Estimation du coefficient de Hurst

Niaussat Victor

Résumé

Un signal y à temps continu est dit présenter des propriétés d ?auto-similarité si et seulement si il existe H, tel que :

$$y(Nt) \stackrel{\mathcal{L}}{=} N^H y(t) \quad \text{(égalité en loi)}$$

avec H le coefficient de Hurst. Le coefficient de Hurst est le paramètre que nous étudierons ici et nous travaillerons sur une des premières méthodes d'estimation statistique mise au point pour évaluer ce coefficient : il s'agit de la méthode de l'Analyse de la plage redimensionnée (R/S).

1 Coefficient de Hurst

L ?exposant de Hurst, classiquement noté H , permet d ?évaluer la dépendance à long terme (long range dépendance, LRD) sur les signaux et séries temporelles. Ce concept peut se rattacher à la mémoire d ?un processus. Un processus est dit à mémoire longue avec un paramètre D s ?il est stationnaire au sens large et si sa fonction d ?autocovariance n ?est pas sommable. Le paramètre D est alors appelé le paramètre de mémoire. Il est lié au coefficient de Hurst par la relation :

$$D = H - \frac{1}{2}$$

De ce fait, un processus est dit :

- mémoire longue, ou présenter une LRD, si $0.5 < H < 1$;
- à mémoire courte si $H = 0.5$;
- à mémoire négative, ou anti-persistant, si $0 < H < 0.5$.

De plus, H vaut respectivement $-0.5, 0$ et 0.5 pour un bruit blanc, un bruit rose et un bruit brownien. Estimer H peut s ?avérer pertinent pour la classification de signaux de différents types, notamment physiologiques.

2 Estimation du coefficient de Hurst par la méthode de l'Analyse de la plage redimensionnée (R/S)

Un certain nombre d'estimateurs de la dépendance à long terme ont été proposés dans la littérature. La plus ancienne et la plus connue est l'analyse dite de plage redimensionnée (R/S) popularisée par Mandelbrot et Wallis que nous allons étudier ici. Elle est basée sur les découvertes hydrologiques précédentes de Hurst en 1951 [1].

L'algorithme se présente comme suivant : Il faut d'abord estimer la dépendance de la plage remise à l' échelle sur la durée n d'observation. Une série chronologique de pleine longueur N est divisée en un certain nombre de séries chronologiques plus courtes de longueur $n = N, N/2, N/4, ...$ La plage moyenne remise à l'échelle est ensuite calculée pour chaque valeur de n. Ensuite :

(1) Calculez la moyenne :

$$m = \frac{1}{n} \sum_{i=1}^{n} X_i$$

(2) Créer une série ajustée à la moyenne :

$$Y_t = X_t - m \quad \forall t = 1, 2, \ldots, n\,.$$

(3) Calculer la série d'écarts cumulés Z :

$$Z_t = \sum_{i=1}^{t} Y_i \quad \forall t = 1, 2, \ldots, n\,.$$

(4) Calculez la plage R :

$$R(n) = max(Z_1, Z_2, \ldots, Z_n) - min(Z_1, Z_2, \ldots, Z_n).$$

(5) Calculer l' écart type S ;

$$S(n) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(X_i - m)^2}.$$

(6). Calculez la plage remise à l'échelle $R(n)/S(n)$ et on moyenne sur toutes les séries temporelles partielles de n. En effet, l'exposant de Hurst est la puissance H dans $\mathbb{E}[R(n)/S(n)] = Cn^H$.

H peut alors s'estimer en faisant un régression linéaire sur le logarithme de la moyenne des $R(n)/S(n)$ et en cherchant le coefficient directeur par rapport à $log(n)$:

$$log(\mathbb{E}[R(n)/S(n)]) = log(c) + Hlog(n)$$

Lo (1991) préconise d'ajuster l'écart-type S pour l'augmentation attendue de la portée R résultant de l' autocorrélation à courte portée dans la série chronologique [2]. Il s'agit de remplacer S par $\hat{S}$, qui est la racine carrée de

$$\hat{S}^2 = S^2 + 2\sum_{j=1}^{q}\left(1 - \frac{j}{q+1}\right)C(j)$$

où q est un décalage maximal sur lequel l'autocorrélation à courte portée pourrait être substantielle et $C(j)$ est l' autocovariance de l'échantillon au décalage j.

