DE JUAN Louis
MEDJAHED Haris
Sive Christophe

Projet finance
Présentation des résultats

Les codes matlab et python sont envoyés en pièce jointe du mail envoyé. (ce sont les même mais en langage informatique sont différents

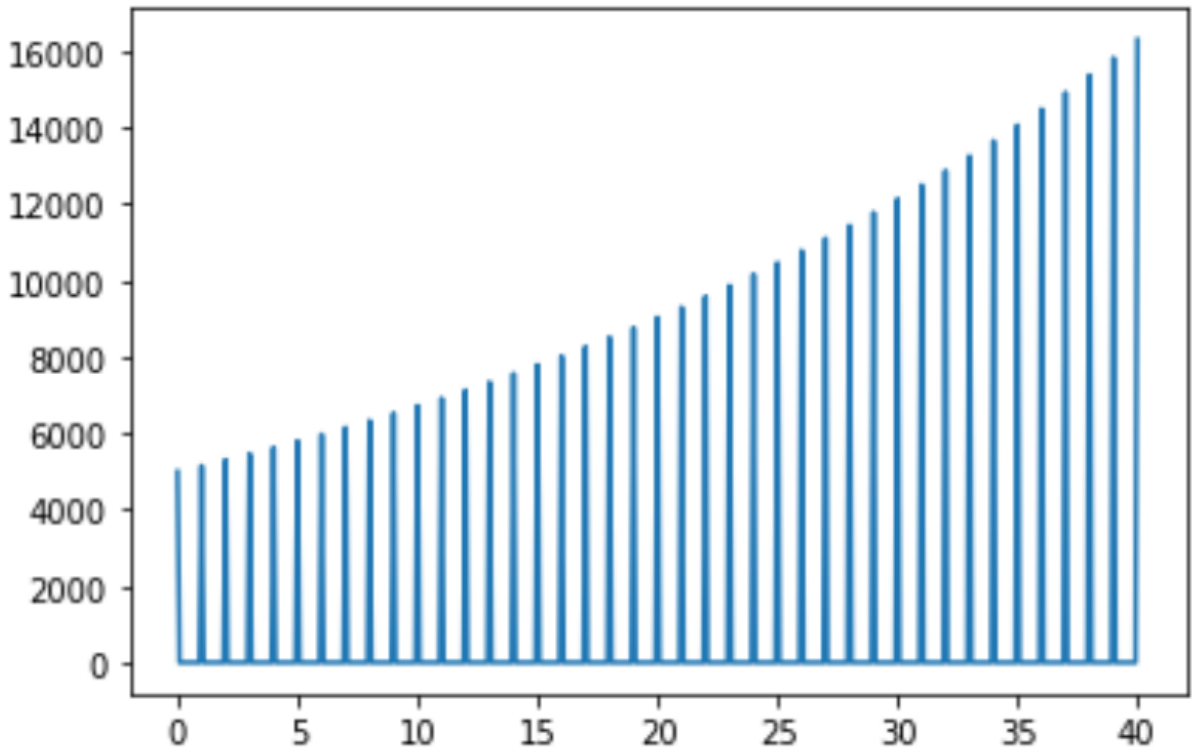
Les contributions au portefeuille sont indépendantes de tout investissement on peut donc les calculer au préalable on peut définir une fonction **calcul_contribution**

```python
def calcul_contribution(income, r_income, r_contrib, vect_temps) :
    '''
        Fonction calcul la contribution annuelle à chaque début d'annees
        params:
            income : salaire initial
            r_income : rendement annuel du salaire
            r_contrib : taux de contribution
            vect_temps : dates
    '''
    result = np.full(shape = vect_temps.shape, fill_value = 0.)
    bond = 1.
    result[0] = income * r_contrib * bond
    for i in np.arange(1,result.shape[0]):
        if (vect_temps[i] == np.int(vect_temps[i])):
            #pour chaque annee

            bond = bond * (1.+r_income)
            result[i] = income * bond * r_contrib

    return result
```

:
si on affiche notre vecteur contribution en fonction du temps :
le ..**shape** renvoie juste la taille du vecteur et le **np.full** crée un vecteur de taille **shape**



Si on somme toutes les contributions cela donne **393 316$**, il serait normal de vouloir que la valeur du portefeuille finale avec les investissements soit supérieure à cette valeur.

