**Computing the Volume of an high-dimensional semi-unsupervised Hierarchical copula**

<table>
<thead>
<tr>
<th>Journal:</th>
<th><em>International Journal of Computer Mathematics</em></th>
</tr>
</thead>
<tbody>
<tr>
<td>Manuscript ID:</td>
<td>GCOM-2010-0395-B.R2</td>
</tr>
<tr>
<td>Manuscript Type:</td>
<td>Original Article</td>
</tr>
<tr>
<td>Date Submitted by the Author:</td>
<td>03-Dec-2010</td>
</tr>
<tr>
<td>Complete List of Authors:</td>
<td>Bernardi, Enrico; University of Bologna, Matemates Romagnoli, Silvia; University of Bologna</td>
</tr>
<tr>
<td>Keywords:</td>
<td>copula volume, Hierarchical copula, clustering methods, high dimensional problems, MatLab code</td>
</tr>
</tbody>
</table>

Note: The following files were submitted by the author for peer review, but cannot be converted to PDF. You must view these files (e.g. movies) online.

IJCM.zip
Computing the Volume of an high-dimensional semi-unsupervised Hierarchical copula

Enrico Bernardi*  Silvia Romagnoli†

December 1, 2010

Abstract

We propose an algorithm for the computation of the volume of a multivariate copula function (and the probability distribution of the counting variable linked to this multidimensional copula function) which is very complex for large dimensions. As it is common practice for large dimensional problem, we restrict to positive orthant dependence and we construct a Hierarchical copula which describes the joint distribution of random variables accounting for dependence among them. This approach approximates a multivariate distribution function of heterogenous variables with a distribution of a fixed number of homogenous clusters, organized through a semi-unsupervised clustering method. These clusters, representing the second level sectors of Hierarchical copula function, are characterized by an into-sector dependence parameter determined by a method which is very similar to the Diversity Score method. The algorithm, implemented in MatLab™ code, is particularly efficient allowing us to treat cases with a large number of variables, as can be seen in our scalability analysis. As an application, we study the problem of valuing the risk exposure of an insurance company, given the marginals i.e. the risks of each policy.

*University of Bologna, Department of Mathematical Economics, Viale Filopanti 5, 40126 Bologna, Italy. Phone: +(39) 0512094345; Fax: +(39) 0512094357. e-mail: enrico.bernardi@unibo.it
†University of Bologna, Department of Mathematical Economics, Viale Filopanti 5, 40126 Bologna, Italy. Phone: +(39) 0512094340; Fax: +(39) 0512094340. e-mail: silvia.romagnoli@unibo.it
Keywords: copula volume, Hierarchical copula, clustering unsupervised method, high dimensional problem, MatLab\textsuperscript{TM} code.

1 Introduction

Copula functions represent a methodology which has recently become the most significant new tool to handle in a flexible way the co-movement between markets, risk factors and other relevant variables studied in finance. A \( n \)-dimensional copula is a function \( C \) from the \( n \)-dimensional hypercube to \([0,1]\) such that: i) \( C \) is grounded; ii) its one-dimensional margins are the identity function; and iii) \( C \) is \( n \)-increasing, (see Nelsen, 2006 and Joe, 1997). The last requirement is strictly linked to the notion of copula's volume; in effect it is equivalent to require the positivity of the copula's volume between two points. In Cherubini and Romagnoli (2009) an algorithm is proposed to compute analytically such volume; since it must evaluate a set of \( 2^n \) terms, that algorithm becomes rapidly unusable for problems of dimension higher than 20. Our main goal here is to present an algorithm allowing to treat large dimensional problems.

This algorithm is composed by two parts: the first one is the preparing phase while the second is the computational one. In the preparing phase we reduce the dimension of the problem. Our algorithm organizes via a semi-unsupervised method (K-means) the variables into a fixed number of groups which will be then considered as clusters of homogeneous variables. Such approximation will be more accurate as the number of clusters that we decide to set, increases. The dependence parameters in the groups, or the number of variables to consider will be determined by a procedure which is in the same spirit of the Diversity Score approach. After this preparation procedure, the computational phase aggregates the groups with a Hierarchical Archimedean copula function since we restrict to the positive orthant dependence, as it’s usual for the high dimensional problem. Along the same lines, Savu and Trede (2006) propose a class of Hierarchical Archimedean copulas to model high-dimensional joint distributions of asset returns organized in supervised clusters.

For the applications of this problem we recall that the computation of the survival copula, frequent in finance because it corresponds to the put-call parity in multidimensional setting (Cherubini et all., 2004), is a particular case of copula volume’s computation. Furthermore we observe that the
volume is involved in the computation of the distribution function of the
counting variable linked to the multidimensional copula.
The paper is organized as follows: in section 2 we present and discuss the
preparing phase or the clustering method and the homogenous approxima-
tion, in section 3 we analyze the computational phase from a theoretical
point of view and finally in section 4 we propose an empirical application
and a scalability analysis of our algorithm. Section 5 concludes. In the
Appendix we present the MatLab™ code for our procedures.

2 The preparing phase

Here we explain the clustering procedure of the original sample and the
homogenous approximation, used in order to reduce the complexity of the
problem.

