High-frequency price data analysis in R

R/Finance 2015

Kris Boudt
VU Brussel/Amsterdam
About myself

• Associate Professor of Finance and Econometrics at Free University of Brussels and Amsterdam;

• Research on developing econometric methodology to solve problems in finance.

• R packages to which I contributed: highfrequency, PeerPerformance, PerformanceAnalytics, PortfolioAnalytics, CIP, DEoptim;
Roadmap (i)

• “There are old traders, there are bold traders, but there are no old bold traders”
• Focus: How to use high frequency price data to understand better the time-varying risk properties of the investment
• Two types of risk: the normal volatility risk and the jump risk
• Topics:
  – Cleaning and aggregation (univariate and multivariate) of tick prices into log-returns
  – Discrete time model for intraday returns:
    • Spot volatility estimation
    • Price jump detection
  – Continuous time model for log-prices
    • Realized volatility estimation
    • Detection of a jump component in realized volatility
  – Forecasting volatility using realized volatility measures.
Roadmap (ii)

• And how to do these analysis with the functions in the R package **highfrequency**
  – Latest version at: [http://r-forge.r-project.org/R/?group_id=1409](http://r-forge.r-project.org/R/?group_id=1409)
  – Main authors are Jonathan Cornelissen (Datacamp), Scott Payseur (UBS) and myself.

Other contributors:
• GSOC:
  • Giang Nguyen
  • Maarten Schermers
• Chris Blakely, Brian Peterson, Eric Zivot
  • You?
WARNING: The functions in highfrequency were initially designed for the Trades and Quotes database but are generally applicable, as long as:

- They are xts-objects;
- Some functions require tdata/qdata:
  
  • tdata: Trade data having at least the column name “PRICE”
  • qdata: Quote data having at least the column names “BID” and “OFR”

```r
> highfrequency:::tdatacheck
function (tdata)
{
  if (!is.xts(tdata)) {
    stop("The argument tdata should be an xts object")
  }
  if (!any(colnames(tdata) == "PRICE") ) {
    stop("The argument tdata should have a PRICE column")
  }
}

<environment: namespace:highfrequency>
> highfrequency:::qdatacheck
function (qdata)
{
  if (!is.xts(qdata)) {
    stop("The argument qdata should be an xts object")
  }
  if (!any(colnames(qdata) == "BID") ) {
    stop("The argument qdata should have a column containing the BID. Could not find that column")
  }
  if (!any(colnames(qdata) == "OFR") ) {
    stop("The argument qdata should have a column containing the ASK / OFR. Could not find that column")
  }
}

<environment: namespace:highfrequency>
```
CLEANING AND AGGREGATION
• High frequency price data analysis: Making sense of too big data
The tick by tick ‘raw’ price series needs to processed in two ways:

- **Data cleaning** to remove some obvious “errors” from the data:
  - Trades and quotes with position size of 0;
  - Trades and quotes with time stamp outside the opening hours of the exchange;
  - Trade prices that are below the best bid are above the best ask
  - Bid quotes that are higher than the ask quote
  - Fat finger errors: A human error caused by pressing the wrong key when using a computer to input data.

- **Aggregation** to the frequency of interest.
<table>
<thead>
<tr>
<th>Function in highfrequency</th>
<th>Aim</th>
<th>Requirement on input data</th>
</tr>
</thead>
<tbody>
<tr>
<td>exchangeHoursOnly</td>
<td>Restrict data to exchange hours</td>
<td>xts</td>
</tr>
<tr>
<td>selectexchange</td>
<td>Restrict data to specific exchange</td>
<td>xts with column &quot;EX&quot;</td>
</tr>
<tr>
<td>autoSelectExchangeTrades</td>
<td>Restrict data to exchange with highest trade volume</td>
<td>xts with column &quot;EX&quot; and &quot;SIZE&quot;</td>
</tr>
<tr>
<td>mergeTradesSameTimestamp</td>
<td>Delete entries with same time stamp and use median price</td>
<td></td>
</tr>
<tr>
<td>rmTradeOutliers</td>
<td>Delete entries with prices above/below ask/bid +/- bid/ask spread</td>
<td>xts with column “PRICE” xts with columns “BID” and “OFR” ➔ Transaction and quote data are matched internally with the function matchTradesQuotes</td>
</tr>
<tr>
<td>rmOutliers</td>
<td>Remove outliers in quote data based on rolling outlier detection in the spread</td>
<td>xts with columns “BID” and “OFR”</td>
</tr>
</tbody>
</table>

