CreditCruncher - Technical Document

Gerard Torrent Gironella

Version 1.1 - R418

Abstract

The CCruncher goal is to compute the credit risk of a portfolio, where investments are fixed income assets, taking into account default correlations between economic sectors. This is done by determining the probability distribution of portfolio loss at time T using the Monte Carlo simulation method and computing risk statistics (Expected Loss, Standard Deviation, Value at Risk, Expected Shortfall).

Keywords: credit risk, Monte Carlo, gaussian copula, Value at Risk, Expected Shortfall.

1 Parameters

This section explains the input parameters required by CCruncher.

1.1 Time nodes

In order to speed up the calculations we precompute asset losses at fixed times. To fix these time nodes we need an *initial date*, the *number of steps* and the *step length*. The probability distribution is computed at date $T = initial\ date + number\ of\ steps \times step\ length$. See figure 1.

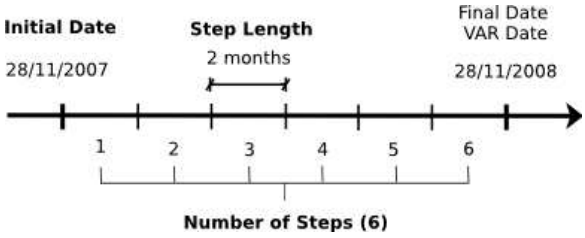

Figure 1: Time nodes

1.2 Ratings and Survival Functions

A rating tells a lender or investor the probability of the subject being able to pay back a loan. A poor rating indicates a high risk of defaulting. Every rating has a survival function associated. This function indicates the probability that a borrower with initial rating X be non-defaulted at time t. The creation of a rating system and the construction of the survival functions are outside the scope of this paper[1].

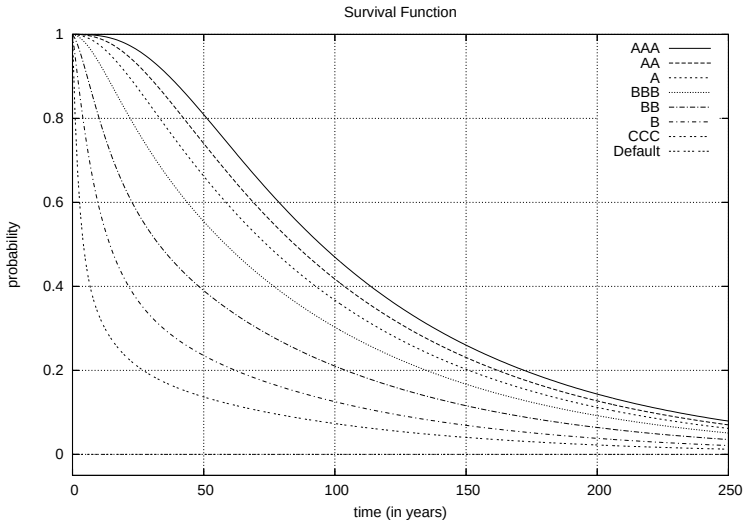

Figure 2: Survival functions

[1]Appendix A.3 shows how to determine the survival functions using the transition matrix.

1.3 Sectors and Correlations

The risk of a credit portfolio depends crucially on default correlations between economic sectors. These sectors are groupings of companies that react similarly to given economic conditions. Example of sectors: energy, financial, technology, media and entertainment, utilities, health care, etc. Default correlations between sectors measures the default dependence between the defined sectors. This can be expressed in tabular form:

	$Sector_1$	$\dots$	$Sector_m$
$Sector_1$	$\rho_{1,1}$	$\dots$	$\rho_{1,m}$
$\vdots$	$\vdots$	$\ddots$	$\vdots$
$Sector_m$	$\rho_{1,m}$	$\dots$	$\rho_{m,m}$

$$\rho_{i,j} = Corr(Sector_i, Sector_j)$$

How to fix the $\rho_{i,j}$ values is outside the scope of this paper.

1.4 Portfolio

Portfolio is composed by borrowers. Each borrower has an initial rating and belongs to a sector. Each borrower have one or more assets. Each asset is defined by its addition date in the portfolio, expected cashflow and forecasted recovery at certain dates. Figure 3 shows the structure of a portfolio.

