**

**市场微观结构**

**课程论文**

|  |  |
| --- | --- |
| **题 目：** | **基于中国股票市场高频数据的波动率的研究** |
| **学生姓名：** | **蔡骁** |
| **学生学号：** | **S191801513** |
| **班 级：** | **应用经济学一班** |

**2020 年 1月 10 日**

**基于中国股票市场高频数据的波动率的研究**

1. **背景概述**

在计算机技术尚未普及时，人们对时间序列数据的研究还以日、周、月等单位的数据为主，这类数据被称为低频数据。低频数据领域也是传统的计量经济学的主要应用范围。但近年来随着计算机和网络技术的迅速发展，时间序列的采样频率越来越密集，让我们进行以时、分、秒为单位的高频数据的实证研究成为可能，有关高频时间序列的金融计量理论孕育而生，推动了金融计量经济学的发展。

在针对时间序列的波动率的研究中，若我们采用低频数据，自回归条件异方差(Autoregressive Conditional Heteroscedasticity, ARCH)模型和随机波动(Stochastic Volatility, SV)模型以及基于ARCH和SV模型的扩展形式，是我们主要的研究工具。高频数据和低频数据相比，它有以下不同的性质：（1）高频数据采样的频率较高，其包含的信息较多，较丰富。但是，（2）非正态性和厚尾。Anderson(1998)研究发现，高频资产的收益率具有十分显著的非正态性，增加采样的频率会增大数据的峰度。（3）高频金融时间序列普遍存在一阶负相关性；（4）日历效应。日历效应是指高频时间序列中的金融变量可能会有在采样频率内表现出稳定的周期性趋势。如，日内模式体现在金融变量在日内的“U”型走势——即两头高于中间的走势。（5）长记忆性。（6）由不同步交易(Asynchronous Trading)、闭市效应(Market Closing Effects)、买卖价格跳跃(Bid - Ask Bounce)等因素引起的市场微观结构噪音能导致高频数据交易价格不服从传统金融理论中资产价格为半鞅过程的假设，市场微观结构噪音对资产价格过程造成影响，使真实价格与资产有效价格产生偏差。

为了研究对高频数据的波动率进行研究，Andersen和Bollerslev提出了已实现波动率(Realized Volatility, RV)，它具有计算方便、无模型的特点。此后，还有其他学者陆续提出了已实现极差波动(RRV)、已实现双幂次变差(RBV)、双时间标度实现波动(TSRV)等波动率估计量。而在已实现波动率之后提出的各波动率估计量则是因为人们发现，在采样频率较高时，市场微观结构噪音会使得已实现波动率不是积分波动率(Integrated Volatility, IV)的无偏估计量，因此，。针对这一问题，一是可以适度地降低采样频率以降低微观结构噪音的影响。但有学者指出，实际中单位时期内得到的观察测数目是有限的，价格依然是被离散记录，存在着价格离散化。计算得到的实现波动含有新的偏差，此为抽样误差。估计量的方差会因为价格离散化的增加而增加，需要我们在估计量偏差和方差之间取舍。另一种降低市场微观结构噪音而更好度量真实的波动率的思路是，不采用上述降低采样频率的措施而采用更准确的指标来度量，如使用已实现极差(RRV), 已实现双幂次变差(RBV), 已实现极差双幂次变差(RRBV)、双时间标度实现波动(TSRV)等更为精确的估计量。

在对波动率进行度量之后，若需要对波动率进行预测，则需要对波动率进行建模。Müller（1993）提出了异质市场假说，异质市场假说认为——由于信息的不对称性，市场上的交易者可分为噪音交易者、不知情交易者和知情交易者，这三种不同类型的交易即便对同一信息的反应和预期也都不同，当异质交易者数量增大时，会造成报价的差异增大，产生巨大差异的订单流。Corsi（2004）基于异质市场假说提出了使用不同时间跨度的已实现波动率来进行建模，把短期交易者、中期交易者和长期交易者对应到不同的时间段——日、周和月，从而实现了用不同时间跨度的带滞后期的已实现收益率来刻画不同市场参与者的订单对已实现波动率的影响，此模型可以很好地刻画波动率的长记忆特征。

本文选取2016年1月3日至2016年12月30日的华谊集团（600623）五分钟的高频数据进行分析并构建HAR-RV模型。

1. **模型构建**

**2.1 市场微观噪音的度量**

假设有效对数价格过程为一个连续随机波动半鞅，即

其中，为标准布朗运动，和为漂移率和瞬时波动率。是理想市场中反映所有当前信息的市场有效价格（或称均衡价格），但现实市场并不完美，存在着各种市场微观结构摩擦，如买卖价格跳跃，信息非对称导致的价格偏离、价格离散化等等，这些因素造成的市场观测价格与有效价格产生偏差，这一偏差称为市场微观噪音。将市场微观噪音记为，观测价格记为，则：

假设微观结构噪音满足，，，且与相互独立。

则观测收益、有效收益及收益噪音的关系式为：

收益噪音具有零均值的MA(1)结构，并且具有负的一阶序列自相关，因此。真是收益率，因此可观测收益率，假设不存在自相关性，且，

**2.2 已实现波动率**

考虑一个对数价格过程，，其定义在概率空间，其中，是t时刻的信息集，于是，得到以下跳跃扩散过程：

其中，为对数价格的变动，是漂移项，是瞬时波动率，是严格为正的，样本路径是右连续、左极限存在的随机波动过程，表示标准的布朗运动，是一个描述跳跃大小的随机变量，是一个泊松过程，若跳跃发生在内，则，否则。