NB: notre vecteur temps possède 480 valeurs entre 0 et 40, car il y a 12 mois dans une année et chaque valeur qui appartient à N correspond au début de l'année.(38 correspond au mois de janvier de la 38eme année et 38.08 au 2eme mois)

Le brownien géométrique est défini par **dr = u dt + v * sqrt(delta_t) * N(0,1)**

on cherche à calculer de rendement "prévu" pour un actif risqué et non risqué on implémente donc une fonction **brownien géométrique**

```python
def brownien_taux_instantane(mu,vol,temps):
    '''
        Fonction qui renvoie : dr = u dt + v * sqrt(delta_t) * N(0,1)
        params :
            mu : taux de rendement (annuelle)
            v  : volatilité (annuelle)
            delta_t : periode (annuelle)
            taille : taille de la trajectoire à simuler
        returns :
            le rendement instantane
    '''
    taille = temps.shape[0]
    delta_t = np.diff(temps, prepend = 0.)

    result = mu * delta_t + vol * np.sqrt(delta_t) * np.random.normal(0.,1., taille)
    result [0] = 0.

    return result
```

le np.diff renvoie la valeur de temps[i+1]-temps[i]

buy and hold : (coefficient constants)

```python
def calcul_allocation_constante(r_highs,r_lows, a_high, a_low, contributions, npv):
    '''
        Calcul de la valeur de l'investisseme pour une allocation constante
        params :
            r_highs : vecteur de rendement de l'actif risqué
            r_lows  : vecteur de rendement de l'actif non risqué
            a_high : allocatoin sur l'actif risqué
            a_low : allocatoin sur l'actif non risqué
            contributions : contributions sur chaque période
            npv : valeur du portefeuille initial (contribution initiale)
    '''
    r_port = a_high * r_highs[0] + a_low * r_lows[0]
    npv_next = npv * (1 + r_port) + contributions[0]
    for i in np.arange(1, r_highs.shape[0]):
        #les coeffs sont constant
        r_port = a_high * r_highs[i] + a_low * r_lows[0]
        npv_next = npv * (1.+r_port) + contributions[i]
        npv = npv_next
    return npv
```

