Using R for backtesting algorithmic trading strategies on high-frequency data

Piotr Wójcik
pwojcik@wne.uw.edu.pl

University of Warsaw
Faculty of Economic Sciences
Chair of Quantitative Finance

eRum 2016
12–14.10.2016, Poznan

Contents

Experience

- since 2011 teaching on **Quantitative Finance M.Sc. programme** in **Warsaw University** (ranked as the **19^{th} worldwide** by EduUniversal Top 100 in Financial Market)

  http://qf.wne.uw.edu.pl/

- 4.5 years experience as Quantitative Researcher in U.S. hedge fund.

Algorithmic trading strategy

Algorithmic trading strategy is the use of computer programs to automate one or more stages of the trading process:

- **pre-trade analysis** (what to trade) – analysis of properties of assets using market data or financial news,
- **trading signal generation** (when to trade) – identifies trading opportunities based on the pre-trade analysis,
- **trade execution** (how to trade) – executing orders for selected assets.

Some definitions and thoughts

- **backtesting** – checking strategy performance on historical data,
- **optimization** – comparison of (very) many combinations of strategy parameters with respect to selected performance measures (return, risk, drawdown, etc.) done on a part of data (**in-sample**),
- selected variant(s) verified on remaining data (**out-of-sample**),
- **high-frequency data** – intra-day data (e.g. hourly, 1-minute, 1-second, tick data),
- this creates **large amounts of data** (1440 minutes, 86400 seconds each day, ca. 252 trading days every year) and requires **massive computations**.

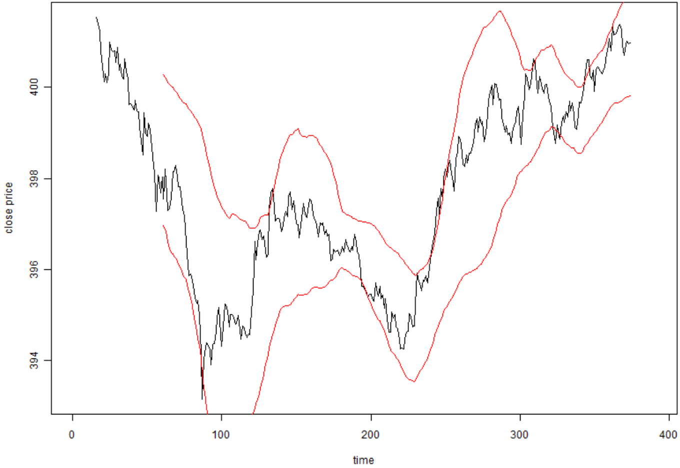
General types of trading strategies

- **trading strategies can be profitable only if asset price is either mean-reverting** or **trending**, otherwise, it is random-walking,
- if one believes that prices are **mean reverting** and that they are currently **low** relative to some reference price, one should **buy** now and plan to sell higher later,
- however, if one believes the prices are **trending** and that they are currently **low**, one should (short) sell now and plan to **buy** at an even lower price later,
- the opposite position should be assumed if one believes that prices are currently high,
- mean-reverting approach can be applied also on a pair of assets which are related to one another (**pair trading**, **spread trading**).

Volatility breakout model

- how to assess if a price change is large enough to indicate a new trend?
- bands are placed a certain distance above and below some measure of a current value (usually smoothed price),
- the distance is determined by recent market volatility (with some multiplier),
- as volatility increases – the bands expand, as it declines – the bands contract,
- one observes how the current price (signal) relates to volatility bands – if bands are crossed, a new position is assumed,
- this approach is called a **volatility breakout** model,
- assumption behind: large moves always begin with small moves.

Volatility breakout model – sample plot



Data and strategy assumptions

- 1-second data for EUR/GBP currency pair,
- quoted 24 hours a day since 00:00 on Monday until 21:00 on Friday (GMT),
- **trend following volatility breakout model**,
- additional assumptions: leave all positions 15 minutes before the weekend and do not assume any position in the first 15 minutes after the weekend,
- **learning data** (**in-sample**) data for 2016-08 (ca. 8.5 mln ticks aggregated to 2.7 mln 1-sec bars),
- **test data** (**out-of-sample**) data for 2016-09 (ca. 8.6 mln ticks aggregated to 2.6 mln 1-sec bars),
- transactional costs assumed: 1 basis point per trade,
- performance assessment based on **annualized net Sharpe Ratio**,

Strategy parameters considered

signal compared with slow MA $+/-$ multiplier $\times$ volatility

- MA type: SMA or EMA,
- MA memory/half-life for the signal: 10, 20, 30, 40, 50, 60, 90, 120 seconds,
- MA memory/half-life for the slow MA: 60, 90, 120, 150, 180, 240, 300, 360, 420, 480, 600 seconds,
- volatility measure: standard deviation (SD), median absolute deviation (MAD),
- volatility memory: 60, 90, 120, 150, 180, 240, 300, 360, 480, 600 seconds,
- volatility mutliplier: from 0.5 to 3.5 by 0.25.