定义二次变差QV(Quadratic Variation)为：

其中,是第i次跳跃出现的时间。等式右边的第一项是积分波动率，积分波动率是对价格在时间区间内波动的很好的度量，这是我们需要度量的真实波动率。但是，想得到它并不简单。根据伊藤引理，在[t-1,t]时间内，对数价格

于是，

其中，为对数价格收益率。Barndorff-Nielsen and Shephard (2002a,b) 提出了已实现波动率，我们在一天内对价格取样M次,每日的已实现波动率为：

当取样频率持续增加，

虽然，采样的频率越高，已实现波动率越接近于二次变差，但是它不是积分波动率的一致估计量。我们可以通过构造如已实现极差(RRV), 已实现双幂次变差(RBV)等波动率的估计量来更精确地度量实际的波动率，或者可以适当地降低采样的频率。经过国内外众多学者的反复且细致的实证研究，发现由5min的采样频率得到的数据而计算出来的已实现波动率最具合理性。因此，有鉴于此，本文选取的是时间间隔为五分钟的高频数据。

**2.2 高频数据的波动率模型**

基于异质市场假说，Corsi提出了异质自回归已实现波动率(Heterogeneous autoregressive Model of Realized Volatility, HAR-RV)模型，该模型永不同时间间隔的已实现波动率来表示不同交易频率的交易者，来显示不同交易者的组成和其对市场波动率的影响程度。该模型可表示为：

（1）

本文参照Patton和Sheppard (2015) 的做法，采用不重叠的区间，采用HAR模型的平方根形式。

（2）

其中，h表示不同的预测区间；表示随机扰动项，分别表示日，周及月的已实现累计平均波动率，即：

依次类推，

这里，我们选用过去20天已实现波动率的平均数作为计算月累计平均已实现波动率，因为上海证券交易所的总交易所天数大概在240天，即平均每月交易日为20。

**3 数据与实证**

本文选取2016年1月3日至2016年12月29日的伊利股份（600623）的高频数据进行分析，总共有239个交易日以及1054296个价格数据，数据来源WIND数据库。

**3.1 使用秒级高频收益率并消除市场微观结构噪音的影响，计算每日已实现波动率**

本文采取子采样法进行降噪处理，参考刘梦瑶（2018）的做法，为了避免某一日数据的极端性，根据波动率计算公式以及二尺度实际波动率的计算公式，取J=1, K=20（约每1min的稀疏采样一次），得到二尺度实际波动率TSRV。

我们假设噪声过程是独立于的平稳过程，假设存在。是对数交易价格，是潜在的真实价格。

存在常数，

现在，我们关注积分波动率：

可以通过子样本来进行波动率的估计，我们定义平均滞后j阶的已实现波动率为：

对于，为：

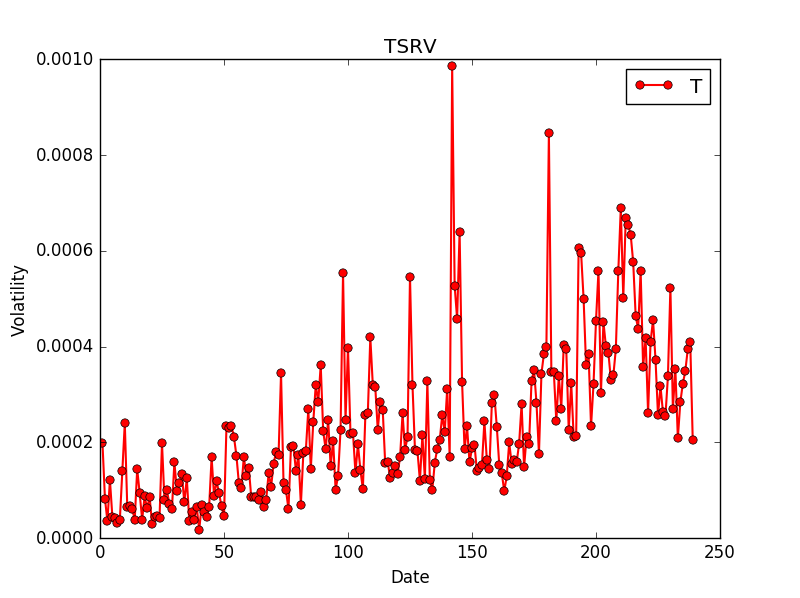
这里， ,

本文，取J=1，K=20。

下面对TSRV进行描述性统计：

**表3-1 描述性统计**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **最小值** | **最大值** | **均值** | **中位数** | **标准差** |
|  | 1.6509e-05 | 9.8622e-04 | 2.3348e-04 | 1.9638e-04 | 1.5889e-04 |