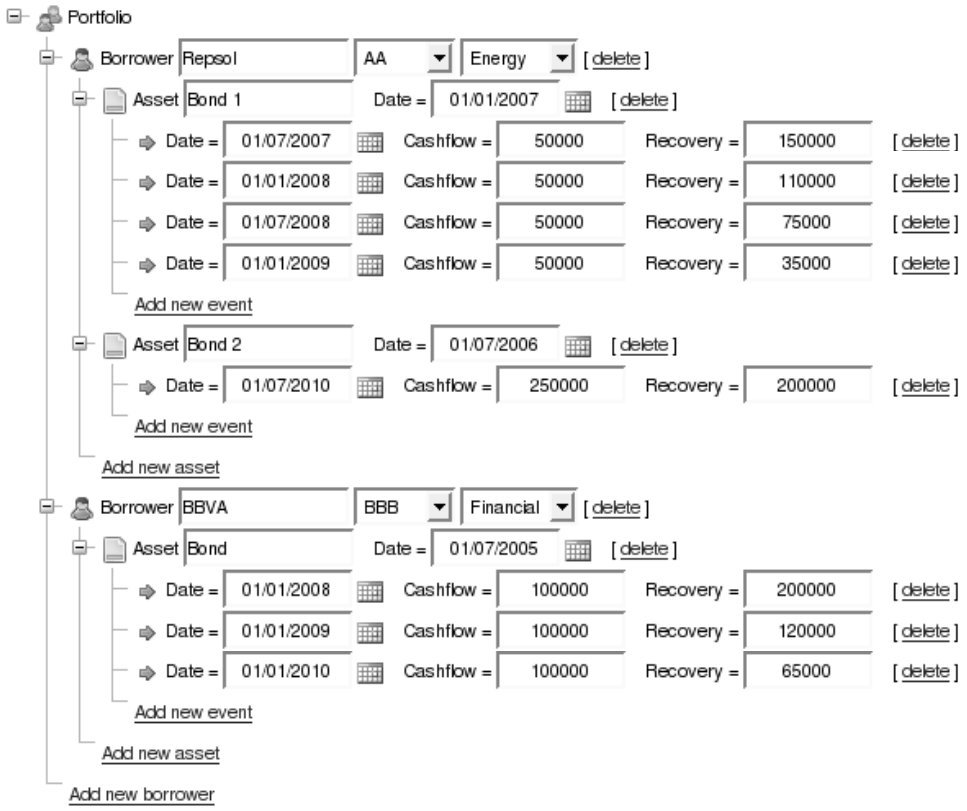

Figure 3: Portfolio example

Cashflow indicates the cash given to the borrower (negative amounts) and the cash received from the borrower (positive amounts) at each date (see figure 4).

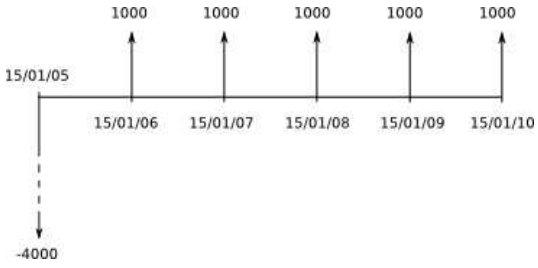

Figure 4: Asset cashflow example

Recovery indicates the cash recovered (after taking legal action against the borrower) if the borrower defaults at a certain date (see figure 5).

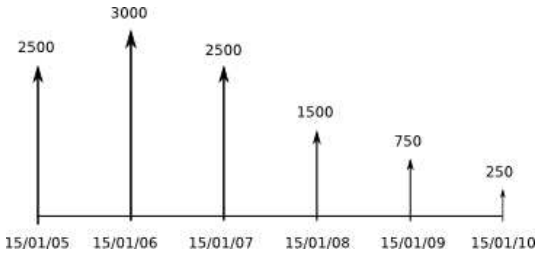

Figure 5: Asset recovery example

If a borrower defaults, the loss due to this default is the sum of all remaining cashflows at default time minus the recovery at default time.

2 Resolution

This section explains how CCruncher computes the credit risk of a portfolio, as defined in the abstract.

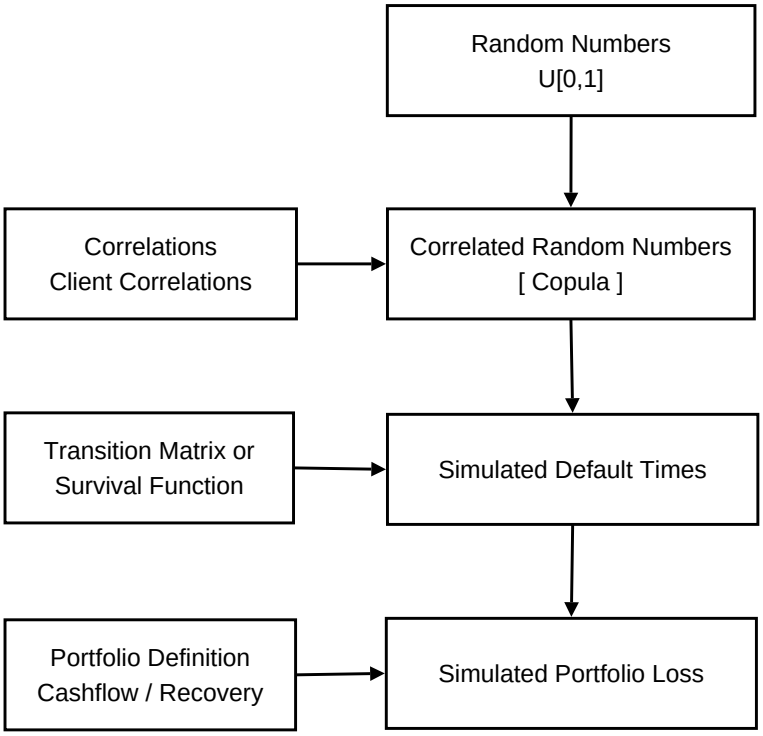

Figure 6: Monte Carlo simulation schema

2.1 Borrowers correlation matrix

We need to translate from default correlations between sectors to default correlations between borrowers. Let us suppose that we have n borrowers and m sectors. Each borrower belongs to a sector. Correlations between sectors are known (see 1.3):

