Manuel :

Stage L'analyse du marché des options à la Bourse de Paris à la Belle Epoque,

1899-1914

1 – statistiquesGenerales.py

# 8 - 12

Les 1ères lignes du code importent les différents modules nécessaires aux calculs des statistiques.

* Pandas : Pandas est une [bibliothèque](https://fr.wikipedia.org/wiki/Biblioth%C3%A8que_logicielle) écrite pour le langage de programmation [Python](https://fr.wikipedia.org/wiki/Python_(langage)) permettant la manipulation et l'[analyse des données](https://fr.wikipedia.org/wiki/Analyse_des_donn%C3%A9es).
* Numpy : NumPy est une bibliothèque pour langage de programmation [Python](https://fr.wikipedia.org/wiki/Python_(langage)), destinée à manipuler des [matrices](https://fr.wikipedia.org/wiki/Matrice_(math%C3%A9matiques)) ou tableaux multidimensionnels ainsi que des fonctions mathématiques opérant sur ces tableaux.
* Matplotlib : Matplotlib est une bibliothèque du langage de programmation [Python](https://fr.wikipedia.org/wiki/Python_(langage)) destinée à tracer et visualiser des données sous formes de graphiques.
* Datetime : Le module [datetime](https://docs.python.org/fr/3/library/datetime.html#module-datetime) fournit des classes permettant de manipuler les dates et les heures.
* Statistics : Ce module fournit des fonctions pour calculer différentes statistiques numériques sur des données.

# 14

Cette ligne permet d’importer les données depuis les fichiers Excel que l’on veut traiter. Le 1er argument de la fonction read\_excel correspond au chemin de l’emplacement du fichier Excel. Il faut donc modifier cette ligne pour écrire le chemin correspondant à celui de votre ordinateur. Le fichier choisit correspond aux échanges à la bourse de Paris des 11 stocks les plus liquides.

# 17 – 27

Il est arrivé 26 fois que le titre ne s’échange pas ni le jour même, ni la veille. Dans ce cas, le cours de compensation a été utilisé. Le cours de compensation n’est pas un prix de transaction mais un prix calculé par les agents de change permettant de liquider les opérations à prime qui se dénouent ce jour. C’est à cela que correspond ces lignes.

# 29 – 94

## def nb\_titres\_ech(affichage=0):

Cette fonction permet d’afficher le nombre d’échanges réalisées pour les 11 stocks les plus liquides de la base de données sur la période 1898-1914. Cette fonction affiche aussi la moyenne, les quartiles ainsi que l’écart-type. Cette fonction renvoie un tableau de 11 éléments tab\_stock\_id correspondant aux 11 ID des 11 stocks, ainsi qu’un tableau qui contient le nombre de titres échangés par stocks tab\_nb\_titre\_ech (exemple tab\_nb\_titre\_ech [0] contient le nombre d’échanges réalisés pour le stock tab\_stock\_id [0] i.e. stock 985). Elle renvoie ces deux tableaux car ils sont utiles pour d’autres fonctions dans ce fichier.

Comme cette fonction affiche des graphiques et qu’on a besoin de l’appeler dans d’autres procédures SANS utiliser ces graphiques, nous avons inclus un paramètre d’affichage. Si ce paramètre est à 0 alors les graphiques seront affiché. Dans le cas contraire ils ne le seront pas. Le paramètre est instancié à 1 dans une autre procédure qui doit elle aussi afficher des graphiques. L’affichage des deux graphiques en même temps provoque des interférences et affiche n’importe quoi d’où la nécessité d’avoir un paramètre qui contrôle l’affichage.

# 97 – 159

## def duree\_maturite():

cette fonction permet de calculer la durée des maturités pour toutes les options échangées à la Bourse de Paris sur la période. On affiche également la moyenne ainsi que l’écart-type. Un graphique a été réalisé pour plus de lisibilité. Cette fonction ne renvoie rien.

# 162 – 222

## def stat\_gen():

Diverses statistiques (moyenne, quartiles, écarts-types, etc.) sur toutes les colonnes « utiles » de la base de données. La description de ces colonnes est indiquée dans le fichier StatistiquesDescriptives. Cette fonction ne renvoie rien.

# 225 - 296

## def funct\_noTrade():

Cette fonction affiche en graphique les fréquences de « no trade » pour chacun des 11 stocks les plus liquides de la Bourse de Paris sur la période 1898-1914. La définition d’un « no trade » est expliqué dans le fichier StatistiquesDescriptives. Cette fonction appelle la procédure nb\_titres\_ech (1) définie précédemment (sans afficher les graphiques liés à celle-ci). Elle affiche également la moyenne, les quartiles ainsi que l’écart-type.

Cette fonction ne renvoie rien.

# 300 – 312

## def TSR() :

Cette fonction affiche la valeur en % du taux sans risque sur la période 1896 – 1914. Une définition du taux sans risque est donnée dans le rapport de stage.

# 315 – 400

Cette partie-là est une partie destinée aux calculs de la volatilité historique passé. Cette section a été plus ou moins abandonnée au cours de mes avancés au profit d’un calcul de la volatilité réalisée glissante sur plusieurs fenêtres. Je l’ai néanmoins laissé en commentaire au cas ou un futur stagiaire et/ou chercheur souhaiterait s’en servir pour calculer la volatilité historique passé.

