Processo de difusão com Saltos

Francis Araujo

June 13, 2021

impa
Instituto de Matemática
Pura e Aplicada

Empiricamente, os retornos de ativos arriscados apresentam caudas mais pesadas em relação à distribuição normal (excesso de curtose), hipótese adotada pelo modelo de Black & Scholes. Merton (1976) introduz o conceito de processo de difusão com saltos para explicar tais fenômenos.

Dinâmica do preço do ativo subjacente:

$$dX_t = (\mu - \delta)X_{t^-}dt + \sigma X_{t^-}dB_t + (J_t - 1)\, X_{t^-}dN_t \tag{1}$$

Sob a medida Q, a dinâmica de X_t é determinada por:

$$dX_t = [r - \delta - \lambda(\bar{J} - 1)]X_{t^-}dt + \sigma X_{t^-}d\tilde{B}_t + (J_t - 1)\, X_{t^-}dN_t \tag{2}$$

Onde:
(I) B_t é Browniano padrão
(II) $\bar{J} = E^P\,(J_t) = \exp\left(\mu_J + \frac{1}{2}\sigma_J^2\right)$
(III) $\tilde{B}_t = B_t + \theta t \quad \text{e} \quad \theta = \frac{\mu - r + \lambda(\bar{J} - 1)}{\sigma}$
(IV) $N_t \sim \mathrm{Poi}(\lambda t)$
(V) $J_i \sim \log N\left(\mu_J, \sigma_J^2\right), \quad i = 1, 2, \ldots, N_{T-t}$

Defina $y_t = log(X_t); \;\; Z_1 \sim N(0,1); \;\; Z_2 \sim N(\mu_j, \sigma_j^2); \; n_t = N_{(t+\Delta t)-t}$

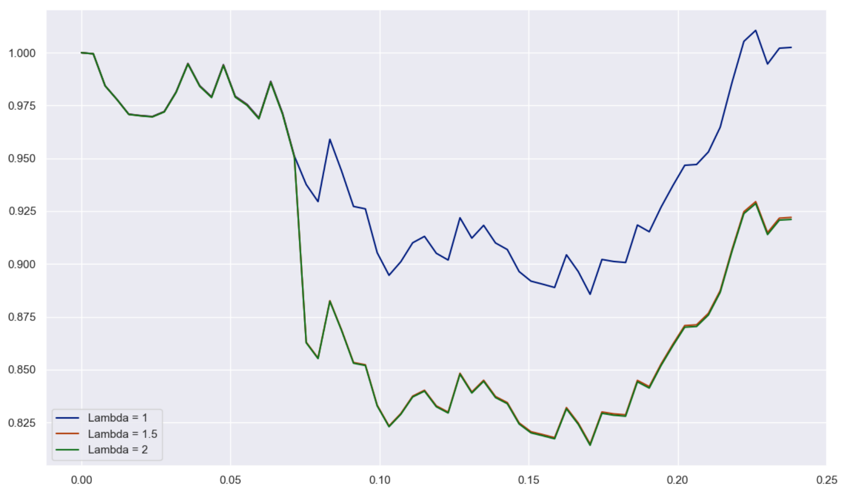
$$
\begin{aligned}
dX_t &= [r - \delta - \lambda(\bar{J} - 1)]X_{t^-}dt + \sigma X_{t^-}d\tilde{B}_t + (J_t - 1)\,X_{t^-}dN_t \\
\frac{dX_t}{X_{t^-}} &= [r - \delta - \lambda(\bar{J} - 1)]dt + \sigma d\tilde{B}_t + (J_t - 1)\,dN_t \\
d(log(X_t)) &= [r - \delta - \lambda(\bar{J} - 1)]dt + \sigma d\tilde{B}_t + (J_t - 1)\,dN_t \\
&= [r - \delta - \lambda(\bar{J} - 1) - \frac{1}{2}\sigma^2]dt + \sigma d\tilde{B}_t + log(J_t)\cdot dN_t \text{ (Expansão de Taylor)} \\
&= [r - \delta - \lambda(\bar{J} - 1) - \frac{1}{2}\sigma^2]dt + \sigma d\tilde{B}_t + \sum_{i=1}^{n_t} log(J_t)
\end{aligned}
$$

Portanto,

$$
y_{t+\Delta t} = y_t + [r - \delta - \lambda(\bar{J} - 1) - \frac{1}{2}\sigma^2]\Delta t \; + \sigma\sqrt{\Delta t}\cdot Z_1 + \sum_{i=1}^{n_t} Z_2
$$

Presentation Outline

Fixando as variáveis aleatórias através da função seed. As variações no parâmetro lambda resultam:



Intuitivamente, quanto maior o lambda, maior será a frequência dos saltos.

