## **量化建模分析入门（上）**

原创 KeKe， 张国平 VNPY  2019-10-26

**一、数据分析的必要性**

**（一）策略开发离不开数据分析。**

首先，数据质量的好坏决定策略实盘的可行性，例如下载BITMEX TESTNET的仿真XBTUSD历史数据，却用来研究实盘策略，就会导致典型的“Garbage in, garbage out”。

其次，下载的历史数据不应该出现某段时间缺失的情况，这就要求建模分析的第一步是对行情数据进行画图，通过人眼初步判断数据的好坏质量：即无缺失的行情部分、也无明显的“异常”数据点。

确定数据质量可靠后，就可以分析数据特性，从而决定其适用于开发哪种类型的策略。若数据自相关性弱，属于平稳时间序列，符合正态分布，可以考虑用作统计套利；若数据自相关性强，且非平稳时间序列，有明显的肥尾特征，则可以用来开发趋势跟踪类的策略（CTA）。

**（二）CTA策略开发前的建模分析流程。**

1. 对行情数据画图，确定无断点；

2. 进行随机性检验，确定数据具有非纯随机性；

3. 进行平稳性检验，确定时间序列的不平稳性；

4. 画出自相关图，确定时间序列自相关性强；

5. 相对波动率分析，确定其平均波动率大于固定交易成本（手续费）；

6. 行情变化率分析，确定其符合肥尾分布；

7. 计算相关计算指标（DataFrame格式），可以与行情图对比分析；

8. 合成更高维度的K线，重复前面1~7步。

**二、数据分析流程**

**第1步：对行情数据进行画图。**

首先，调用vn.py数据库模块的database\_manager.load\_bar\_data函数从数据库载入数据到内存中。

load\_bar\_data函数的输入参数有5个：

symbol：合约代码，由用户定义，字符串格式，如“XBTUSD”;

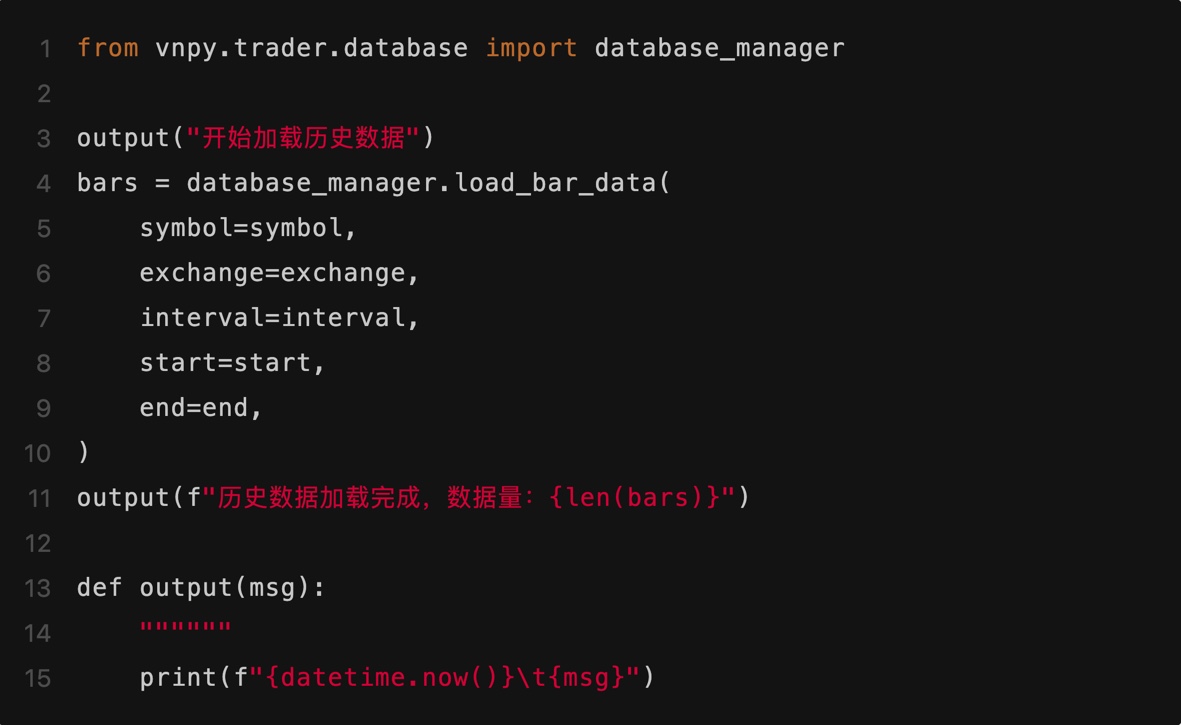
exchange：交易所代码，Exchange枚举值格式，如Exchange.BITMEX;

