

# Rapport de pré-étude Value at Risk et Modèle de GARCH

Benjamin BOUGUET      Damien CARDUNER  
Paul CHAIGNON      Eric CHAUTY      Xavier FRABOULET  
Clément GAUTRAIS      Ulysse GOARANT

24 octobre 2013

Hamdi RAISSI - Quentin GIAI GIANETTO  
4ème Année - Département Informatique  
INSA de Rennes

# Table des matières

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>Contexte mathématique</b>	<b>4</b>
2.1	Introduction . . . . .	4
2.2	Définitions . . . . .	4
2.3	Particularités statistiques des séries financières . . . . .	5
2.4	Modèle GARCH . . . . .	6
2.4.1	Le modèle GARCH . . . . .	6
2.4.2	Conditions d'application du modèle GARCH . . . . .	6
2.4.3	Lien avec la Value at Risk . . . . .	7
2.5	Calcul de la Value at Risk . . . . .	8
2.5.1	Par la méthode historique . . . . .	8
2.5.2	Par la méthode RiskMetrics . . . . .	8
2.5.3	Par la méthode bootstrap . . . . .	9
<b>3</b>	<b>Etat de l'art</b>	<b>10</b>
3.1	Add-ins Excel . . . . .	11
3.2	Logiciels indépendants . . . . .	11
<b>4</b>	<b>Description du logiciel</b>	<b>14</b>
4.1	Fonctionnalités . . . . .	14
4.1.1	Importation des données . . . . .	14
4.1.2	Exportation des données . . . . .	15
4.1.3	Gestion des portefeuilles . . . . .	15
4.1.4	Application d'opérations sur les données . . . . .	16
4.1.5	Algorithmes et interprétations . . . . .	16
4.1.6	Affichage des données et des résultats . . . . .	17
4.1.7	Vérification en backtesting . . . . .	18
4.2	Interface graphique . . . . .	19

<b>5</b>	<b>Organisation du projet</b>	<b>22</b>
5.1	Outils employés . . . . .	22
5.1.1	Système d'exploitation visé . . . . .	22
5.1.2	Interfaçage avec Excel . . . . .	22
5.1.3	Choix du langage . . . . .	22
5.1.4	Interfaçage avec R . . . . .	23
5.1.5	Git, la gestion de versions . . . . .	23
5.2	Planning et répartitions . . . . .	24
5.2.1	Planning du projet . . . . .	24
5.2.2	Répartition du travail . . . . .	24
5.3	Licence . . . . .	24
5.3.1	Les licences libres . . . . .	25
5.3.2	Tableau recapitulatif . . . . .	26
5.3.3	Le choix de la licence . . . . .	26
	<b>Annexes</b>	<b>28</b>
<b>A</b>	<b>En-tête de la Licence LGPL</b>	<b>29</b>

# Chapitre 1

## Introduction

Malgré l'augmentation de la volatilité des marchés financiers, le développement de produits dérivés et surtout une série de faillites et de krachs boursiers, il faudra attendre 1996 et les accords de Bâle I, puis Bâle II en 2001, pour que des dispositions bancaires internationales obligent les banques à se prémunir des pertes imprévues.

De plus, bien qu'aucun outil de mesure de risque n'ait été imposé par les accords, les institutions financières se mirent à développer leurs outils. En effet, des pénalités peuvent être appliquées aux banques ayant une mauvaise gestion des risques.

Bien que les accords de Bâle concernent plus les risques systémiques du secteur bancaire que la finance de marché à proprement dit, ce premier accord de Bâle sera le point de départ de la réalisation et l'utilisation massive d'un nouvel indicateur, considéré très rapidement par les institutions financières comme un standard dans l'évaluation des risques financiers, la Value-At-Risk (VaR).

La VaR dans les accords de Bâle est plutôt utilisée pour déterminer une limite de fonds propres que les banques doivent pouvoir conserver afin de ne pas faire faillite. En finance de marché, la VaR est utilisée pour mesurer un risque de marché, se couvrir contre ce risque et, in fine, optimiser le rendement d'un portefeuille de titres.

Démocratisée par la banque JP Morgan en 1993 via la méthode RiskMetrics, la notion de VaR a été utilisée pour la première fois dans les années 1980 par la banque Bankers Trust, sur les marchés financiers américains. Jusqu'alors il était coutume d'utiliser la variance ou le drawdown, qui sont d'autres mesures de risque de marché.

L'objectif de ce projet est de développer une application permettant d'utiliser des outils statistiques pour le calcul de risques en finance. Destiné aux acteurs de la finance, ce logiciel devra présenter une interface simple à uti-

liser. L'utilisateur pourra importer des données brutes telles que les cours des actions du CAC 40, gérer différents portefeuilles et comparer les résultats des différentes méthodes statistiques, ainsi que calculer en particulier la VaR. Pour permettre d'évaluer quels sont les modèles statistiques les plus adaptés aux données d'entrée, un ensemble de tests et de contrôles seront possibles sur les résultats obtenus selon les différents algorithmes de calcul.

Le développement de ce logiciel tiendra compte de l'existant. Plusieurs logiciels permettent déjà de calculer la VaR et d'utiliser différentes méthodes statistiques pour calculer des risques financiers, que ce soient des modules à intégrer à Excel ou des logiciels spécialisés. L'avantage que présentera notre programme se situe principalement dans les fonctionnalités de backtesting et de tests de comparaison sur les différentes méthodes statistiques. De plus, l'interface sera plus ergonomique, plus simple du fait du nombre de fonctionnalités (inférieur à celui d'Excel par exemple). Le logiciel produit sera libre et implémentera des fonctionnalités inédites ou normalement payantes.

Ce rapport de pré-étude vient donner les conclusions de la phase d'analyse du projet. Dans un premier temps, nous nous attacherons à exposer le contexte mathématique dans lequel évolue le projet. Cela inclut les séries temporelles, leurs fondements mathématiques ainsi que les différentes méthodes qui seront utilisées dans le projet.

La seconde partie du rapport tiendra lieu de cahier des charges du logiciel. Nous y détaillerons notamment les différentes fonctionnalités souhaitées. Nous évoquerons la manière dont nous voyons leur intégration dans le logiciel.

Dans la dernière partie du rapport, nous présenterons les différents outils et technologies utilisés, puis nous détaillerons le planning prévisionnel ainsi que la répartition des tâches envisagés. Pour finir, nous nous interrogerons sur la licence sous laquelle nous placerons notre logiciel, ce qui déterminera notamment les conditions de sa réutilisation pour de futurs développements.