And several others: noZeroPrices, noZeroQuotes, mergeQuotesSameTimestamp, rmNegativeSpread
Aggregation

• High frequency price data analysis consists of zooming in on the intraday price data obtained typically as tick data (which occur at irregular times) aggregated at some frequency:

![Diagram of Aggregation]

- Daily aggregation
- Aggregation over higher frequency intervals

Tick by tick price data
Two types of aggregation from tick data:

- **Calendar time** based sampling: Every 10 minutes, Every minute, Every second, Every milisecond ➔ Prices are observed at regularly spaced time intervals

- **Transaction** based sampling (also called business time sampling): Every 10 trades, every trade (tick data).

The choice of the sampling frequency may be a function of, among other things, the liquidity of the stock: Illiquid stocks are infrequently traded implying many zero returns at very high frequencies.
Calendar time based sampling

- Function `aggregatets`: From transaction time to a **fixed calendar time based frequency**, e.g. every 5 seconds:
  - Default: previous tick: take the last price observed in the interval: [start,end] (i.e. excluding the value at the end time of the interval)
  - Alternative: take the mean value

```r
data("sample_tdata");
ts = sample_tdata$PRICE;
# Previous tick aggregation to the 5-seconds sampling frequency:
tsagg5secs = aggregatets(ts, on="seconds", k=5);
head(tsagg5secs);
```
Note: Loss of observation, but more tractable, and less market microstructure noise issues
Business time based sampling

• From transaction time to a **fixed business time based frequency**, e.g. every 5 ticks:

```r
    ts[seq(1, length(ts), 5)]
```
Note: Loss of observation, but more tractable, and less market microstructure noise issues.
• For multivariate analysis, such a univariate aggregation scheme is often not suited due to **non-synchronicity** in the trades of different assets:
  – If one stock has traded, but the other has not, it would seem as there is no relationship, while in fact there is one, but we have not observed it yet.
  – Epss effect: Because trades occur in discrete time, when sampling at ultrahighfrequency observation times, the correlation is biased towards zero.
Multivariate synchronization: Refresh times

- From nonsynchronous transaction times based observations of multiple series to common observations: next observation is when there has been a new observation for all series.

\[ \tau_1 \] is the time it has taken until the three assets have traded, i.e. all the posted prices have been updated.

\[ \tau_2 \] is the first time when all the prices are again refreshed.

#suppose irregular timepoints:
start = as.POSIXct("2010-01-01 09:30:00")
ta = start + c(1,2,4,5,9,14);
tb = start + c(1,3,6,7,8,9,10,11,15);
tc = start + c(1,2,3,5,7,8,10,13);
#yielding the following timeseries:
a = as.xts(1:length(ta),order.by=ta);
b = as.xts(1:length(tb),order.by=tb);
c = as.xts(1:length(tc),order.by=tc);
#Calculate the synchronized timeseries: 
refreshTime(list(a,b,c))
Note:

- The least liquid stock will determine the sampling grid: you risk to lose many observations.
- Even if same liquidity, because of random arrivals, large data losses in high dimensions. Curse of dimensionality!
DISCRETE TIME MODEL FOR THE LOG-RETURNS
Let us consider first a discrete time location-scale model for the high-frequency returns;

Notation:
- $P(s)$ is the price at time $s$
- $p(s) = \log(P(s))$ is the natural logarithm of the price
- We assume for the moment equispaced intraday returns and denote the $i$-th return on day $t$ as $r_{t,i}$
- The length of one day is normalized to $[0, 1]$
- Assume $M$ observations in a day, then the time between two observations is $\Delta = 1/M$
- The $i$-th return on day $t$ is given by:

$$r_{t,i} = p(t - 1 + i\Delta) - p(t - 1 + (i - 1)\Delta)$$
A discrete time conditional location-scale model for the highfrequency log-return

• Conditional on the information available at the end of the previous intraday time interval $I_{t,i-1}$:

\[ r_{t,i} = \mu_{t,i} \Delta + \sigma_{t,i} \sqrt{\Delta} z_{t,i} \]

– with $\mu_{t,i}$ the conditional mean at the daily level (also called drift)
– $\sigma_{t,i}$ the conditional volatility at the daily level (also called spot volatility)
– $z_{t,i}$ standard white noise (i.e. iid with mean 0 and variance 1, typically assumed to be Gaussian).
Estimation?

