Notes détaillées sur le fonctionnement de l'outil

1. Fichier Excel 1 : Base de données

1.1. Importation

Ce module contient 1 macro *importation* et sert à importer l'historique des cours des actions du 01/01/2004 au 23/02/2024. Elle est à lancer en premier afin de regrouper dans un même classeur chaque feuille de cours en fonction de son type : actions françaises, indices, actions tech, bons du trésor US 30 ans. Cela va nous permettre d'aller chercher les données directement par la suite.
Dans le code VBA, on importe les feuilles une par une grâce à la méthode "Application.GetOpenFilename". Il est important pour l'utilisateur d'importer chaque feuille dans le bon ordre : *FrenchStocks*, *Indexes*, *TechStocks* puis *Rates*.

1.2. Runs

Dans ce module, nous allons chercher à calculer les répartitions des runs pour chaque action disponible dans notre univers d'investissement.

- **Fonction fn_Stat_Run**

Dans un premier temps, on crée une fonction fn_Stat_Run qui permet de calculer le nombre de runs et la fréquence des cours. Cette fonction prend en argument un vecteur c() qui comprendra toutes les valeurs enregistrées pour un titre.

Tout d'abord, on calcule le nombre de valeurs enregistrées pour le titre en stockant la longueur du vecteur des cours c() dans nbre_periodes. On redimensionne le vecteur des Runs() à cette valeur.

On crée une boucle pour itérer sur chaque période, de la 2 à la dernière. On calcul la variation subie par le cours entre cette période et la précédente, la valeur est stockée dans *val*. Puis, on crée une condition sur le signe entre cette variation et la précédente (*val*val_prec*) afin de savoir si le cours a changé de sens ou non. Si ce n'est pas le cas, on incrémente *durée* de 1, ce qui signifie que le cours n'a pas changé pendant une période supplémentaire. Dans le cas contraire, on incrémente le nombre de runs à la valeur de *duree* de 1. Si cette durée est supérieure à au maximum enregistré récemment, *max_duree* prend cette valeur. Puis, *var* devient la valeur précédente *var_prec*.

Puis, on redimensionne le vecteur Runs() à la taille du maximum de runs enregistrés, soit *max_duree*. On fait la même chose pour le vecteur freq() qui va contenir la fréquence des runs pour chaque valeur de runs.

Pour cela, on crée un autre boucle pour itérer sur chaque valeur de durée de runs. Pour chacune, on calcule la fréquence des runs tels que freq[j] = Runs[j]/tot, avec tot la somme de tous les runs calculée grâce à la fonction Sum.

Le résultat de la fonction renvoyée sous la forme d'un Array contenant Runs et freq().

Dans un deuxième temps, on crée une Sub Runs qui va permettre d'utiliser la fonction que nous venons de créer et l'appliquer pour tous les titres que nous avons à notre disposition sous la forme d'un tableau des runs.

- **Sub Runs**

Nous commençons par y renommer la première feuille encore vierge par « Actions ». Nous calculons grâce à la méthode .End(xlUp).Row le nombre de dates disponibles.

Puis grâce à boucle, nous calculons la dimension de la série s'arrêtant pour chaque action au 30/12/2004, date à laquelle on va se référer pour le prix de vente des titres car étant la dernière enregistrée à la date de l'achat.

Nous créons une autre boucle qui va itérer sur nos quatre feuilles de cours. À l'intérieur, nous créons une autre boucle pour itérer sur les titres de la feuille sélectionnée c'est-à-dire les colonnes. Puis nous créons à nouveau une boucle à l'intérieur afin d'itérer sur les valeurs prises par le titre à chaque date jusqu'au 30/01/2004.

À titre informatif, nous avons aussi voulu renseigner en ligne 12 les taux de rendement des titres sur la période concernée. Pour cela les taux de rendements sont stockés dans le vecteur r() et calculés selon la formule r = (Rendement à t – Rendement à (t-1)) / Rendement à (t-1). Puis, on appelle la fonction fn_Stat_Runs afin de calculer la distribution des runs pour calculer le nombre et la fréquence de runs des cours. Ces valeurs sont reportées dans le tableau des runs.

En dessous, nous avons aussi voulu calculer la fréquence des runs supérieurs ou égaux à 3 grâce à la fonction Sum. Ce résultat sera utile pour la composition de notre portefeuille d'actions ultérieurement.