Total number of **42640 combinations of parameters** considered.

Comparison of time efficiency of different approaches in R

```
benchmark(SMA(dane$EURGBP, 10),
          SMA(coredata(dane$EURGBP), 10),
          runmean(dane$EURGBP, 10),
          runmean(coredata(dane$EURGBP), 10),
          rollapply(dane$EURGBP, 10,
                    FUN = function(x) mean(x, na.rm = T),
                    align = "right"),
          rollapply(coredata(dane$EURGBP), 10,
                    FUN = function(x) mean(x, na.rm = T),
                    align = "right"),
          rollmeanr(dane$EURGBP, 10),
          rollmeanr(coredata(dane$EURGBP), 10)
          )
```

Time efficiency of different MA or **rolling** stdev functions (in milliseconds)

	function (package)	applied on	replications	elapsed	relative
MOVING AVERAGE	runmean (caTools)	numeric vector	100	6.63	1.00
	SMA (TTR)	numeric vector	100	15.95	2.41
	SMA (TTR)	xts	100	32.47	4.90
	rollmean (zoo)	numeric vector	100	597.86	90.18
	runmean (caTools)	xts	100	1278.23	192.80
	rollmean (zoo)	xts	100	4072.89	614.31
	rollapply, mean (zoo)	numeric vector	100	7168.56	1081.23
	rollapply, mean (zoo)	xts	100	11743.48	1771.26
STD DEV	runsd (caTools)	numeric vector	100	34.73	1.00
	runSD (TTR)	numeric vector	100	85.36	2.46
	runSD (TTR)	xts	100	142.49	4.10
	runsd (caTools)	xts	100	1401.65	40.36
	rollapply, sd (zoo)	numeric vector	100	11977.17	344.87
	rollapply, sd (zoo)	xts	100	15949.98	459.26

Note: 2.7 mln observations, 100 replications each.

Loops in R are slow and C++ might help (Rcpp)

pure R

```r
positionR<-function(signal, lower, upper, pos_flat, strategy = "m
{
  # lets first create a vector of 0s
  position<-rep(0, length(signal))

  for (i in 2:length(signal))
  {
    if ( pos_flat[i] == 1 ) position[i] <- 0
    else
    { # check if values are nonmissing (otherwise calculations nc
      if (!is.na(signal[i-1]) &
            !is.na(upper[i-1]) &
            !is.na(lower[i-1]))
      {
        # what if previous position was 0
        if (position[i-1] == 0){
          if (signal[i-1] > upper[i-1]){position[i] <- -1}
          if (signal[i-1] < lower[i-1]){position[i] <- 1}
        } else if (position[i-1] == -1){
          # what if previous position was -1
          if (signal[i-1] > lower[i-1]){position[i] <- -1}
          if (signal[i-1] < lower[i-1]){position[i] <- 1}
        } else if (position[i-1] == 1){
          # what if previous position was 1
          if (signal[i-1] < upper[i-1]){position[i] <- 1}
          if (signal[i-1] > upper[i-1]){position[i] <- -1}
        }
      } else position[i] <- position[i-1]
      # if anything is missing, keep previous position
    }
  }
  # reverse the position if we use a momentum ("mom") strategy
  if(strategy == "mom") position <- (-position)

  return(position)
}
```