# Chapitre 2

## Contexte mathématique

### 2.1 Introduction

L'introduction de l'obligation pour les banques de se prémunir contre d'éventuelles pertes financières a naturellement entraîné l'utilisation de différents concepts mathématiques. En effet, pour les banques, l'enjeu est d'immobiliser le minimum de ressources financières, tout en garantissant un niveau de risque satisfaisant. Il est donc naturel que des notions de statistiques et de probabilités soient introduites afin quantifier le risque. Ces notions sont présentées par la suite.

### 2.2 Définitions

Pour pouvoir quantifier le risque, il faut tout d'abord modéliser les variations des valeurs boursières. Pour cela, on utilise des séries temporelles. Une série temporelle est une suite de valeurs numériques représentant l'évolution d'une quantité au cours du temps. La valeur de  $x$  à l'instant  $t$  sera ainsi notée  $x_t$ . L'introduction de la notion de temps doit être accompagnée de la définition d'une unité de temps. Dans notre cas, l'unité de temps est le jour. Si nous sommes à l'instant  $t$ ,  $t - 1$  correspond à la veille et  $t + 1$  au lendemain.

Pour estimer les risques financiers, différentes techniques existent. L'une des plus utilisées actuellement est la Value-at-Risk (VaR). Elle permet de mesurer les risques associés à un portefeuille. Plus précisément, elle permet d'estimer la perte maximale associée à un portefeuille pour un niveau de confiance et un horizon de temps donnés. Elle peut être calculée selon plusieurs méthodes. Nous en présentons trois par la suite.

On définit  $V_t$  comme le prix de vente du portefeuille à l'instant  $t$ . Le log-rendement correspond au logarithme du rapport de deux prix de vente suc-

cessifs :  $r_t = \log \left( \frac{V_t}{V_{t-1}} \right)$

## 2.3 Particularités statistiques des séries financières

La modélisation des séries financières est particulièrement complexe car celles-ci ont des comportements atypiques.

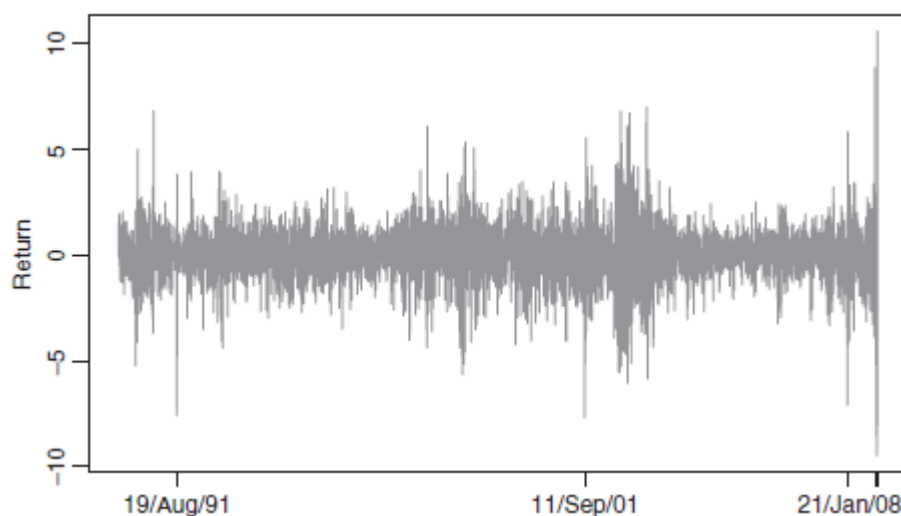


FIGURE 2.1 – Log rendement du CAC 40 entre 1990 et 2008

Tout d'abord, les valeurs extrêmes ont tendance à se regrouper ; on parle alors de *Volatility clustering*. En effet, lorsque les cours sont instables, la probabilité qu'ils le soient le lendemain est assez élevée ; de même lorsqu'ils sont stables, ils ont plus tendance à le rester. De plus, les queues de distributions des rendements sont épaisses. Cela signifie que les valeurs extrêmes ont une probabilité non négligeable d'apparaître. Cela est donc plus compliqué à modéliser : les rendements ne suivent pas une loi normale par exemple. Un autre effet qui peut compliquer la modélisation est l'effet de levier. Cet effet correspond au fait qu'une même variation des cours n'aura pas la même conséquence, suivant qu'elle soit positive ou négative. En effet, une baisse des cours entrainera une instabilité plus importante qu'une variation positive de même amplitude. L'influence de la variation des rendements n'est donc pas symétrique. Enfin, une certaine saisonnalité des rendements est parfois observée. En effet, suivant la période de l'année ou suivant le jour, ils seront plus ou moins volatiles. Il n'existe pas à ce jour de modèle permettant de

modéliser l'ensemble de ces particularités. Mais comme celles-ci n'ont pas toutes la même importance, des modèles plus simples permettent de modéliser correctement l'évolution des rendements financiers.

Il faut également noter qu'il existe d'autres particularités (corrélation entre les différentes classes d'actifs notamment) mais elles ne sont pas expliquées ici car elles ne seront pas prises en compte dans notre logiciel.

## 2.4 Modèle GARCH

### 2.4.1 Le modèle GARCH

Afin de modéliser l'évolution des rendements, différents modèles ont été introduits. Un des modèles les plus utilisés est le modèle GARCH. Il est défini par deux équations :

$$\begin{aligned} r_t &= \sigma_t \eta_t, \eta_t \text{ iid}(0, 1) \\ \sigma_t^2 &= \omega + \alpha * r_{t-1}^2 + \beta * \sigma_{t-1}^2 \end{aligned}$$

avec  $\omega, \alpha, \beta > 0$

La première équation modélise le fait que le rendement s'exprime comme le produit d'une variance  $\sigma_t$  (calculée à partir des observations précédentes) avec une variable aléatoire  $\eta_t$ . Cette dernière quantité permet de modéliser les événements imprévisibles.

La seconde équation porte sur le calcul de la variance. Ce calcul est effectué par rapport aux rendements et aux variances précédents. Les  $\alpha$  et  $\beta$  sont présents afin de moduler l'influence des informations (rendements et variances).

Le modèle GARCH présenté ici est assez simple et il en existe d'autres prenant en compte des données plus anciennes que celles obtenues au temps  $t - 1$ . Il permet néanmoins de bien modéliser le *Volatility clustering*. En revanche, il ne tient pas compte de l'asymétrie de la variation des rendements. Il existe donc d'autres modèles, basés sur le GARCH, qui permettent de mieux modéliser certaines particularités de séries financières.

### 2.4.2 Conditions d'application du modèle GARCH

Comme tout modèle, GARCH doit être appliqué sous certaines conditions. Tout d'abord, il est nécessaire de vérifier qu'il n'existe pas de corrélation entre les rendements à des temps différents. Pour cela on calcule, à partir des données collectées, la covariance empirique et on teste sa nullité.