2.1 The clustering method

The primary objective of cluster analysis is to partition a given data of mul-
tidimensional vectors (patterns) into so-called homogeneous clusters such
that patterns within a cluster are more similar to each other than patterns
belonging to different clusters. The best known and most widely used algo-
rithm to reach this goal is K-means. Lately neural networks, for example,
competitive-learning networks, self-organizing feature maps and adaptive
resonance theory networks have been also often used to cluster data. Each
approach has its own pros and cons. In the application chosen in section 4
we opted for K-means but the method proposed may be implemented simi-
larly grouping the data with other clustering methods.
K-means clustering is a partitioning method, i.e. the function K-means par-
titions the observations in our data into K mutually exclusive clusters and
returns a vector of indices pointing to the clusters which have been assigned
to each observation.
K-means treats each observation in our data as an object having a location
in space. It finds a partition in which objects within each cluster are as close
to each other as possible, and as far of objects in other clusters as possible.
Each cluster in the partition is defined by its member objects and by its centroid, i.e. the point to which the sum of distances from all objects in that cluster is minimal. K-means computes cluster centroids differently for each distance measure, to minimize the sum with respect to the chosen measure (it is the mean of the points in each cluster for our choice of distance measure).

K-means then uses an iterative algorithm that minimizes the sum of distances from each object to its cluster centroid, over all clusters. This algorithm moves objects between clusters until the sum cannot be decreased further. The result is a set of clusters that are as compact and well-separated as possible.

K-means is a semi-supervised method since it requires to choose a distance metric and a number of clusters further discussed in the following subsections. Moreover in section 4, we propose a comparison between our algorithm implemented with K-means and with a completely unsupervised clustering method, i.e. with Kohonen self-organizing maps (SOM).

2.1.1 Metric selection

In order to mathematically identify clusters in a data set, it is usually necessary to first define a measure of similarity or proximity which will establish a rule for assigning patterns to the domain of a particular cluster centered. As it is to be expected, one of the most popular is the Euclidean distance which is one of the most common Minkowski distance metric. By using the Euclidean distance, K-means algorithm tends to detect hyperspherical-shaped clusters. Since clusters can be of arbitrary shapes and size, the Minkowsky metrics may not be a good choice a priori. In Su and Chou (2001) a new non-metric measure is proposed, based on the idea of symmetry which can be used to group a given data set into a set of clusters of different geometrical structures. In Wang et al. (2006) through observing the characteristic property of data clustering, a new data-dependent dissimilarity measure is designed, called density-sensitive distance metric, which has the property of elongating the distance among points in different high density region and then can reflect the characters of data clustering. The density-sensitive K-means algorithm can identify non-convex clustering structures, thus generalizing the application area of the original K-means algorithm. In the same spirit Xing et al. (2002) determines the metric as
a parameterized Mahalanobis distance which solves a convex-optimization
problem for a given data set.
On the other hand other researchers have focussed more on the performance
of a metric in a randomized data set, than in the choice of a particular met-
ic, arguing that in the case of absence of noise of any kind, virtually all
metrics are equivalent. In Francois et al. (2005) it is suggested to choose
the metric according to the shape of the noise that is assumed on the data.
While it is known that the Euclidean metric is optimal in presence of white
Gaussian noise, other type of noise require other metric such as the frac-
tional norm which is proved to be a better dissimilarity estimator than the
Euclidean norm when a highly colored noise is present.
Given the kind of data that we are clustering in this problem, we choose to
work with the squared Euclidean distance as a distance measure. In section
4 we discuss some experiments on the performance of this choice for some
randomized data.

2.1.2 Cluster Number Selection
The choice of the number of groups, given the dimensionality of the problem,
plays an important role since it affects the between and within dependence
parameters and then also the precision of the method. This is a critical
problem which involves the consideration of the trade-off between the com-
plicity of the problem in term of time of computing and the precision of
the method. Several papers address the problem of cluster number selection
by using a K-means approach that exploits local changes of internal validity
indices to split or merge clusters. In Pelleg and Moore (2000) this issue
is addressed by developing a dynamic K-means variant known as X-means
that performs bisecting splits of clusters where the resulting clustering solu-
tion is addressed by the Bayesian Information Criterion. Here the bisecting
K-means algorithm splits the data samples into two disjoint clusters and if
the split increases the overall fitness measured by internal validity indices,
the cluster is split and the bisecting K-means continues recursively. If none
of the bisecting steps of each existing cluster leads to an improved result,
the algorithm stops and returns the current clusters as result. In Muhir and
Granitzer (2009) a comparison is proposed between an X-means method
with several internal multivariate validity indices that measure cluster qual-
ity based on the total scatter of between-cluster and within-cluster sum of
squares, as the Calinski and Harabasz index (see Calinski and Harabasz, 1974), the Hartingan index (see Hartigan, 1985) and the Krzanowski and Lai index (see Krzanowski and Lai, 1985).