interval：K线周期，Interval枚举值格式，如Interval.MINUTE;

start：数据开始时间，datetime格式，如datetime(2019, 4, 1)；

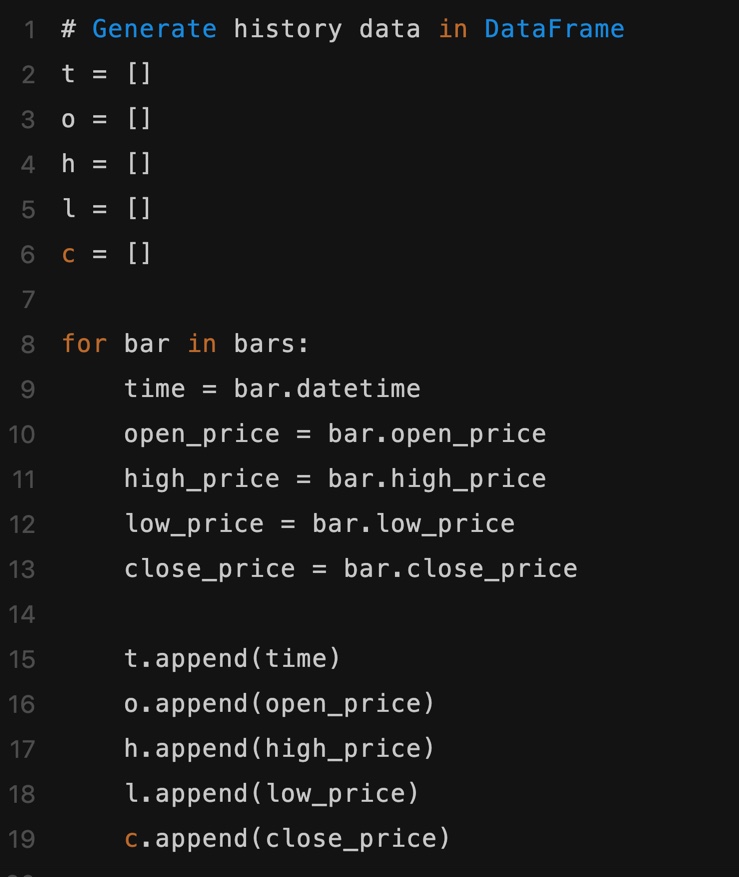
end：数据结束时间，datetiem格式，如datetime(2019, 10, 30)；

load\_bar\_data函数输出的是一个包含系列BarData格式行情数据的列表bars。



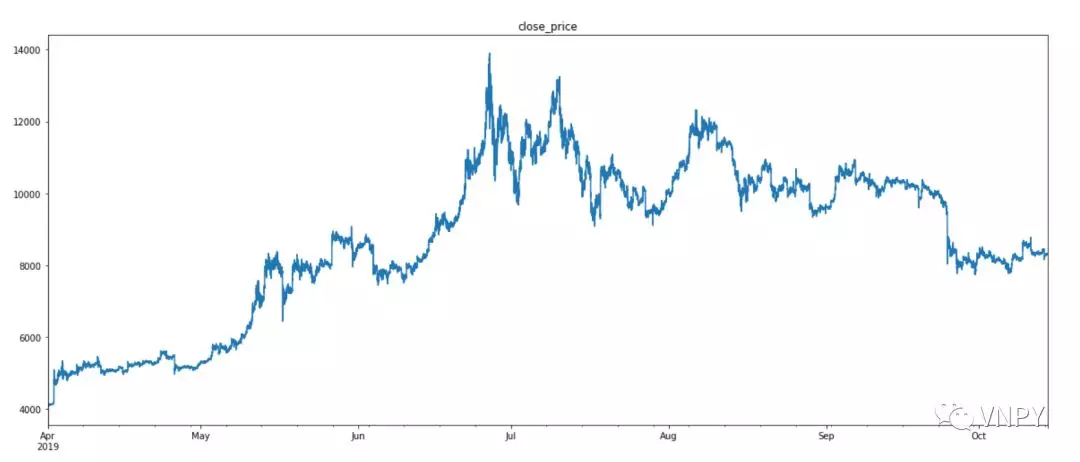
用for循环读取bars列表中的BarData数据，然后分别缓存时间、开盘价、最高价、最低价、收盘价到专门的列表中，最终合成DataFrame， 设置DataFrame索引为K线数据的时间。