• At high frequencies, we can assume the drift to be 0:

\[ \mu_{t,i} \Delta = 0 \]

• The function spotvol provides an estimate of \( \sigma_{t,i} \):
  – Non-parametric: local kernel estimator;
  – Semi-parametric: volatility is assumed to be given by the product between:
    • A stochastic daily volatility level \( \sigma_t \)
    • A deterministic intraday period process, corresponding to the U-shape \( f_i \):

\[ \sigma_{t,i} = f_i \sigma_t \]
data(sample_real5minprices)
plot(spotvol(sample_real5minprices))

Time series of 5-minute spot volatilities for 60 days:

Corresponding components:
* Intraday periodic component that only de
• Extensions in spotvol:
  – Allow for a stochastic component in the periodic pattern;
  – Robust estimators that account for price jumps in the estimation:

\[ r_{t,i} = \sigma_{t,i} \sqrt{\Delta} z_{t,i} + j_{t,i} \]

– And use this to detect the price jumps by identifying a jump when:

\[ \left| \frac{r_{t,i}}{\sigma_{t,i} \sqrt{\Delta}} \right| \text{ is very large} \]
# illustration for day 9 in the example data sample_real5minprices
# plot the price series and the corresponding jump test statistics

d=9 ; par(mfrow=c(2,1),mar=c(3,2,2,1))
plot(sample_real5minprices[d*79+(1:79)],main="prices")
plot(abs(diff(log(sample_real5minprices[d*79+(1:79)]))[(−1)])/spotvol(sample_real5minprices)$spot[d*79+(1:79)],type="h",
main="jump test statistics")
• How large?
  – Take a high critical value to avoid false detections
  – But not too high to avoid that you lose power to detect the jumps
REALIZED VOLATILITY ESTIMATION
• In addition to the estimation of the local spot volatility, it is important to also estimate the variability over a longer time window, such as one day.
• Noisy measure of daily variability: squared daily return
• Potentially more efficient measures use intraday data:
  – daily price range
  – Realized variance: sum of squared intraday returns
  – ...

➤ However, to understand what parameter they actually estimate it is important to have a model for the intraday price evolution: **Continuous-time brownian semimartingale model with jumps.**

➤ The observed prices are discrete time realizations of that continuous-time process.
From discrete to continuous time model

- Asymptotic analysis: what happens if $M \to \infty$, that is, when $\Delta \to 0$. 
From discrete to continuous time

- Discrete time conditional location scale model

\[ r_{t,i} = \mu_{t,i} \Delta + \sigma_{t,i} \sqrt{\Delta} z_{t,i} \]

- Continuous time brownian semi-martingale diffusion

\[ dp_s = \mu_s \, ds + \sigma_s \, dw_s \]

\( \mu \) is the drift parameter  \( \sigma \) is the spot volatility parameter
• A continuous time stochastic process \( \{w_t\} \) is a Brownian motion (Wiener process) if it satisfies that its increments are iid normal with variance equal to the time change.

• More precisely, for any time \( s \), we have that
\[
w_s - w_{s-\Delta} = z_s \sqrt{\Delta}
\]
where \( z \) is a standard normal random variable.