To further emphasize how the implementation of the K-means technique applies to our problem, we propose to determine the best cardinality of groups given a subjective tolerance $\delta$, defined as the minimum number of groups such that the maximum distance, on the chosen metric, of each data to the relative centroid is not greater than the tolerance. In this way, if the distance of a data is greater than the tolerance, one more group is made. Formally, if we work on the Euclidean metric, we can define the $\delta$-optimal number of groups, $n_\delta$, as the argument of the following optimization problem with constrain:

$$
n_\delta = \arg\min_{u_j - c_i \leq \delta, \forall i, j} (CER(i))
$$

where $u_j$ are the data, $c_i$ are the centroids and the clustering error (CER) to minimize is

$$
CER(i) = \sum_{w=1}^{i} \sum_{j=1}^{n_i} (u_j - c_i)^2
$$

where $n_i$ is the number of data in the $i$-th group. We can observe that CER is very similar to the clustering error defined as sum of confusions in Wang et al. (2006).

We can notice that the selection of $n_\delta$ may be included in the K-means procedure with a simple if/else code-block.

### 2.2 The Homogenous approximation

In this subsection we discuss the homogeneous and heterogeneous group, so it is useful to first specify what is the meaning in this context that the elements inside the group are or are not equal in value.

The clustering method assembles some groups which must be considered homogenous to simplify the computation in case of high dimensional problems. To this aim we develop a procedure allowing us to consider that the variables are homogenous. In order to do that we must compensate the homogenous assumption with a change in the dependence parameter inside the group or in the size of the group. This problem is similar to that of
reducing the actual portfolio of collateral to a synthetic pool of identical assets, each defaulting independently, in the analysis of cash flow collateralized debt obligation. The Diversity Score, which is defined as the number of assets in the synthetic pool, was firstly proposed in Cifuentes and O’Connor (1996) and following in the Moody’s alternative version in Cifuentes et al. (1999) and some applications of the model was discussed in Schorin and Weinreich (1999) and in Duffie and Gârleanu (2001).

To determine the dependence parameters we propose a method along the lines of the Diversity Score approach, which is inapplicable in our particular setting though. Embrechts et al. (2003) have shown that there must be a higher degree of dependence for variates linked at a lower level, than between those linked only at higher level to satisfy the conditions for a proper multidimensional distribution.

For this reason we can’t transform the groups into homogenous and independent samples but we must consider only the homogenous transformations. In Rouvinez (2003) the “equivalent number of assets” is defined, which is an alternative method to the Diversity score one, where the independent request is dropped.

If we compensate with the dependence parameter, we have the following:

**Definition 1** The Equivalent Dependence Parameter (EQDP) for the copula function $C$ of group $i$, $i = 1, \ldots, n$ is $\alpha_i$ such that:

$$C(u_1^i, u_2^i, \ldots, u_m^i, \alpha_i) = C(c_1, \ldots, c_i, \alpha_i)$$

where $u_j^i, j = 1, \ldots, m$ are the heterogenous marginals of the group $i$, $C(c_1, \ldots, c_i, \alpha_i)$ is an $m$-variate copula with homogeneous marginals $c_i$ equal to the centroid of the $i$-th cluster and $\alpha_i$ is the standard estimation of the dependence parameter within the heterogenous group for the same copula function.

On the other hand, if we decide to change the number of the variables to be considered into the homogenous group, we have the following:

**Definition 2** The Equivalent Number of Variables (ENV) for the copula function $C$ of group $i$, $i = 1, \ldots, n$ is the dimension $m_i$ of copula $C$ after the homogeneous approximation:

$$C(u_1^i, u_2^i, \ldots, u_m^i, \alpha_i) = C(c_1, \ldots, c_i, \alpha_i)$$
where \( u_j, j = 1, \ldots, m \) are the heterogenous marginals of group \( i \), \( C(c_1, \ldots, c_i, \tilde{\alpha}_i) \) is an \( m_i \)-variate copula with homogeneous marginals \( c_i \) equal to the centroid of the \( i \)-th cluster and \( \tilde{\alpha}_i \) is the standard estimation of the dependence parameter within the heterogenous group for the same copula function.

3 The Computational phase

We recall that the volume of copula function in \( n \) dimensions may be represented in a combinatorial form which is well suited for computational purposes (Cherubini and Romagnoli, 2009). Given a copula function \( C \), the volume of the \( n \)-dimensional box \( S = [u, v] \) with \( u, v \in [0, 1]^n, u \leq v \), may be represented as:

\[
V_C(S) = \sum_{i=0}^{2^n-1} (-1)^{k(i)} C(c_n(p_n(i)))
\]

where \( p_n(i) \) denotes the \( n \)-dimensional vector of zeros and ones corresponding to the binary representation of \( i \), \( k(i) \) is the number of elements equal to one in \( p_n(i) \), and \( c_n(p_n(i)) \) is an \( n \)-dimensional vector such that \( c_j = v_j \) if \( p_j = 0 \) and \( c_j = u_j \) if \( p_j = 1 \), where \( u_j, v_j \) and \( c_j \) denote the \( j \)-th element of the corresponding vector.

We refer to the previous representation where we introduce a Hierarchical copula function. These copulas extend the class of Archimedean copulas to account for non-exchangeability (Joe, 1997; Mc Neil et al., 2005; Savu and Trede, 2006)

\[
HC(u_1, u_2, \ldots, u_n) \equiv \phi_s^{-1}(\phi_s \diamond \phi_s^{-1}(\phi_s^1(u_1)+\phi_s^1(u_2))+\ldots+\phi_s \diamond \phi_s^{-1}(\phi_s^1(u_{n-1})+\phi_s^1(u_n)))
\]

where \( \phi_s \) represents the generator of the Archimedean copula representing the dependence between groups, \( \phi_j, j = 1, \ldots, n \) represents the dependence within groups and \( \diamond \) is the compound operator. We can apply the previous representation of the volume of copula function to the Hierarchical class.