	$Sector_1$	$\dots$	$Sector_m$
$Sector_1$	$\rho_{1,1}$	$\dots$	$\rho_{1,m}$
$\vdots$	$\vdots$	$\ddots$	$\vdots$
$Sector_m$	$\rho_{1,m}$	$\dots$	$\rho_{m,m}$

$$\rho_{i,j} = Corr(Sector_i, Sector_j)$$

We sort the borrowers so that we find those of sector 1 at the beginning and those of sector m at the end. Then we create the borrowers correlation matrix taking as correlation between two borrowers the correlation between their sectors (see figure 7).

$$\begin{pmatrix}
1 & \cdots & \rho_{1,1} & & \rho_{1,k} & \cdots & \rho_{1,k} & & \rho_{1,m} & \cdots & \rho_{1,m} \\
\vdots & \ddots & \vdots & & \vdots & & \vdots & & \vdots & & \vdots \\
\rho_{1,1} & \cdots & 1 & & \rho_{1,k} & \cdots & \rho_{1,k} & & \rho_{1,m} & \cdots & \rho_{1,m} \\
 & & & \ddots & & & & & & & \\
\rho_{1,k} & \cdots & \rho_{1,k} & & 1 & \cdots & \rho_{k,k} & & \rho_{k,m} & \cdots & \rho_{k,m} \\
\vdots & \ddots & \vdots & & \vdots & \ddots & \vdots & & \vdots & & \vdots \\
\rho_{1,k} & \cdots & \rho_{1,} & & \rho_{k,k} & \cdots & 1 & & \rho_{k,m} & \cdots & \rho_{k,m} \\
 & & & & & & & \ddots & & & \\
\rho_{1,m} & \cdots & \rho_{1,m} & & \rho_{k,m} & \cdots & \rho_{k,m} & & 1 & \cdots & \rho_{m,m} \\
\vdots & \ddots & \vdots & & \vdots & \ddots & \vdots & & \vdots & \ddots & \vdots \\
\rho_{1,m} & \cdots & \rho_{1,m} & & \rho_{k,m} & \cdots & \rho_{k,m} & & \rho_{m,m} & \cdots & 1
\end{pmatrix}$$

Figure 7: Borrowers correlation matrix

Observe that this is a correlation matrix (symmetric, definite positive, $|\rho_{i,j}| < 1$, $|\rho_{i,i}| = 1$) composed by blocks. We will use this characteristic to adapt the Cholesky algorithm with the aim to improve memory size and speed.

2.2 Mapping asset losses at fixed time nodes

Fixed time nodes can be different from cashflow dates. Because we want CCruncher to be fast in performing Monte Carlo simulation we precompute the losses of each asset at fixed time nodes. The following paragraphs tell how to compute losses at fixed time nodes.

2.2.1 Asset recovery at fixed time nodes

We consider the recovery at time node k as the closest asset recovery by right. If time node k is previous to the first asset event then the recovery value at time node k is 0. If time node k is later to the last asset event then the recovery value at time node k is 0. Figure 8 shows the recoveries displayed in figure 5 mapped to, in this case, half-yearly time nodes.

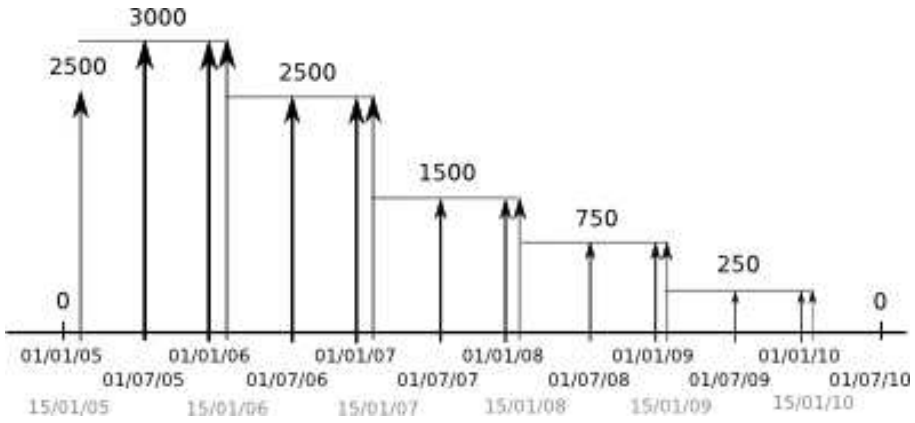

Figure 8: Asset recoveries mapped to fixed time nodes

2.2.2 Asset losses at fixed time nodes

If borrower defaults before the date of asset risk assumption then loss is 0. If borrower defaults after asset finalization date then loss is 0. If borrower defaults at time node k then loss is the averaged sum of all pending cashflows minus the recovery at time node k. Figure 9 shows asset losses mapped to, in this case, half-yearly time nodes of the asset displayed in figures 4, 5 and 8.