对收盘价进行画图，设置图的尺寸和标题。然后用肉眼初步确认时间序列图无数据缺失或者明显“异常”行情。



对收盘价进行画图，设置图的尺寸和标题。然后用肉眼初步确认时间序列图无数据缺失或者明显“异常”行情。

output("第一步:画出行情图，检查数据断点")self.orignal["close"].plot(figsize=(20, 8), title="close\_price")plt.show()



**第2步：随机性检验**

调用statsmodels库的acorr\_ljungbox函数，对收盘价进行白噪声检验，函数返回一个p值，p值越大表示原假设成立的可能性越大，即数据是随机的可能性越大。

一般p值与0.05进行对比：

若p值>0.05，证明数据具有纯随机性：不具备某些数据特征，没有数据分析意义；

若p值<0.05，证明数据不具有纯随机性：具备某些数据特性，进行数据分析有利于开发出适应其统计规律的交易策略。

from statsmodels.stats.diagnostic import acorr\_ljungbox  
def random\_test(close\_price): """ 白噪声检验 """ acorr\_result = acorr\_ljungbox(close\_price, lags=1) p\_value = acorr\_result[1] if p\_value < 0.05: output("第二步：随机性检验：非纯随机性") else: output("第二步：随机性检验：纯随机性") output(f"白噪声检验结果:{acorr\_result}\n")



**第3步：平稳性检验**

同样，调用statsmodels库的adfuller函数对收盘价进行单位根检验，函数返回的是一个字典，我们对字典里面的字段进行判断：

若ADF值>10% 统计量，说明原假设成立：存在单位根，时间序列不平稳；

若ADF值<10% 统计量，说明原假设不成立：不存在单位根，时间序列平稳。

CTA策略研究的是非平稳性时间序列，平稳时间序列适用于期货价差的统计套利。

from statsmodels.tsa.stattools import adfuller as ADF  
def stability\_test(close\_price): """ 平稳性检验 """ statitstic = ADF(close\_price) t\_s = statitstic[1] t\_c = statitstic[4]["10%"]  
 if t\_s > t\_c: output("第三步：平稳性检验：存在单位根，时间序列不平稳") else: output("第三步：平稳性检验：不存在单位根，时间序列平稳")  
 output(f"ADF检验结果：{statitstic}\n")