**Proposition 3** (Hierarchical copula volume) Given a Hierarchical copula function \( HC \) with \( k \) groups of dimension \( m_s, s = 1, \ldots, k \) respectively such that \( \sum_{s=1}^k m_s = n \), the volume of the \( n \)-dimensional box \( S = [u, v] \) with
\( \mathbf{u}, \mathbf{v} \in [0, 1]^n, \mathbf{u} \leq \mathbf{v} \), may be represented as:

\[
V_{HC}(S) = \sum_{i=0}^{n} (-1)^i \sum_{j=1}^{D^r(i,k)} HC(c(p_{i,k}(j)))
\]

where the coordinates defining the box \( S \) are obtained by the matrix \( \tilde{\mathbf{u}}, \tilde{\mathbf{v}} \in [0,1]^k \times [0,1]^{\max\{m_s, s = 1, \ldots, k\}} \) whose \((s, w)\)-th element corresponds to the \( w \)-th position in the \( s \)-th group\(^1\), where \( D^r(i,k) \) denotes the number of the ordered compatible combinatorial distributions (o.c.c.ds in the following)\(^2\) of the integer \( i \) into \( k \) groups, \( p_{i,k}(j) \) is a \( k \times \max(m_s, s = 1, \ldots, k) \) matrix of zeros and ones corresponding to the \( j \)-th o.c.c.d. of \( i \) ones into \( k \) groups of dimensions \( m_s, s = 1, \ldots, k \), \( c(p_{i,k}(j)) \) is a \( k \times \max(m_s, s = 1, \ldots, k) \) matrix such that \( c_{s,w} = \tilde{v}_{s,w} \) if \( p_{s,w} = 0 \) and \( c_{s,w} = \tilde{u}_{s,w} \) if \( p_{s,w} = 1 \), where \( \tilde{u}_{s,w}, \tilde{v}_{s,w} \) and \( c_{s,w} \) denote the \((s, w)\)-th element of the corresponding matrix and \( HC(c(p_{i,k}(j))) \) is the Hierarchical copula computed for the \( j \)-th o.c.c.d.

Remark 4 We observe that \( D(i,k) \) can be determined by assuming to have a number of groups equal to \( n \), i.e. to the sum of groups’ cardinalities, each one with cardinality equal to 1, so that one can take into account the order of the groups. Then, computing \( D(i,k) = n^i \), we must filter these o.c.d. with respect to the groups’ cardinalities, reducing to the compatible ones, i.e. \( D^r(i,k) \).

For example, if we have two groups each one with cardinality three, the number of o.c.d.s of \( i = 1 \) is \( D(1,2) = D^r(1,2) = 6 \) since all the o.c.d.s are compatibles with the groups’ cardinalities. For all \( i \) we assume to order, in some way, the o.c.d.s according to some order index \( j \). This way, the second o.c.c.d. for \( i = 1 \) and two groups, each one with cardinality three, corresponds (for the chosen ordering criteria which is the row’s order here) to

\[
\mathbf{p}_{1,2}(2) = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}
\]

where the first row corresponds to the first group and the second to the second

\(^1\)The coordinates of the box \( S \) are computed as \( \mathbf{u} = \text{col}(\tilde{\mathbf{u}}), \mathbf{v} = \text{col}(\tilde{\mathbf{v}}) \), where the function \( \text{col} : [0,1]^{\max(m_s, s = 1, \ldots, k)} \times [0,1]^k \rightarrow [0,1]^n \) sets orderly all the columns of a matrix into the same column.

\(^2\)We refer to \( D(i,k) \) as the number of the ways in which one can distribute the integer \( i \) into \( k \) groups taking into account the order into the groups (o.c.d.s for short). If we also take into account the cardinalities of the groups, we refer to \( D^r(i,k) \) as the number of the ordered compatible (for groups’ cardinalities) combinatorial distributions.
one. We observe that we can choose any ordering criteria since the final order of the o.c.c.ds is not relevant.

We can observe that the problem is simplified assuming that the groups are homogeneous.

**Proposition 5** (Hierarchical Homogeneous copula volume) Given a Hierarchical copula function $HHC$ with $k$ homogeneous groups of dimension $m_s, s = 1, \ldots, k$ respectively such that $\sum_{s=1}^{k} m_s = n$, the volume of the $k$-dimensional box $S = [u, v]$ with $u, v \in [0, 1]^k$, $u \leq v$, may be represented as:

$$V_{HHC}(S) = \sum_{i=0}^{n} (-1)^i \sum_{j=1}^{\hat{D}(i,k)} HHC(c(p_{i,k}(j)))$$

where $\hat{D}(i,k)$ denotes the number of compatible combinatorial distribution (c.c.ds in the following)\(^3\) of the integer $i$ into $k$ groups, $p_{i,k}(j)$ is a $k \times \max(m_s, s = 1, \ldots, k)$ matrix of zeros and ones corresponding to the $j$-th c.c.d. of $i$ ones into $k$ groups of dimensions $m_s, s = 1, \ldots, k$, $c(p_{i,k}(j))$ is a $k \times \max(m_s, s = 1, \ldots, k)$ matrix such that $c_{s,w} = v_s$ if $p_{s,w} = 0$ and $c_{s,w} = u_s$ if $p_{s,w} = 1$, where $u_s, v_s$ are the $s$-th element of the corresponding vector and $c_{s,w}$ denote the $(s,w)$-th element of the corresponding matrix and $HHC(c(p_{i,k}(j)))$ is the hierarchical copula computed for the $j$-th c.c.d. accounting for the permutation of the elements into the groups.