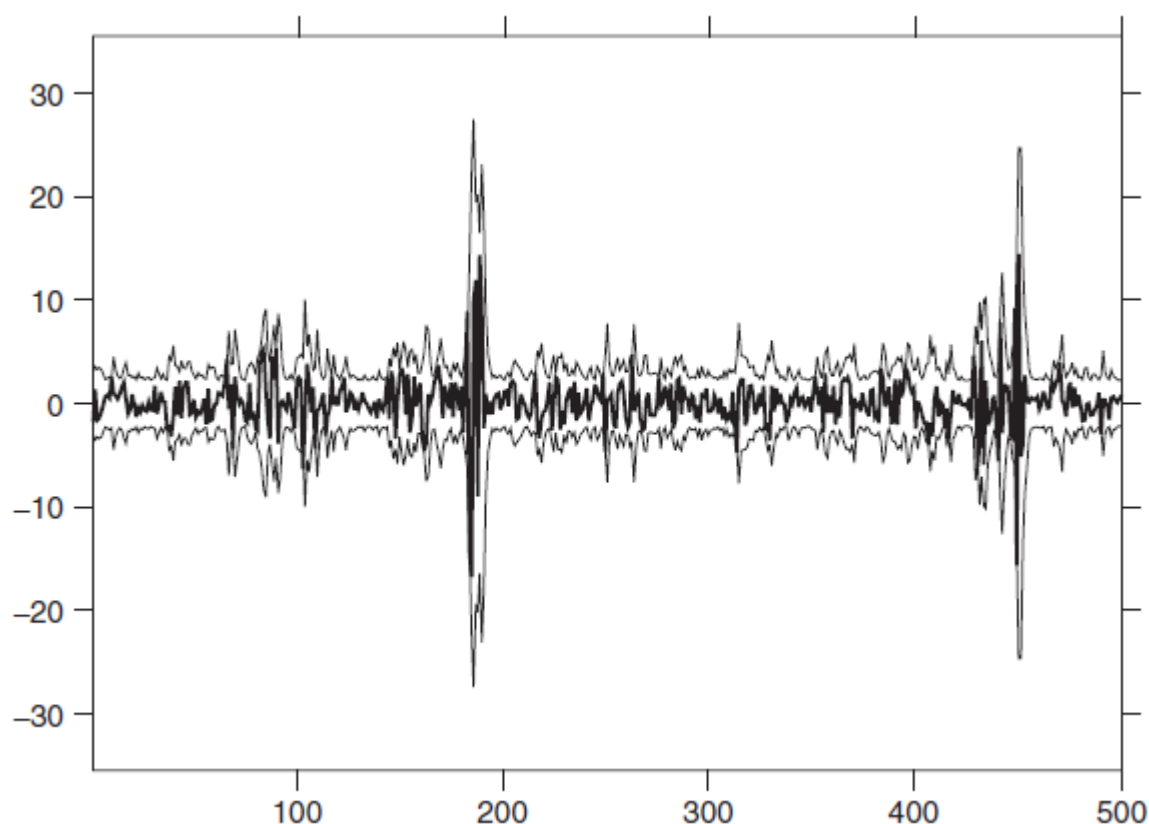


FIGURE 2.2 – Exemple de prévision par un modèle GARCH

Une autre condition est que la covariance calculée à partir des rendements au carré soit supérieure à zéro. Pour vérifier cela, on calcule, à partir des données collectées, la covariance empirique des rendements au carré et on vérifie sa positivité.

Toutes ces différentes conditions sont basées sur des résultats théoriques et elles peuvent également être vérifiées en traçant différents graphiques.

Il existe encore d'autres conditions d'applications (notamment sur les valeurs des  $\alpha$  et  $\beta$ ), mais celles-ci ne seront pas abordées ici car elles sont plus complexes.

### 2.4.3 Lien avec la Value at Risk

La modélisation GARCH nous permet ainsi de modéliser l'évolution des rendements. Il nous faut maintenant lier ces rendements à la VaR. Par défi-

nition, on a :

$$\alpha = P(VaR_{t,h}(\alpha) < (V_t - V_{t+h})) \text{ si } V_{t+h} < V_t. \text{ Sinon, } VaR_{t+h} = 0$$

Cela exprime bien le fait que la probabilité que la valeur de la perte ( $V_t - V_{t+h}$ ) dépasse la VaR vaut  $\alpha$ .

Si on développe cette expression on obtient :

$$\alpha = P(VaR_{t,h}(\alpha) < -(V_{t+h} - V_t)) \quad (2.1)$$

$$= P\left(\frac{-VaR_{t,h}(\alpha)}{V_t} > \frac{(V_{t+h} - V_t)}{V_t}\right) \quad (2.2)$$

$$= P\left(1 - \frac{VaR_{t,h}(\alpha)}{V_t} > \frac{V_{t+h}}{V_{t+h-1}} * \frac{V_{t+h-1}}{V_{t+h-2}} * \dots * \frac{V_{t+1}}{V_t}\right) \quad (2.3)$$

$$= P\left(\log\left(1 - \frac{VaR_{t,h}(\alpha)}{V_t}\right) > r_{t+h} + r_{t+h-1} + r_{t+h-2} + \dots + r_{t+1}\right) \quad (2.4)$$

On arrive donc à montrer que  $VaR_{t+h}(\alpha) = 1 - \exp(q_t(h, \alpha)) * V_t$ , avec  $q_t(h, \alpha)$  le quantile d'ordre  $\alpha$  de  $r_{t+h} + r_{t+h-1} + r_{t+h-2} + \dots + r_{t+1}$ . On voit donc ici que la VaR est calculée à partir de rendements futurs et donc encore non observés. Il est donc nécessaire d'estimer ces rendements à l'aide d'un modèle GARCH bien ajusté aux données observées.

## 2.5 Calcul de la Value at Risk

### 2.5.1 Par la méthode historique

Historiquement, la VaR se calcule à partir de l'histogramme des rendements. En effet, la méthode originale consiste à prendre la valeur limite correspondant à un certain pourcentage des rendements les plus faibles. Le risque  $\alpha$  évoqué précédemment est alors ce pourcentage. Cela revient donc à prendre le quantile d'ordre  $\alpha$  de la distribution de la valeur du portefeuille.

### 2.5.2 Par la méthode RiskMetrics

L'une des méthodes les plus utilisées pour estimer la VaR est la méthode RiskMetrics. Elle correspond à un modèle GARCH particulier. En effet, avec RiskMetrics, la valeur de la variance est calculée uniquement par rapport aux données obtenues au temps  $t - 1$ , dans notre cas la veille. De plus, la somme des coefficients  $\alpha$  et  $\beta$  est égale à 1. Enfin, le terme constant de variance ( $\omega$ ) est nul. On a donc le modèle suivant :

$$r_t = \sigma_t \eta_t, \eta_t \text{ iid}(0, 1)$$

$$\sigma_t^2 = (1 - \lambda) * r_{t-1}^2 + \lambda * \sigma_{t-1}^2$$

L'avantage de cette méthode simplifiée est que le calcul de la VaR est direct. Ce modèle, bien qu'il ne soit pas le plus fiable, est néanmoins très utilisé, grâce à sa simplicité.

Contrairement au modèle GARCH, la valeur de l'unique coefficient  $\lambda$  n'est pas calculée avec les données précédentes. En effet, il est souvent fixé à une valeur, obtenue par expérience, de 0.94.

### 2.5.3 Par la méthode bootstrap

En dehors des deux méthodes précédentes, le calcul de la VaR ne peut se faire de façon directe. En effet, comme précisé plus haut, le calcul de la VaR dépend d'un quantile d'une loi inconnue<sup>1</sup>. Comme ce quantile ne peut être déterminé de façon théorique, il doit être estimé. Cette estimation est réalisée par la méthode *bootstrap*. Cette méthode consiste à générer un grand nombre scénarios et à calculer la somme des rendements pour chacun d'entre eux. Ainsi, on obtient un nombre assez important de sommes de rendements. Comme ces sommes suivent la loi inconnue que l'on cherche pour calculer la VaR, on peut les utiliser pour obtenir le quantile qui nous intéresse (soit directement, soit en cherchant à estimer la fonction de densité).

Cependant, pour que la loi obtenue par les scénarios soit pertinente, ces scénarios sont générés à partir de données déjà observées. De ce fait, la première étape du *bootstrap* consiste à calculer les résidus  $\eta_t$  sur des données observées. Ensuite, on réalise un tirage avec remise entre les différents résidus pour générer les scénarios. Lorsque l'on a tiré aléatoirement un résidu, on peut calculer le rendement associé ainsi que la volatilité à l'étape suivante. En répétant ces étapes, on obtient donc différents scénarios.

---

1. La loi de la somme des rendements n'est pas connue à priori

# Chapitre 3

## Etat de l'art

Nous présentons dans cette partie les différents logiciels disponibles qui intègrent un calcul de la VaR. Il ne s'agit bien sûr pas d'une liste exhaustive, mais nous nous efforçons de présenter les principaux types de solutions existantes.

De nombreux logiciels d'évaluation de risques financiers sont disponibles sur le marché. Si certains sont principalement centrés sur le calcul de la Value-at-Risk, la majorité intègre son calcul parmi un ensemble bien plus vaste de fonctionnalités. Aussi, il est parfois difficile de se faire une idée d'un logiciel donné car, pour pouvoir effectivement le tester, il est nécessaire d'être un client potentiel ce qui n'est bien évidemment pas le cas ici. De plus, les méthodes du calcul de la VaR employées par certains logiciels sont parfois mal explicitées. Il existe deux catégories principales de logiciels : des *add-ins Excel* et des logiciels indépendants. La première catégorie semble assez convoitée par les clients puisqu'elle représente la majorité des solutions proposées. Son intérêt réside dans la possibilité de pouvoir intégrer la solution proposée avec des outils ou des données existants. On peut aussi noter que la méthode de calcul de la VaR la plus communément répandue semble être celle de Monte Carlo. Cette méthode se différencie de celle de GARCH dans la mesure où les scénarios sont élaborés à partir d'une loi de probabilité et non de l'exploitation des données historiques par la méthode *bootstrap*.

Les logiciels plus généralistes tels que *Matlab*, *SAS* ou *R* permettent bien sûr la programmation du calcul de la Value-at-Risk selon les différentes méthodes<sup>1</sup>. Cependant, ces premiers sont exclus d'office de l'étude car ils ne répondent pas aux éléments du cahier des charges concernant la facilité d'utilisation et l'interface intuitive.

---

1. Des modules spécifiques au domaine de la finance sont même disponibles pour certains

## 3.1 Add-ins Excel

De nombreux éditeurs tels que *FEA*, *Hoadley*, *Palisade*, *PortofolioScience* ou encore *VoseSoftware* proposent des logiciels d'évaluation de risques financiers intégrés à *Excel*. Ces logiciels sont assez similaires et diffèrent principalement du point de vue du nombre de fonctionnalités, des méthodes de calcul de la VaR et de leur intégration dans *Excel*<sup>2</sup>. Dans tous les cas, nous nous sommes focalisés sur les fonctionnalités plutôt que sur l'interface étant donné que notre solution n'a pas pour objectif de s'intégrer à *Excel*.

Parmi les éditeurs cités, nous avons choisi de présenter la solution développée par *Portofolio Science*. Elle propose la composition de portefeuilles et un calcul direct de la VaR selon les méthodes suivantes :

- Basée sur la volatilité
- Delta-normal
- Historique
- Historique dégradée
- Monte Carlo

Elle possède beaucoup d'autres fonctionnalités telles que différentes décompositions de la VaR, une analyse avancée de la volatilité selon une méthode GARCH, l'élaboration de matrices de corrélation et covariance, ainsi que la possibilité d'effectuer des scénarios de stress selon plusieurs méthodes.

Bien que *Hoadley* et *PortofolioScience* intègrent des prévisions de la volatilité selon une méthode GARCH, aucune des solutions listées ne propose un calcul de la VaR selon une méthode GARCH.

## 3.2 Logiciels indépendants

L'autre catégorie principale de logiciels proposant le calcul de la Value-at-Risk se présentent sous la forme de logiciels indépendants par contraste avec la première catégorie présentée. Il s'agit généralement de suites logicielles de taille importante dont le calcul de la VaR n'est en fait qu'une fonctionnalité parmi beaucoup d'autres.

L'éditeur *Rho-Works* distribue gratuitement le logiciel *EC-VaR* pour un usage éducatif ou personnel. L'importation des données se fait à partir de fichier CSV ou XLS. Il permet la composition de portefeuilles d'actifs. Le logiciel implémente le calcul de la VaR selon les méthodes suivantes :

- VaR historique
- VaR conditionnelle
- BetaVaR

---

2. Ils exploitent plus ou moins bien le ruban ou ils utilisent des fenêtres supplémentaires

- Component VaR

Le logiciel offre un onglet par méthode de calcul. Chaque onglet présente une vue graphique relative à la méthode utilisée. On voit ici la distribution des rendements historiques avec en rouge les 5% plus faibles.

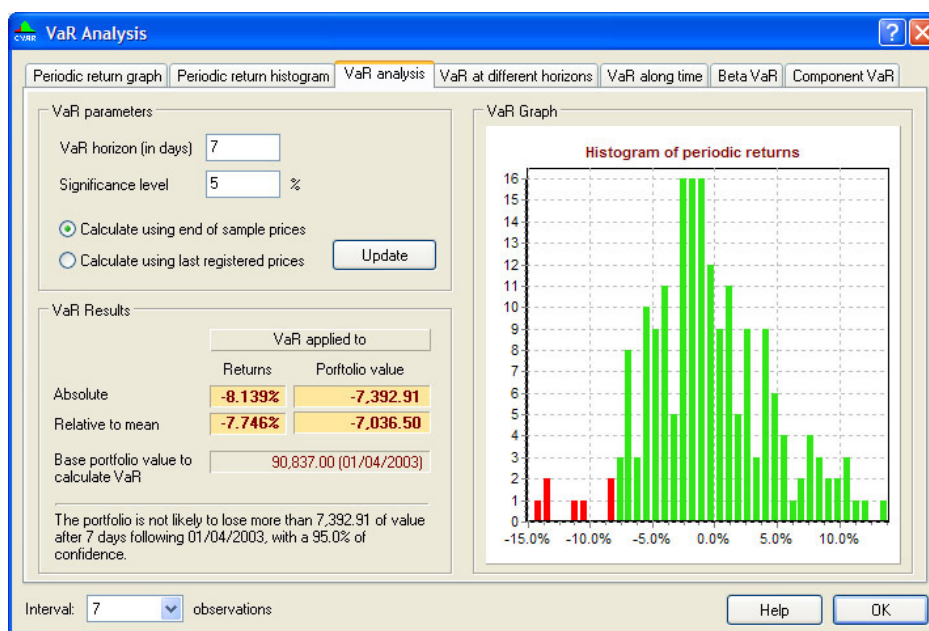


FIGURE 3.1 – Vue de *EC-VaR* relative à la VaR calculée selon la méthode historique

Bien qu'il soit probable que l'interface de notre logiciel soit différente, le logiciel ici présenté semble le plus proche de notre solution en terme de fonctionnalités, aux méthodes de calcul de la VaR près.

Certains éditeurs ont mis l'accent sur la performance des calculs. C'est le cas de *Quartetfs* qui publie le logiciel *ActivePivot* capable d'un calcul très rapide de la VaR pour un usage en temps réel. Son utilisation se fait à travers une interface web. Il est aussi possible de calculer une VaR marginale, d'analyser les queues de distribution et d'effectuer des simulations de scénarios de stress. Cependant, une telle solution ne correspond pas véritablement à notre besoin : on ne souhaite pas vouloir calculer la VaR pour un usage en temps réel car le modèle de GARCH est mieux adapté pour des prévisions quotidiennes. Dans la même catégorie, *Mors Software* propose l'add-on *Mors VaR* à ses logiciels existants *Mors liquidity management* et *MORS treasury and trading solutions* pour un calcul performant de la VaR (historique et Monte Carlo). Il possède une fonctionnalité de backtesting. Cependant, l'add-on est une fois de plus orienté vers la performance et ne correspond donc pas au

besoin de notre client.

Parmi les logiciels présentés, il n'en existe finalement aucun capable de calculer directement la VaR selon la méthode GARCH. Un seul est gratuit et aucun n'est libre. Il est probable que les acteurs de la finance intéressés par l'utilisation d'une telle méthode fassent développer des solutions sur mesure.

# Chapitre 4

## Description du logiciel

### 4.1 Fonctionnalités

#### 4.1.1 Importation des données

Le principal objectif du logiciel sera de réaliser des calculs de risques sur des portefeuilles d'actifs. Il devra donc permettre à l'utilisateur d'importer des données de manière transparente depuis différentes sources. L'importation pourra être réalisée grâce à un menu ou à un "drag and drop".

Le module d'importation transformera automatiquement les informations issues de ces sources en des données utilisables par le logiciel. Plusieurs types de sources existent : les sites web comme Yahoo Finance ou Bloomberg, ou encore des logiciels comme DataStream. Cette abondance de sources implique un problème : aucune de ces bases de données utilisent le même standard pour stocker l'information. Dans la suite, les formats de fichiers utilisables et les possibilités offertes en terme d'importation seront présentés.

#### **Les formats de fichiers**

Trois formats de fichiers sont très utilisés dans le domaine de la finance et ils devront par conséquent être pris en charge par l'application :

- le format texte brut ;
- le format CSV ;
- le format Excel : son importation nécessitera d'utiliser des API fournies par Microsoft.

Le format CSV sera le plus simple à importer puisqu'il est standardisé. En effet, les colonnes sont séparées par des virgules, des points-virgules, etc ; et, chaque n-uplet de valeurs occupe une ligne.



## Problèmes liés aux formats de données

Une fois les données lues, le module d'importation devra choisir les variables utiles aux calculs et enlever celles inutiles. Par exemple, les données proposées par Yahoo Finance donnent des informations sur les volumes d'actions échangés. Or, cette information n'est pas utile pour les calculs. C'est pourquoi ce module devra sélectionner si possible automatiquement les colonnes utiles, ou proposer à l'utilisateur de les sélectionner à la main.

La date peut aussi poser problème. En effet, elle peut être au format français (jj-mm-aaaa), anglais (mm-jj-aaaa) ou international (aaaa-mm-jj). Le logiciel devra donc prendre en charge ces différents formats de dates.

### 4.1.2 Exportation des données

Le logiciel offrira différentes options en terme d'exportation et de sauvegarde :

- exporter les données brutes obtenues lors de l'importation en différents formats : texte brut, CSV et Excel ;
- exporter les portefeuilles d'actifs constitués (cf. 4.1.3) dans ces mêmes formats ;
- fournir un document récapitulant les résultats liés à un calcul ;
- sauvegarder une session<sup>1</sup>.

### 4.1.3 Gestion des portefeuilles

La gestion de portefeuilles ou d'actifs, consiste à construire un ensemble de titres boursiers dont il faudra faire fructifier la valeur au cours du temps. Les actifs que nous aurons seront les données importées via l'outil d'importation, par exemple les données du CAC 40. Pour faire fructifier un portefeuille, le gestionnaire doit limiter les risques qui sont liés aux actions, tout en pariant sur une montée de leur cours. Pour ces raisons, notre logiciel permettra une utilisation directe des différents modèles sur les portefeuilles constitués.

#### Constitution des portefeuilles

Pour créer un portefeuille, il est nécessaire d'importer des données via l'outil d'importation. Ces données peuvent représenter diverses entreprises et/ou différentes places boursières.

Dès lors que les données seront importées, il suffira de sélectionner celles que l'utilisateur désire, puis de les insérer dans un portefeuille existant ou un nouveau.

---

1. L'ensemble des portefeuilles en cours d'utilisation, les calculs effectués...

Pour créer un nouveau portefeuille, nous avons pensé à plusieurs options :

- On peut imaginer la présence d’un bouton “Ajout” sur lequel il suffit de cliquer pour afficher une fenêtre permettant de personnaliser le portefeuille allant être créé : intitulé, date, etc.
- Un second bouton permettant d’insérer les données dans un portefeuille existant ; une fenêtre devra alors apparaître pour laisser le choix à l’utilisateur du portefeuille destinataire des données sélectionnées.
- Un clic droit sur les données sélectionnées pourra aussi permettre la création d’un nouveau portefeuille ainsi que l’insertion des données dans un portefeuille existant. Ceci donnera accès aux mêmes fenêtres de paramètres que les boutons d’action.
- Le déplacement de données de portefeuilles vers d’autres portefeuilles se fera par un simple glisser-déposer de la source jusqu’à la destination.

### **Affichage des portefeuilles**

La liste des portefeuilles créés sera en permanence présente sur l’écran de l’utilisateur, via une barre verticale à gauche de celui-ci. Regroupés si besoin en liste dépliant, il suffit de cliquer sur le portefeuille que l’on désire, pour que les données qu’il contient s’affichent à l’écran, accompagné des différents résultats des calculs effectués sur celles-ci.

#### **4.1.4 Application d’opérations sur les données**

Après importation des données, l’utilisateur doit pouvoir appliquer des opérations mathématiques sur les données. Il doit par exemple pouvoir effectuer des calculs de :

- moyenne ;
- variance ;
- kurtosis ;
- valeurs extrêmes.

Toutes ces valeurs seront calculées et affichées en même temps.

#### **4.1.5 Algorithmes et interprétations**

Le but de notre logiciel est de permettre de calculer la Value-at-Risk. Nous voulons donc simplifier l’obtention de cette information pour l’utilisateur. L’algorithme permettant d’établir la modélisation GARCH nous sera fourni en langage R. Les autres algorithmes découlent directement des formules mathématiques vues précédemment.

Nous allons proposer le calcul de la Value-at-Risk suivant trois méthodes. A chaque fois les calculs pourront être faits sur l’ensemble des données ou

seulement sur celles sélectionnées. Pour chaque méthode, un niveau de risque pourra être choisi.

La première méthode passe par une modélisation GARCH (cf. 2.4.1). Parce que cette dernière peut prendre du temps à calculer, nous offrirons la possibilité de la réaliser sans continuer directement avec un calcul de la Value-at-Risk. Dans ce cas, la modélisation sera conservée dans les résultats et la Value-at-Risk pourra être calculée depuis cette dernière par la suite. Avant le calcul de la modélisation, nous donnerons à l'utilisateur la possibilité de préciser le nombre d'éléments prélevés pour le *bootstrap* (cf. 2.5.3) et donc, sa précision.

Les deux autres méthodes utilisent respectivement le modèle Riskmetrics (cf. 2.5.2) et la méthode historique (cf. 2.5.1). A chaque fois, seul le niveau de risque sera demandé.

Deux tests seront proposés pour vérifier la validité de la modélisation GARCH : un test de corrélation des données simple et à l'ordre deux (cf. 2.4.2). Ces deux tests seront proposés avant le calcul de la modélisation mais pourront aussi être faits, via le menu, indépendamment de la modélisation.

Enfin, pour chaque option saisie par l'utilisateur, la valeur par défaut pourra être modifiée. L'idée est ici de permettre à l'utilisateur d'aller plus vite la fois suivante. Par exemple, si l'utilisateur ne veut jamais effectuer les tests de corrélation avant de calculer la modélisation GARCH, il pourra choisir de mettre les deux cases à cocher pour les deux tests par défaut à "décochées".

#### 4.1.6 Affichage des données et des résultats

Dans notre projet nous allons distinguer deux types d'affichage. En effet, il y a d'une part l'affichage des données, et d'autre part l'affichage des résultats.

L'affichage des données doit permettre de représenter au mieux les différents portefeuilles importés. Pour rappel, ces portefeuilles seront créés par l'intermédiaire de liste de données. L'idée la plus naturelle est donc d'afficher les portefeuilles à l'aide de tableaux quadrillés. Dans le but d'améliorer l'ergonomie générale du logiciel, plusieurs fonctionnalités permettant de manipuler les colonnes et les lignes du tableau seront implémentées. Il faudra par exemple pouvoir modifier directement les données contenues dans le tableau. Il devra également être possible de faire passer une colonne du quadrillage associé à une liste de données, dans un autre quadrillage de manière simple, en harmonisant les données. Pour faire cela, l'utilisation du "glisser-déposer" est une solution envisagée.

Un autre aspect important de l'affichage concerne la présentation des résultats. L'objectif est d'offrir à l'utilisateur le plus de souplesse possible. Sur

l'ensemble des graphiques on laissera donc à l'utilisateur la possibilité d'afficher plusieurs courbes s'il le souhaite, ou encore de sélectionner les couleurs s'il le souhaite.

Le logiciel proposera aussi la création d'un histogramme à partir des données d'une colonne.

Nous avons également souhaité proposer un affichage sous forme de textes des différents résultats calculés. Dans la partie affichage des résultats l'utilisateur aura donc accès à un rapport récapitulant l'ensemble des résultats calculés par les différents algorithmes.

Le client a souhaité que le logiciel puisse fournir une matrice de corrélation. Ce type de représentation des données est très utile car il permet de déterminer très rapidement la tendance globale du marché.

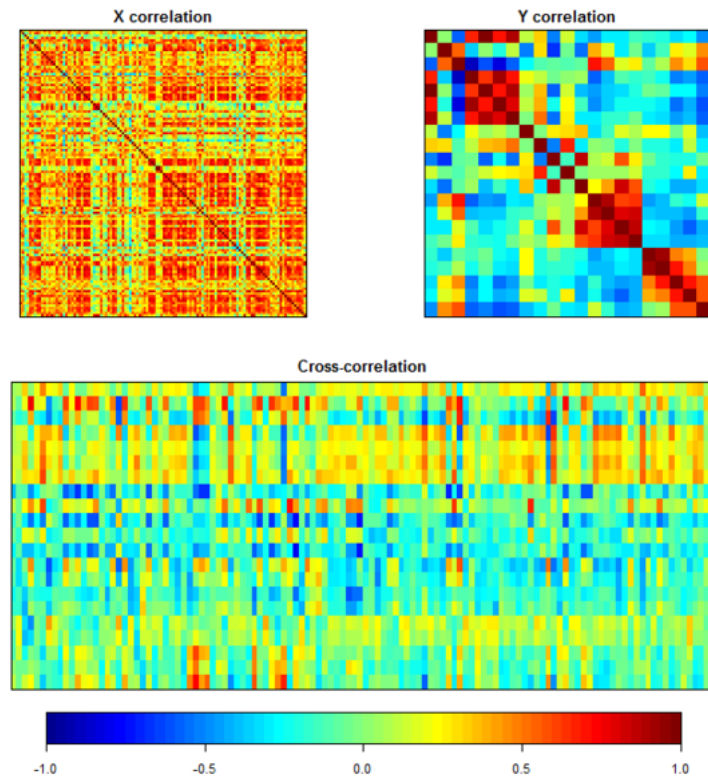


FIGURE 4.1 – Matrice de corrélation d'indices boursiers

#### 4.1.7 Vérification en backtesting

Le backtesting consiste à évaluer l'exactitude d'un modèle de calcul de la Value-at-Risk en se basant sur des données historiques. Il permet ainsi

