User manual for *cluster*

Reuben Settergren
Department of Computing, Imperial College
`reuben@doc.ic.ac.uk`
January 22, 2009

Introduction

The program *cluster* accepts statistical information (mean/variance) on asset behaviour as input, performs a large number of simulations of the assets, and "summarizes" that large number of simulations by clustering the results and outputting the centers of each cluster, in the form of a multistage scenario tree (formatted for input to multistage portfolio optimizer *foliage*).

Input file

The format of a file input to *cluster* is divided into sections by keywords. A small example is provided in Fig. 1. The sections are

ASSETS The number of assets provided here determines the dimension the program will expect for the statistical information provided below.

STAGES The depth of the scenario tree to be generated. The number provided here determines the dimension the program will expect for BRANCHING below.

COVARIANCE The lower-triangular part of a postitive (semi-)definite matrix describing the historical (or desired future) deviation of the assets from the exponential growth curve defined by growth rates below.

INITIAL Asset prices "today", i.e. at the root of the scenario tree. If the output will be given in terms of returns (default) instead of prices, INITIAL values could be nonsense values, like all 1.

GROWTH Exponential growth rates for each asset (presumably fitted from historical data).

BRANCHING The number of branches from each node, at each depth in the scenario tree. Although there is no theoretical reason it must be so, in this implementation, the tree must be symmetric: all nodes at the same level must have the same number of children.

END Signifies the end of input.

Other than ASSETS and STAGES being first and second, the order of the other keywords is not important, nor are linebreaks or other whitespace (i.e. the triangular look of the covariance matrix in Fig. 1 is for the reader's convenience; it could just as well have been all on one line).

```
ASSETS 3
STAGES 3
COVARIANCE
78.9339
-13.5393 23.3747
4.58791  43.9313 405.026
INITIAL
296.43  118.5  336
GROWTH
-0.000330405 -0.00663024 0.00693373
BRANCHING
4 4 4
END
```

Figure 1: Sample input file

Command-line options

There are a large number of parameters which may be specified by the user via command-line switches. If the user is happy with the default settings, it is only necessary to supply the first three.

-f *filename* Takes as an argument the name of the input file (formatted as described in the previous section)

-o *filename* Takes as an argument the name of the output file (formatted as input to *foliage*).

-n N Takes as an argument N, the number of simulations to perform. The default is 100, which is not enough for any real-sized problem.

-SEQ Use the sequential method of scenario generation, as opposed to the default of parallel (as described in the paper). With -SEQ, N scenarios are branched from every node in the tree, and then clustered. Thus, the branching from each node in the tree should take the same amount of time. In the default parallel mode, N scenarios are branched from the root, and after clustering, each child keeps only the scenarios in its own cluster. This way the number of scenarios is held constant at N per level, and each depth of the tree should take the same amount of time. It seems like parallel is obviously faster, but because of lower resolution at the leaves, the methods are not comparable, because larger N need to be used for parallel than for sequential generation. Experiment with values of N and -SEQ or not until you are happy.

-SOBOL Takes no argument. Uses low-discrepancy quasirandom Sobol series instead of default of IID pseudorandom variates.

-ANTITHETIC Uses the method of antithetic variates with the IID pseudorandom variates, i.e. performs only $N/2$, and the other half are antithetic to the first half.

-m M Takes as an argument the minimium ratio M of scenarios/leaves. Before doing any simulations at all, a calculation is done to see whether N is large enough to allow M scenarios per leaf after clustering is done. If not, it complains and exits, suggesting a larger value of N (this will happen if you forget to specify N larger than its default of 100). During simulation and clustering, M is also used as a guide to reject clusterings which are good for now, but will not be divisible enough further down the tree. Default for this ratio is $M = 3.0$ scenarios per leaf.

-r R Takes as an argument the minimum ratio R of largest/smallest cluster sizes. This is a way to control extremely high/low probability events in the scenario tree, as the probabilities are directly proportional to the cluster sizes. Setting R too small (close to 1), will slow the program down, and possibly even cause it to give up due to repeated failure to find a clustering with similar-enough sized clusters. Default for this ratio is $R = 5.0$.

-p Takes no argument. If present, this switch causes output to be in terms of asset prices, not the default of asset returns (all prices are divided today/yesterday before output).

-NORMAL Takes no argument. If present, the program reverts to its original stupid behaviour of having "tomorrow" prices be normally distributed around their target (instead of the default of lognormal).

-d D Takes as an argument the threshold D below which the dampening function is activated. This capability was installed when -NORMAL was the default method for random variable generation, because a combination of high variances and negative growth rates often led to negative prices. Since -NORMAL is a theoretically stupid way to do things, you shouldn't worry about this feature. But just in case, if you specify damping at -d 0.8, for instance, "tomorrow" prices at least at large as 80% of "today" prices will remain unchanged, and prices that drop more than 20% will be passed through an exponential filter that squeezes the range $[-\infty, D * \mathit{today}]$ into $[0, D * \mathit{today}]$. If you don't specify -NORMAL, then specifying a damping threshold with -d has no effect.

-q Takes no argument. Causes the silent mode of operation.

-v Takes no argument. Verbose mode prints out more informative output to the screen. The default is to print, on a new line for each level of the scenario tree, '.' for each failure to cluster a branching, and '|' when success is achieved. In verbose mode, the details of each failed and successful clustering is printed.

Output File

As noted above, the program writes the generated scenario tree into the specified output file. One thing to note is that it makes a few arbitrary decisions about parts of this file other than the SCENARIOS part generated by this program.

First is the BENCHMARKS section. With no means of specifying benchmarks to the cluster program, it simply guesses at an equally divided benchmark of $1/N$ per asset.

Next is the COVARIANCES section. Simply passing on the given covariance matrix would leave them scaled in terms of the prices, not for returns. So the covariance matrix is scaled down first (for internal workings of *cluster* as well) to $\Lambda \otimes INI^{-1} INI^{-T}$, where INI is the provided vector of INITIAL prices.

Last, and probably most important, is the BOUNDS section, for which cluster always gives the fairly loose default bounds of 0–50% for each asset balance (w), and absolute bounds of ± 0.5 on the value of each transaction (x).