE curves, and seem to support your statements from the question?

3. *Principal components* Compute the principal components of the returns. You may use `prcomp` function. Read the documentation carefully, you want the analysis to both center and scale the log returns.

   (a) (5) Make a barplot to report the weights of the first principal component (which should be a length 28 vector). It will be easier to interpret your plot if you sort the weights prior to plotting. Use the `las=1` or `las=2` option of `barplot` to make readable perpendicular axis labels. Comment on any notable patterns.

   (b) (5) Plot the projection on to the first principal component against date. Comment on any notable patterns. How does this compare to your earlier plot of log returns?

   (c) (5) Plot the eigenvalues of the pricipal component analysis. (One way to do this is to `plot` the object returned by `princomp`.) How many components do you think are actually necessary to capture a lot of the variance? How much of the variance is your first principal component capturing?

   (d) (5) The second principal component captures the direction of largest variance from your one-dimensional model. Make another sorted barplot of the weights of the second principal component. Do you see any pattern in the weights?

4. Fit a one-factor model.

   (a) (5) Report the vector of factor loadings. (Again, this will be most easily reported visually.) Comment on any notable patterns, and compare it to the first principal component.

   (b) (5) Plot the factor score against the date. Comment on any notable patterns, and compare to the projection on the first principal component.

5. (10) Use case bootstrapping (bootstrapping days) to provide 90% confidence intervals for the factor loadings of the one-factor model. Report the results as a figure rather than a table.

6. The `tricky_prices` data frame contains closing prices for the same stocks and two additional stocks. Again, convert these prices to log return values. Run principal component analysis on the returns for these 30 stocks, as you did above.

(a) (5) Plot the eigenvalues; how many components does it look like you might need?

(b) (10) Look at your weights, projections onto the first principal component, and closing prices. What changed when the new stocks were added, and why? Use plots and words to explain what happened.

RUBRIC (10): The text is laid out cleanly, with clear divisions between problems and sub-problems. The writing itself is well-organized, free of grammatical and other mechanical errors, and easy to follow. Questions which ask for a plot or table are answered with both the figure itself and the command (or commands) use to make the plot. Plots are carefully labeled, with informative and legible titles, axis labels, and (if called for) sub-titles and legends; they are placed near the text of the corresponding problem. All quantitative and mathematical claims are supported by appropriate derivations, included in the text, or calculations in code. Numerical results are reported to appropriate precision. Code is properly integrated with a tool like R Markdown or knitr, and both the knitted file and the source file are submitted. The code is indented, commented, and uses meaningful names. All code is relevant to the text; there are no dangling or useless commands. All parts of all problems are answered with actual coherent sentences, and never with raw computer code or its output.