3 Implémentation en Python

Pour illustrer cela, nous allons implémenter cela en Python. Je me suis aidé des codes implémentés en Python par Mottl sur Github [3].
J'ai tracé trois analyse R/S pour un mouvement brownien, un signal persistant et un signal anti persistant.

3.1 Analyse RS du Brownien

Le mouvement brownien est un signal à mémoire courte. Les valeurs futures sont indépendant des précédentes et on s'attend à avoir un coefficient de Hurst qui vaut $H = 0.5$.

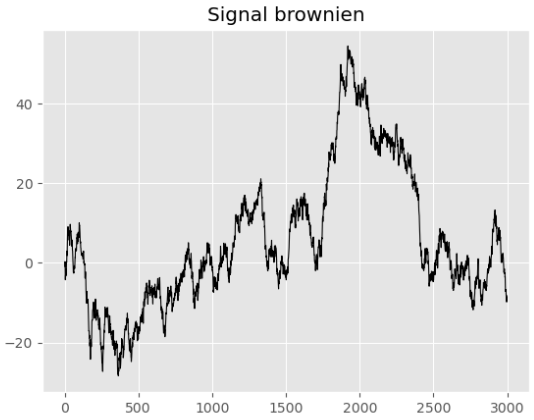

FIGURE 1 – Signal Brownien

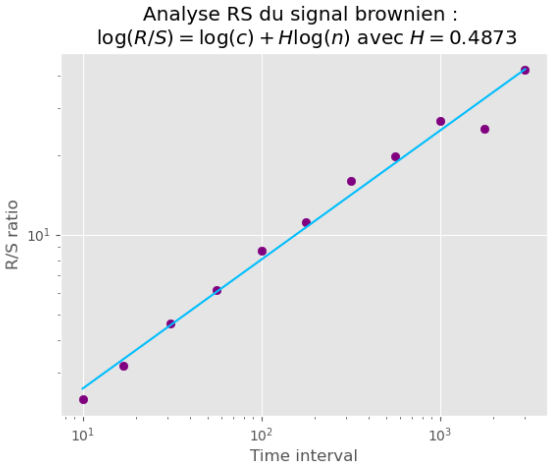

FIGURE 2 – Analyse RS du Brownien

3.2 Analyse RS d'un signal persistant

Un signal persistant est un signal à mémoire longue. Il a tendance à conserver la même "tendance" par rapport aux valeurs qui le précède. On s'attend à avoir un coefficent de Hurst $0.5 < H < 1$.