Média = 0.95; Variância = 0.9%. Gráfico com 5 trajetórias:

Conforme aumento do número de simulações, ocorre redução de variância e convergência da estimação. Características dos valores finais (X_T):

Sample Size	***Average***	***Variance***	***Lower Bound***	***Upper Bound***
5	0.95675	0.00958	0.92013	1.00337
500	1.00922	0.01113	1.00521	1.01485
5000	1.00914	0.01026	1.00792	1.01083
10000	1.00693	0.01061	1.00605	1.00815

Note que os valores finais seguem assintoticamente a distribuição lognormal. Esse fato foi utilizado para construção dos intervalos de confiança.

Seja $N_{T-t} = n_t$. Pela equação (3) do enunciado:

$$X_T = X_t \exp\left[\left(r - \delta - \lambda(\bar{J} - 1) - \frac{1}{2}\sigma^2\right)(T - t) + \sigma \tilde{B}_{T-t}\right] \prod_{i=1}^{N_{T-t}} J_i$$

Isto implica que,

$$log(X_T) = log(X_t) + \left(r - \delta - \lambda(\bar{J} - 1) - \frac{1}{2}\sigma^2\right)(T - t) + \sigma \tilde{B}_{T-t} + \sum_{i=1}^{N_{T-t}} log(J_i)$$

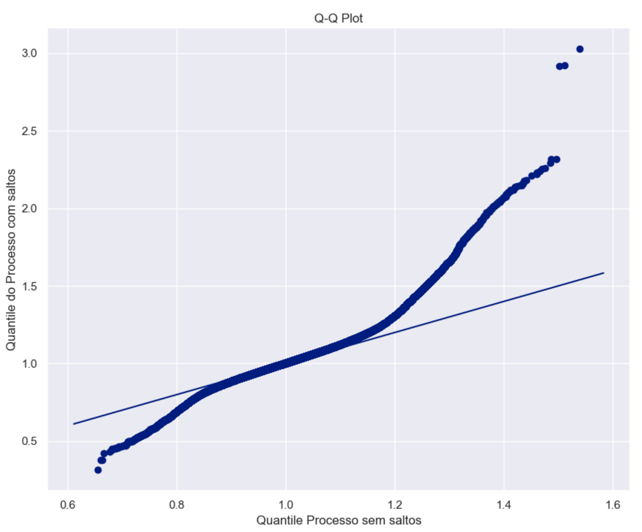
A variância condicional do processo:

$$Var[log(X_T)|X_t] = \sigma(T - t) + n_t \cdot \sigma_j^2$$

A variância do processo é amplificada ao adicionar o componente dos saltos.

Comparando Distribuições (Q-Q Plot)

A distribuição de processos com saltos possui ambas caudas mais pesadas. A dinâmica dos saltos aumenta a probabilidade de eventos extremos.



Conclusões

- Evidência Empírica: Observações extremas nas séries de preços
- Elevado custo computacional
- Método de Monte Carlo permite determinar os intervalos de confiança

Presentation Outline

- Aiube, F. (2013), Modelos Quantitativos em Finanças: Com Enfoque em Commodities.
- Merton (1976), Option pricing when underlying stock returns are discontinuous