**第4步：画出自相关图**

同样，调用statsmodels库的plot\_acf函数和plot\_pacf函数对收盘价画自相关和偏自相关图：

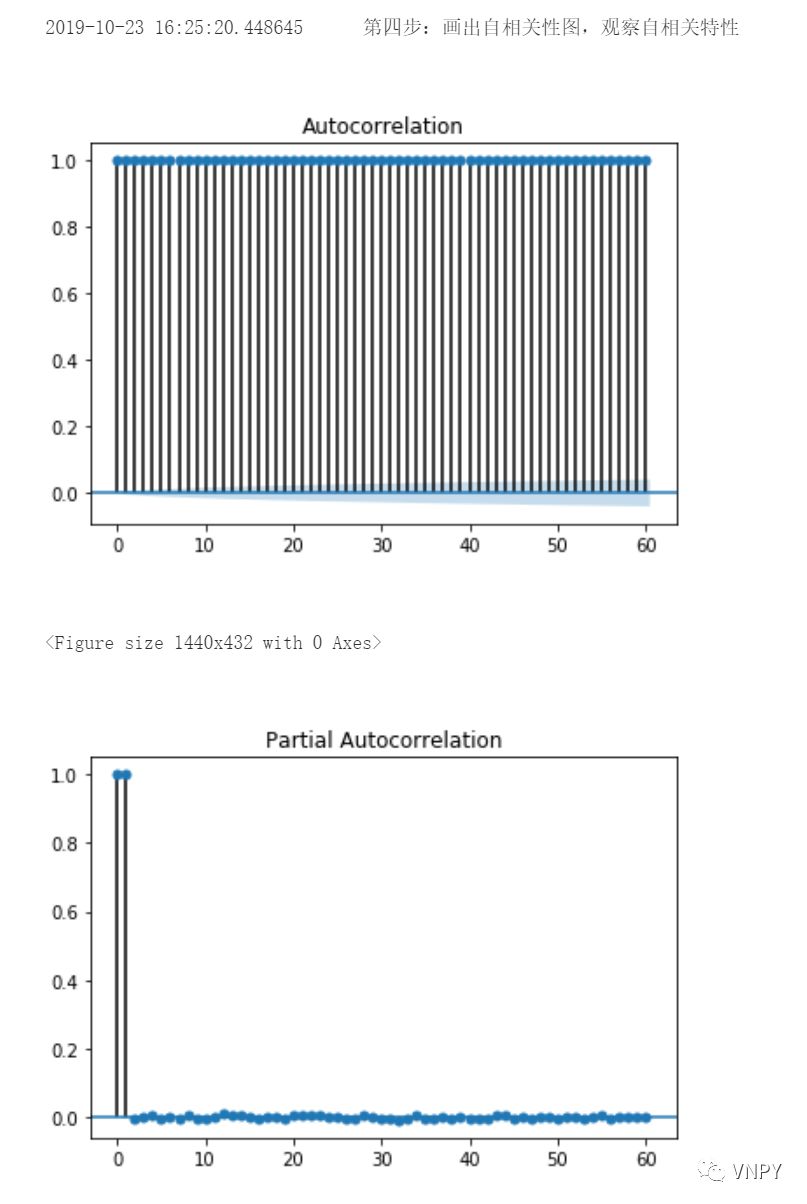
自相关图：统计相关性总结了两个变量之间的关系强度，即描述了一个观测值与另一个观测值之间的自相关，包括直接和间接的相关性信息。这种关系的惯性将继续到之后的滞后值，随着效应被削弱而在某个点上缩小到没有。

偏自相关图：偏自相关是剔除干扰后时间序列观察与先前时间步长时间序列观察之间关系的总结，即只描述观测值与其滞后之间的直接关系。可能超过k的滞后值不会再有相关性。

置信区间被画成圆锥形。默认情况下，置信区间被设置为95％，这表明，圆锥之外的值很可能是相关的，而不是统计上的意外。

也就是说，圆锥以外的值越多，时间序列自相关性越强，越适用于研究CTA策略。

from statsmodels.graphics.tsaplots import plot\_acf, plot\_pacf  
def autocorrelation\_test(close\_price): """ 自相关性检验 """ output("第四步：画出自相关性图，观察自相关特性")  
    plot\_acf(close\_price, lags=60) plt.show()  
    plot\_pacf(close\_price, lags=60).show() plt.show()