**图3-1 二尺度实际波动率TSRV**

**3.2 使用5分钟收益率并忽略市场微观结构噪音的影响，计算每日已实现波动率**

**3.2.1 高频数据向低频数据转化**

假设一天有个秒级价格数据，将一天的交易时间分为，共个小区间，秒级数据转换为5分钟数据的做法是：

为第个时间区间内价格数据的个数，是第个时间区间内第个价格数据。简而言之，就是对每个5分钟的时间区间内的价格取平均数，得到5分钟级的价格序列，再计算波动率。

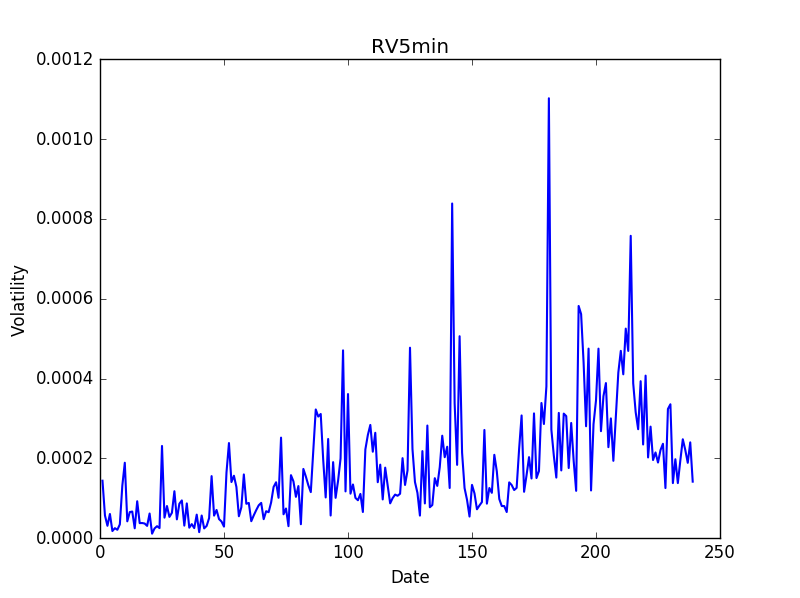
**3.2.2 计算已实现波动率**

在5分钟的时间维度下，根据Lily Y. Liu（2015），5分钟的RV是很好的估计量，研究时，我们根据文献和课程论文题目的要求，使用5分钟的已实现波动率，忽略市场微观结构噪音：

下面对5分钟的已实现波动率进行描述性统计：

**表3-2 描述性统计**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **最小值** | **最大值** | **均值** | **中位数** | **标准差** |
|  | 1.1260e-05 | 1.1027e-03 | 1.7811e-04 | 1.3984e-04 | 1.4510e-04 |



**图3-2 5分钟已实现波动率RV**

**3.3 运用HAR等模型，试说明：使用高频数据并消除噪音影响，能否提高对波动率的预测能力**

考虑如下HAR族：

:

消除市场微观结构噪音后：

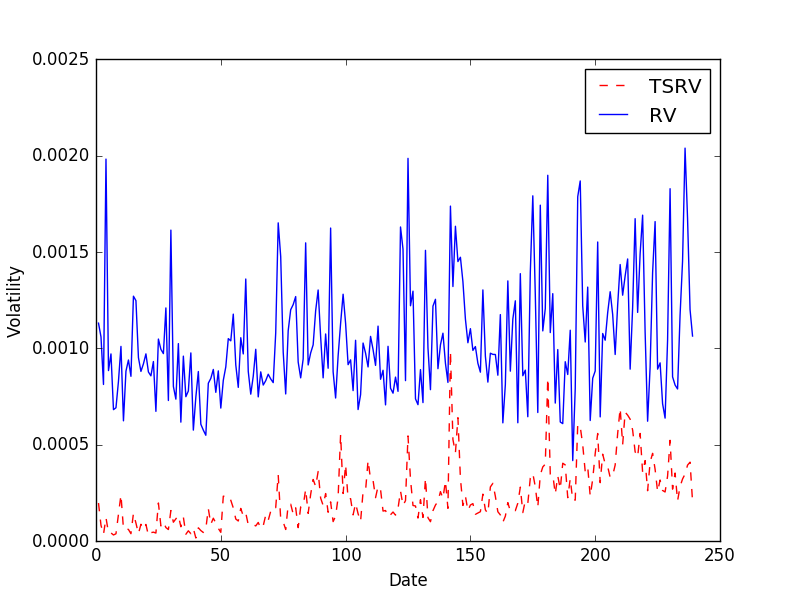
:

本文选取h=1，通过比较六个模型的均方误差（MSE）来比较波动率的预测能力。MSE越小，预测能力越好。

在使用回归分析之前，先计算没有剔除噪音的已实现波动率即秒级数据下的RV。

**表3-3 描述性统计**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **最小值** | **最大值** | **均值** | **中位数** | **标准差** |
|  | 1.6509e-05 | 9.8622e-04 | 2.3348e-04 | 1.9638e-04 | 1.5889e-04 |
|  | 4.181e-04 | 2.0398e-03 | 1.0399e-03 | 9.7162e-04 | 3.0788e-04 |



**图3-3 2017伊利股份已实现波动率RV与二尺度实际波动率TSRV的对比**

由图和基本的描述性统计可知，TSRV降噪效果显著，均值、中位数、标准差均比未降噪时要好，而且二尺度实际波动率TSRV更加平滑。

**表3-4 六个均方误差**

|  |  |  |  |
| --- | --- | --- | --- |
| **MSE** | | | |
|  | 8.7986e-02 |  | 1.2326e-08 |
|  | 1.9629e-05 |  | 9.5890e-06 |
|  | 7.4022e-02 |  | 1.7403e-01 |