FIM

```python
In [ ]: import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

def merton_model(X_0, T, r,delta, sigma, Lambda, mu_j, sigma_j, Time_steps, simulations,seed):

    # Define semente aleatória como parâmetro
    np.random.seed(seed)

    # Computa o tamanho do intervalo de tempo
    dt = T / Time_steps

    # Define a matriz de preços
    price_path = np.zeros([simulations, Time_steps+1])

    # Substitui a primeira coluna pelo vetor de preços iniciais, em versão log
    price_path[:,0] = np.log(X_0)

    # Define variáveis aleatórias
    N1 = np.random.normal(size=[simulations, Time_steps])
    N2 = np.random.normal(mu_j, sigma_j, size=[simulations, Time_steps])
    N_poisson = np.random.poisson(Lambda * dt, size=[simulations, Time_steps])

    # Define parâmetros compostos a serem utilizados
    J_bar = np.exp((mu_j - ((sigma_j ** 2)/ 2)))

    # Define a dinâmica de movimento do preço (formato logaritmo); preenche cada elemento da matriz após t = 0;
    for i in range(Time_steps):

        price_path[:, i + 1] = price_path[:, i] + ((r - delta - (Lambda * (J_bar - 1)) - (0.5 * sigma**2)) * (dt)) +
(sigma * np.sqrt(dt) * (N1[:, i])) + N_poisson[:,i]*N2[:,i]

    # Aplica exponencial na matriz
    exp_price_path = np.exp(price_path)

    # Delimita os valores para os preços finais
    last_price = exp_price_path[:, -1]

    # Computa a média para simulação de MC
    mean_process = np.mean(last_price)

    # Computa a variância da simulação de MC
    std_hat = np.std(last_price)

    variance_process = std_hat**2

    # Calcula o intervalo de confiança (alpha = 5%)

    CI_upper = mean_process + (np.quantile(last_price,0.95) * std_hat) / np.sqrt(simulations)

    CI_lower = mean_process - (np.quantile(last_price,0.05) * std_hat) / np.sqrt(simulations)

    # Divide o eixo x
    time_grid = np.linspace(0, T, Time_steps + 1)

    return time_grid,exp_price_path, last_price, mean_process, variance_process ,CI_lower,CI_upper,N_poisson

#### Sensibilidade -  Lambda ###

lambda_case1 = merton_model(X_0=1, T= 60 / 252, r=0.035, delta=0.01, sigma=0.2, Lambda=1, mu_j=0.01, sigma_j=0.05, T
ime_steps=60, simulations=1, seed=215)

lambda_case2 = merton_model(X_0=1, T=60 / 252, r=0.035, delta=0.01, sigma=0.2, Lambda=1.5, mu_j=0.01, sigma_j=0.05,
Time_steps=60, simulations=1, seed=215)

lambda_case3 = merton_model(X_0=1, T=60 / 252, r=0.035, delta=0.01, sigma=0.2, Lambda=2, mu_j=0.01, sigma_j=0.05, Ti
me_steps=60, simulations=1, seed=215)

sns.set(palette='dark')
plt.figure(figsize=(10, 8))
ax = plt.axes()
ax.plot(lambda_case1[0], lambda_case1[1].transpose(), label='Lambda = 1')
ax.plot(lambda_case2[0], lambda_case2[1].transpose(), label='Lambda = 1.5')
ax.plot(lambda_case3[0], lambda_case3[1].transpose(), label='Lambda = 2')
ax.set_xlabel('# Years')
ax.set_ylabel('Price')
plt.legend()
plt.show()

########################################################################################################

#### Sensibilidade -  Volatilidade Salto ########

volatility_case1 = merton_model(X_0=1, T= 60 / 252,r=0.035, delta=0.01, sigma=0.2, Lambda=1, mu_j=0.01, sigma_j=0.05
, Time_steps=60, simulations=1, seed=219)

volatility_case2 = merton_model(X_0=1, T= 60 / 252, r=0.035, delta=0.01, sigma=0.2, Lambda=1, mu_j=0.01, sigma_j=0.5
, Time_steps=60, simulations=1, seed=219)

sns.set(palette='dark')
plt.figure(figsize=(10, 8))
az = plt.axes()
az.plot(volatility_case1[0], volatility_case1[1].transpose(), label='Volatilidade 5%')
az.plot(volatility_case2[0], volatility_case2[1].transpose(), label='Volatilidade 50%' )