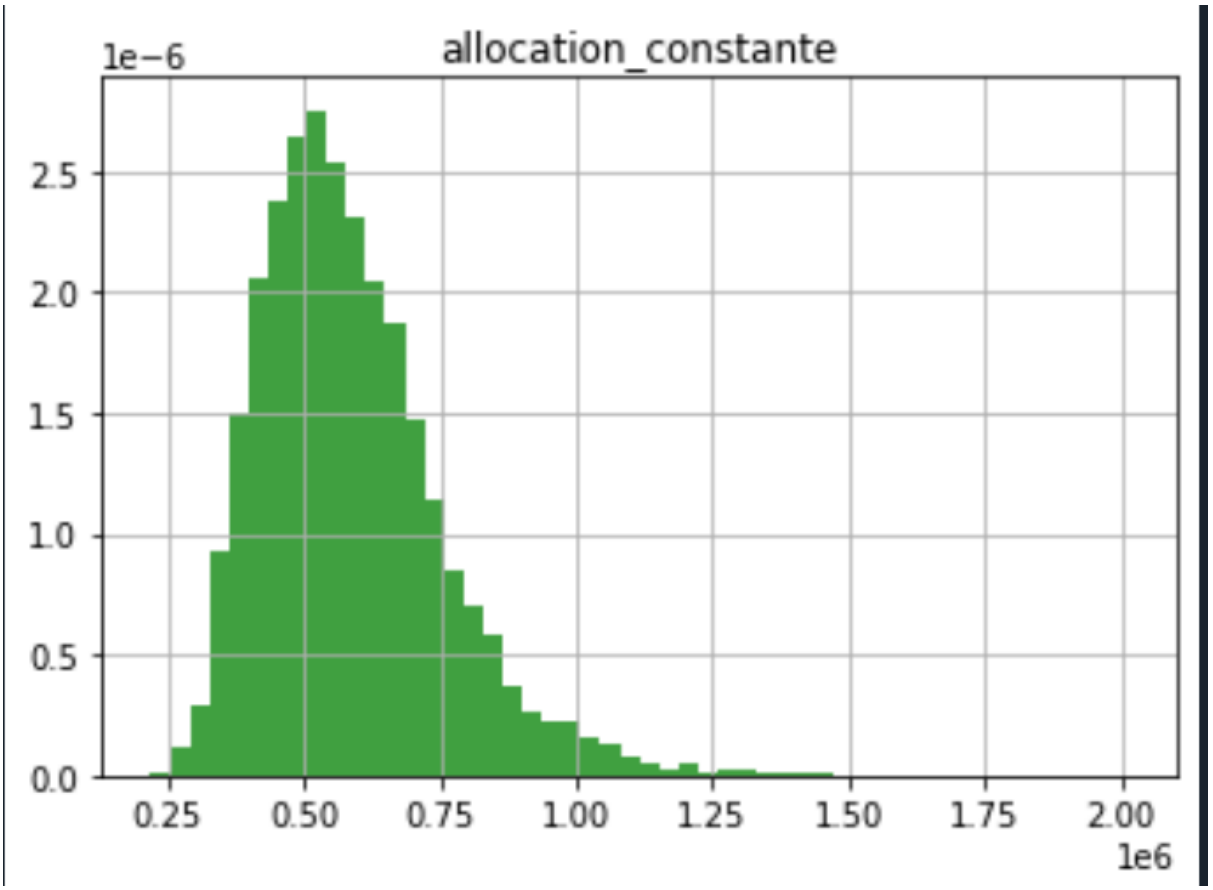
pour 10000 vecteurs rendement de l'actif risqué et rendement de l'actif non risqué :
avec 0.6 pour l'actif sans risque et 0.4 pour l'actif risqué, on trouve :

médiane : 560 017$

Moyenne : 620 290$** **Percentile_0.05 : 486 077$

Ecart_type : 84 545 $** **Percentile_0.95 : 729 920$

histogramme des valeurs de portefeuille finales pour le buy and hold :



rebalancing (coefficients variables, temporels):

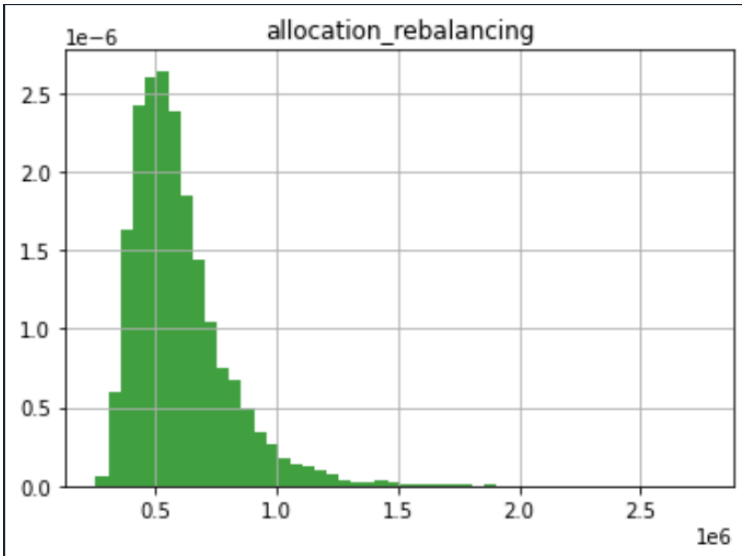
```python
def calcul_allocation_temporelle(r_highs,r_lows, a_high, a_low, temps, contributions, npv):
    '''
        Calcul de la valeur de l'investisseme pour une allocation en fonction du temps
        params :
            r_highs : vecteur de rendement de l'actif risqué
            r_lows  : vecteur de rendement de l'actif non risqué
            a_high : allocatoin sur l'actif risqué
            a_low : allocatoin sur l'actif non risqué
            temps : liste des années
            contributions : contributions sur chaque période
            npv : valeur du portefeuille initial (contribution initiale)
    '''
    r_port = a_high * r_highs[0] + a_low * r_lows[0]
    npv_next = npv * (1 + r_port) + contributions[0]
    for i in np.arange(1, r_highs.shape[0]):
        r_port = a_high * r_highs[i] + a_low * r_lows[0]
        npv_next = npv * (1.+r_port) + contributions[i]
        npv = npv_next
#
#     Mise a jour des coefficients
#
        if (temps[i] == np.int(temps[i])): #permet juste de prendre le premier jour des années
            if (temps[i]<20):
                a_high=0.8
                a_low=0.2
            if(20<=temps[i]<30):
                a_high=0.5
                a_low=0.5
            if(temps[i]>=30):
                a_high=0.15
                a_low=0.85


    return npv
```