la comparaison des différentes méthodes d'évaluation de la VaR pour un ensemble d'actifs et une période donnés.

Pour tester un modèle particulier, on détermine d'abord les paramètres du modèle sur une durée de test. On enregistre ensuite le nombre de fois où la perte a été supérieure à la prévision. Il est ainsi possible de déterminer quelle méthode de calcul de la VaR a fourni les meilleurs résultats pour un ensemble d'actifs et une durée donnés.

Par exemple, pour tester l'exactitude du modèle GARCH à l'horizon d'un jour, on commence par déterminer les paramètres du modèle sur une période donnée. On regarde ensuite si la perte effective de l'actif a été ou non supérieure à la VaR. On répète ensuite ce processus sur une durée déterminée par l'utilisateur. L'horizon de temps peut bien évidemment être changé, le principe reste le même. A la fin du backtesting, on obtient le nombre de jours où la perte a été effectivement supérieure à la VaR. Ce nombre peut être par exemple de 5 pour une durée totale de test de 100 jours. L'utilisateur peut ainsi estimer la fiabilité du modèle. Il peut alors choisir la méthode de calcul de la VaR la plus adaptée à ses données en analysant les résultats obtenus.

## 4.2 Interface graphique



FIGURE 4.2 – Interface graphique : onglet données

1. Barre d'outils pour un accès rapide aux fonctionnalités les plus utilisées.
2. Volet de selection des portefeuilles
3. Affichage des données dans une grille



FIGURE 4.3 – Interface graphique : onglet résultats

#### 4. Affichage des résultats

# Chapitre 5

## Organisation du projet

### 5.1 Outils employés

#### 5.1.1 Système d'exploitation visé

Nous avons établi rapidement que notre logiciel ne serait développé que pour Windows. En effet, ce système d'exploitation est très majoritairement utilisé dans les salles de marché.

#### 5.1.2 Interfaçage avec Excel

De nombreuses fonctionnalités de notre logiciel sont celles d'un simple logiciel tableur. Nous devons entre autres importer des données, les afficher sous formes de tableaux et de graphes ainsi que les exporter et être capable de travailler sur les tableaux facilement. Toutes ces fonctionnalités sont supportées par des logiciels tels qu'Excel ou OpenOffice Calc. Parce que les salles de marchés en sont largement équipées, nous avons choisis d'utiliser Excel. Notre logiciel implémentera donc toutes les fonctionnalités ci-dessus en s'appuyant sur le tableur de Microsoft.

#### 5.1.3 Choix du langage

Le choix du langage que nous utiliserons pour développer le logiciel est déterminé par notre besoin précédent. Comme nous avons choisi d'utiliser Excel, nous allons devoir utiliser la plateforme de développement de Microsoft, Visual Studio. L'intégration d'Excel peut se faire en C# et C++. Nous avons choisi de développer en C++ car nous y aurons été formés d'ici là.



### 5.1.4 Interfaçage avec R

Nous allons avoir besoin du logiciel R pour effectuer certains calculs, notamment pour effectuer la modélisation GARCH. Cette dernière peut être assez compliquée à calculer et R permet de le faire beaucoup plus simplement. L'autre solution serait de réécrire l'algorithme avec le langage que nous utiliserons.

Pour exécuter du code R depuis un autre langage plusieurs solutions s'offrent à nous. La première pourrait être de réaliser un exécutable à partir du script R. Le programme pourrait ensuite être appelé comme n'importe quel autre programme. Cependant comme R est un langage interprété, réaliser un exécutable qui soit indépendant du logiciel R à partir d'un script se révèle assez compliqué. La commande R CMD LINK permet cependant de le faire d'après la documentation R [2].

Une seconde solution serait d'appeler R en ligne de commande depuis notre programme C++ pour interpréter le script R. Cette solution nécessite cependant d'installer le logiciel R.

Comme R est un langage interprété, une troisième solution existe. Celle-ci consiste à exécuter le code R directement depuis notre code C++. Différents packages existent pour cela. Plusieurs d'entre eux - notamment StatConnector et RDCOMServer - utilisent le mécanisme (D)COM de Windows pour communiquer entre des applications (TODO Ajouter à la biblio : <http://cran.r-project.org/doc/manuals/R-exts.html#Embedding-R-under-Windows>).

Les deux dernières solutions nécessitent d'installer le logiciel R sur l'ordinateur du client. La dernière solution est la plus communément utilisée et donc la plus documentée.

### 5.1.5 Git, la gestion de versions

Nous développerons à l'aide du logiciel de gestion de versions décentralisé Git. Le choix se pose essentiellement entre Git et Apache Subversion (SVN). Nous avons choisi d'utiliser Git premièrement parce que nous y avons été formés. De plus, Git nous laissera plus de liberté de mouvement pour effectuer notre suivi en local sans forcément avoir à directement soumettre les modifications au serveur.

Nous n'utiliserons cependant pas complètement le côté décentralisé de Git puisque nous nous appuyerons sur un serveur GitHub ou Bitbucket (TODO Passer en bas de page : suivant si on veut que notre code source soit visible ou non du public).

## 5.2 Planning et répartitions

### 5.2.1 Planning du projet

Le projet se déroulant sur plusieurs mois, il est nécessaire de pouvoir organiser les tâches de manière efficace afin d'anticiper les éventuels problèmes et donc leurs solutions. Comme prévu dans le déroulement du projet, nous allons utiliser le logiciel Microsoft Project pour la planification et la gestion de projet.

L'utilisation de cet outil a plusieurs objectifs. D'une part nous allons pouvoir identifier de manière efficace les zones problématiques de notre planification. Cela nous permettra de tester différentes possibilités d'organisations, ainsi que de visualiser efficacement les changements que l'on doit appliquer à notre planification. D'autre part, MS Project est largement répandu dans le monde de l'entreprise. Acquérir de l'expérience sur ce logiciel dans le cadre de notre formation est donc une excellente opportunité.