plt.legend()
plt.show()


########################################################################################################

#### Momentos do Processo com Salto - QUESTÁO 2 ########


### Caso com 5 simulações

momentos_case1 = merton_model(X_0=1, T= 60 / 252, r=0.035, delta=0.01, sigma=0.2, Lambda=1, mu_j=0.01, sigma_j=0.05,
Time_steps=60, simulations=5, seed=215)

mean = momentos_case1[3]

variance = momentos_case1[4]

lower_bound =  momentos_case1[5]

upper_bound = momentos_case1[6]

sns.set(palette='dark')
plt.figure(figsize=(10, 8))
mc = plt.axes()
mc.plot(momentos_case1[0], momentos_case1[1].transpose())
mc.set_xlabel('# Years')
mc.set_ylabel('Price')
plt.show()

print('Média com 5 simulações: ' + str(mean))

print('Variância com 5 simulações: ' + str(variance))

print('Intervalo inferior com 5 simulações: ' + str(lower_bound))

print('Intervalo superior com 5 simulações: ' + str(upper_bound))



### Caso com 10.000 simulações

momentos_case2 = merton_model(X_0=1, T= 60 / 252, r=0.035, delta=0.01, sigma=0.2, Lambda=1, mu_j=0.01, sigma_j=0.05,
Time_steps=60, simulations=10000, seed=None)

mean_c2 = momentos_case2[3]

variance_c2 = momentos_case2[4]

lower_bound_c2 =  momentos_case2[5]

upper_bound_c2 = momentos_case2[6]


print('Média com 10000 simulações: ' + str(mean_c2))

print('Variância com 10000 simulações: ' + str(variance_c2))

print('Intervalo inferior com 10000 simulações: ' + str(lower_bound_c2))

print('Intervalo superior com 10000 simulações: ' + str(upper_bound_c2))


###### Casos Extras

## 500 simulacões

momentos_case3 = merton_model(X_0=1, T= 60 / 252, r=0.035, delta=0.01, sigma=0.2, Lambda=1, mu_j=0.01, sigma_j=0.05,
Time_steps=60, simulations=500, seed=None)

mean_c3 = momentos_case3[3]

variance_c3 = momentos_case3[4]

lower_bound_c3 =  momentos_case3[5]

upper_bound_c3 = momentos_case3[6]

print('Média com 500 simulações: ' + str(mean_c3))

print('Variância com 500 simulações: ' + str(variance_c3))

print('Intervalo inferior com 500 simulações: ' + str(lower_bound_c3))

print('Intervalo superior com 500 simulações: ' + str(upper_bound_c3))


## 5000 simulacões

momentos_case4 = merton_model(X_0=1, T= 60 / 252, r=0.035, delta=0.01, sigma=0.2, Lambda=1, mu_j=0.01, sigma_j=0.05,
Time_steps=60, simulations=5000, seed=None)

mean_c4 = momentos_case4[3]

variance_c4 = momentos_case4[4]

lower_bound_c4 =  momentos_case4[5]

upper_bound_c4 = momentos_case4[6]

print('Média com 5000 simulações: ' + str(mean_c4))

print('Variância com 5000 simulações: ' + str(variance_c4))

print('Intervalo inferior com 5000 simulações: ' + str(lower_bound_c4))

print('Intervalo superior com 5000 simulações: ' + str(upper_bound_c4))


########################################################################################################

### Processo com salto

jump_process = merton_model(X_0=1, T= 60 / 252, r=0.035, delta=0.01, sigma=0.2, Lambda=1, mu_j=0.01, sigma_j=0.2, Ti
me_steps=60, simulations=100000, seed=219)

### Processo sem salto

zero_jumps = merton_model(X_0=1, T= 60 / 252, r=0.035, delta=0.01, sigma=0.2, Lambda=1, mu_j=0, sigma_j=0, Time_step
s=60, simulations=100000, seed=219)

#### Q-Q Plot (Final Price) ###
sns.set(palette='dark')
plt.figure(figsize=(10, 8))
qq_gp = plt.axes()
qq_gp.scatter(np.sort(zero_jumps[2]), np.sort(jump_process[2]))
qq_gp.set_title('Q-Q Plot')
qq_gp.set_ylabel('Quantile do Processo com saltos')
qq_gp.set_xlabel('Quantile Processo sem saltos')
x = np.linspace(*qq_gp.get_xlim())
qq_gp.plot(x, x)
plt.show()
########################################
```