2 – volatiliteImplicite.py

# 10 – 14

Import des modules. Définition déjà donné précédemment.

# 17

Import des données du fichier Excel

# 31 – 37

## def d\_BS(sigma, S, K, r, t, q):

Calcul des paramètre d1 et d2 des équations de Black and Scholes. Les paramètres sont : sigma = la volatilité, S = le prix du sous-jacent, K = le strike, r = le taux sans risque, t = la maturité et q = les dividendes annualisés.

Cette fonction renvoie les valeurs de d1 et d2.

# 40 - 44

## def call\_price\_BS(sigma, S, K, r, t, q, d1, d2):

Calcul du prix du call selon le modèle de Black and Scholes en utilisant notamment les paramètres d1 et d2 calculés précédemment. Pour plus d’informations sur le modèle, consulter le rapport de stage.

Cette fonction renvoie le prix C du call.

# 47 – 62

## def d\_bach(sigma, S, K, r, t, q):

Calcul du paramètre dn du modèle de Bachelier.

Renvoie la valeur de dn.

# 65 - 72

## def call\_price\_bach(sigma, S, K, r, t, q, dn):

Calcul du prix d’un call selon le modèle de Bachelier (voir le rapport de stage pour plus de détails). Certains calculs de prix génèrent une « math error » que je n’ai pas eu le temps de résoudre. C’est pourquoi j’ai décidé en l.69 de levé une exception en cas de problème.

Cette fonction renvoie donc le prix du call selon le modèle de Bachelier.

# 77 - 173

## def newton\_raphson(S, K, r, t, q = 0, C0 = 0, method = 0):

Cette fonction calcule la volatilité implicite de façon numérique en utilisant l’algorithme de Newton-Raphson. Les étapes de cet algorithme sont expliquées clairement dans le rapport de stage. La fonction permet de calculer le prix du call selon le modèle de Black & Scholes (method=0) ou selon le modèle de Bachelier (method=1).

Cette fonction renvoie la valeur de la volatilité implicite calculé par l’algorithme avec une précision de 10-3.

# 179 – 190

Même explication qu’en l.17-27 du fichier statistiquesGenerales.py.

# 199 – 290

## def calcul\_vol\_impli\_BS(id\_stock):

Calcul de la volatilité implicite pour tous les prix de la base de données et détermination du prix du call en utilisant la volatilité implicite calculée précédemment (cette étape sert juste à montrer que l’algorithme fonctionne bien ; en effet la valeur du call calculé doit correspondre à la valeur du prix observé sur les marchés).

La procédure à en paramètre id\_stock. En effet, la fonction renvoie les volatilités implicites calculés pour un seul stock sur la période 1898 – 1914 ainsi que les dates correspondantes. Exemple : calcul\_vol\_impli\_Bach(985) renverra un tableau des volatilités implicites du stock 985 sur les 16 ans, et un autre tableau avec chaque date ou il y a eu un échange du stock 985.

# 298 – 389

## def calcul\_vol\_impli\_Bach(id\_stock):

Même chose que précédemment mais en utilisant le modèle de Bachelier.

3 – volatiliteRealiseeGlissante.py

# 9 – 12

Import des modules.

# 16

Import des données de l’Excel.

# 23 - 47

## def rempli\_tableau\_prix\_date(id\_stock):

prend en paramètre l’id\_stock que l’on veut. Fonction qui va chercher toutes les dates d’échanges ainsi que les prix pour l’ID en question.

Elle renvoie donc les prix et les dates.

# 50 - 60

## def calcul\_date\_vol\_deux\_ans\_avant(tab\_date, fenetre):

Fonction qui cherche la date extrême i.e. la première pour pouvoir calculer la volatilité réalisée passée glissante. (exemple pour un stock qui s’échange depuis le 1/08/1898 et que l’on veut calculer la volatilité passée glissante, fenêtre 2 ans, cette fonction renverra 1/08/1900).

# 64 – 71

## def calcul\_date\_vol\_deux\_ans\_apres(tab\_date, fenetre):

Même principe mais pour la volatilité réalisée future ; qui renverra donc la dernière date (moins la fenêtre).

# 77 – 89

## def valeur\_date(id\_stock, fenetre):

Rassemblement des deux dates trouvées dans une seule fonction. Renvoie les deux dates.

# 96 – 166

## def calcul\_volatilite(id\_stock, fenetre):

Cette fonction calcule les volatilités réalisées passé et future pour le stock de notre choix dans la fenêtre de notre choix. Le calcul de cette volatilité est détaillé dans le rapport de stage. On affiche les deux graphiques superposés.

main.py

Dans le main on lance toutes les fonctions nécessaires à la visualisation des différentes statistiques et des différentes volatilités. A l’heure actuelle, je n’ai pas réussi à afficher plusieurs graphiques en même temps. Il faut donc lancer les fonctions une par une et laisser les autres en commentaires.

L’id\_stock et la fenêtre sont modifiable en ligne 36 et 37. Un récapitulatif des différents stocks proposés dans la base de données et affichés juste au-dessus.