À la sortie de la toute première boucle et jusqu'à la fin de la Sub, il s'agit de la mise en page du tableau des runs.

1.3. Actions

Ici, on se sert de la feuille « Actions » créée dans le module Runs. Après avoir calculé les répartitions des runs des titres à disposition de notre client pour investissement, ce module va nous permettre de sélectionner ceux à retenir afin de déterminer la composition du portefeuille d'actions. Ce code permet ainsi d'établir notre stratégie d'investissement dans les actions, telle que l'on ne va retenir que les titres ayant au moins 75% de leurs runs répartis entre 3 et 7. Nous souhaitons que le budget total alloué à cet investissement soit d'au moins 500 000€.

Pour cela, on crée une Sub Selection.

Elle est d'abord composée d'une boucle parcourant les colonnes (= titres), et qui sélectionne ceux ayant au moins 75%. On crée un nouveau tableau plus bas sur la feuille qui contient les intitulés des titres des actions retenues.

Ensuite, nous avons créé une autre boucle afin qu'à la ligne suivante du tableau soit reporté les cours de l'action de chaque titre retenu à la date du 30/12/2004, soit la dernière date enregistrée à la date de l'investissement et donc le prix d'achat de l'action pour une part. Pour cela, on crée une autre boucle à l'intérieure de la première qui va itérer sur les 4 feuilles de cours, et on utilise la méthode .Find afin de trouver dans la première ligne de la feuille de nom du titre recherché et de retenir l'adresse où il se trouve. Puis, la méthode Offset nous permet de trouver la valeur dans la colonne du titre à la ligne 53, c'est-à-dire celle au 30/12/2004.

Puis, on décide du nombre de parts à acheter dans chaque titre, c'est-à-dire la composition du portefeuille. Pour cela, on alloue un budget total de 500 000€ en actions (*budget_total*). Nous avons décidé d'un portefeuille équipondéré entre nos 10 (*nbre_*titres) titres sélectionné, tel que le budget investi dans un titre est de *budget_titre = budget_total / nbre_*titres. Puis nous instaurons une boucle pour itérer sur nos 10 titres. Pour chaque titre, nous augmentons le nombre de parts dans un titre jusqu'à ce que la valeur investie dans le titre ait dépassé *budget_titre* = 50 000€. Nous n'avons pas choisi une égalité parfaite afin d'optimiser notre programme, du fait de la faible probabilité d'atteindre exactement cette valeur. Ainsi notre programme fait en sorte d'atteindre une valeur la plus proche possible de la valeur exacte sans être trop lourde pour notre programme. Le nombre de parts de l'action est reporté en ligne 20 et le budget exact investi en ligne 21.

À l'aide de la fonction Sum, nous avons instauré une colonne « Total » à la fin du tableau des actions afin de renseigner le nombre de parts possédées par le client au total et la valeur exacte investie dans les actions.

La fin de la Sub sert à la mise en page du tableau d'investissement en actions que nous avons créé sur la feuille « Actions », ainsi qu'à appeler la macro Obli qui servira à la composition du portefeuille d'obligations.

1.4. Obligations

Ce module a pour objectif de créer 10 obligations pour notre client sur la base d'un budget que nous avons établi de manière aléatoire et qui doit représenter moins de la moitié du portefeuille, soit moins de 500 000€. De plus, la moitié des obligations doit verser un coupon semi-annuel.

Dans ce module, nous avons trois procédures ainsi qu'un module de classe.

- **Module de classe**

Ce module de classe définit une structure pour représenter une obligation financière et fournit des méthodes pour calculer différentes mesures financières importantes associées à cette obligation.
Ce module va nous permettre par la suite d'effectuer des calculs afin de déterminer toutes les informations de nos obligations. Ainsi, il nous permet :

- De déclarer des propriétés de la classe :
  Public Maturite As Integer : Représente la durée de vie de l'obligation en nombre d'années.
  Public Nominal As Long : cela représente la valeur nominale de l'obligation.
  Public TxCoupon As Double : cela représente le taux de coupon de l'obligation (sous forme décimale).
  Public Periodicity As Integer : cela représente la périodicité des paiements de coupon (en nombre de fois par an).
  Public RiskFreeRate As Double : cela représente le taux sans risque associé à l'obligation (sous forme décimale).