**Remark 6** We can observe that $\hat{D}(i,k)$ can be computed by a number of permutations with repetitions, i.e.

$$\hat{D}(i,k) = \binom{n+k-1}{k-1}$$

For example if we assume two groups each one with cardinality three, the number of c.d.s of $i = 1$ is $\hat{D}(1,2) = \hat{D}(1,2) = 2$ since all the c.d.s are compatibles with the groups’ cardinalities. In this framework these two c.d.s correspond to the following situations:

\(^3\)We refer to $\hat{D}(i,k)$ as the number of the ways in which one can distribute the integer $i$ into $k$ groups without taking into account the order of the groups (c.d.s for short). If we take into account the groups’ cardinalities, then we refer to $\hat{D}(i,k)$ as the number of the compatible (for cardinalities’ groups) combinatorial distributions.
1. the integer 1 is placed in the first group, independently of the position (for convention in the first place);

2. the integer 1 is placed in the second group, independently of the position (for convention in the first place).

If we order these c.c.ds, then the second c.c.d. for $i = 1$ distributed into two groups each one with cardinality three, corresponds (for the chosen ordering criteria given here by the column’s order) to

$$p_{1,2}(2) = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

We can observe that the computation of the Survival copula function is a particular case of the volume computation:

**Corollary 7** (Survival copula function in $n$ dimension) In the same setting of Proposition 5, the survival of the Hierarchical copula function $HHC$ with marginals $u \in [0,1]^k$, may be represented as:

$$V_{HHC}(S) = \sum_{i=0}^{n} (-1)^i \sum_{j=1}^{D^*(i,k)} HHC(c(p_{i,k}(j)))$$

where the box $S = [u,e]$ and where $e = (1,1,\ldots,1)$ is the unit vector in $\mathbb{R}^k$.

The volume of a copula is involved into the computation of the probability distribution of the counting variable linked to copula function:

**Definition 8** (Counting variable linked to $n$- dimensional copula function) In the same setting of Proposition 5, the counting variable linked to the copula function $HHC(c(p_{i,k}(j)))$ is given by

$$r(j) = \sharp(p_{i,k}(j))$$

which counts the number of elements equal to one in the matrix $p_{i,k}(j)$ for the $j$-th c.c.d.. Clearly we have $r(j) = 1, \forall j$. 

11
We can recover the distribution of the counting variable linked to the \(n\)-dimensional copula function. To do this we observe that a number \(l\) must correspond, through his probability function, to a number in \([0, 1]\) by considering all the c.c.d.s of \(l\) ones into \(k\) groups (we enumerate this c.c.d. with the index \(j\)). The \(j\)-th c.c.d., corresponding to the matrix \(p_{l;k}(j)\), generates a pair of coordinates\(^4\), i.e. the \(j\)-th c.c.d. generates the box \(S^j\) for which we compute the volume. The sum of the volumes computed for all the coordinates generated by all the c.c.d.s for the same number of ones, i.e. for the same \(l\) ones into \(k\) groups, represents the probability to count \(l\) ones, or better to have \(l\) events whose probabilities are the marginal centroids.

**Corollary 9** (Probability distribution of the counting variable linked to the \(n\)-dimensional copula) In the same setting of Proposition 5, the probability distribution of the counting variable linked to the \(n\)-dimensional copula \(HHC\), with \(k\) homogeneous groups of dimension \(m_s, s = 1, \ldots, k\) respectively such that \(\sum_{s=1}^{k} m_s = n\) and with centroids \(u \in [0, 1]^k\), is the function \(F_r : [0, n] \rightarrow [0, 1]\) such that:

\[
F_r(i) = \sum_{l=0}^{i} \sum_{j=1}^{D^r(l,k)} V_{HHC}(S^j(p_{l;k}(j)))
\]

where \(V_{HHC}(S^j)\) is the volume of the hierarchical homogeneous copula computed for the box \(S^j = [u^j, v^j] \in \mathbb{R}^k \times \mathbb{R}^k\), determined for the \(j\)-th c.c.d. of \(l\) ones into \(k\) groups such that, if \(p_{s,w} = 0, u^j_s = 0\) and \(v^j_s = u_s\), if \(p_{s,w} = 1, u^j_s = u_s\) and \(v^j_s = 1\), where \(p_{l;k}(j)\) is a \(k \times \max(m_s, s = 1, \ldots, k)\) matrix of zeros and ones corresponding to the \(j\)-th c.c.d. of \(l\) ones into \(k\) groups of dimensions \(m_s, s = 1, \ldots, k\) and finally accounting for the permutation of the elements into the groups.

Referring to the representation of copula’s volume of Proposition 5, in the Appendix we lay out the computational phase detailing the MatLab\textsuperscript{TM} code of our algorithm.