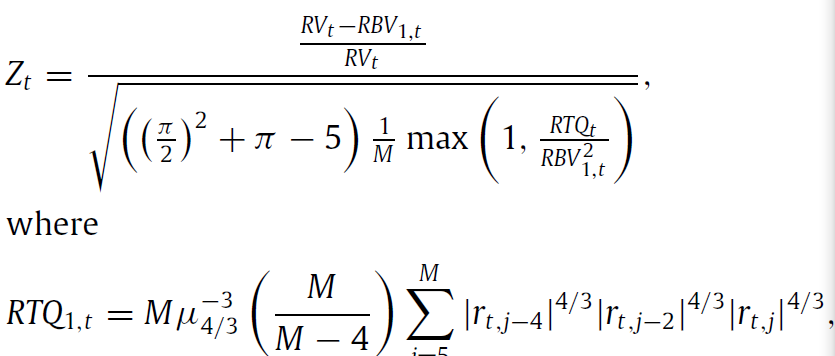
由上表可知，将降噪前和降噪后的预测能力相比，除了效果不如对应的模型，其他的开平方根和基准模型表现的预测能力更胜于消除噪音前。

**3.4 试计算由收益率跳跃产生的已实现波动率。在每日的已实现波动率中，由收益率跳跃产生的波动比例有多高**

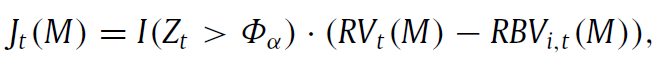
定义二次幂变差：

进而得出日已实现波动率的非连续部分：

使用Z统计量捕捉显著的跳跃：

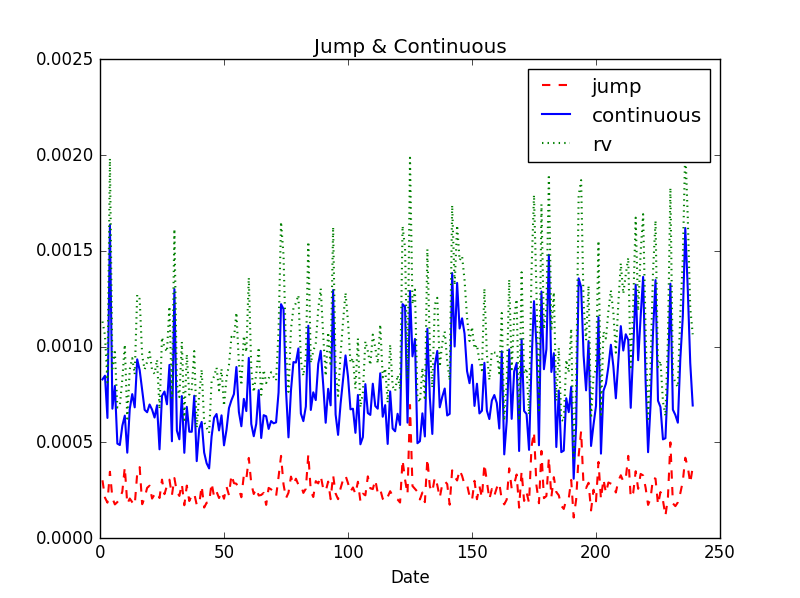






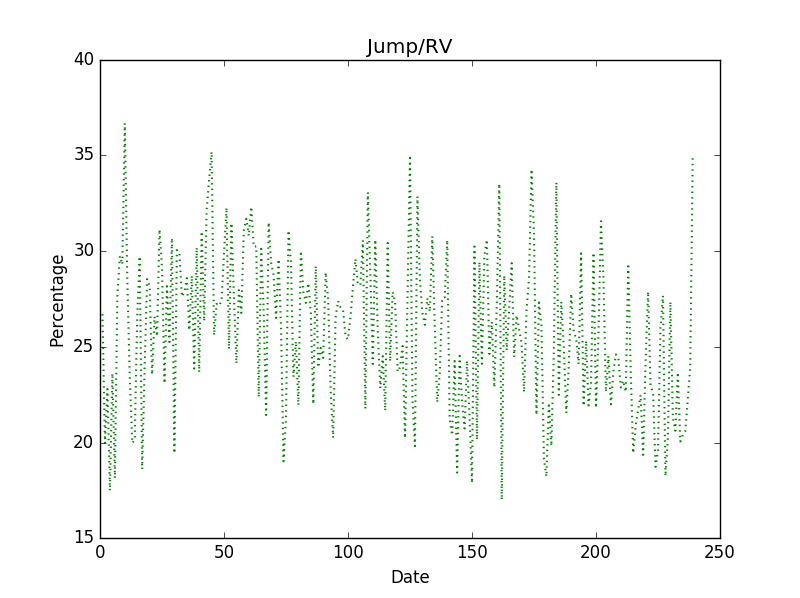
其中，就是我们要求的显著的跳跃部分。在采样趋于无穷大的情况下，已实现波动率可认为是连续部分和非连续部分的和。