C++/Rcpp

```cpp
/* initiate vector of positions with 0s */
for( int i=0; i<n; i++)
  {pos[i] = 0;}

    /* position calculation */
          for( int i=1; i<n; i++)
           {
            if ( pos_flat[i]==1 )
             {pos[i]=0;}
             else
             { /* if anything is missing, keep previous position */
              if ((NA_p[i-1] || NA_u[i-1] || NA_l[i-1]))
              {pos[i] = pos[i-1];}
              else
              {
                 /* what if previous position was 0 */
                 if (pos[i-1]==0){
                   if (signal[i-1] > upper[i-1])
                     {pos[i] = -1;}
                   if (signal[i-1] < lower[i-1])
                     {pos[i] = 1;}
                 } else if (pos[i-1]==-1){
                   /* what if previous position was -1 */
                   if (signal[i-1] > lower[i-1])
                     {pos[i] = -1;}
                   if (signal[i-1] < lower[i-1])
                     {pos[i] = 1;}
                 } else if (pos[i-1]==1){
                   /* what if previous position was 1 */
                   if (signal[i-1] < upper[i-1])
                    {pos[i] = 1;}
                   if (signal[i-1] > upper[i-1])
                    {pos[i] = -1;}
                 }
              }
             }
           }
/* reverse the position if we use a momentum ("mom") strategy */
if(strategy[0]=="mom")
   {for( int i=0; i<n; i++) {pos[i]=(-pos[i]);}}

/* return the vector of positions */
  return pos;
```

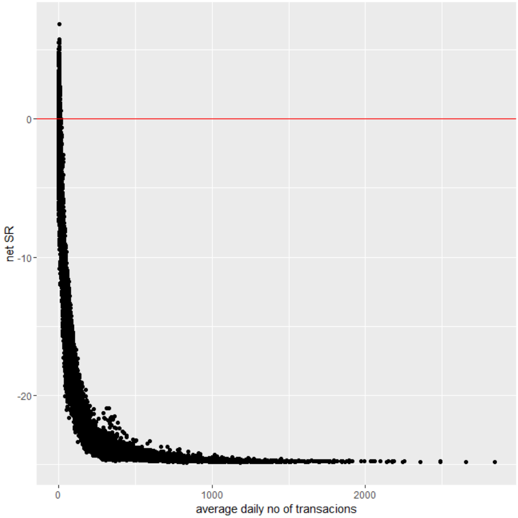
Time efficiency of a loop calculating positions (in milliseconds)

approach	applied on	replications	elapsed	relative
Rcpp	numeric vector	100	14.82	1.00
Rcpp	xts	100	21.22	1.43
R	numeric vector	100	1437.45	96.99

Note: 2.7 mln observations, 100 replications each.

In case of more complex statistical models the use of RcppArmadillo library strongly recommended to improve time efficiency of calculations.

In-sample profitability (net SR) vs trading frequency (average daily trades)



- 9.5% combinations had net SR>0,
- 8.5% combinations had net SR>1,

Results for top 10 in-sample combinations (based on net SR)

parameters				in-sample (2016-08)		
signal	slowMA	volat	m	gross SR	net SR	av daily ntrans
SMA50	SMA60	mad480	3	8.30	6.83	3.3
SMA50	SMA60	mad480	2.75	8.29	6.82	3.3
EMA20	EMA60	sd60	3.25	7.61	5.75	5.5
SMA50	SMA60	mad480	2.5	7.12	5.61	3.5
SMA50	SMA60	mad480	3.25	6.96	5.59	3.2
SMA30	SMA60	sd600	2.25	5.62	5.51	0.2
SMA30	SMA60	sd480	2.25	5.53	5.50	0.1
EMA50	EMA90	sd120	1	6.74	5.49	3.9
EMA90	EMA120	mad480	2.5	5.90	5.47	1.0
EMA90	EMA120	mad480	2.75	5.90	5.47	1.0

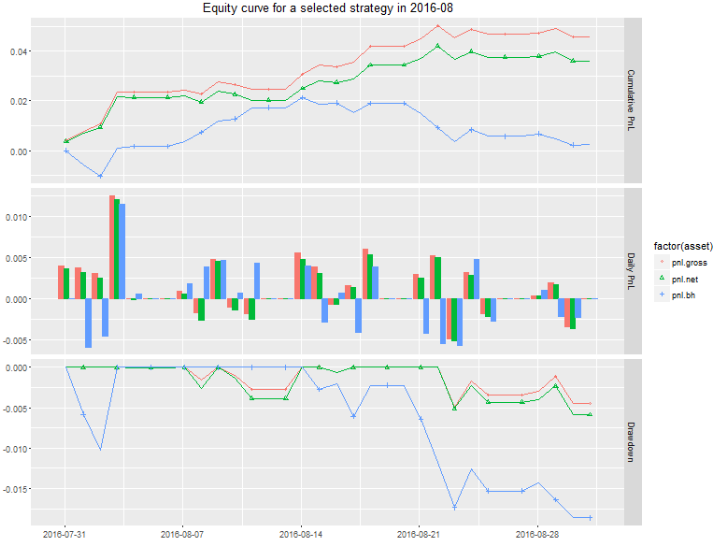
- very high in-sample net Sharpe Ratios in all cases,
- relatively small trading frequency – at most 3–5 trades a day is not a ”real” high frequency trading.