### 5.2.2 Répartition du travail

Pour l'élaboration de ce projet, de multiples tâches sont à prévoir et il est intéressant de se poser la question dès maintenant de leur répartition. Tout au long de notre travail, il y aura deux postes toujours nécessaires, le « chef de projet » et le « décompte du temps ». Lors de la première phase d'analyse nous avons déjà attribué ces postes à deux personnes, mais notre objectif est de redéfinir à chaque changement de phase le poste de chef de projet afin que chacun puisse découvrir les différentes facettes de la gestion d'un projet.

Nous allons développer notre projet en suivant le modèle du cycle en V. D'octobre à décembre l'ensemble des membres du groupe prendront part aux phases de spécifications fonctionnelles et de conception logicielle. Ensuite de janvier à avril quatre membre du groupes prendront part aux phases de codage, de tests unitaire, de tests de vérification et enfin à la validation.

## 5.3 Licence

Une licence est un contrat par lequel le propriétaire du logiciel définit les conditions dans lesquelles le programme peut être utilisé, diffusé et modifié. Les licences peuvent être séparées en deux grandes familles :

- Les licences libres permettent d'utiliser, étudier, modifier, dupliquer ou diffuser librement le logiciel sous certaines réserves - préciser le nom de l'auteur par exemple ;

- les licences propriétaires assurent à son propriétaire un contrôle sur l'évolution, l'usage et la modification du produit.

Pour ce projet, une licence libre sera utilisée afin de permettre à un développeur extérieur de reprendre le code, de le modifier et éventuellement d'ajouter de nouvelles fonctionnalités.

### 5.3.1 Les licences libres

Les trois licences décrites par la suite ont été choisies parce qu'elles représentent très bien la diversité des licences libres. De plus, elles sont très utilisées dans le monde du logiciel libre.

#### Licence General Public License

La licence General Public License (GPL) est sans doute la plus connue de toutes les licences libres [3]. Elle a pour principes :

- la liberté d'exécuter le logiciel, pour n'importe quel usage ;
- la liberté d'étudier le fonctionnement d'un programme et de l'adapter à ses besoins, ce qui passe par l'accès aux codes sources ;
- la liberté de redistribuer des copies ;
- la liberté d'améliorer le programme et de rendre publiques les modifications afin que l'ensemble de la communauté en bénéficie.

L'utilisateur a le droit de modifier le code source uniquement s'il le redistribue sous la même licence GPL (principe du copyleft).

#### Licence Lesser General Public License

La licence Lesser General Public License (LGPL) est similaire à la GPL [4]. La seule différence réside dans le fait qu'elle autorise l'utilisateur à utiliser le code dans un logiciel qui n'est pas sous licence LGPL. De ce fait une personne voulant intégrer du code LGPL dans un logiciel propriétaire pourra le faire mais à la seule condition qu'il ne modifie pas ce code. Si celui-ci est modifié, il devra être mis sous une licence LGPL. Cette licence est surtout utilisée dans le cadre de bibliothèques, comme GTK+ par exemple.

#### Licence Berkeley Software Distribution

La licence Berkeley Software Distribution (BSD) est la licence offrant le plus de liberté [5]. Seul le nom de l'auteur original est demandé. Ces licences permettent de redistribuer sous une autre licence (pas de copyleft). La philosophie des licences libres reste intacte. L'utilisateur peut copier, utiliser, modifier et distribuer un programme sous licence BSD.

### 5.3.2 Tableau recapitulatif

	<b>GPL</b>	<b>LGPL</b>	<b>BSD</b>
<b>Type</b>	Libre	Libre	Libre
<b>Copyleft</b>	oui	oui	non
<b>Propriétaire</b>	Auteur	Auteur	Auteur
<b>Source obligatoire</b>	oui	oui	oui
<b>Inclusion dans un logiciel propriétaire</b>	non	non	oui
<b>Modification</b>	autorisée	autorisée	autorisée
<b>Utilisation gratuite</b>	fréquent	fréquent	fréquent
<b>Utilisation payante</b>	autorisée	autorisée	autorisée
<b>Remarques</b>	Ne peut être lié (ou inclus) à un logiciel ayant une licence autre que GPL.	Peut être lié à un logiciel ayant une licence autre que LGPL.	Le nom des auteurs ne peut être utilisé pour promouvoir un produit dérivé

### 5.3.3 Le choix de la licence

La licence retenue pour ce projet est la LGPL. La GPL n'a pas été choisie car celle-ci est trop restrictive. En effet, l'utilisation d'un code sous licence GPL force l'utilisation de cette licence sur l'ensemble du logiciel. La LGPL n'a pas cet inconvénient : le code source de ce projet pourra être réutilisé dans un logiciel propriétaire sous réserve de publier les modifications apportées sous LGPL. Enfin, la licence BSD n'a pas non plus été retenue car elle n'oblige pas la diffusion du code modifié.

Pour valider la licence il faut mettre un entête dans chacun des fichiers sources. La documentation devra mentionner la licence. De plus un fichier nommé LICENCE devra apparaître dans un fichier de la distribution, il contiendra le texte de la licence. Enfin, un lien dans l'interface devra amener vers un rappel de licence et le texte de la licence. En annexe A, figure un modèle d'en-tête de licence.

# Glossaire

**portefeuille** Un portefeuille est une composition de plusieurs actifs financiers (actions, obligations, matières premières, etc).

# Annexes

# Annexe A

## En-tête de la Licence LGPL

Copyright (C) 2013 Benjamin Bouguet, Damien Carduner, Paul Chaignon, Eric Chauty, Xavier Fraboulet, Clement Gautrais, Ulysse Goarant

This program is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

# Bibliographie

- [1] C. Francq and J.M. Zakoian. *GARCH Models : Structure, Statistical Inference and Financial Applications*. Wiley, 2010.
- [2] Create executable programs. <http://stat.ethz.ch/R-manual/R-devel/library/utils/html/LINK.html>, décembre 2012. Consulté le 9 octobre 2013.
- [3] Free Software Foundation. Licence gnu general public license (gpl). <http://www.gnu.org/copyleft/gpl.html>, 06 2007. Consulté le 27 septembre 2013.
- [4] Free Software Foundation. Licence gnu lesser general public license (lgpl). <http://www.gnu.org/copyleft/lgpl.html>, juin 2007. Consulté le 27 septembre 2013.
- [5] Anonyme. Licence berkeley software distribution license (bsd). <http://www.freebsd.org/copyright/license.html>, 1995. Consulté le 27 septembre 2013.