下图为已实现波动率中的跳跃部分和连续部分：



**图3-4**

跳跃部分占的百分比为：



**3.5 若剔除跳跃产生的波动，并消除噪音的影响，HAR模型对波动率连续部分的预测能力是否高于问题3的实证结果？**

尝试剔除噪音和波动（用第1问的TSRV，减去跳跃的大小），得到新的指标，记为C，将负值调整为零。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **HRV** | | | | | |
|  | 8.7986e-02 |  | 1.2326e-08 |  | 5.5514e-09 |
|  | 1.9629e-05 |  | 9.5890e-06 |  | 1.5607e-05 |
|  | 7.4022e-02 |  | 1.7403e-01 |  | 失效 |

模型表现最好，能够提高对连续部分的预测能力，但是和不能提高对连续部分的预测能力。

**3.6 收益率跳跃产生的波动是否与订单流有显著的关系？**

每日订单流OD=每日的买单手数-卖单手数

[r,p]=corr(jump,od)

相关系数r= 0.2098

相关系数显著性检验：p= 0.0011<1% 说明收益率跳跃和订单流有着十分显著的关系。

**参考文献**

[1] Aït-Sahalia, Yacine, Mykland, Per A., Zhang, Lan, 2005. A tale of two time scales:determining integrated volatility with noisy high-frequency data. Journal of the American Statistical Association 100, 1394–1411.

[2] Patton, A. J., & Sheppard, K. (2015). Good volatility, bad volatility: Signed jumps and the persistence of volatility.Review of Economics and Statistics, 97, 683–697.

[3] Barndorff-Nielsen, Ole E., Shephard, Neil, 2006. Econometrics of testing for jumps in financial economics using bipower variation. Journal of Financial Econometrics, 4, 1–30.

[4] Andersen T G , Dobrev D , Schaumburg E . Jump-robust volatility estimation using nearest neighbor truncation[J]. Journal of Econometrics, 2012, 169(1), 75-93

[5] Liu L Y , Patton A J , Sheppard K . Does anything beat 5-minute RV? A comparison of realized measures across multiple asset classes[J]. Journal of Econometrics, 2015, 187(1):293-311.

[6]闫会强,夏霄松,金浩.HAR族模型对波动率的预测精度比较及其SPA检验——基于沪深300指数高频数据[J].经济论坛,2017(11):75-84.

[7]马锋,魏宇,黄登仕.基于符号收益和跳跃变差的高频波动率模型[J].管理科学学报,2017,20(10):31-43.

[8]智冬晓,许晓娟.基于高频数据实现波动算法的改进与评估[J].统计与决策,2015(17):9-13.

[9]张虎,周迪.基于波动和收益分解的股市风险收益关系检验——以2003—2012年上证指数高频数据为例[J].西部论坛,2014,24(05):80-89.

[10]马锋,魏宇,黄登仕.基于符号收益和跳跃变差的高频波动率模型[J].管理科学学报,2017,20(10):31-43.

[11]马锋. 高频数据视角下非参数波动率建模、预测及其评价研究[D].西南交通大学,2016.

**附录**

|  |
| --- |
| 数据清洗 |
| import os  import datetime  # 时间戳  def get\_timestamp(date):  return datetime.datetime.strptime(date, "%Y-%m-%d").timestamp()  #stamp转datetime 格式:list  def stamp2datetime(date\_stamp):  dt = [datetime.datetime.fromtimestamp(x) for x in date\_stamp]  return dt  #datatime转str 格式：list  def datetime2str(dt):  date\_str = [x.strftime("%Y-%m-%d") for x in dt]  return date\_str  my\_path = "D:\\HNU\\Market Microstructure\\YiLi"  files\_name = os.listdir(my\_path)  ##切片处理  fdates2017 = []  for fname in files\_name:  #原文件名 fdates  fdates, fextension = os.path.splitext(fname)  fdates = fdates[8:]  fdates2017.append(fdates)  fdates2017\_stamp = [get\_timestamp(x) for x in fdates2017]  fdates2017\_stamp = sorted(fdates2017\_stamp)  #时间戳转换成datatime格式  fdates2017 = stamp2datetime(fdates2017\_stamp)  #datatime转成str  fdates2017\_str = datetime2str(fdates2017)  if \_\_name\_\_ == "\_\_main\_\_":  #批量更改文件名  for i in files\_name:  old\_files = os.path.join(my\_path, i)  fdates, fextension = os.path.splitext(i)  fdates = fdates[8:]  f\_sta = get\_timestamp(fdates)  f\_d = datetime.datetime.fromtimestamp(f\_sta)  f\_d\_str = f\_d.strftime("%Y-%m-%d")  new\_files = os.path.join(my\_path, f\_d\_str + '.xlsx')  os.rename(old\_files, new\_files) |

"""

Created on Wed Dec 11 11:28:15 2019

@author: Cai Xiao 翻版必究

"""

import math

import pandas as pd

import matplotlib

import matplotlib.pyplot as plt

import os

import statsmodels as sm #statsmodels用不了，statsmodels.api可以

import numpy as np

#plt.rcParams['figure.figsize'] = (8.0, 4.0) # 设置figure\_size尺寸

#plt.rcParams['savefig.dpi'] = 2000 #图片像素

plt.rcParams['figure.dpi'] = 200 #分辨率

#plt.savefig(‘plot123\_2.png’, dpi=200)#指定分辨率

my\_path = "D:\\HNU\\Market Microstructure\\YiLi" #股票地址

save\_path = "D:\\HNU\\Market Microstructure" #计算结果保存本地文件夹

files\_name = os.listdir(my\_path) #文件名加后缀

#读取日期

fdates2017 = []

for fname in files\_name:

fdates, fextension = os.path.splitext(fname)

fdates2017.append(fdates)