mediane 557 606$

Moyenne :599 008$** **Percentile_0.05 : 369 849$

Ecart_type : 200 790 $** **Percentile_0.95 : 967 204$



Cppi :

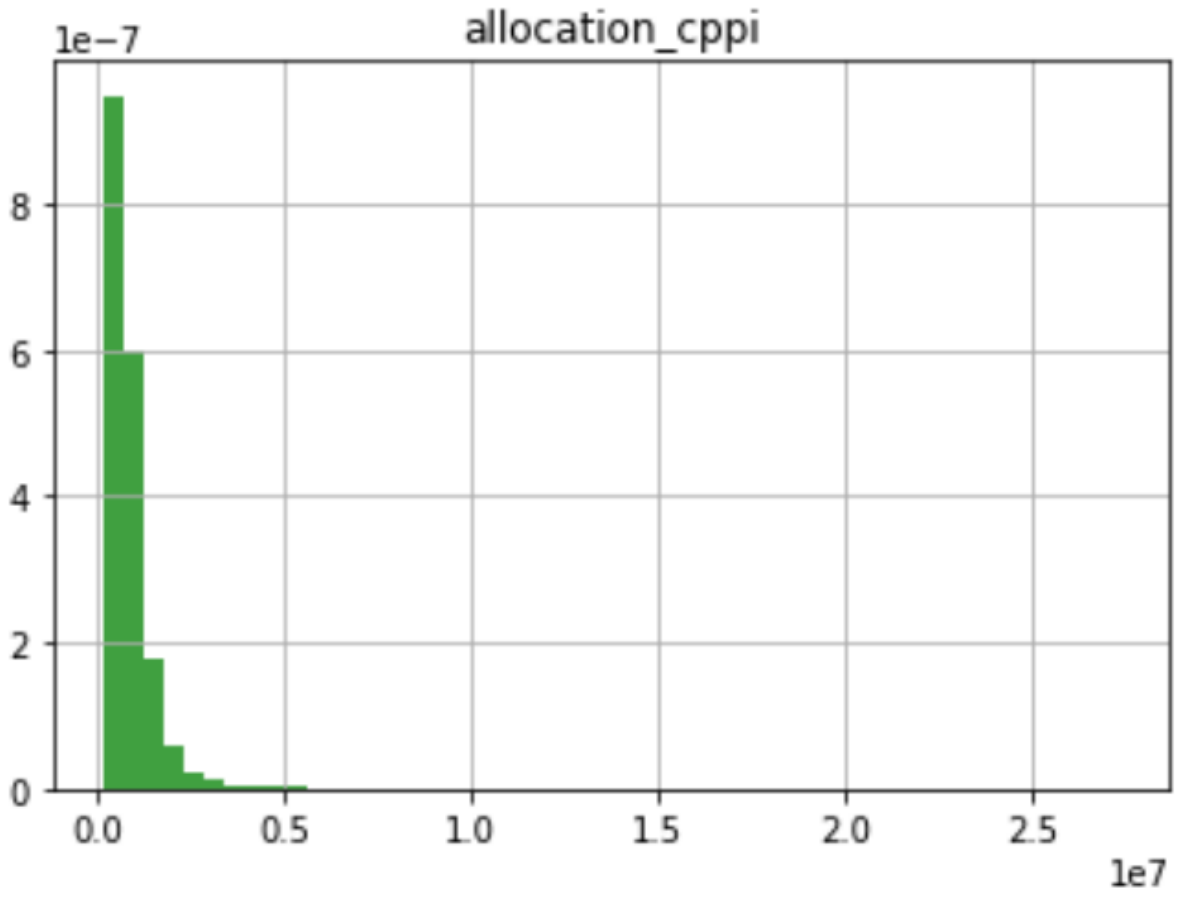
```
def calcul_allocation_cppi(r_highs,r_lows, m_high, pr_floor, contributions, temps, npv):
    '''
        Calcul de la valeur de l'investissement pour une allocation cppi
        params :
            r_highs : vecteur de rendement de l'actif risqué
            r_lows  : vecteur de rendement de l'actif non risqué
            m : multiplicateur d'allocation sur l'actif risque
            pr_floor : niveau du floor lorsque : il sera mis à jour à chaque année
            temps : vecteur de temps
            contributions : contributions sur chaque période
            npv : valeur du portefeuille initial (contribution initiale)
    '''
#
# Calcul du coussin et des allocations actif risqué et actif sans risque
#
    npv_next = npv + contributions[0]
    npv = npv_next
    floor = npv * pr_floor
    coussin = npv - floor
    a_high = min(max(0,coussin / npv * m_high),1)
    a_low = 1. - a_high
```

```
    for i in np.arange(1, r_highs.shape[0]):
        r_port = a_high * r_highs[i] + a_low * r_lows[0]
        npv_next = npv * (1.+r_port) + contributions[i]
        npv = npv_next
#
# Mise a jour du floor (uniquement annuellement)
#
        if (temps[i] == np.int(temps[i])):
            floor_new = npv * pr_floor
            if (floor_new > floor) :
                floor = floor_new
#
# Mise a jour des nouvelles allocations : il faut les calculer après de calculer
#le rendement du portefeuille pour la date suivante
# Sinon pas causal
#
        coussin = npv - floor
        a_high = min(max(0,coussin / npv * m_high),1)
        a_low = 1. - a_high


    return npv
```