  D'initialiser les propriétés de l'obligation lors de sa création avec la sub InitOblig

  - Un calcul automatique de nos coupons avec la propriété Coupon, pour cela, il nous faut faire la différence entre les coupons annuels et semi-annuels en utilisant l'instruction « if, else ». Pour les coupons annuels, on fait le calcul : Coupon = taux de coupon x nominal. Pour les coupons semi-annuels, on fait le calcul : Coupon = ((1 + Taux de Coupon) ^ (1 / Périodicité) - 1) x Nominal

  - Un calcul de la valeur des obligations avec la propriété BondValue. Tout d'abord, on commence par initialiser la valeur d'une obligation à 0. Pour effectuer ce calcul, on va effectuer une boucle pour chaque période avec l'instruction « for, next » avec i allant 1/ période jusqu'à atteindre la maturité de l'obligation avec un pas de 1/période. Lorsque l'on atteint la dernière période alors le numérateur sera égal au Coupon + Nominal, sinon uniquement au coupon. Pour se faire, on utilise l' instruction « if, else ».Le dénominateur quant à lui vaut (1 +Taux sans risque) ^ i. Enfin la valeur de l'obligation vaut la somme de la valeur de l'obligation de toutes les boucles divisées par le dénominateur.

  - De calculer la duration de Macaulay grâce à la propriété MacaulayDuration. On commence par initialiser cette duration à 0. Pour effectuer ce calcul, on va effectuer une boucle pour chaque période avec l'instruction « for, next » avec i allant 1/ période jusqu'à atteindre la maturité de l'obligation avec un pas de 1/période. Dans chaque boucle, nous le cashflow sera soit égal à Coupon / Périodicité + Nominal si i est égal à la maturité, sinon à Coupon / Périodicité. Pour ce faire, nous utilisons une instruction « if, else ». Enfin la duration de

Macaulay est égal à la somme pour chaque période du calcul suivant : i * (CashFlow / ((1 +Taux sans risque) ^ i)) /Valeur de l'obligation.

- De calculer la duration modifiée de Macaulay grâce à la propriété Modified Macaulay Duration. Cette propriété calcule la durée de Macaulay modifiée de l'obligation, qui est la durée de Macaulay ajustée pour tenir compte de la périodicité des paiements de coupon. Elle est donc égale à la duration de macaulay divisée par (1 + Taux sans risque/ Périodicité)

- **Sub Obligation**

Pour obtenir notre portefeuille d'obligations, nous avons en premier lieu décidé de choisir le portefeuille d'action, afin de connaître la partie du budget restant que nous pouvons allouer à nos obligations. On récupère donc le budget directement comme argument dans la sub. De plus, nous fixons notre nombre de titres à 10.
Cette sub consiste à mettre le nom de nos énoncés dans notre tableau d'obligation ainsi qu'à appeler les deux sub suivantes que nous allons détailler par la suite et qui se nomment *Bonds* et *Applystyle*.

- **Sub Bonds**

Cette sub va permettre de créer les obligations en fonction du nombre spécifié (pas forcément 10) et du budget.
Après avoir attribué les variables et défini les dimensions, nous allons effectuer une boucle « do, loop » pour créer des obligations tant que la valeur totale reste inférieure au budget. Ainsi, à chaque itération de la boucle, une nouvelle obligation est créée de manière aléatoire grâce à la fonction *Randomize* et *rnd()*. Pour ce faire, on va utiliser les caractéristiques du module de classe.
Le nominal de chaque obligation est déterminé grâce aux nombres de titres et au budget total, il est calculé comme suit :

- Bond.Nominal = ((budget / nbTitres) * (0.5 + (Rnd()))* 10 / nbTitres

Puis, on va simuler les taux de coupons et les maturités avec les calculs suivant :

- Bond.Maturite = Int(Rnd() * 20) + 1
- Bond.TxCoupon = 0.01 + Int(Rnd() * 10) / 100

Par ailleurs, le taux sans risque pour chaque obligation est la même et est égal à 5%

Par la suite, on détermine pour chaque obligation sa périodicité : la moitié des obligations aura une périodicité de 1 et l'autre moitié de 2.
On ajoute par la suite chaque obligation dans une collection que l'on nomme *BondPtf*. Puis, nous faisons apparaître toutes ces informations dans un tableau sur Excel à l'aide de la propriété Array.
La boucle s'effectue tant que la valeur totale des obligations représente plus de la moitié de la valeur du portefeuille (qui est mise à jour à chaque itération, i.e. qui augmente à chaque boucle de la valeur de la nouvelle obligation créée).