\(^4\)The rule which explains how the coordinates are generated, is presented in the following Corollary.
3.1 Copula’s volume vs Probability distribution

We observe that we have found all the elements which may be used to compute the copula’s volume but also the distribution of counting variable linked to the multidimensional copula function. In this case, each row of the matrix 0-1 (\( p_{4,3}(2) \) in the example in the Appendix), generates the coordinates of a volume corresponding to a probability and then, by aggregation, corresponding to an element of the probability distribution of the counting variable (see corollary 9).

Finally, we remark that the computation of the survival copula function is a particular case of volume computation (see Corollary 7).

4 Empirical Application

We consider here an insurance company, that for simplicity operates only in the life branch. The company evaluates the marginal probabilities of death of each insured subject but the total probability of default isn’t simply assessable for a high number of policies. We face here the problem of determining the survival copula function corresponding to the probability of total survivals. We observe that for large-dimensional problems like the one we are dealing with, it’s necessary to reduce its complexity and we propose here the application of a semi-supervised clustering method. In this example we consider 80 policies.

K-mean clustering method based on the Euclidean metric with tolerance \( \delta = 0.00039 \), generates 5 groups with the following cardinalities:

\[
\text{clusterVector} = [11 \ 31 \ 30 \ 6 \ 2]
\]

represented by the histogram in Figure 2. Figure 1 represents the marginal death probabilities of the sample and underlines the clusters created by K-means approach. These groups are assumed homogeneous with marginals equal to the centroids of the same group, which represents the vector \( \mathbf{u} \) in the Corollary 5:

\[
\text{means} = [0.003427 \ 0.001085 \ 0.001845 \ 0.00607855 \ 0.0171348]
\]
Figure 1: Marginal Death probabilities of the sample with evidence of K-means clusters.
The dependence parameters within and between the groups are assumed equal to the EQDP in the sense specified before:

\[ \alpha_{\text{within}} = [0.1, 2, 3, 0.4, 0.8] \quad \alpha_{\text{between}} = 0.01 \]

In this application, we find a Survival copula function equal to 0.9121 as a sum of 80 terms with sign represented in Figure 4.

We can fix the number of groups and the dependence parameters determined before and calculate the Survival copula function for a sample where we take off an element with marginal equal to the last, following the order depicted in Figure 1. Working backwards till a cardinality of the sample equal to 40, we can observe in Figure 4 the behavior of the Survival copula.

---

5We assume to know the estimations of the real parameters within and between the groups which may be estimated by an econometric methodology, like as the maximum likelihood method, possibly simplifying the problem by considering the limiting distributions proposed in Schönbucher (2004) for Archimedean copulas with granularity adjustment, and then compensated to account for the homogeneous approximation as suggested in Definition 1.

6We must recall the representation of the Survival copula function proposed in Corollary 7. The terms represented in Figure 3 are these inside the first sum in the Corollary.
Figure 3: The components with sign of the HH Survival copula function for a sample of dimension 80.
Figure 4: The HH Survival copula of a sample clustered in 5 groups for an increasing dimension of the sample.
Table 1: K-means HH Survival copula vs CR Survival for several number of groups. Clayton copula with $\alpha_{\text{within}} = 0.1$ and $\alpha_{\text{between}} = 0.01$ computed on a heterogeneous sample of dimension 20.

<table>
<thead>
<tr>
<th>Number of groups</th>
<th>K-means HH Survival</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0.9588</td>
<td>0.0039</td>
</tr>
<tr>
<td>4</td>
<td>0.9565</td>
<td>0.0016</td>
</tr>
<tr>
<td>5</td>
<td>0.9560</td>
<td>0.0011</td>
</tr>
<tr>
<td>10</td>
<td>0.9553</td>
<td>0.0004</td>
</tr>
</tbody>
</table>

function with respect to the size of the sample. As expected, the Survival copula function decreases for larger samples.

In order to validate our procedure, we make a comparison with the algorithm CR proposed in Cherubini and Romagnoli (2009) that we consider as our Benchmark, i.e. the truth. We consider the maximum dimension which allow to calculate the truth, i.e. we consider a sample of 20 heterogeneous elements (which correspond to the first 20 elements of our sample), and compare it to the hierarchical homogeneous survival (HH Survival) calculated with our procedure for several number of groups. The clusters are made with K-means and with Kohonen self-organizing map (SOM) and the distance is computed on the Euclidean metric. In table 1 we can observe the performance of our K-means procedure which clearly increases with the number of the groups.

In table 2 and 3 we investigate the performance of our procedure in a randomized data set. We start with the same data set used in table 1 randomized by a white Gaussian noise and by a colored Gaussian noise, i.e. a white Gaussian noise affecting only few marginals. The clusters are made by K-means based on the Euclidean distance. We compare the CR Survival with the K-means HH Survival, computed on a randomized data by a Monte Carlo procedure. In a random content we compute the K-means HH Survival for a maximum number of groups equal to 5 since K-means can’t preserve a greater number of clusters during the iterations, reducing it when someone become empty.