• Martingale property: best possible prediction of future value is most recently observed value
\[
E[w_t \mid w_s] = w_s, \forall t \geq s
\]
Simulation of Brownian motions

```r
set.seed(1234)
# simulate brownian motion
M = 10000
delta = 1/M
z1 = rnorm(M)*sqrt(delta)
z2 = rnorm(M)*sqrt(delta)
z3 = rnorm(M)*sqrt(delta)
z4 = rnorm(M)*sqrt(delta)
w1 = cumsum(z1); w2 = cumsum(z2); w3 = cumsum(z3); w4 = cumsum(z4)
vt = seq(0,1,length.out=M)
plot( vt, w1 , ylim = c( min(c(w1,w2,w3,w4)), max(c(w1,w2,w3,w4)) ) ,
   col="blue" , type="l
", xlab="time",ylab="realizations of Brownian
motion")
lines( vt , w2 , col="red")
lines( vt , w3 , col="green")
lines( vt , w4 , col="orange")```

Including jumps

- For simplicity, assume jumps have finite activity: this means that for each interval the number of jumps that can happen is finite.
- The cumulative number of jumps is given by a count process $q_s$ (e.g., Binomial).
- The magnitude of the jump is given by the process $\kappa_s$.
- The Brownian semimartingale with finite activity jumps is then:

$$dp_s = \mu_s ds + \sigma_s dw_s + \kappa_s dq_s$$
• Different ex post measures of variability can be considered:

  – Quadratic variation: Total variation;
  – Integrated variance: Only the smooth variation;
  – Jump variance: Only the jump variation;
  – Jump tests: Is there a significant jump variability observed on a given day?
Realized variance

- Realized variance is the sum of squared intraday returns
  \[ RV = \sum_{i=1}^{M} r_{t,i}^2 \]
- When \( \Delta \to 0 \), the realized variance converges to the quadratic variation, which under the BSMFAJ model equals the integrated variance + sum of squared intraday jumps:
  \[ QV = \lim_{M \to \infty} \sum_{i=1}^{M} r_{t,i}^2 = \int_{0}^{1} \sigma_s^2 ds + \sum_i K_i^2 \]
Realized bipower variation

- Sometimes we only wish to estimate the integrated variance
- Jumps have finite activity: the probability that two contiguous returns have a jump component is 0 almost surely.
- Two continuous returns have almost the same spot variance
- The impact of the product between a “continuous” return and a return with a jump component is negligible
- Hence the realized bipower variation is consistent for the Ivar

\[ RBV = \frac{\pi}{2} \sum_{i=2}^{M} |r_{t,i}| |r_{t,i-1}| \xrightarrow{\Delta \to 0} \int_{0}^{1} \sigma_s^2 \, ds \]

(the correction factor π/2 corresponds to the inverse of the square of the expected value of a standard normal random variable)
• Since:

\[ RV = \sum_{i=1}^{M} r_{t,i}^2 \xrightarrow{\Delta \to 0} \int_0^1 \sigma_s^2 \, ds + \sum_i \kappa_i^2 \]

\[ RBV = \frac{\pi}{2} \sum_{i=2}^{M} | r_{t,i} \| r_{t,i-1} | \xrightarrow{\Delta \to 0} \int_0^1 \sigma_s^2 \, ds \]

• We have that:

\[ RV - RBV \xrightarrow{\Delta \to 0} \sum_i \kappa_i^2 \]
Other robust estimators exist

• MedRV

\[
\text{medRV} = c \sum_{i=2}^{M-1} \text{median}(|r_{t,i-1}|, |r_{t,i-1}|, |r_{t,i}|)^2 \xrightarrow{\Delta \to 0} \int_0^1 \sigma_s^2 ds
\]

(c is a correction factor to ensure consistency)

• ROWVar

\[
\text{ROWVar} = c_k \sum_{i=1}^M r_{t,i}^2 I\left[ \frac{|r_{t,i}|}{\sigma_{t,i} \sqrt{\Delta}} \leq k \right] \xrightarrow{\Delta \to 0} \int_0^1 \sigma_s^2 ds
\]
• Because of estimation error, the jump robust estimator and the total variation estimator will always give a different number in finite samples;

• How to decide whether that difference is large enough to say that there has been a price jump?
Jump test

- **H₀**: no jump on day t
- **Hₐ**: at least one jump on day t
- Under **H₀** the RV and robust alternatives estimate the same quantity (IV): the difference is estimation error that is normally distributed around 0 and variance proportional to the integrated quarticity

\[
\sqrt{M} (RV - RBV) \xrightarrow{d} N(0, \theta \int_0^1 \sigma_s^4 ds)
\]
Estimation of integrated quarticity

- MedRQ

\[ medRQ = c \sum_{i=2}^{M-1} \text{median}(|r_{t,i-1}|, |r_{t,i-1}|, |r_{t,i}|)^4 \xrightarrow{\Delta \to 0} \int_0^1 \sigma_s^4 ds \]

(c is a correction factor to ensure consistency)

- Then the jump test statistic is:

\[ \sqrt{M} \frac{(RV - \hat{IV})}{\sqrt{\theta \hat{IQ}}} \xrightarrow{d} N(0,1) \]
Example

data(sample_tdata)

`BNSjumpTest` (sample_tdata$PRICE[1:79], QVestimator= "RV", IVestimator= "medRV", IQestimator = "medRQ", type= "linear", makeReturns = TRUE)

```r
> data(sample_tdata)
> BNSjumpTest(sample_tdata$PRICE[1:79], QVestimator= "RV", IVestimator= "medRV", +
>        IQestimator = "medRQ", type= "linear", makeReturns = TRUE)
$sztest
[1] -0.6748374