On diminue notre budget restant de la valeur de notre portefeuille d'actions et une obligation supplémentaire est utilisée pour compléter le budget restant. Sa maturité et sa périodicité valent 1, son taux de coupon vaut 0 et son taux sans risque vaut 5%. Quant à son nominal, il est égal au montant du portefeuille restant multiplié par (1+5%). On calcule ainsi la nouvelle valeur de notre portefeuille. Le budget restant à la fin de l'opération est donc égal à 0.

- **Sub ApplyStyle**

Tout d'abord, cette sub va donner les dimensions du tableau où seront écrits tous nos résultats. On va également trier nos obligations par ordre croissant en fonction des durations (de la plus basse à la plus importante).
Elle va permettre d'effectuer toute la mise en page, notamment le style des intitulés, de mettre des bordures à notre tableau, de changer la couleur d'arrière-plan de certaines cellules, d'ajuster les colonnes et enfin d'enlever le quadrillage.

2. Fichier Excel 2 : Outil

Modules :

2.1. Importation

Ce module a pour but d'importer chaque feuille de cours sur Excel comme nous l'avions fait pour le premier fichier Excel. On va ainsi copier toutes les feuilles du fichier de la base de données sur ce document Excel qui étaient au nombre de 6. Puis, on ferme la base de données et on supprime la première feuille Excel qui était vide.
Concernant la feuille contenant les actions, nous supprimons toutes les lignes où étaient répertoriés les runs afin de ne conserver que notre tableau du portefeuille d'actions.
Sur une nouvelle feuille que l'on crée, on va effectuer une boucle afin de récupérer uniquement les cours d'actions que nous avons utilisés dans notre portefeuille d'actions.
Ainsi, nous aurons sur une nouvelle feuille tous les cours que l'on prend en compte et on ajoute enfin la colonne des dates.
Suite à cela, on va supprimer les 4 feuilles de cours que nous avons utilisées afin de n'en garder qu'une seule, regroupant ainsi les titres qui nous intéressent.
Enfin, nous supprimons toutes les dates et donc cours qui se trouve avant le 30/12/2004.
On effectue par la suite la mise en page de la feuille de cours.

2.2. Outil

Ce module ne contient qu'une seule sub du même nom et a pour but de mesurer l'évolution de la valeur du portefeuille d'actions et d'obligations entre la date la plus ancienne (le 30/12/2004) et les 2 dernières dates saisies par l'utilisateur.
Premièrement, on enregistre dans les variables date_ et date_2 les 2 dernières dates saisies par l'utilisateur. Pour cela, on attribue à *date_2* la valeur de la cellule (14,12) de la feuille Obligation et on demande via une InputBox à l'utilisateur la *date_* que l'on inscrit également ensuite dans la cellule (14,12) de la feuille obligations pour qu'elle soit attribuée à date_2 lors de la prochaine exécution du code. On convertit ensuite ces dates en format numérique pour pouvoir les rechercher par la fonction FIND.
On commence par une première boucle sur les actions où l'on va reporter les valeurs calculées lors de la dernière exécution du code sur la ligne correspondant à celle de la date_2 dans l'optique de faire passer date_ à date_2 (si c'est la première exécution, alors rien ne sera reporté). On initialise ensuite un nouvel objet de notre classe d'action et via un constructeur de type Call(...) on rentre toutes les données nécessaires au calcul des rendements. Toutes ces informations sont déjà présentes sur la feuille sauf la valeur du cours à la date 1 (date_). Pour trouver cette valeur, on reporte préalablement (avant l'appel du constructeur) cette valeur dans le tableau grâce à la fonction FIND en passant par un variant *Adresse*. Si la date entrée ne correspond à aucune date de la feuille cours, alors on recherchera la date précédente jusqu'à tomber sur une date présente sur la feuille *Cours*. On ajoute finalement l'objet à la collection *Action* et on calcule la valeur par poche (titre) avec des formules simples.On termine finalement la boucle.