Results for top 10 in-sample combinations (based on net SR)

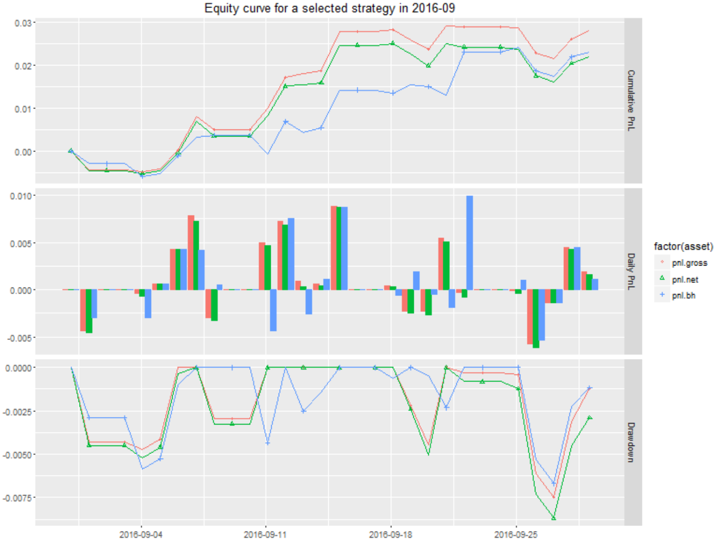
parameters				in-sample (2016-08)			out-of-sample (2016-09)		
signal	slowMA	volat	m	gross SR	net SR	av daily ntrans	gross SR	net SR	av daily ntrans
SMA50	SMA60	mad480	3	8.30	6.83	3.3	-5.05	-5.39	0.8
SMA50	SMA60	mad480	2.75	8.29	6.82	3.3	-2.10	-2.48	1.0
EMA20	EMA60	sd60	3.25	7.61	5.75	5.5	1.82	0.70	3.5
SMA50	SMA60	mad480	2.5	7.12	5.61	3.5	-2.09	-2.46	1.0
SMA50	SMA60	mad480	3.25	6.96	5.59	3.2	-2.19	-2.45	0.6
SMA30	SMA60	sd600	2.25	5.62	5.51	0.2	-3.88	-4.13	0.6
SMA30	SMA60	sd480	2.25	5.53	5.50	0.1	-3.36	-3.50	0.3
EMA50	EMA90	sd120	1	6.74	5.49	3.9	4.32	3.45	2.6
EMA90	EMA120	mad480	2.5	5.90	5.47	1.0	1.22	1.02	0.5
EMA90	EMA120	mad480	2.75	5.90	5.47	1.0	0.29	0.08	0.5

- most of in-sample top 10 strategies unsuccessful in testing period (out-of-sample),
- only one (row 8) relatively stable – still might be a coincidence.

Cumulative profit-and-loss (PnL) of a selected strategy in 2016-08



Cumulative profit-and-loss (PnL) of a selected strategy in 2016-09



Worth to keep in mind. . .

- it is quite easy to find a strategy which works well on learning data (in-sample),
- the most challenging part is designing a strategy which is consistently profitable also in the out-of-sample period and when implemented in real trading,
- **past performance is never a guarantee of future returns**,
- high-frequency trading is difficult, but can generate stable profits under various market conditions,
- large datasets and large number of considered strategy variants require efficient cmputational tools, which R provides with the help of C++,
- strategies made public soon become obsolete,
- best-performing strategies are those kept in the strictest confidence.

R in Finance – annual conference (`www.rinfinance.com`)

R/Finance home agenda register travel committee previous

R/Finance 2016: Applied Finance with R

May 20 & 21, Chicago, IL, USA

The conference is sold out but has a waitlist.

> home(2016)

The eighth annual R/Finance conference for applied finance using **R**, the premier free software system for statistical computation and graphics, will be held on May 20 and 21, 2016 in Chicago, IL, USA at the **University of Illinois at Chicago**. The two-day conference will cover topics including portfolio management, time series analysis, advanced risk tools, high-performance computing, market microstructure, and econometrics. All will be discussed within the context of using R as a primary tool for financial risk management, portfolio construction, and trading. Over the past seven years, R/Finance has included attendees from around the world. It featured presentations from prominent academics and practitioners, and we expect another exciting line-up for 2016.