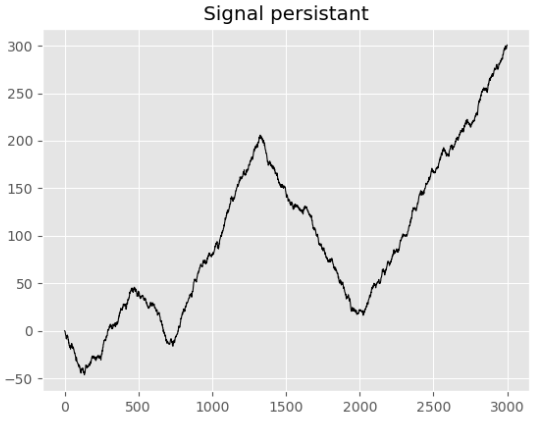

FIGURE 3 – Signal persistant

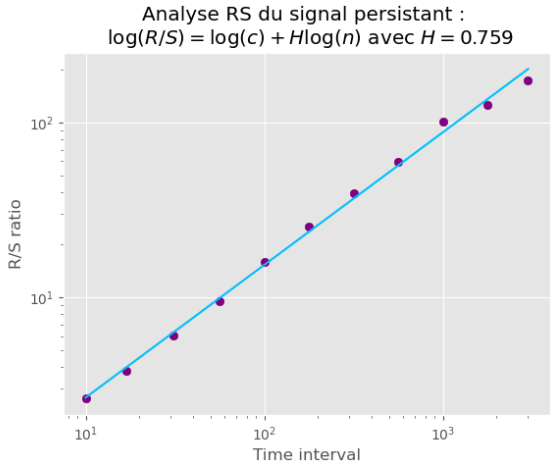

FIGURE 4 – Analyse RS d'un signal persistant

3.3 Analyse RS d'un signal anti persistant

Un signal persistant est un signal à mémoire négative. Il a tendance à inverser sa "tendance" par rapport aux valeurs qui le précède. On s'attend à avoir un coefficent de Hurst $0 < H < 0.5$.