On passe ensuite sur une nouvelle boucle concernant les obligations. Cette boucle est moins complexe puisqu'il n'y a pas de cours à aller chercher et car les propriétés de l'objet *Obli* permettent de réaliser directement tous les calculs. On va donc simplement créer un nouvel objet *Obli* et refaire 2 fois la même méthode dans la boucle pour chaque date (date_ et date_2). Pour cela, on appelle le constructeur via la méthode Call(...) dont les 2 derniers arguments sont la date de départ puis la date de fin. Donc on va faire une première fois avec la plus ancienne date (le 30/12/2004) et la date 1 (date_) et une autre avec la plus ancienne date et la date 2 (date_2). À chaque fois, on reporte dans un tableau la valeur actualisée, les cashflows perçus ainsi que la duration via les propriétés de l'objet *Obli*. Finalement, on calcule le montant de cash flows perçus entre les 2 dates saisies que l'on reporte sur la dernière colonne du dernier tableau et l'on termine la boucle.
On calcule finalement les totaux et on met en page la feuille avant d'afficher via une MsgBox la valeur totale du portefeuille à la date 1. La sub se termine sur l'appel de la sub *Volatilite*.

2.3. Volatilité

Après avoir initialisé les variables, on formate la cellule de la feuille obligation comprenant la nouvelle date rentrée par l'utilisateur afin qu'elle soit en format numérique. Puis on calcule grâce à la feuille *Cours*, le nombre d'actions et de dates que l'on a. Enfin, on formate également toutes les dates de la feuille *Cours* au format numérique.

Suite à la date saisie par l'utilisateur dans l'outil, on va effectuer une boucle sur les actions. On va pour chaque action, rechercher la nouvelle date rentrée par l'utilisateur dans la feuille de cours afin d'en récupérer son cours à cette date. Puis, on va utiliser la fonction que nous expliquerons plus bas pour calculer la matrice covariance-variance des titres.
Une fois la matrice réalisée, on va reformater la case contenant la nouvelle date rentrée par l'utilisateur de la feuille obligation et la colonne des dates de la feuille cours au format date. Puis nous allons mettre le nom de chaque titre autour de la matrice variance covariance pour plus de clarté lors de la lecture du document par le client.

Par la suite, on va calculer le budget en pourcentage qui est alloué pour chaque titre, on va également recopier la variance de chaque titre que l'on trouve dans la matrice précédente et grâce à cela, calculer la volatilité.
Pour calculer la variance du portefeuille par produit matriciel, on va appliquer la fonction de "Application.WorksheetFunction" *MMult* ainsi que *Transpose*. Cela va nous permettre de non pas obtenir une matrice comme on obtiendrait en utilisant seulement *MMult* mais d'obtenir un vecteur. Puis, suite à ces résultats, on calcule la volatilité du portefeuille en faisant la racine carrée de la variance.
Suite à cela, on va ajuster les cellules avec du formatage et mettre tous les titres dont nous avons besoin pour nommer les lignes afin que le client puisse se repérer. On va enfin mettre en page la feuille *Composition Action* en mettant des bordures et de la couleur dans les tableaux.

Dans ce module, on a également une fonction *Cov* qui sert à retourner une matrice de covariance. Tout d'abord, on définit le nombre de titres que l'on a. Puis, on redimensionne *Result* de la taille qu'aura la matrice.
La double boucle parcourt chaque paire de colonnes dans la plage de données et calcule la covariance entre les données de chaque paire à l'aide de la fonction Covariance_S de la classe WorksheetFunction. Les données de chaque colonne sont stockées dans plage_1 et plage_2.

Modules de classe :

2.4. ActionClass

Ce module de classe permet de créer une classe d'actions dont on va se servir pour la composition du portefeuille des actions.

- **Sub *InitAction***

Cette Sub est un "constructeur" et permet de construire une classe *Action* à laquelle on va attribuer un nom (ticker), le nombre de parts possédé, le budget investi dans l'action, la date du tout premier cours (30/12/2004), la date rentrée par l'investisseur et la date rentrée par l'investisseur la fois précédente.

- **Property Get *Rend1***

Permet de calculer le rendement de l'action entre son achat au 30/12/2004 et la date rentrée par l'investisseur.

- **Property Get *Rend2***

Permet de calculer le rendement de l'action entre la date rentrée par l'investisseur et la date rentrée par l'investisseur la fois précédente, dans le cas où l'ancienne date rentrée est postérieure à la nouvelle.

- **Property Get *Rend3***

Permet de calculer le rendement de l'action entre la date rentrée par l'investisseur et la date rentrée par l'investisseur la fois précédente, dans le cas où l'ancienne date rentrée est antérieure à la nouvelle.