For 2016, we once again invite you to submit complete papers in pdf format for consideration. We will also consider one-page abstracts (in txt or pdf format) although more complete papers are preferred. We welcome submissions for both full talks and abbreviated "lightning talks". Both academic and practitioner proposals related to R are encouraged. Presenters are strongly encouraged to provide working R code to accompany the presentation/paper. Data sets should also be made public for the purposes of reproducibility (though we realize this may be limited due to contracts with data vendors). Preference may be given to presenters who have released R

We are very excited about the keynote speakers for 2016:

- Patrick Burns,
- Frank Diebold,
- Tarek Eldin, and
- Rishi Narang.

The inaugural 2009 conference featured keynotes by Patrick Burns, Robert Grossman, David Kane, Roger Koenker, David Ruppert, Diethelm Wuertz, and Eric Zivot, as well as a number of excellent presentations.

The 2010 conference contained keynotes by Bernhard Pfaff, Ralph Vince, Marc Wildi, and Achim Zeileis. This was followed in 2011 with keynotes by Meb Faber, Stefano Iacus, John Bollinger and Louis Kates.

The 2012 conference had keynotes from Blair Hull, Paul Gilbert, Rob McCulloch, and Simon Urbanek. The 2013 conference featured keynotes by Sanjiv Das, Attilio Meucci, Ryan Sheftel, and Ruey Tsay.

Selected commercial intraday data sources

- NYSE Trade and Quotes (TAQ) – interfaced in R by RTAQ package,
- Bloomberg (American equities) – interfaced in R by Rblpapi package,
- Olsen & Associates (FX),
- Interactive Brokers – interfaced in R by IBrokers package
- tickdata – `https://www.tickdata.com/historical-market-data-products/`,
- quantquote – `https://quantquote.com/`,
- Πtrading.com – `http://pitrading.com/historical-data.html`,
- kibot – `http://www.kibot.com/`

Selected free intraday data sources

- Ducascopy (all major currency pairs, some commodities, indices
  and many stocks),
- stooq (5-minute and 1-hourly data for last month from many exchanges),
- Bossa (Polish stocks, bonds and futures),
- TrueFX (intraday data for currency pairs) – interfaced in R by TFX package,
- HistData.com (intraday data for currency pairs),
- Google finance (indirectly).

Selected blogs on trading and R

- The R Trader – http://www.thertrader.com/
- QuantStrat TradeR – https://quantstrattrader.wordpress.com/
- Gekko Quant – http://gekkoquant.com/
- Dekalog Blog – http://dekalogblog.blogspot.co.uk/
- FOSS Trading – http://blog.fosstrading.com
- R-bloggers – https://www.r-bloggers.com/search/trading/
- and many other

Useful R packages

- zoo, xts, chron – dealing with time series, dates and times,
- quantmod – nice charting,
- IBrokers, Rblpapi, TFX – R interface to different data providers,
- tseries – sharpe, maxdrawdown and functions related to irregular time-series objects,
- PerformanceAnalytics – many strategy performance measures, charting functions and functions for measuring co-movements,
- TTR, caTools – many efficient functions for calculations on a rolling window,
- Rcpp – provides C++ classes that greatly facilitate interfacing C or C++ code in R,
- inline – allows to define R functions with in-lined C, C++ (or Fortran) code,
- RcppArmadillo – Rcpp Integration for the 'Armadillo' Templated Linear Algebra Library – a C++ linear algebra library,
- rbenchmark, microbenchmark – comparison of time efficiency of different approaches.

References

Dziubinski, M. P. (2016), 'Getting the most out of Rcpp: High-performance C++ in practice'. R in Finance 2016.

Eddelbuettel, D. and Francois, R. (2011), "Rcpp: Seamless R and C++ integration", *Journal of Statistical Software* , Vol. 8, pp. 1–18.

Eddelbuettel, D., Francois, R. and Bates, D. (2016), 'RcppArmadillo: Rcpp integration for Armadillo templated linear algebra library'. R package version 0.6.700.6.0, URL http://CRAN.R-Project.org/package=RcppArmadillo.

Georgakopoulos, H. (2015), *Quantitative Trading with R. Understanding Mathematical and Computational Tools from a Quant's Perspective*, Palgrave Macmillan.

Paciorek, C. (2014), 'C++ for statisticians, with a focus on interfacing from R and R packages'. University of California, Department of Statistics workshop, http://www.stat.berkeley.edu/scf/paciorek-cppWorkshop.pdf.

Thank you for your attention!

pwojcik@wne.uw.edu.pl
http://wne.uw.edu.pl/pwojcik