We can observe that for an higher number of groups, the error decreases for the white randomization but increases for the colored one. In line with Francois et all. (2005), we can assert that our approach, and therefore the
Table 2: K-means HH Survival copula vs CR Survival for several number of groups. Clayton copula with $\alpha_{\text{within}} = 0.1$ and $\alpha_{\text{between}} = 0.01$ computed on a white randomized heterogeneous sample of dimension 20.

<table>
<thead>
<tr>
<th>Number of groups</th>
<th>K-means HH Survival/white random</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0.9587</td>
<td>0.0038</td>
</tr>
<tr>
<td>4</td>
<td>0.9566</td>
<td>0.0017</td>
</tr>
<tr>
<td>5</td>
<td>0.9563</td>
<td>0.0014</td>
</tr>
</tbody>
</table>

Table 3: K-means HH Survival copula vs CR Survival for several number of groups. Clayton copula with $\alpha_{\text{within}} = 0.1$ and $\alpha_{\text{between}} = 0.01$ computed on a colored randomized heterogeneous sample of dimension 20.

<table>
<thead>
<tr>
<th>Number of groups</th>
<th>K-means HH Survival/colored random</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0.9274</td>
<td>0.0275</td>
</tr>
<tr>
<td>4</td>
<td>0.9237</td>
<td>0.0312</td>
</tr>
<tr>
<td>5</td>
<td>0.9229</td>
<td>0.0320</td>
</tr>
</tbody>
</table>

K-means approach based on the Euclidean metric, can be a good choice in the presence of white Gaussian noise but the performance is not so good if we assume a colored randomization of data.

In table 4 we compare the CR Survival with the Kohonen SOM HH Survival computed on the 20-th dimensional data set, allowing also for white and colored randomization. Kohonen SOM groups the data on a map with six neurons and, yet after 200 iterations, the grid tends to approximate the data distribution. In the random data experiments, we compute SOM HH Survival by a Monte Carlo procedure.

If we compare table 4 with the previous tables relative to K-means applications, we observe that SOM with a small number of nodes behave in a way that is similar to K-means while, as it’s pointed in Kohonen and Hondela

<table>
<thead>
<tr>
<th>Scenario</th>
<th>SOM HH Survival</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deterministic</td>
<td>0.9558</td>
<td>0.0009</td>
</tr>
<tr>
<td>White random</td>
<td>0.9559</td>
<td>0.001</td>
</tr>
<tr>
<td>Colored random</td>
<td>0.9226</td>
<td>0.035</td>
</tr>
</tbody>
</table>

Table 4: SOM HH Survival copula vs CR Survival in several scenarios. Clayton copula with $\alpha_{\text{within}} = 0.1$ and $\alpha_{\text{between}} = 0.01$. 
Table 5: CPU computation time in seconds. K-means HH Probability Distribution of the left queue (till 5 percent) of a variable of dimension $n = 100$

<table>
<thead>
<tr>
<th>N. of groups</th>
<th>numerosity of the groups</th>
<th>Cpu time</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>20</td>
<td>0.5</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>8</td>
</tr>
<tr>
<td>20</td>
<td>5</td>
<td>224</td>
</tr>
</tbody>
</table>

Table 6: CPU computation time in seconds. K-means HH Probability Distribution of the left queue (till 10 percent) of a variable of dimension $n = 60$

<table>
<thead>
<tr>
<th>N. of groups</th>
<th>numerosity of the groups</th>
<th>Cpu time</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>12</td>
<td>0.8</td>
</tr>
<tr>
<td>10</td>
<td>6</td>
<td>19.17</td>
</tr>
<tr>
<td>15</td>
<td>4</td>
<td>166</td>
</tr>
</tbody>
</table>

(2007), larger self-organizing maps rearrange data in a way that is fundamentally topological in character.

An interesting question concerns the scalability of the algorithm, i.e. the maximum dimension to which our techniques can be extended. In K-means HH algorithm the scalability must be analyzed with respect to the number of groups and the total dimension of the problem. The MatLab\textsuperscript{TM} algorithm was launched on a Dell Dimension DXP061 workstation, Intel(R) 2CPU, 6400@213 GHz, 213 GHz, 2.00 GB RAM. Table 5 reports the computing time for the first 5 percent of the probability distribution of a variable of dimension $n = 100$ for different numbers of groups. Table 6 reports the computing time for the first 10 percent of the probability distribution of a variable of dimension $n = 60$ for different numbers of groups. We can observe that the number of groups is the critical variable; the CPU time increases exponentially with the number of group till 15, for the next divisor, i.e. 20, problems of memory stack become unmanageable.
5 Conclusion

We propose a procedure to compute the volume of a multivariate copula function in high dimensional problems. Our approach is articulated in two steps:

- we cluster, by a semi-supervised method, the variables in a fixed number of homogeneous groups by compensating with the into-sectors and between-sectors dependence parameters to take account for the homogeneous assumption;
- we compute a Hierarchical copula function describing the joint distribution of random variables and its volume.

The empirical example presented above explains the strong potentiality of this procedure, as can be observed in the scalability analysis reported, and points also to the feasible aspect of the algorithm strongly dependent on the memory stack resources of MatLab™ program.

A Appendix

A.1 The combinatorial problem

To compute the volume of copula function, we must solve an interesting combinatorial problem. In fact we are concerned with the computing of the c.c.ds of an integer into the groups already determined by the clustering procedure. We must consider the c.c.ds for all the integers from zero to the dimension of the problem.

```matlab
plist = zeros(0,n);
for i = 0:m
    temp = partitions(total_sum - i*c, ... candidate_set(1:end-1)), ... max_count(1:end-1), fixed_count-i);
```
A.2 Testing compatibility with the groups

Moving beyond the purely combinatorial framework, we now select the c.ds consistent with the groups’ cardinalities generated by K-means.

```matlab
% This function tests the compatibility of the c.c.ds in MPerm
% with the groups' cardinalities.

function [SizeThreshold] = SizeThreshold(MPerm, p)

% If the number of ones in the group is bigger than its cardinality, the row is set to zero and then erased.
nc = size(MPerm, 2);
for j = 1:size(MPerm, 1)
    for i = 1:nc
        if (MPerm(j, i) > p(1, i))
            MPerm(j, :) = 0;
        else
            MPerm(j, i) = MPerm(j, i);
        end
    end
end
SizeThreshold = MPerm;
```

A.3 Preparing the Copula computation

Giving the c.c.ds’ matrix, we compute the copula function into the groups and so we create a matrix of dimension equal to the number of groups $\times$ the maximum dimension of the groups (this is the matrix $p_{i,k}(j)$ in Proposition 5) where we distribute $i$ ones into $k$ groups according to the $j$-th c.c.d.. For example if the second row of the c.c.ds’ matrix for the integer 4 into 3 groups with the same dimension equal to 3, is the following:

$$
\begin{bmatrix}
0 & 1 & 3
\end{bmatrix}
$$
and we order following the row’s order, the corresponding 0-1 matrix is:

\[
p_{4,3}(2) = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}
\]

Then we must substitute the elements of the matrix \(p_{4,3}(2)\) with the coordinates requested to compute our object, following the methods suggested by Proposition 5 and the following Corollaries 7 and 9. If we want to compute the copula’s volume, we must substitute the element of \(p_{4,3}(2)\) with the corresponding element of the vectors defining the box. For example if the box is \(S = [col(0.3 \times I), col(0.5 \times I)]\) where \(I\) is the identity matrix in \(\mathbb{R}^3\), the matrix \(p_{4,3}(2)\) becomes:

\[
c(p_{4,3}(2)) = \begin{bmatrix} 0.5 & 0.5 & 0.5 \\ 0.3 & 0.5 & 0.5 \\ 0.3 & 0.3 & 0.3 \end{bmatrix}
\]

% This function returns a matrix A (it's c(p_i,k(j))) from the
% 0-1 matrix np according to the following rule: if we hit on a zero, we substitute the corresponding element of vector v,
% whereas if we hit on a one we substitute the corresponding element of vector u.
% np is a 0-1 matrix (each row is a c.c.d.)
% sizecluster is the vector of the groups's cardinalities
% u and v are the vectors of volume's coordinates
function [Amat]=mmv(np,u,v,sizecluster)
    n = size(v,2);
    for i=1:size(np,2)
        A(i,:)= [ones(1,np(1,i)) ones(1,max(sizecluster)-np(1,i))*0];
    end
    for i=1:size(A,1)
        for j=1:size(A,2)
            if (A(i,j)==0)
                A(i,j)= v(1,i);
            else
                A(i,j)= u(1,i);
            end
        end
    end
    Amat=A;

Following this calling order, the c.c.d. considered here is the second one, i.e. \(j = 2\).
A.4 Hierarchical Copula computation

We compute the copula function into the groups; we aggregate the marginal probabilities with an Archimedean copula (in the following MatLab™ code we compute the Clayton’s copula function with parameter equal to the EQDP of the corresponding group). This must be done for each row of the c.c.d. matrix.

```matlab
% This function computes the Clayton's hierarchical second level copula, and the product of binomial coefficients allows to take account for the position into the groups.
% alpha is the vector of the parameters inside the groups
% alpha_0 is the parameters among groups
% numones is the c.c.d.'s matrix for the same number of
% deaths into the groups
% sizecluster is the vector of the groups' cardinalities
% u and v are the vectors of volume's coordinates
% Given a row corresponding to a c.c.d., this function creates a matrix with a number of rows equal to the number of groups and a number of columns equal to the max cardinality of groups (it's p_ik(j))
% The function G_Clayton computes the copula's generators for each element of matrix Amat and for the parameter of the corresponding group.
% Here we sum over the rows of matrix Cmat till the cardinality of the corresponding group.

function [HCM, pfactor] = claytonLevel2(numones, u, v, sizecluster, alpha, alpha_0)

% numclusters=size(numones,2);

Amat=mmv(numones, u, v, sizecluster);
for j=1:numclusters
    for i=1:sizecluster(1,j)
        Cmat(j,i)=G_Clayton(Amat(j,i), alpha(1,j));
    end
end
Cmat=[Cmat(:,:)];

% Here we sum over the rows of matrix Cmat till the cardinality of the corresponding group.
```

Finally we compute the copula function at the first level, i.e. we aggregate the second level copula functions with an Archimedean copula with parameter equal to the corresponding EQDP, accounting for the position of the element into the groups.

Acknowledgements: The authors would like to thanks our colleagues and friends Umberto Cherubini and Sabrina Mulinacci for useful comments and support and two anonymous referees which allows to improve this work.

References