# 读取文件

prices\_all\_year = pd.Series() #创建空的Series

ln\_returns\_all\_year = pd.Series()

rv\_day = pd.Series() #每日的日内已实现波动率

# Order-Flow

orderflow = []

for i in files\_name:

files\_path = os.path.join(my\_path, i)

data = pd.read\_excel(files\_path, index\_col=False) #有表头，列无index

buy = data.iloc[:, 14]

sell = data.iloc[:, 15]

buy\_sum = sum(buy.values)

sell\_sum = sum(sell.values)

orderflow\_meitian = buy\_sum - sell\_sum

print(orderflow\_meitian)

orderflow.append(orderflow\_meitian)

np.save(save\_path + 'orderflow.npy', orderflow)

# 每天有几个对数收益率数据 （价格数据 - 1）

M\_eachday = []

for j in files\_name:

files\_path = os.path.join(my\_path, j)

data = pd.read\_excel(files\_path, index\_col=False)

M\_temp = data.shape[0] - 1

M\_eachday.append(M\_temp)

np.save('D:\\HNU\\Market Microstructure\\每日对数收益率个数.npy', M\_eachday)

# 保存收益率文件

save\_path\_return\_tick = 'D:\\HNU\\Market Microstructure\\YiLi\_returns'

for i in files\_name:

files\_path = os.path.join(my\_path, i)

data = pd.read\_excel(files\_path, index\_col=False) #有表头，列无index

prices = data.iloc[:, 1]

ln\_prices = pd.Series([math.log(x) for x in prices])

temp = ln\_prices.diff()[1:]

save\_path\_tick\_data = os.path.join('D:\\HNU\\Market Microstructure\\YiLi\_returns', i[:10])

np.save(save\_path\_tick\_data, temp)

# Question 4 计算考虑噪音和跳跃时的已实现波动率 (tick data)

for i in files\_name:

files\_path = os.path.join(my\_path, i)

data = pd.read\_excel(files\_path, index\_col=False) #有表头，列无index

prices = data.iloc[:, 1]

ln\_prices = pd.Series([math.log(x) for x in prices])

temp = ln\_prices.diff()[1:]

a = 0

for x in temp.values:

a += x\*\*2

rv = pd.Series([a])

#print(sm.tsa.stattools.adfuller(ln\_returns.diff()[1:]))

prices\_all\_year = prices\_all\_year.append(prices)

rv\_day = rv\_day.append(rv)

rv\_day.index = fdates2017

#print(rv\_day)

#plt.figure(dpi=144)

#plt.grid(linestyle='-.')

#plt.plot(range(1,240), rv\_day, 'p-.')

#month = ['2017-01','2017-02','2017-03','2017-04','2017-05','2017-06'

# ,'2017-07','2017-08','2017-09','2017-10','2017-11','2017-12']

#month\_index = [0, 18, 36, 59, 73, 92, 114, 135, 158, 179, 196, 218]

#plt.xticks(month\_index, month, rotation=0)

# Question 1 计算RBV (tick data)

rbv\_day = pd.Series()

for j in files\_name:

files\_path = os.path.join(my\_path, j)

data = pd.read\_excel(files\_path, index\_col=False) #有表头，列无index

prices = data.iloc[:, 1]

ln\_prices = pd.Series([math.log(x) for x in prices])

temp = ln\_prices.diff()[1:]

b = 0

M = temp.values.shape[0] #一天内有多少个收益率数据

for x in range(2, M):

b += abs(temp.values[x]) \* abs(temp.values[x-2])

b = b \* (math.pi/2) \* (M/(M-2))

print(b)

rbv = pd.Series([b])

#print(sm.tsa.stattools.adfuller(ln\_returns.diff()[1:]))

rbv\_day = rbv\_day.append(rbv)

rbv\_day.index = fdates2017

#保存文件

np.save('D:\\HNU\\Market Microstructure\\rv\_day\_tick.npy', rv\_day)

np.save('D:\\HNU\\Market Microstructure\\rbv\_day\_tick.npy', rbv\_day)

np.save('D:\\HNU\\Market Microstructure\\fdates2017.npy', fdates2017)

np.save('D:\\HNU\\Market Microstructure\\files\_name.npy', files\_name)

# Question 2 5min data

fivemin\_per = 5 / (24 \* 60)

am\_start = 9.5 / 24

am\_end = 11.5 / 24

pm\_start = 13.0 / 24

pm\_end = 15.0 / 24

rv\_5min = []

for i in files\_name:

save\_path\_5min = "D:\\HNU\\Market Microstructure\\YiLi\_5min"

files\_path = os.path.join(my\_path, i)

data = pd.read\_excel(files\_path, index\_col=False)

tick\_time = data.iloc[:, 0]

prices = data.iloc[:, 1]