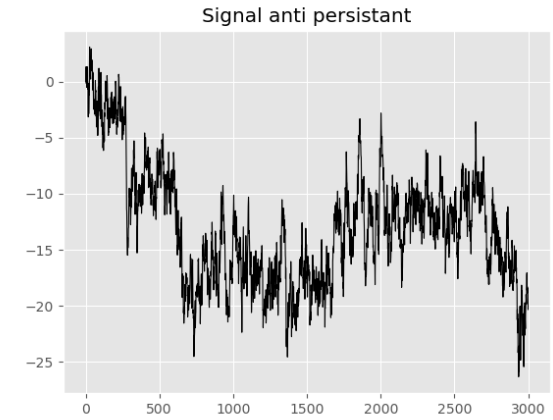

FIGURE 5 – Signal Brownien

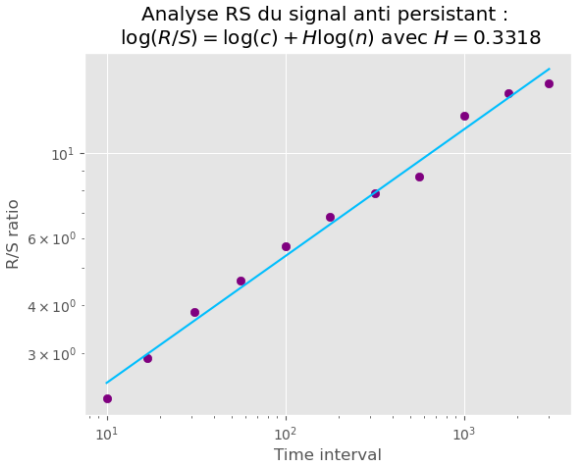

FIGURE 6 – Analyse RS d'un signal anti persistant

4 Code

```
import matplotlib.pyplot as plt
import random as rd
import numpy as np
import math

def __to_inc(x):
    incs = x[1:] - x[:-1]
    return incs

def __to_pct(x):
    pcts = x[1:] / x[:-1] - 1.
    return pcts

def __get_simplified_RS(series):

    incs = __to_inc(series)
    R = max(series) - min(series)  # range in absolute values
    S = np.std(incs, ddof=1)
    if R == 0 or S == 0:
        return 0  # return 0 to skip this interval due undefined R/S
    return(R,S)


def compute_Hc(series,min_window=10, max_window=None,):
    RS_func = __get_simplified_RS
    err = np.geterr()
    np.seterr(all='raise')

    max_window = max_window or len(series)-1
    window_sizes = list(map(
        lambda x: int(10**x),
        np.arange(math.log10(min_window), math.log10(max_window), 0.25)))
    window_sizes.append(len(series))

    RS = []
    for w in window_sizes:
        rs = []
        for start in range(0, len(series), w):
            if (start+w)>len(series):
                break
            _ = RS_func(series[start:start+w])
            if _ != 0:
                rs.append(_)
        RS.append(np.mean(rs))

    A = np.vstack([np.log10(window_sizes), np.ones(len(RS))]).T
    H, c = np.linalg.lstsq(A, np.log10(RS), rcond=-1)[0]
    np.seterr(**err)

    c = 10**c
    return H, c , [window_sizes,RS]

def random_walk(length, proba=0.5, min_lookback=1, max_lookback=100, cumprod=False):

    assert(min_lookback>=1)
    assert(max_lookback>=min_lookback)

    if max_lookback > length:
        max_lookback = length
        warnings.warn("max_lookback parameter has been set to the length of the random walk
    series.")

    if not cumprod:  # ordinary increments
        series = [0.] * length  # array of prices
        for i in range(1, length):
            if i < min_lookback + 1:
                direction = np.sign(np.random.randn())
            else:
                lookback = np.random.randint(min_lookback, min(i-1, max_lookback)+1)
                direction = np.sign(series[i-1] - series[i-1-lookback]) * np.sign(proba - np.
    random.uniform())
            series[i] = series[i-1] + np.fabs(np.random.randn()) * direction
    else:  # percent changes
        series = [1.] * length  # array of prices
```

```
        for i in range(1, length):
            if i < min_lookback + 1:
                direction = np.sign(np.random.randn())
            else:
                lookback = np.random.randint(min_lookback, min(i-1, max_lookback)+1)
                direction = np.sign(series[i-1] / series[i-1-lookback] - 1.) * np.sign(proba - np.random.uniform())
            series[i] = series[i-1] * np.fabs(1 + np.random.randn()/1000. * direction)

    return np.array(series)


def gen_MB(n):
    return(np.cumsum(np.random.normal(0,1,size=n)))

dic = {"brownien":0.5,"persistant":0.7,"anti persistant":0.3}
for k in ["brownien","persistant","anti persistant"] :
    Signal = random_walk(3000,proba=dic[k])
    H,c,data=compute_Hc(Signal)


    # Plot
    plt.style.use("ggplot")
    f, ax = plt.subplots()
    ax.plot(data[0], c*data[0]**H, color="deepskyblue")
    ax.scatter(data[0], data[1], color="purple")
    ax.set_xscale('log')
    ax.set_yscale('log')
    ax.set_xlabel('Time interval')
    ax.set_ylabel('R/S ratio')
    ax.set_title(f"Analyse RS du signal {k} : \n $\log(R/S) = \log(c) + H\log(n)$ avec $H=${round(H,4)}")
    plt.savefig(f"RS_{k}.png")

    plt.figure()
    plt.plot(Signal,linewidth=0.8,color="black")
    plt.title(f"Signal {k}")
    plt.savefig(f"Signal_{k}.png")
    print("H={:.4f}, c={:.4f}".format(H,c))
```

5 Conclusion

Ainsi, la méthode d'analyse de la plage redimensionnée (R/S) permet d'estimer le coefficient de Hurst. C'est une méthode moyennement efficace. Il existe de nouvelles méthodes se basant sur le DFA (*Detrended Fluctuation Analysis*), introduite par Peng en 1991 [4] qui sont plus performantes. Ce sont des estimateurs qui sont encore aujourd'hui sujet à de nombreuses recherches.

On a plus d'informations autour des résultats de convergence de la méthode [5].

Références

[1] Harold Edwin Hurst. « Long-term storage capacity of reservoirs ». In : *Transactions of the American society of civil engineers* 116.1 (1951), p. 770-799.

[2] Andrew W Lo. « Long-term memory in stock market prices ». In : *Econometrica : Journal of the Econometric Society* (1991), p. 1279-1313.

[3] Dmitry Mottl. *Hurst exponent evaluation and R/S-analysis*. 2018. URL : https://github.com/Mottl/hurst.

[4] C.-K. Peng et al. « Mosaic organization of DNA nucleotides ». In : *Phys. Rev. E* 49 (2 fév. 1994), p. 1685-1689. DOI : 10.1103/PhysRevE.49.1685. URL : https://link.aps.org/doi/10.1103/PhysRevE.49.1685.

[5] David M. Mason. « The Hurst phenomenon and the rescaled range statistic ». In : *Stochastic Processes and their Applications* 126.12 (2016). In Memoriam : Evarist Giné, p. 3790-3807. ISSN : 0304-4149. DOI : https://doi.org/10.1016/j.spa.2016.04.008. URL : https://www.sciencedirect.com/science/article/pii/S0304414916300308.