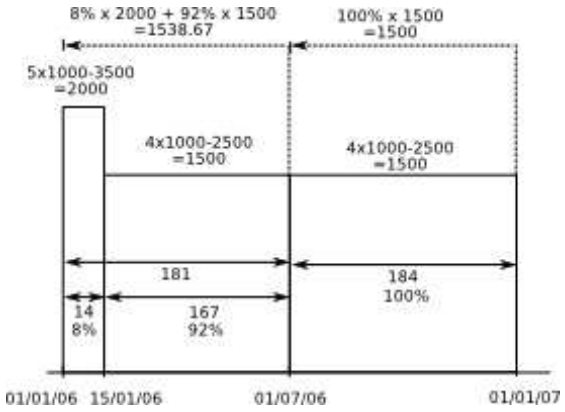

Figure 9: Asset losses mapped to time nodes 01/07/06 and 01/01/07

2.3 Monte Carlo simulation

Monte Carlo methods [5] are a class of computational algorithms for simulating the behavior of various physical and mathematical systems. Each simulation con-

sists of computing a random default time for each borrower and then calculate the loss of the portfolio. We need that default times fulfill two conditions: the borrower's survival functions and the borrower's correlation matrix. This is achieved by simulating a copula. A copula [6] [8] is a multivariate random variable where each component is a uniform $U[0,1]$. Lets go step by step:

2.3.1 Random numbers generation

Each simulation requires a set of random numbers between $[0,1]$ and correlated by the borrowers' correlation matrix, Σ_1 (see 2.1, figure 7). We generate these numbers using the gaussian copula simulation algorithm:

Step 1. We create the covariance[2] matrix Σ_2 mapping Σ_1 component by component:

$$\Sigma_2 = \begin{pmatrix} 2sin(\frac{\pi}{6}) & 2sin(\rho_{12}\frac{\pi}{6}) & \dots & 2sin(\rho_{1n}\frac{\pi}{6}) \\ 2sin(\rho_{12}\frac{\pi}{6}) & 2sin(\frac{\pi}{6}) & \dots & 2sin(\rho_{2n}\frac{\pi}{6}) \\ \vdots & \vdots & \ddots & \vdots \\ 2sin(\rho_{1n}\frac{\pi}{6}) & 2sin(\rho_{2n}\frac{\pi}{6}) & \dots & 2sin(\frac{\pi}{6}) \end{pmatrix}$$

Observe that Σ_2 have diagonal elements with 1 because $2sin(\frac{\pi}{6}) = 1$.

Step 2. We apply Cholesky to Σ_2, then $\Sigma_2 = B \cdot B^\top$, where:

$$B = \begin{pmatrix} b_{11} & 0 & \dots & 0 \\ b_{21} & b_{22} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ b_{n1} & b_{n2} & \dots & b_{nn} \end{pmatrix}$$

Step 3. We simulate[3] a $N(0,1)$ n times:

$$\vec{Y} = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} \qquad y_k \sim N(0,1) \text{ independents}$$

[2]Correlation and covariance matrix are the same because diagonal elements are 1.
[3]The algorithm used to simulate a $N(0,1)$ is explained in appendix A.1

Step 4. We simulate a normal multivariate $\vec{Z} \sim N(\vec{0}, \Sigma_2)$ doing:

$$B \cdot \vec{Y} = \begin{pmatrix} b_{11} & 0 & \dots & 0 \\ b_{21} & b_{22} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ b_{n1} & b_{n2} & \dots & b_{nn} \end{pmatrix} \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} = \begin{pmatrix} z_1 \\ \vdots \\ z_n \end{pmatrix} = \vec{Z}$$

Step 5. Finally we obtain the copula $\vec{U}$ doing:

$$\begin{pmatrix} \Phi(z_1) \\ \vdots \\ \Phi(z_n) \end{pmatrix} = \begin{pmatrix} u_1 \\ \vdots \\ u_n \end{pmatrix} = \vec{U}$$

where $\Phi(x)$ is the $N(0,1)$ cumulative distribution function.

$$\Phi(x) = \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi}} e^{-\frac{t^2}{2}} dt$$

Steps 1 and 2 are done only one time before the simulations. The rest of steps are performed in each Monte Carlo simulation.

In appendix A.2 is described a Cholesky decomposition adapted to block matrix. It allows to work with matrix bigger than 50000×50000.

2.3.2 Default times generation

Given the borrower k and the random number $u_k \in [0,1]$ (generated using a copula in the previous step) we simulate the default time considering the inverse of the borrower survival function at u_k. Figure 10 shows this in a graphic manner.

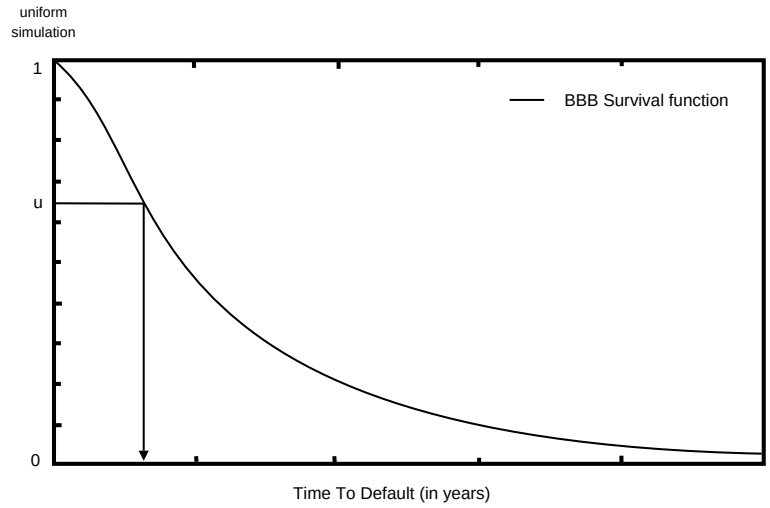

Figure 10: Default time generation with initial rating BBB

2.3.3 Portfolio loss evaluation

At this point, given any borrower, we have a simulated default time. We map this default time to the closest time node to the right. In this node we have the precomputed losses of that borrower's assets. To obtain the simulated value of the portfolio loss we sum all asset losses and we keep this value in a list. Afterwards we will use this list to compute risk.