**第5步：相对波动率分析**

调用talib库的ATR函数计算1分钟K线的绝对波动率；

通过收盘价\*手续费计算固定成本；

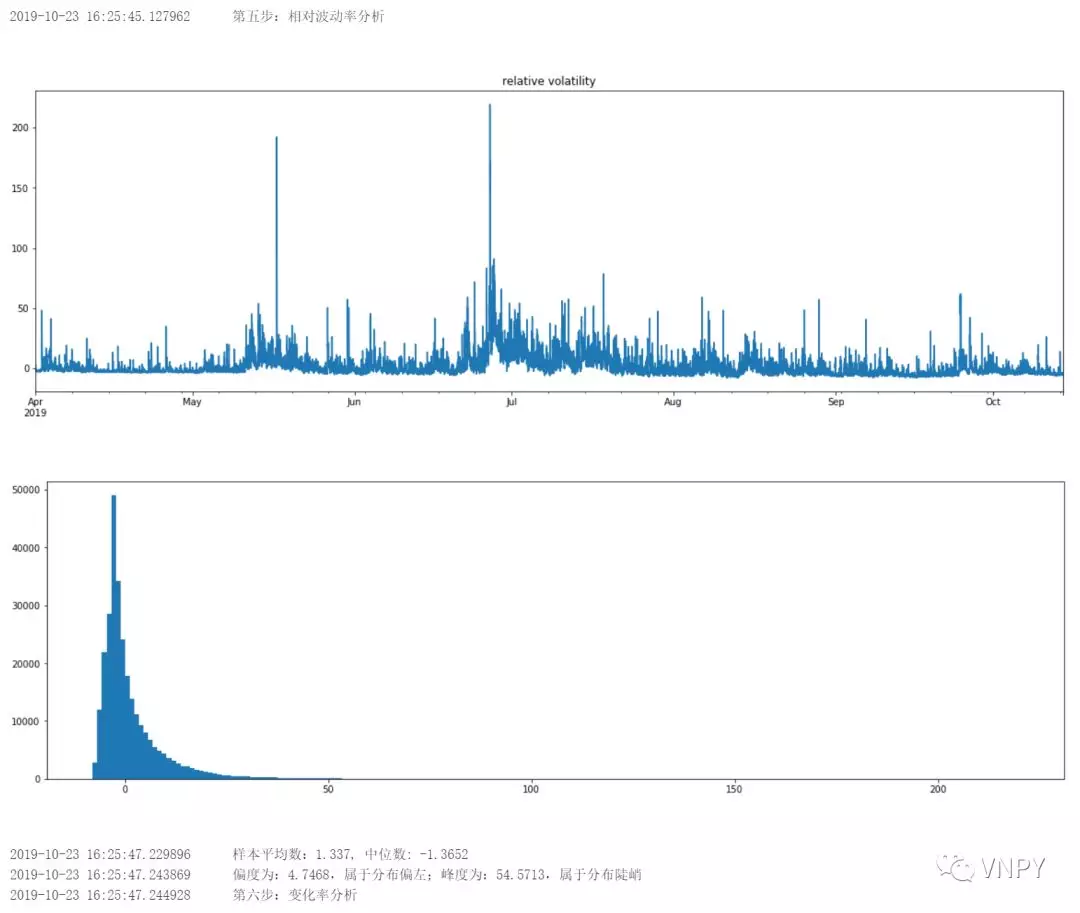
相对波动率=绝对波动率-固定成本；

对相对波动率进行画图：时间序列图和频率分布图；

对相对波动率进行描述统计分析，得到平均数、中位数、偏度、峰度

若相对波动率分布属于右偏（肥尾在右边），且分布陡峭，在统计学上具有尖峰肥尾的特色，适用于研究CTA策略。

def relative\_volatility\_analysis(self, df: DataFrame = None): """ 相对波动率 """ output("第五步：相对波动率分析") df["volatility"] = talib.ATR( np.array(df['high']), np.array(df['low']), np.array(df['close']), self.window\_volatility )  
 df["fixed\_cost"] = df["close"] \* self.rate df["relative\_vol"] = df["volatility"] - df["fixed\_cost"]  
 df["relative\_vol"].plot(figsize=(20, 6), title="relative volatility") plt.show()  
 df["relative\_vol"].hist(bins=200, figsize=(20, 6), grid=False) plt.show()  
 statitstic\_info(df["relative\_vol"])  
def statitstic\_info(df): """ 描述统计信息 """ mean = round(df.mean(), 4) median = round(df.median(), 4) output(f"样本平均数：{mean}, 中位数: {median}")  
 skew = round(df.skew(), 4) kurt = round(df.kurt(), 4)  
 if skew == 0: skew\_attribute = "对称分布" elif skew > 0: skew\_attribute = "分布偏左" else: skew\_attribute = "分布偏右"  
 if kurt == 0: kurt\_attribute = "正态分布" elif kurt > 0: kurt\_attribute = "分布陡峭" else: kurt\_attribute = "分布平缓"  
 output(f"偏度为：{skew}，属于{skew\_attribute}；峰度为：{kurt}，属于{kur t\_attribute}\n")



**第6步：变化率分析**

计算百分比变化率 = 100\*（收盘价- 上一根收盘价）/上一根收盘价；

对变化率进行画图：时间序列图和频率分布图；

对变化率进行描述统计分析，得到平均数、中位数、偏度、峰度。

def growth\_analysis(self, df: DataFrame = None): """ 百分比K线变化率 """ output("第六步：变化率分析") df["pre\_close"] = df["close"].shift(1).fillna(0) df["g%"] = 100 \* (df["close"] - df["pre\_close"]) / df["close"]  
 df["g%"].plot(figsize=(20, 6), title="growth", ylim=(-5, 5)) plt.show()  
 df["g%"].hist(bins=200, figsize=(20, 6), grid=False) plt.show()  
 statitstic\_info(df["g%"])