M = tick\_time.values.shape[0] #看看与多少个价格（或时间）数据

ii = 0

prices\_5min = list()

for j in range(0, 24):

prices\_5\_temp = 0

ii\_begin = ii

while (round(tick\_time.values[ii],6) >= round(am\_start+j\*fivemin\_per,6)) and (round(tick\_time.values[ii],6) <= round(am\_start+(j+1)\*fivemin\_per,6)):

prices\_5\_temp += prices.values[ii]

ii += 1

ii\_end = ii - 1

temp\_num = ii\_end - ii\_begin + 1

if temp\_num:

prices\_5\_avg = prices\_5\_temp / temp\_num

prices\_5min.append(prices\_5\_avg)

else:

prices\_5min.append(prices\_5min[j-1])

j\_num = len(prices\_5min) #上午有多少个五分钟价格数据

for j in range(0, 24):

prices\_5\_temp = 0

ii\_begin = ii

while (round(tick\_time.values[ii],6) >= round(pm\_start+j\*fivemin\_per,6)) and (round(tick\_time.values[ii],6) <= round(pm\_start+(j+1)\*fivemin\_per,6)):

if ii == M-1:

ii\_end = ii

prices\_5\_temp += prices.values[ii]

break

prices\_5\_temp += prices.values[ii]

ii += 1

ii\_end = ii - 1

temp\_num = ii\_end - ii\_begin + 1

if temp\_num:

prices\_5\_avg = prices\_5\_temp / temp\_num

if prices\_5\_avg:

prices\_5min.append(prices\_5\_avg)

else:

prices\_5min.append(prices\_5min[j+j\_num-1])

#计算 5min\_rv

ln\_prices\_5min = []

num\_5min = 0

returns\_5min = []

rerurns\_5min\_square = []

ln\_prices\_5min = [math.log(prices) for prices in prices\_5min ]

num\_5min = len(ln\_prices\_5min)

returns\_5min = [ln\_prices\_5min[j+1] - ln\_prices\_5min[j] for j in range(0, num\_5min-1)]

rerurns\_5min\_square = [x\*\*2 for x in returns\_5min]

rv\_5min.append(sum(rerurns\_5min\_square))

save\_5min = os.path.join(save\_path\_5min, i[:10])

np.save(save\_5min, prices\_5min)

rv\_5min00 = {'rv\_5min':rv\_5min}

rv\_5min\_df = pd.DataFrame(rv\_5min00)

rv\_5min\_df.index = fdates2017

np.save('D:\\HNU\\Market Microstructure\\rv\_5min.npy', rv\_5min\_df)

# 示性函数 indicator function

#Calculate Z-stat

# Calculate C

# Calculate J

rv\_day = np.load('D:\\HNU\\Market Microstructure\\rv\_day\_tick.npy')

rbv\_day = np.load('D:\\HNU\\Market Microstructure\\rbv\_day\_tick.npy')

JUMP = []

CONTI = []

alpha = 0.001

nppf = 3.090232306

#显著性水平为0.999，求分位点

#示性函数

def ind\_dayu(x):

if x > nppf:

i = 1

else:

i = 0

return i

def ind\_xiaoyu(x):

if x <= nppf:

i = 1

else:

i = 0

return i

for t in range(0, 239):

# Calculate rtq

#跳跃部分

jump = 0

#连续部分

conti = 0

npy\_name = fdates2017[t] + '.npy'

npv\_r\_tick\_path = os.path.join(save\_path\_return\_tick, npy\_name)

temp\_return = np.load(npv\_r\_tick\_path)

rtq\_sigma = 0

M\_temp = M\_eachday[t]

constant\_part = M\_temp \* (M\_temp / (M\_temp - 4)) \* ((2\*\*(2/3)) \* (math.gamma(7/6) / math.gamma(1/2)))\*\*(-3)

for j in range(4, M\_temp):

rtq\_sigma += abs(temp\_return[j-4])\*\*(4/3) \* abs(temp\_return[j-2])\*\*(4/3) \* abs(temp\_return[j])\*\*(4/3)

rtq = constant\_part \* rtq\_sigma

# Calculate Z-stat

z\_stat = M\_temp \*\* (1/2) \* ((rv\_day[t]-rbv\_day[t])/rv\_day[t])/((((math.pi/2)\*\*2+math.pi-5)\*max(1,(rtq/rbv\_day[t]\*\*2)))\*\*(1/2))

jump = ind\_dayu(z\_stat) \* (rv\_day[t] - rbv\_day[t])

JUMP.append(jump)

conti = ind\_dayu(z\_stat) \* rbv\_day[t] + ind\_xiaoyu(z\_stat) \* rv\_day[t]

CONTI.append(conti)

np.save('D:\\HNU\\Market Microstructure\\JUMP.npy', JUMP)

np.save('D:\\HNU\\Market Microstructure\\CONTI.npy', CONTI)

jt = np.load('D:\\HNU\\Market Microstructure\\JUMP.npy')

ct = np.load('D:\\HNU\\Market Microstructure\\CONTI.npy')

tsrv = np.load('D:\\HNU\\Market Microstructure\\tsrv.npy')