2.4 Risk computation

After N simulations (eg. 20000, 50000 or more) we have a list of numbers, $x_1, ..., x_N$, where each number represents a simulated portfolio loss. The more values, the more accuracy in the results. All risk statistics (eg. Expected Loss) have an error margin that will be estimated. CCruncher uses *R package*[4] to perform the statistical computations described below.

First of all we can approximate the probability distribution of the portfolio loss creating an histogram with the simulated values.

[4]http://www.r-project.org

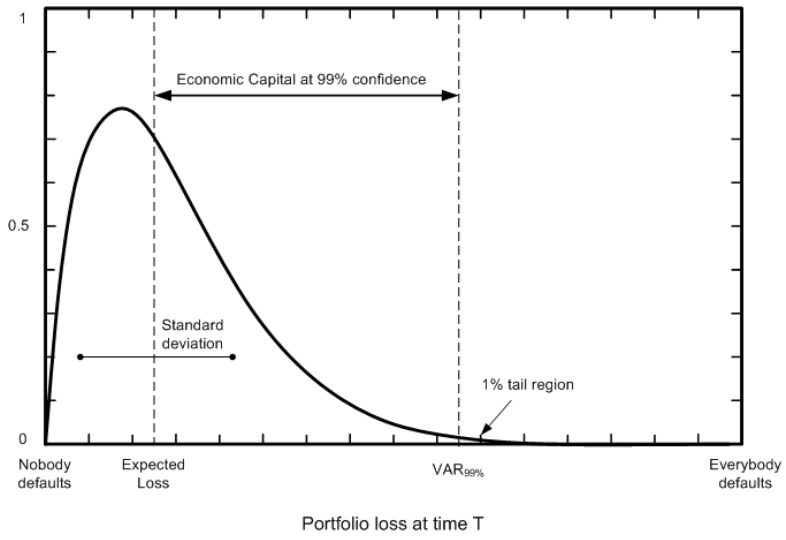

Figure 11: Portfolio loss at time T

2.4.1 Expected Loss

Expected Loss is the probability distribution mean of the portfolio loss. Central Limit Theorem [7] grants that:

$$\mu = \widehat{\mu} \pm \phi^{-1}\left(\frac{1-\alpha}{2}\right) \cdot \frac{\widehat{\sigma}}{\sqrt{N}}$$

where α is the error confidence level, ϕ^{-1} the N(0,1) inverse cumulative distribution function and $\widehat{\mu}$ and $\widehat{\sigma}$ are the mean and stddev estimators:

$$\widehat{\mu} = \frac{1}{N}\sum_{i=1}^{N} x_i$$

$$\widehat{\sigma} = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}\left(x_i - \widehat{\mu}\right)^2} = \sqrt{\frac{1}{N-1}\left(\sum_{i=1}^{N} x_i^2 - \frac{\left(\sum_{i=1}^{N} x_i\right)^2}{N}\right)}$$

2.4.2 Portfolio Loss Standard Deviation

Another usual risk statistic is the standard deviation of the portfolio loss. Central Limit Theorem [7] grants that:

$$\sigma = \widehat{\sigma} \pm \phi^{-1}\left(\frac{1-\alpha}{2}\right) \cdot \frac{\widehat{\sigma}}{\sqrt{2N}}$$

where α is the error confidence level, ϕ^{-1} the N(0,1) inverse cumulative distribution function and $\widehat{\sigma}$ is the stddev estimator:

$$\widehat{\sigma} = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}\left(x_i - \widehat{\mu}\right)^2} = \sqrt{\frac{1}{N-1}\left(\sum_{i=1}^{N} x_i^2 - \frac{\left(\sum_{i=1}^{N} x_i\right)^2}{N}\right)}$$

2.4.3 Value At Risk

Value at Risk [4] is the most used risk value. We call it VAR_β where β is the VAR confidence level (eg. VAR at 95%). VAR is another form to say quantile. Then $VAR_\beta = q_\beta = \inf\{x | F(x) \geq \beta\}$.

$$VAR_\beta = \widehat{q_\beta} \pm \phi^{-1}\left(\frac{1-\alpha}{2}\right) \cdot \text{stderr}(q_\beta)$$

where α is the error confidence level, β is the VAR confidence level, ϕ^{-1} the N(0,1) inverse cumulative distribution function, $\widehat{q_\beta}$ is the quantile estimator, and stderr(q_β) is the estimation of the standard error.

$$\widehat{q_\beta} = x_{k:N}$$

where,

- k fulfills $\frac{k}{N} \leq \beta < \frac{k+1}{N}$
- $x_{k:N}$ is the k-th element of ascendent sorted values

We determine stderr(q_β) using Maritz-Jarret method described in [2].