Moyenne :852 760$
Ecart_type : 694 114 $
mediane : 676 978$

Percentile_0.05 : 318 618$
Percentile_0.95 : 1 944 782$



Les 3 stratégies de portefeuilles sont intéressantes chacune propose des avantages et inconvénients, le **buy and hold** paraît être la stratégie la plus “safe” , pas très risquée vu que l'écart type est faible ,même que le percentile à 5% est supérieur à la somme des contributions, donc avec le buy and hold il semble que l’on soit quasi sur d’avoir un retour sur investissement positif.

le **rebalancing** est une stratégie correcte, qui peut faire gagner un peu plus que le buy and hold mais qui peut s'avérer perdante dans 5/10 % des cas, néanmoins cette stratégie a l'avantage d'être risquée lorsque l’on est jeune et moins risqué lorsque l’on s’approche de la retraite, donc au final, le risque est moindre car durant les premières années la valeur du portefeuille est plus faible donc on parie risqué sur moins d’argent alors que plus vieux les contributions sont plus grandes ainsi que la valeur de portefeuille, et cette fois ci on prend moins de risque.

le cppi est le plus risqué (gros écart type), Selon les goûts le cppi peut être plus attrayant que le buy and hold, le jeu peut rapporter gros , et les pertes ont l’air moindre, il suffit de regarder la médiane. qui est supérieure pour le cppi, et même si

le percentile a 5% est assez faible 310 000$ contre 480 000$ pour le buy and hold, celui à 95% est monstrueux.

La stratégie que prendra le client dépend de sa façon de penser
si le client pense plutôt **au pire des cas,** il choisira le **buy and hold**
si le client pense plutôt **au meilleur des cas** , il choisira le **cppi**

et entre les 2 il y a le rebalancing.