Pricing under Rough Volatility Models Lab Report

Vega Institute Foundation

Spring 2022

Abstract

In the present paper we investigate the roughness of the Russian stock market in the context of the rough stochastic volatility model. We obtain the estimation of the Hurst parameter and the Zumbach effect. We also obtain an estimation of a bias using modern approaches in Data Science and Applied Statistics. In the end we form the conclusion that the rough volatility model is a XXX working model for the Russian stock market and formulate the future research horizon.

Keywords: Rough Volatility, RFSV Model, Zumbach Effect, Hurst Parameter, Equity, MOEX

Contents

1 Introduction

2 Description of the Model

2.1 Nessessary Definitions from Stochastic Calculus

Definition 2.1. The fractional Brownian motion (fBm) $(W_t^H)_{t\in\mathbb{R}_+}$ with Hurst parameter $H \in (0,1)$ is a Gaussian process with the following properties:

1. $W_0^H = 0$,
2. $\mathbb{E}\left[W_t^H\right] \equiv 0$,
3. $\mathbb{E}\left[W_s^H W_t^H\right] = \frac{1}{2}\left(t^{2H} + s^{2H} - |t-s|^{2H}\right)$

2.2 Rough Fractional Stochastic Volatility Model

Our base article is [1]

$$dS_t = \alpha S_t dt + \sigma_t S_t dW_t, \tag{2.1}$$

$$d\log\sigma_t = \nu dW_t^H - \alpha(\log\sigma_t - m)dt \tag{2.2}$$

3 Statistical Analysis

3.1 Data Preprocessing

In our work we used two types of data: high-frequency data from a trading terminal (1m data), and candles from Yahoo Finance (1 day). We made the following preprocessing steps:

1. Created a mean price variable for each time interval ($\frac{high+low}{2}$ for HF, $\frac{open+close}{2}$ for candles)
2. Estimated the realized volatility (RV) as an standard deviation of the assets' log-returns, calculated from mean prices (40 minutes estimation for HF, 20 days for candles)

3.2 Model Parameters Estimation

4 Conclusion

5 Appendix

5.1 Estimation Code

```
def m(q, Delta, volatility_array):
    retval = 0
    N = volatility_array.size-Delta

    for i in range(0, N, Delta):
        retval += pow(abs(math.log(
                volatility_array[i+Delta]/volatility_array[i])), q)

    retval /= int(volatility_array.size/Delta)
    return retval

def ACov(volatility_array, Delta):
    retval = 0
    N = volatility_array.size - Delta

    for i in range(0, N):
        retval+= volatility_array[i]*volatility_array[i+Delta]

    retval /= N
    return math.log(retval)

def skew(x1, x2, y1, y2):
    return (y2 - y1) / (x2 - x1)

def analyse_volatility(name):
    if show_pics:
        print("Report on " + name)
    thresh = 3200
    df = pd.read_csv(name, sep="\t")
    name = name.replace(".csv", '')
    df["Mean"] = 0.5*(df["High"]+df["Low"])

    log_returns = np.zeros(df.shape[0])
    for i in range(thresh, df.shape[0]):
        log_returns[i] = math.log(df["Mean"][i]/df["Mean"][i-1])

    df["log_returns"] = log_returns
    df["rlz_var"] = df["log_returns"].rolling(minutes_count).var()
    df["rlz_vol"] = np.sqrt(df["rlz_var"])

    if show_pics:
        df["rlz_vol"].plot(title="RVol of " + name + " estimated by "
                                    + str(minutes_count) + " mins")
        plt.xlabel("Time")
        plt.ylabel("rv60")
        plt.show()
```

```
volatility_array = df["rlz_vol"][minutes_count+thresh-1:].to_numpy()
zetaq            = np.zeros((2, num_of_q))

for I in range(0, num_of_q):
    graph_data = np.zeros((2, pD-sD))
    q          = step_of_q*(1+I)
    line_start = math.log(sD)
    line_stop  = math.log(pD)

    for Delta in range(sD, pD):
        graph_data[0, Delta-sD] = math.log(Delta)
        graph_data[1, Delta-sD] = math.log(m(q, Delta, volatility_array))

    linear_model    = np.polyfit(graph_data[0],graph_data[1], 1)
    linear_model_fn = np.poly1d(linear_model)
    x_s             = np.arange(line_start, line_stop, 0.1)

    if show_pics:
        plt.plot(x_s, linear_model_fn(x_s))
        plt.scatter(graph_data[0], graph_data[1])

    skew_of_linear_model = skew(line_start,
                                line_stop,
                                linear_model_fn(line_start),
                                linear_model_fn(line_stop))
    title = name

    if show_pics:
        plt.title(title)
        plt.xlabel("$\log \Delta$")
        plt.ylabel("$\log m$")
        if not one_pic:
            title += ", q = " + str(q) + ", skew = "
                                       + str(skew_of_linear_model)
            plt.title(title)
            plt.show()

    zetaq[0, I] = q
    zetaq[1, I] = skew_of_linear_model

linear_model_H    = np.polyfit(zetaq[0], zetaq[1], 1)
linear_model_H_fn = np.poly1d(linear_model_H)
x_s               = np.arange(0, step_of_q*(num_of_q+1), step_of_q)

if show_pics:
    if one_pic:
        plt.show()
    plt.plot(x_s, linear_model_H_fn(x_s), color="red")
    plt.scatter(zetaq[0], zetaq[1])
```

```
    plt.title("$H$ parameter estimation for " + name)
    plt.xlabel("$q$")
    plt.ylabel("$\zeta_q$")
    plt.show()

H_est = skew(0,
             step_of_q*(num_of_q)+1,
             linear_model_H_fn(0),
             linear_model_H_fn(step_of_q*(num_of_q)+1))
print("Estimated H parameter for " + name + " is equal to " + str(H_est))

sz = 50
graph_data = np.zeros((2, sz))

for Delta in range(1, sz+1):
    graph_data[0, Delta-1] = Delta**(2*H_est)
    graph_data[1, Delta-1] = ACov(volatility_array, Delta)

linear_model    = np.polyfit(graph_data[0],graph_data[1], 1)
linear_model_fn = np.poly1d(linear_model)
x_s             = np.arange(1, (sz+1)**(2*H_est), 0.1)

if show_pics:
    plt.plot(x_s, linear_model_fn(x_s), color="red")
    plt.scatter(graph_data[0], graph_data[1])
    plt.xlabel("$\Delta^{ 2H }$")
    plt.ylabel("$\log E[\sigma_t \sigma_{t+\Delta}]$")
    plt.show()

for Delta in range(1, sz+1):
    graph_data[0, Delta-1] = math.log(Delta)

linear_model    = np.polyfit(graph_data[0],graph_data[1], 1)
linear_model_fn = np.poly1d(linear_model)
x_s             = np.arange(0, math.log(sz+1), 0.1)

if show_pics:
    plt.plot(x_s, linear_model_fn(x_s), color="red")
    plt.scatter(graph_data[0], graph_data[1])
    plt.xlabel("$\log \Delta$")
    plt.ylabel("$\log E[\sigma_t \sigma_{t+\Delta}]$")
    plt.show()

return H_est
```

5.2 Results for Additional Assets

References

[1] Jim Gatheral, Thibault Jaisson, and Mathieu Rosenbaum. "Volatility is rough". In: *Quantitative Finance* 18.6 (2014), pp. 933–949. arXiv: 1410.3394. URL: http://arxiv.org/abs/1410.3394.