$$\begin{aligned} M &= [N\beta + 0.5] \\ a &= M - 1 \\ b &= N - M \\ W_i &= B(a, b, \frac{i+1}{N}) - B(a, b, \frac{i}{N}) \\ C_k &= \sum_{i=1}^{N} W_i \cdot x_i \end{aligned}$$

where $[x]$ is the integer part of x and $B(a, b, x)$ is the incomplete beta function:

$$B(a, b, x) = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} \int_0^x t^{a-1}(1-t)^{b-1} dt$$

Then,

$$\text{stderr}(q_\beta) = \sqrt{C_2 - C_1^2}$$

2.4.4 Expected Shortfall

VAR is not a distance because it does not fulfill the sub-additive property [3], $VAR(A+B) \nleq VAR(A) + VAR(B)$. Expected Shortfall is a consistent risk measure [1] similar to VAR. It can be described as the average of the $\beta\%$ worst losses.

$$ES_\beta = \widehat{ES_\beta} \pm \phi^{-1}\left(\frac{1-\alpha}{2}\right) \cdot \text{stderr}(ES_\beta)$$

where α is the error confidence level, β is the ES confidence level, ϕ^{-1} the N(0,1) inverse cumulative distribution function, $\widehat{ES_\beta}$ is the ES estimator and stderr(ES_β) is the estimation of the standard error.

We select the simulation portfolio loss values $(x_1, ..., x_N)$ that are bigger than VAR_β.

$$y_1, y_2, y_3, \cdots, y_K \qquad \text{where} \quad y_i > VAR_\beta$$

Then,

$$\widehat{ES_\beta} = \frac{1}{K}\sum_{i=1}^{K} y_i$$

$$\text{stderr}(ES_\beta) = \frac{\sqrt{\frac{1}{K-1}\sum_{i=1}^{K}\left(y_i - \widehat{ES_\beta}\right)^2}}{\sqrt{K}} = \frac{\sqrt{\frac{1}{K-1}\left(\sum_{i=1}^{K} y_i^2 - \frac{\left(\sum_{i=1}^{K} y_i\right)^2}{K}\right)}}{\sqrt{K}}$$

2.4.5 Economic Capital

We can compute the Economic Capital at confidence level β as:

$$\text{Economic Capital} = VAR_\beta - \text{Expected Loss}$$

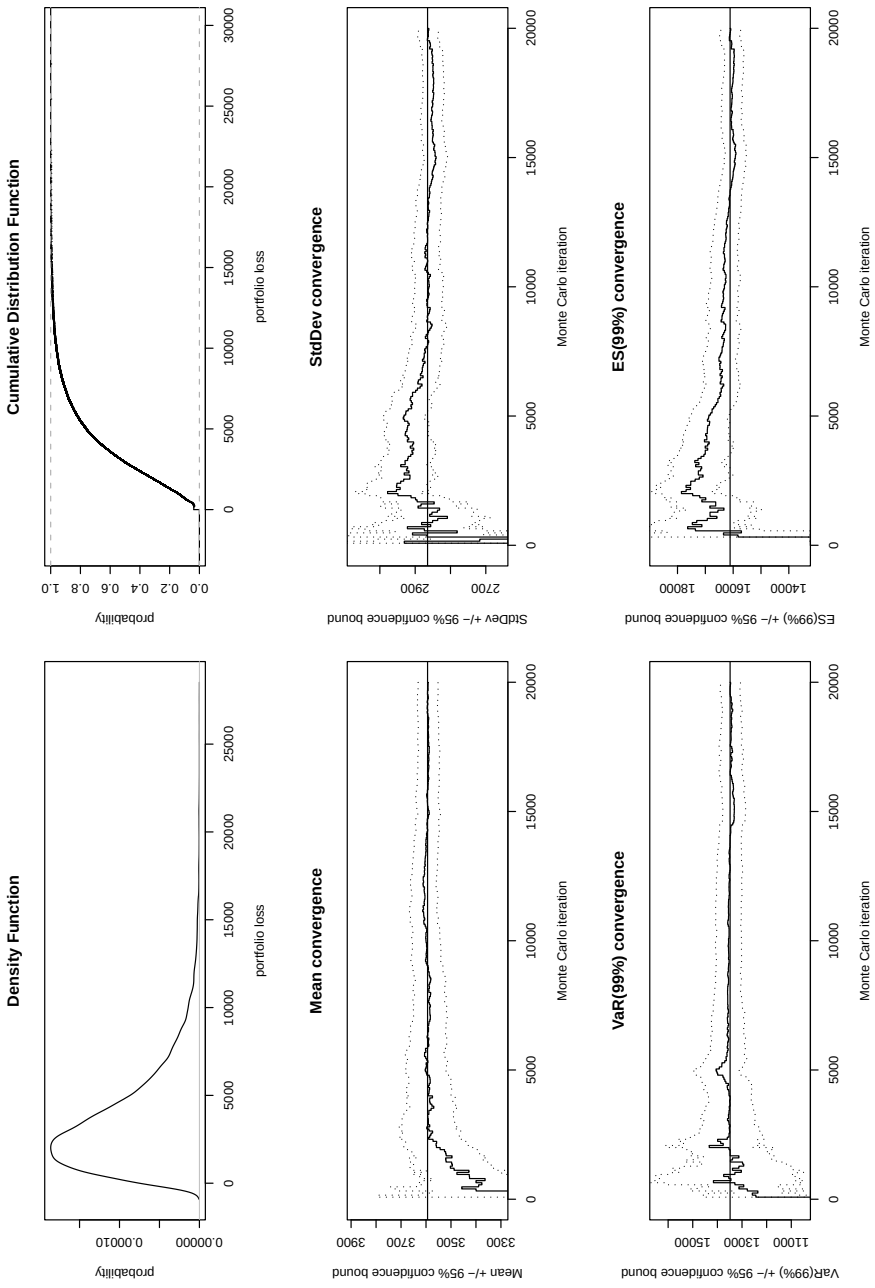

Figure 12: CCruncher results

3 Other considerations

CCruncher takes into consideration other concepts that have not been exposed up to this moment with the purpose of simplifying the content.

3.1 Risk aggregation

CCruncher simulates the whole portfolio loss at time T. It also allows (in the same execution) the simulation of subportfolios by defining one or more aggregators. Consequently you can determine which are the borrowers (or products or branches or regions or assets, etc.) that increase the risk of your portfolio. Every subportfolio has its own list of simulated values that can be processed to obtain the risk indicators for this subportfolio.

3.2 Yield curve

CCruncher can consider that money value decreases along the time following a fixed curve (eg. yield curve). CCruncher allows the input of the yield curve and and takes it into account in all its computations. The coeficient that determines the money value decreases along the time is given by the compound interest formula:

$$\Upsilon(r, t_0, t_1) = (1 + r)^{(t_1 - t_0)}$$

3.3 Antithetic technique

The random number generation using a copula is time expensive. Gaussian copula is symmetric, that means that $(u_1, u_2, \cdots, u_N)$ is equiprobable to $(1 - u_1, 1 - u_2, \cdots, 1 - u_N)$. In CCruncher's antithetic mode, each copula generation is used 2 times, $(u_1, u_2, \cdots, u_N)$ and $(1 - u_1, 1 - u_2, \cdots, 1 - u_N)$, reducing to the half the number of generated copulas.

3.4 Parallel computing

Monte Carlo problems are embarrassingly parallel problems. In the jargon of parallel computing, an embarrassingly parallel workload (or embarrassingly parallel problem) is one for which no particular effort is needed to segment the problem into a very large number of parallel tasks, and there is no essential dependency (or communication) between those parallel tasks.

In CCruncher the master sends the RNG seed to each slave ($seed_i = seed + i \cdot 30001$) and waits for results (the simulated values of the portfolio loss) from slaves

that store them in a file. When one of the two stop criteria is achieved (maximum execution time exceded or maximum number of simulations reached) the master sends a stop signal to the slaves and leaves.

A Appendices

A.1 How to simulate a N(0,1)

In order to simulate $Z \sim N(\mu, \sigma^2)$ values we use:

$$z = \mu + \sigma \cdot \sqrt{-2ln(u_1)} \cdot cos(2\pi \cdot u_2) \qquad u_1, u_2 \sim U[0,1]$$

Where u_i are generated by a Mersenne Twister random number generator.

A.2 Cholesky decomposition for block matrix

Cholesky algorithm decomposes a symmetric positive definite matrix into a lower triangular matrix and the transpose of the lower triangular matrix. Algorithm description can be found in *Numerical Recipes in C*[5].

$$A = U^\top \cdot U$$

If we have a portfolio of 50000 borrowers, the correlation matrix size will be a 50000×50000. This requires up to 19 Gb. of RAM memory. The multiplication of this matrix by a vector implies 2500000000 multiplications. This is far too much. So we adapt the Cholesky algorithm in order to consider that the borrowers' correlation matrix is a block matrix with 1's in diagonal. For instance:

$$A = \left(\begin{array}{cccc|ccc} 1 & 0.5 & 0.5 & 0.5 & 0.1 & 0.1 & 0.1 \\ 0.5 & 1 & 0.5 & 0.5 & 0.1 & 0.1 & 0.1 \\ 0.5 & 0.5 & 1 & 0.5 & 0.1 & 0.1 & 0.1 \\ 0.5 & 0.5 & 0.5 & 1 & 0.1 & 0.1 & 0.1 \\ \hline 0.1 & 0.1 & 0.1 & 0.1 & 1 & 0.3 & 0.3 \\ 0.1 & 0.1 & 0.1 & 0.1 & 0.3 & 1 & 0.3 \\ 0.1 & 0.1 & 0.1 & 0.1 & 0.3 & 0.3 & 1 \end{array}\right)$$

We decompose the previous matrix using the standard Cholesky decomposition:

[5] http://www.nr.com

$$U = \left(\begin{array}{cccc|ccc} 1.00000 & 0.50000 & 0.50000 & 0.50000 & 0.10000 & 0.10000 & 0.10000 \\ 0 & 0.86603 & 0.28868 & 0.28868 & 0.05774 & 0.05774 & 0.05774 \\ 0 & 0 & 0.81650 & 0.20412 & 0.04082 & 0.04082 & 0.04082 \\ 0 & 0 & 0 & 0.79057 & 0.03162 & 0.03162 & 0.03162 \\ \hline 0 & 0 & 0 & 0 & 0.99197 & 0.28630 & 0.28630 \\ 0 & 0 & 0 & 0 & 0 & 0.94975 & 0.21272 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.92563 \end{array}\right)$$