# 描述性统计 最大值 最小值 平均值 中位数 标准差

def basic\_stats(x):

max00 = np.max(x)

min00 = np.min(x)

avg00 = np.mean(x)

mid00 = np.median(x)

std00 = np.std(x)

return max00, min00, avg00, mid00, std00

plt.figure(dpi=200)

plt.plot(range(1,240),jt,'r--', linewidth=1.5)

plt.title('Jump & Continuous')

plt.plot(range(1,240),ct,'b-', linewidth=1.5)

plt.plot(range(1,240),rv\_day,'g:', linewidth=1.5)

plt.legend(['jump', 'continuous', 'rv']) # 图例

plt.xlabel('Date')

plt.ylabel('')

plt.show()

#跳跃部分占比

ptg = []

for i in range(0, 239):

ptg00 = jt[i] / rv\_day[i]

ptg.append(100\*ptg00)

plt.plot(range(1,240),ptg,'g:', linewidth=1.5)

plt.xlabel('Date')

plt.ylabel('Percentage')

plt.title('Jump/RV')

plt.show()

clean = []

for i in range(0,239):

clean00 = tsrv[i] + rbv\_day[i] - rv\_day[i]

if clean00 < 0 :

clean00 = 0

clean.append(clean00)

import pandas as pd

import numpy as np

import scipy

from scipy.stats import norm

tsrv = np.load("D:\\HNU\\Market Microstructure\\tsrv.npy")

orderflow = np.load("D:\\HNU\\Market Microstructure\\orderflow.npy")

result = pearsonr(tsrv, orderflow)

import math

import pandas as pd

import os

import numpy as np

import matplotlib

import matplotlib.pyplot as plt

my\_path = "D:\\HNU\\Market Microstructure\\YiLi"

save\_path\_ln\_prices = "D:\\HNU\\Market Microstructure\\ln\_prices\_tick"

save\_path\_prices = "D:\\HNU\\Market Microstructure\\prices\_tick"

files\_name = os.listdir(my\_path) #文件名加后缀

#读取日期

fdates2017 = []

for fname in files\_name:

#原文件名 fdates

fdates, fextension = os.path.splitext(fname)

fdates2017.append(fdates)

for i in files\_name:

files\_path = os.path.join(my\_path, i)

data = pd.read\_excel(files\_path, index\_col=False) #有表头，列无index

prices = data.iloc[:, 1]

ln\_prices = pd.Series([math.log(x) for x in prices])

s = os.path.join(save\_path\_ln\_prices, i[:10])

s0 = os.path.join(save\_path\_prices, i[:10])

np.save(s, ln\_prices.values)

np.save(s0, prices.values)

# 刘梦瑶 counteveryday:每日对数收益率个数

kk = 20

jj = 1

tsrv = []

m\_eachday = np.load('D:\\HNU\\Market Microstructure\\每日对数收益率个数.npy')

for i in range(0, 239):

# 导入对数价格

temp\_names = os.path.join("D:\\HNU\\Market Microstructure\\ln\_prices\_tick"

, fdates2017[i] + '.npy')

logprice = np.load(temp\_names)

sparsekk = 0

for j in range(0, m\_eachday[i] + 1 - kk):

sparsekk += (logprice[j+kk] - logprice[j]) \*\* 2

average\_sparsekk = sparsekk / kk

lagjj = 0

for j in range(0, m\_eachday[i] + 1 - jj):

lagjj += (logprice[j+jj] - logprice[j]) \*\* 2

average\_lagjj = lagjj / jj

nbarkk = (m\_eachday[i] - kk) / kk

nbarjj = (m\_eachday[i] - jj) / jj

tsrv\_meitian = (1-nbarkk/nbarjj)\*\*(-1)\*(average\_sparsekk-nbarkk/nbarjj\*average\_lagjj)

tsrv.append(tsrv\_meitian)

np.save('D:\\HNU\\Market Microstructure\\tsrv.npy', tsrv)

rv\_day = np.load('D:\\HNU\\Market Microstructure\\rv\_day\_tick.npy')

plt.figure(dpi=250)

plt.plot(range(1,240),tsrv,'ro-', label = 'TSRV', linewidth=1.5)

plt.title('TSRV')

plt.legend('TSRV')

plt.plot(range(1,240),rv\_day,'b-', label = 'RV')

plt.legend(['TSRV', 'RV']) # 图例

plt.xlabel('Date')

plt.ylabel('Volatility')

plt.show()

HAR模型拟合

HARRVa <- harModel(data=q5,periods=c(1,5,20),Rvest=c("rCov"),type="HARRV",h=1)

HARRVa\_sqrt <- harModel(data=q5,periods=c(1,5,20),Rvest=c("rCov"),type="HARRV",transform="sqrt",h=1)

HARRVa\_log <- harModel(data=q5,periods = c(1,5,20), Rvest=c("rCov"),type="HARRV",transform="log",h=1)

summary(HARRVa)

summary(HARRVa\_sqrt)

summary(HARRVa\_log)

MSE.HARRVa = mean(HARRVa$residuals\*HARRVa$residuals)

MSE.HARRVa\_sqrt = mean(HARRVa\_sqrt$residuals\*HARRVa\_sqrt$residuals)

MSE.HARRVa\_log = mean(HARRVa\_log$residuals\*HARRVa\_log$residuals)

计算相关系数和相关系数的检验：(Matlab)

[r,p]=corr(jump,od)