2.5. ObliValeur

Ce module de classe permet de créer une classe d'obligations dont on va se servir pour la composition du portefeuille des obligations.

- **Sub *InitObli***

Cette Sub permet de construire une classe Obli à laquelle on va attribuer une maturité, un nominal, un taux de coupon, une périodicité, un taux sans risque, une valeur, un coupon, la date du tout premier cours (30/12/2004) et la date rentrée par l'investisseur.

- **Property Get *ValeurAct***

Permet de calculer la valeur actuelle de l'obligation à la date rentrée par l'investisseur (*date_1*). Elle calcule le nombre d'années écoulées en valeur entières depuis le 30/12/2004 (*date_0*), et en déduit le temps restant jusqu'à la maturité de l'obligation.

On instaure ensuite la condition où l'obligation n'est pas encore arrivée à maturité, et on place une boucle afin d'itérer sur toutes les périodes. Dans la boucle, on effectue le calcul de la valeur actuelle.

- **Property Get *CashFlows***

Permet de calculer les cash flows perçus par l'obligation.

On instaure ainsi une condition suivant si le coupon est annuel ou semi-annuel afin de calculer le nombre de périodes écoulées depuis le 30/12/2004 et on stocke la valeur dans *Temps*. On en déduit le temps restant jusqu'à la maturité (*matRestante*) et on distingue selon que l'obligation soit arrivée à maturité ou non dans le calcul des cash flows perçus.

- **Property Get *MacaulayDuration***

Permet de calculer la duration de Macaulay. On procède de la même manière que pour les cash flows pour calculer le nombre de périodes écoulées et le temps restant jusqu'à la maturité de l'obligation. Puis, on utilise une boucle pour itérer sur toutes les périodes. Enfin, on calcule les cash flows perçus de la même manière que dans CashFlows, et on calcule le résultat de la duration.

- **Property Get *ModifiedMacaulayDuration***

Permet de calculer la duration modifiée de Macaulay. Pour cela, on fait appel à la propriété précédente, MacaulayDuration, afin de calculer le résultat.

3. Fichier SQL

Le fichier SQL va permettre de créer également une base de données mais basée sur un critère différent que celui du fichier excel. La commande LOAD DATA INFILE étant désactivée pour des raisons de sécurité sur les systèmes macOS (et difficile à activer), il est nécessaire d'importer le fichier contenant la base de données excel en .csv. Pour cela, il suffit

d’effectuer un clique droit sur la database dans le ruban à droite de MySQL puis de cliquer sur “Table data import Wizard” et d’importer la base en.csv.

Le code va ensuite créer une database dans laquelle il y aura 2 tables. La première (*Rendements*) contiendra les rendements pour chaque titre entre le début d’année et la fin d’année 2004. Pour cela, on a calculé les rendement de chaque titre dans le sous-ensemble de la base contenant 2 lignes (une pour chaque date). L’autre table (*Action*) contiendra uniquement les titres dont le rendement a été positif cette année-là. Ceci s’effectue simplement avec une procédure posant une condition sur la valeur du rendement de chaque titre de la forme SELECT … UNION SELECT.

Table Actions :

ticker	rendement
ACA.PA	0.1875
AMD	0.266667
BNP.PA	0.0816327
FCHI	0.0746562
FTSE	0.0744467
GLE.PA	0.047619
HO.PA	0.346154
N225	0.0632147
SU.PA	0.04
VIE.PA	0.142857

Il est également possible de faire une connexion entre MySQL et Excel en utilisant la méthode ADODB.Connection. Cette approche implique d'écrire des scripts VBA dans Excel pour établir une connexion à la base de données MySQL, exécuter des requêtes SQL et récupérer les résultats dans une feuille Excel. Pour cela, il faut passer par des fonctions Excel : une servant à transformer le classeur en CSV et à l’importer dans SQL et une servant à établir la connexion. Dans cette dernière, il faudra utiliser la méthode *conn.Open*.
Enfin, il faudra créer une autre fonction ayant pour but de sélectionner toute la base de donnée dans mySQL et prenant en argument la connection trouvée par la fonction précédente.
Cependant, la connexion en MySQL et Excel étant impossible sur MacOS, nous avons créé une base Excel que nous avons utilisé dans la suite de notre projet.

Rédigé par VILLEMONTE DE LA CLERGERIE Margaux, PATTRICK Anastasia et VOIRIN Tristan.