We can see that U have repeated elements that can be kept in RAM memory in this way:

$$U = \left|\begin{array}{c|cc} 1.00000 & 0.50000 & 0.10000 \\ 0.86603 & 0.28868 & 0.05774 \\ 0.81650 & 0.20412 & 0.04082 \\ 0.79057 & 0 & 0.03162 \\ 0.99197 & 0 & 0.28630 \\ 0.94975 & 0 & 0.21272 \\ 0.92563 & 0 & 0 \end{array}\right|$$

that is, for each row we keep the diagonal value and the value of each sector. With this strategy the required memory size is $N \times (M + 1)$ where N is the number of borrowers and M the number of sectors. With this consideration the memory required to store a 50000×50000 borrowers correlation matrix is only 4.2 Mb.

We use the fact that matrix U have repeated elements to reduce the number of operations required to multiply U by a vector. Let see an example:

$$\begin{aligned} &(U \cdot x)_2 = 0.86603 \cdot x_2 + 0.28868 \cdot x_3 + 0.28868 \cdot x_4 + \\ &0.05774 \cdot x_5 + 0.05774 \cdot x_6 + 0.05774 \cdot x_7 \\ &= 0.86603 \cdot x_2 + 2 \cdot 0.28868 \cdot (x_3 + x_4) + 3 \cdot 0.05774 \cdot (x_5 + x_6 + x_7) \end{aligned}$$

Considering this, we can reduce the number of operations from N^2 to $N \times (M + 1)$ where N is the number of borrowers and M the number of sectors. In the 50000 borrowers example, the operations number reduces from 2500000000 to only 500000.

A.3 From transition matrix to survival function

The T-years transition matrix gives the probability to jump from rating r_i to rating r_j in a term of T years.

$$M_T = \begin{pmatrix} m_{1,1} & \dots & m_{1,n} \\ \vdots & \ddots & \vdots \\ m_{n,1} & \dots & m_{n,n} \end{pmatrix} \qquad m_{i,j} = P(r_i \rightarrow r_j; T)$$

where n is the number of ratings and $m_{i,j}$ is the probability that a borrower with rating r_i jumps to r_j in T years. Figure 13 shows a transition matrix example where the probability that a borrower with rating AA jumps to rating B in 1 year is 0.14%.

	AAA	AA	A	BBB	BB	B	CCC	Default
AAA	90.81	8.33	0.68	0.06	0.12	0.00	0.00	0.00
AA	0.70	90.65	7.79	0.64	0.06	0.14	0.02	0.00
A	0.09	2.27	91.05	5.52	0.74	0.26	0.01	0.06
BBB	0.02	0.33	5.95	86.93	5.30	1.17	0.12	0.18
BB	0.03	0.14	0.67	7.73	80.53	8.84	1.00	1.06
B	0.00	0.11	0.24	0.43	6.48	83.46	4.07	5.21
CCC	0.22	0.00	0.22	1.30	2.38	11.24	64.86	19.78
Default	0.00	0.00	0.00	0.00	0.00	0.00	0.00	100.00

Figure 13: 1-year transition matrix

Transition matrix can be scaled in time using the following rules:

$$\begin{aligned} M_{T_1+T_2} &= M_{T_1} \cdot M_{T_2} \\ M_{k \cdot T} &= M_T^k \\ M_{\frac{T}{k}} &= \sqrt[k]{M_T} \end{aligned} \tag{1}$$

The root of a matrix can be computed as:

$$M = P^{-1} \cdot D \cdot P \longrightarrow M^\gamma = P^{-1} \cdot D^\gamma \cdot P$$

where P is a matrix composed by the eigenvectors of M and D is a diagonal matrix composed by the eigenvalues of M. The inverse of a matrix can be computed using the LU decomposition as it is explained in *Numerical Recipes in C*[6]:

$$Id = P^{-1} \cdot P = L \cdot U \cdot P$$

[6]http://www.nr.com

This allows us to compute default probability at any time and at any initial rating (that is, the survival functions), doing:

$$Survival(r_i, t) = (M_t)_{i,n}$$

where r_i is the initial rating, t is the time, M_t is the transition matrix for time t (scaled from M_T using (1)) and n is the index of default rating r_n.

References

[1] Dirk Tasche Carlo Acerbi. Expected shortfall: a natural coherent alternative to value at risk. *BIS*, 2001.

[2] Jonathan E. Grindlay Jaesub Hong, Eric M. Schlegel. New spectral classification technique for x-ray sources: quantile analysis. 2004.

[3] Stefan R. Jaschke. Quantile-var is the wrong measure to quantify market risk for regulatory purposes. *BIS*, 2001.

[4] Philippe Jorion. *Value at Risk*. McGraw-Hill, 1997.

[5] Mervyn Marasinghe. Monte carlo methods. Class notes for Iowa State University, Dept. of Statistics.

[6] Alexander McNeil Paul Embrechts and Daniel Straumann. Correlation and dependence in risk management: properties and pitfalls. *RiskLab*, 1999.

[7] Murray R. Spiegel. *Estadística*. Schaum, 1997.

[8] Shaun S. Wang. Aggregation of correlated risk portfolios: Models & algorithms. *CAS Committee on Theory of Risk*.