$critical.value
[1] -1.959964  1.959964

$rhovalues
[1] 0.4997791
```
• Other jump tests implemented in highfrequency:
  – Ait-Shalia and Jacod: AJjumptest,
  – Jian and Oomen: JOjumptest
Extensions: Microstructure noise

• In practice, we don’t observe the efficient price, but the price with some microstructure noise (rounding, bid-ask bounce)

\[ p_{observed}^{t_i} = p_{efficient}^{t_i} + \varepsilon_{t_i} \]

• Then there are three sources of variability to distinguish: IV, jump variance and the noise variance:
  – Two time scale estimator: function (R)TSCov in highfrequency
  – Preaveraging: function MRCov in highfrequency
Extensions: Multivariate analysis

- Asset pricing models, portfolio selection, hedging, arbitrage strategies, value-at-Risk forecasts: they typically need a multivariate approach and require a covariance estimate.

- Same challenges as in the univariate case because of the three sources of variability, in addition to estimation troubles coming from non-synchronous trading.
  - If ignored: leads to underestimation of dependence.

- In highfrequency:
  - Multivariate refresh time sampling
  - Several covariance estimators: rCov, rHYCov, rTSCov, rThresholdCov, rOWCov, MRC,...
FORECASTING THE REALIZED VOLATILITY (OPEN TO CLOSE)
HAR model

• Realized volatilities model the open to close variabilities
• It’s of interest to forecast future open to close variability
• This is done through a **Heterogeneous AutoRegressive model** in which the RV is predicted based on averages of k past RV
  – Lagged RV (k=1)
  – Average RV of the past week (k=5)
  – Average RV of the past month (k=22)

\[
RV_t = \beta_0 + \beta_1 RV_{t-1} + \beta_2 \sum_{i=1}^{5} RV_{t-i} + \beta_3 \sum_{i=1}^{22} RV_{t-i} + \epsilon_t
\]

• Note: parsimonious, linear in parameters, so OLS estimation.
# Forecasting daily Realized volatility for DJI 2008
# using the basic harModel: HARRV and give RVs as input

```r
data(realized_library);
#Get sample daily Realized Volatility data
DJI_RV = DJI_RV[!is.na(DJI_RV)];  #Remove NA's
DJI_RV = DJI_RV['2008'];
x = harModel(data=DJI_RV, periods = c(1,5,22), RVest = c("rCov"), type="HARRV", h=1, transform=NULL);
summary(x);
plot(x);
```

50
```
> summary(x);

Call:
"RV1 = beta0 + beta1 * RV1 + beta2 * RV5 + beta3 * RV22"

Residuals:
   Min     1Q   Median     3Q    Max
-0.0017683 -0.0000626 -0.0000427 -0.0000087  0.0044331

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
beta0     4.432e-05   3.695e-05   1.200   0.2315
beta1    1.586e-01    8.089e-02   1.960   0.0512 .
beta2    6.213e-01    1.362e-01   4.560  8.36e-06 ***
beta3    8.721e-02    1.217e-01   0.716   0.4745
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.0004344 on 227 degrees of freedom
Multiple R-squared:  0.4679, Adjusted R-squared:  0.4608
F-statistic: 66.53 on 3 and 227 DF,  p-value: < 2.2e-16
```
• Several extensions of the HAR model: including jumps (type == "HARRVJ"), leverage effects (not implemented yet).

• Limitation: open to close variability

• For forecasting close to close variance using realized measures: see the heavyModel in highfrequency (non-linear: QML estimation).
THE END IS NEAR!
Roadmap (i)

• “There are old traders, there are bold traders, but there are no old bold traders”
• Focus: How to use high frequency price data to understand better the time-varying risk properties of the investment
• Two types of risk: the normal volatility risk and the jump risk
• Topics:
  – Cleaning and aggregation (univariate and multivariate) of tick prices into log-returns
  – Discrete time model for intraday returns:
    • Spot volatility estimation
    • Price jump detection
  – Continuous time model for log-prices
    • Realized volatility estimation
    • Detection of a jump component in realized volatility
  – Forecasting volatility using realized volatility measures.
Roadmap (ii)

- And how to do these analysis with the functions in the R package **highfrequency**
  - Latest version at: [http://r-forge.r-project.org/R/?group_id=1409](http://r-forge.r-project.org/R/?group_id=1409)
- Other functionality: Calculation of liquidity measures (effective spreads, depth imbalance, etc.)
- Convert large multiday files from WRDS, TAQ, Tickdata into xts objects organized by day
- Realized higher order moments: rSkew, rKurt
- **Your contribution?**
References

• Plenty!
• Some of my own: