AlgoTCA - GSoC 2019

by Vito Lestingi, Jasen K Mackie, Brian G Peterson

Abstract As part of the R organization's participation in the Google Summer of Code 2019 we replicate, and to some degree extend, the pre and post-trade TCA models as disclosed in "The Science of Algorithmic Trading and Portfolio Management" – Robert L. Kissell. The main focus of this 3-month programme was to build an Implementation Shortfall TCA model and a range of benchmark price performance models (VWAP, PWP, RPM, Arrival Cost etc), models for statistically comparing different algorithmic execution performances and lastly replicating market impact models, Kissell's I-Star model and the Almgren-Chriss model. Using 11 months of JSE data and 16200 observations we are able to train a non-linear least squares estimation function which converges on a plausible solution, allowing us to validate the model which can be used in production for larger datasets with presumably lower variance.

Introduction

Post-trade Transaction Cost Analysis (TCA) is critical for assessing the efficiency with which trading decisions are implemented, especially for the institutional investor whose trade ideas can span multiple time periods to implement and which risks adverse market impact if not managed accordingly. Pre-trade and intra-day TCA is an equally vital component for making optimal trading decisions based on market conditions and trading constraints. The blotter package in R is an ideal package in the R ecosystem for hosting TCA functionality since it serves as the transaction infrastructure for building and analysing simulated and production trading systems. In this article we describe the functions implemented as part of this GSoC 2019 project in 3 parts, namely: 1. Post-trade TCA models, 2. Models for the statistical comparison of algorithmic executions, 3. Pre-trade Market Impact estimation using the I* model as documented in "The Science of Algorithmic Trading and Portfolio Management" – Robert L. Kissell, originally developed by Kissell and Malamut in 1998. Where appropriate we include examples, elaborate on extensions, document the journey of a supposed analyst and communicate results and findings relevant to a user. Lastly we comment on interesting paths of future work.

1. Post Trade TCA

Implementation Shortfall

Implementation Shortfall is a measure that represents the total cost of executing an investment idea. Implementation Shortfall is calculated as the difference between the paper return of a portfolio where all shares are assumed to have transacted at the manager's decision price and the actual return of the portfolio using actual transaction prices and shares executed.

Implementation Shortfall can be implemented in 4 ways:

1. Assuming Complete Execution, implying a zero opportunity cost component. This is the default method for *Implementation Shortfall*

The simplest formulation for the Complete Execution IS method is:

$$IS = SP_{avg} - SP_d + fees$$

Note that we add fees, as a positive metric indicates a cost.

2. Using an Opportunity Cost (Perold 1988) component, where not all shares originally allocated for trading are finally executed. Opportunity Cost is the cost of not executing a portion of the originally allocated S shares for execution. This could be due to limit price constraints or a lack of liquidity.

The formulation for Opportunity Cost is:

$$(S - \sum s_j)(P_n - P_d)$$

The Implementation Shortfall formulation of Perold (1988) can be written as:

$$IS = \sum s_j(P_{avg} - P_d) + (S - \sum s_j)(P_n - P_d) + fees$$

3. Expanded Implementation Shortfall (Wayne Wagner)

Wayne Wagner's implementation categorizes costs into delay, trading and opportunity related costs. Assuming P_d is the decision price, P_0 is the price when trading begins (ideally Arrival Price, defined as the mid-price at order arrival, alternatively Last Price at order arrival or failing that data availability then first transaction price), and P_n is the price at the end of trading. The Expanded IS can be written as:

$$(P_n - P_d) = (P_n - P_0) + (P_0 - P_d)$$

If you substitute the RHS into Perold's IS, then IS can be written as:

$$IS = (\sum s_j p_j - \sum s_j P_d) + (S - \sum s_j)((P_n - P_0) + (P_0 - P_d)) + fees$$

This formula can be re-written into a separate delay, trading and opportunity cost related component as follows:

$$IS_{expanded} = S(P_0 - P_d) + (\sum s_j)(P_{avg} - P_0) + (S - \sum s_j)(P_n - P_0) + fees$$

where each term (excluding fees) reflects the delay, trading and opportunity cost components respectively. Wagner's method allows for an additional decomposition of the "Delay related cost" into the *opportunity delay cost* component and the *trading delay cost* component. Our implementation provides this breakdown in the output.

4. Market Activity IS, which assumes the analyst is unaware of the manager's decision price. This method is equivalent to the Wagner formulation except that the first term is excluded in order to assess only market activity IS:

$$IS_{MktAct} = (\sum s_j)(P_{avg} - P_0) + (S - \sum s_j)(P_n - P_0) + fees$$

Implementation Shortfall - Examples

For the IS examples we borrow from the code in the function help documentation. The first 2 examples below illustrate the output for the Complete Execution and Market Activity implementations using identical datasets. The Implementation Shortfall quantums are identical as we assume complete execution in both instances including the Market Activity IS, although you will notice this version returns additional info when compared with the Complete Execution version, including filled and unfilled units, the opportunity cost component and fees.

```
### Complete Execution IS
impShortfall("testport", "test_txns",
          paQty=5000,
          priceStart=10,
          priceEnd=11,
          arrPrice=10,
          method='Complete')

#>      Symbol   Method Paper.Ret Actual.Ret Shortfall
#> 1 test_txns Complete      5000       2400      2600

### Market Activity IS
impShortfall("testport", "test_txns",
          paQty=5000,
          priceEnd=11,
          arrPrice = 10,
          method='Market')

#>      Symbol Method t.Txn.Qty u.Txn.Qty Trade.Cost Opp.Cost Fees Shortfall
#> 1 test_txns Market      5000         0       2500        0  100      2600
```

The next 2 examples illustrate the Perold and Wagner versions, respectively, again using identical datasets in both cases. We assume only 4,000 of the originially allocated 5,000 units eventually trade.

For the Perold output, we see the breakdown between Execution Cost, Opportunity Cost and Fees separately.

```
### Perold
impShortfall("testport", "test_txns",
          paQty=5000,
          priceStart=10,
          priceEnd=11,
          arrPrice=10,
          method='Perold')

#>      Symbol Method t.Txn.Qty u.Txn.Qty Exe.Cost Opp.Cost Fees Shortfall
#> 1 test_txns Perold      4000      1000     2000     1000   80      3080
```

In the case of the Wagner implementation, we see the extra breakdown compared with Perold, ultimately equating to the same hypothetical Implementation Shortfall measure.

```
### Wagner
impShortfall("testport", "test_txns",
          paQty=5000,
          priceStart=10,
          priceEnd=11,
          arrPrice=10.25,
          method='Wagner')

#>      Symbol Method t.Txn.Qty u.Txn.Qty Opp.Delay Trade.Delay Delay.Cost
#> 1 test_txns Wagner      4000      1000       250        1000       1250
#>   Trade.Cost Opp.Cost Fees Shortfall
#> 1       1000      750   80      3080
```

Benchmark Price Performance - Examples

We built a single function for computing TCA benchmarks where the benchmark price is one of: Arrival Price, Day's Open/Close/Other, Participation Weighted Price (PWP), Volume Weighted Average Price (VWAP) or a qualitative score with the Realtive Performance Measure (RPM). When all methods are used in conjunction, an analyst can derive measures for each benchmark giving a different view of the resultant execution. In addition to giving an overall score, we have built an S3 method for plotting the performance through the life of the execution which allows an analyst to identify specific time periods that warrant additional review.

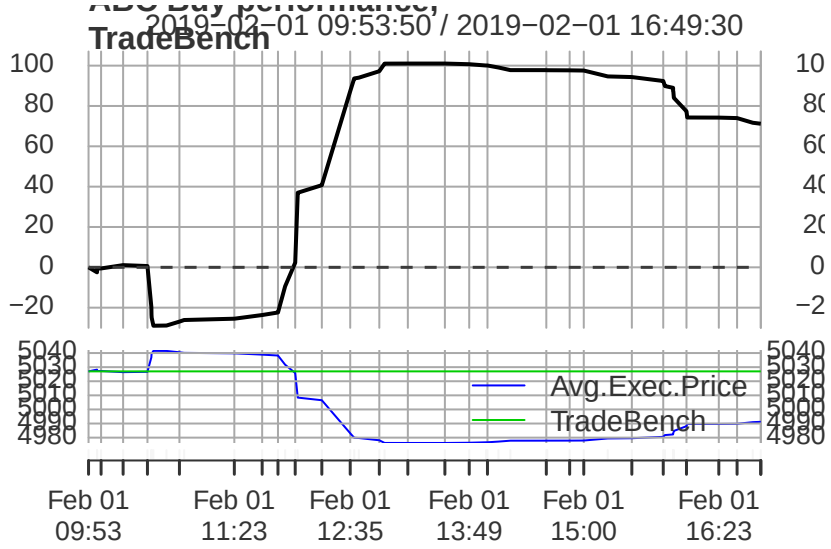
benchmark='TradeBench'

For all 4 methods we use the same sampled set of public trades data to simulate a private execution. In all cases, a positive value indicates outperformance and a negative value, underperformance. In the case of `benchmark='TradeBench'` the benchmark price is the first transaction price in the execution, as a proxy for the Arrival Price. For a true Arrival Cost benchmark using the actual Arrival Price, the user can use `benchmark='MktBench'` with the Arrival Price specified for with the 'priceToBench' argument.

```
benchTradeBench <- benchTradePerf('abc.port.day', 'ABC', side = 1, benchmark = 'TradeBench', MktData = ABC.da
last(benchTradeBench$Trades.TradeBench.Perf)

#>                  Dates Symbol Side Avg.Exec.Price TradeBench Performance
#> 50 2019-02-01 16:49:30    ABC  Buy       4991.218       5027    71.17988

plot(benchTradeBench, benchmark = 'TradeBench')
```



benchmark='MktBench'

As alluded to previously, with `benchmark='MktBench'` the user can specify "Open", "Close" or a numeric price value for benchmarking.

```
# performance against daily open price
benchMktBenchOpen <- benchTradePerf('abc.port.day', 'ABC', side = 1, benchmark = 'MktBench',
                                    type = list(price = 'Open'), MktData = ABC.day[1])
last(benchMktBenchOpen$Trades.MktBench.Perf)

#>                  Dates Symbol Side Avg.Exec.Price MktBench.Open
#> 50 2019-02-01 16:49:30    ABC  Buy       4991.218          5000
#>    Performance
#> 50    17.56425

plot(benchMktBenchOpen, benchmark = 'MktBench')


# performance against daily closing price
benchMktBenchClose <- benchTradePerf('abc.port.day', 'ABC', side = 1, benchmark = 'MktBench',
                                     type = list(price = 'Close'), MktData = ABC.day[nrow(ABC.day)])
last(benchMktBenchClose$Trades.MktBench.Perf)

#>                  Dates Symbol Side Avg.Exec.Price MktBench.Close
#> 50 2019-02-01 16:49:30    ABC  Buy       4991.218           5037
#>    Performance
#> 50    90.89166

plot(benchMktBenchClose, benchmark = 'MktBench')
```

benchmark='VWAP'

A widely used benchmark is the *Volume Weighted Average Price* (VWAP) benchmark. The benchmark is defined as:

$$VWAP = \frac{\sum P_j Q_j}{\sum Q_j}$$

where P_j is the market price and Q_j the market volume, during j trading periods activity of the market. Two different types of VWAP benchmarks are included in the present function, the Interval VWAP and the Full VWAP. Referring to the former as the VWAP where the j market trading periods considered are the ones during which the order is being executed, whereas the latter includes all the j market periods from order execution beginning to last transaction. The VWAP benchmark varies by timespan considered and is commonly used as a proxy for fair market price. It can differ by data vendors specific market data filtering. There are recognized drawbacks of this benchamrk. First of all,

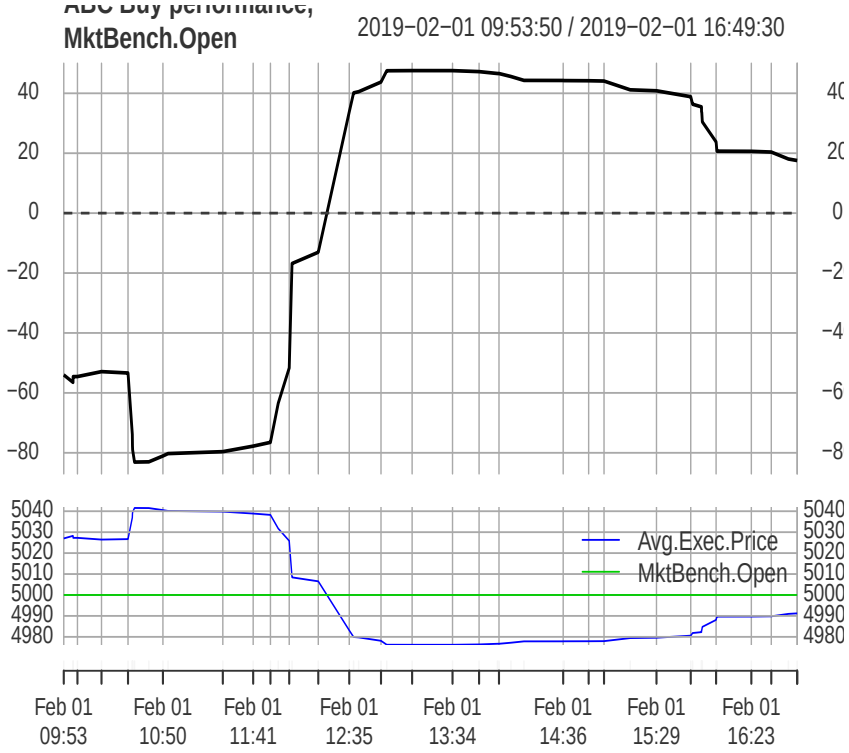


Figure 1: Performance vs. open price

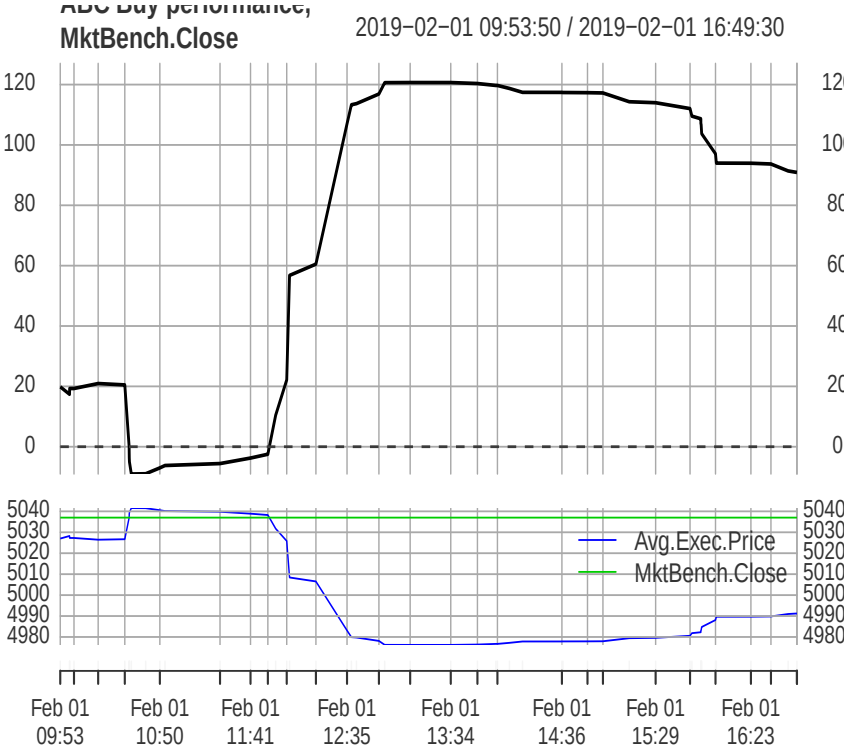


Figure 2: Performance vs. close price

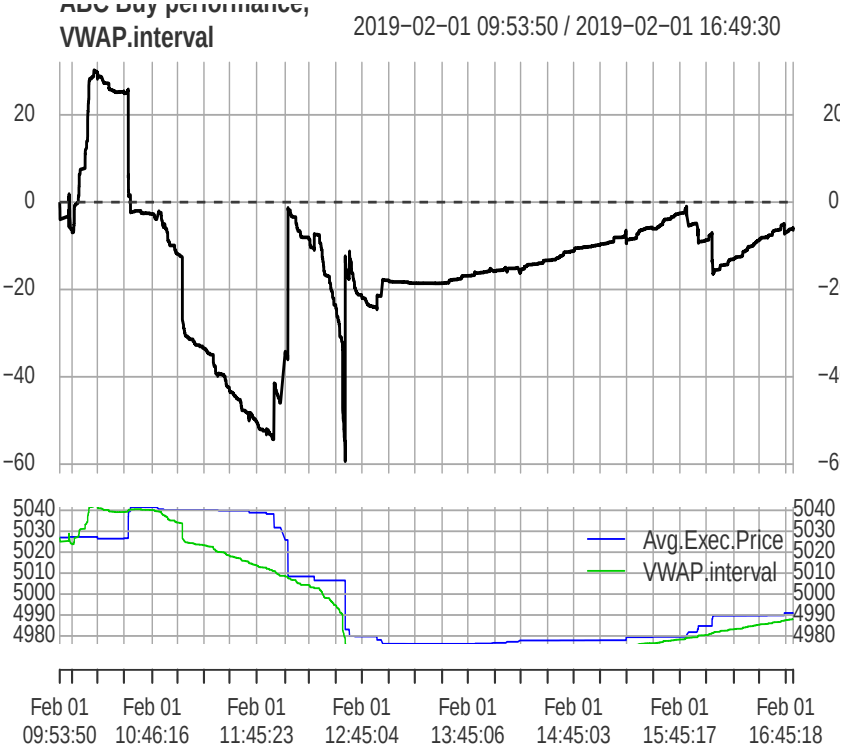


Figure 3: Performance vs. interval-VWAP

the larger the order the closer the execution will be to VWAP. Second, where large block trades occur these could skew the benchmark. Lastly, it is not an indicated comparison across stocks or different days for the same stock.

```
benchVWAPinterv <- benchTradePerf('abc.port.day', 'ABC', side = 1, benchmark = 'VWAP',
                                  type = list(vwap = 'interval'), MktData = ABC.day)
last(benchVWAPinterv$Trades.VWAP.Perf)

#>                   Dates Symbol Side Avg.Exec.Price VWAP.interval
#> 7091 2019-02-01 16:49:30    ABC  Buy       4991.218      4988.044
#>      Performance
#> 7091   -6.363777

plot(benchVWAPinterv, benchmark = 'MktBench')
```

benchmark='PWP'

A variation of the VWAP benchmark is given by the Participation Weighted Price (PWP) benchmark, where the weighting is with respect to the PWP shares:

$$PWP_{shares} = \frac{Tradedshares}{POV}$$

POV refers to the percentage of volume.

The PWP benchwark is:

$$PWP_{price} = \frac{\sum P_h Q_h}{\sum Q_h}$$

where h are the periods from the arrival time of the order into the market until when the PWP shares are completely executed. As the VWAP, the PWP benchmark provides a glimpse into market fair prices. However, this benchmark have limitations similar to the VWAP. It is subject to manipulation in that the market price can be kept inflated by larger orders. Furthermore, as the VWAP, it is not comparable between stocks or across days for the same stock. Also, the benchmark may be biased by temporary impact dissipation.

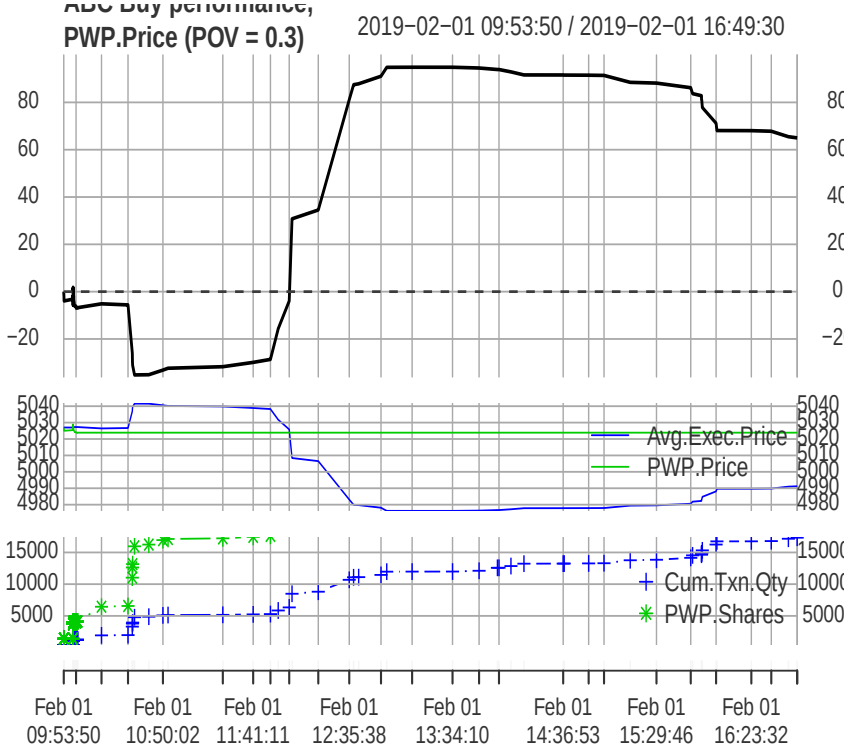


Figure 4: Performance vs. PWP price

```
benchPWP <- benchTradePerf('abc.port.day', 'ABC', side = 1, benchmark = 'PWP', POV = 0.3, MktData = ABC.day)
last(benchPWP$Trades.PWP.Perf)

#>                   Dates Symbol Side Cum.Txn.Qty POV PWP.Shares
#> 164 2019-02-01 16:49:30    ABC  Buy       17322 0.3      57740
#>     Avg.Exec.Price PWP.Price Performance
#> 164       4991.218  5023.869    64.99206

plot(benchPWP, benchmark = 'PWP', main = "")
```

benchmark='RPM'

Lastly, the Relative Performance Measure (RPM), which differs from the PnL metrics above, is a percentile ranking of trading activity. Its expression depends on the side of the trade:

$$RPM_{buy} = \frac{Totalvolume + VolumeatP > P_{avg} - VolumeatP < P_{avg}}{2 * Totalvolume}$$
$$RPM_{sell} = \frac{Totalvolume + VolumeatP < P_{avg} - VolumeatP > P_{avg}}{2 * Totalvolume}$$

where P is the market price specified. The an RPM over 50% is considered as an indication of superior trades, more precisely the RPM can be mapped to a qualitative score of the trades:

RPM	Quality
$0 \leq RPM < 20$	Fair
$20 \leq RPM < 40$	Poor
$40 \leq RPM \leq 60$	Average
$60 < RPM \leq 80$	Good
$80 < RPM \leq 100$	Excellent

This measure is considered as preferred to the VWAP metric because it overcomes some of its drawbacks: it can be used to compare performance across different stocks, days, and volatility; it is not less influenced by large blocks trade at extreme prices.

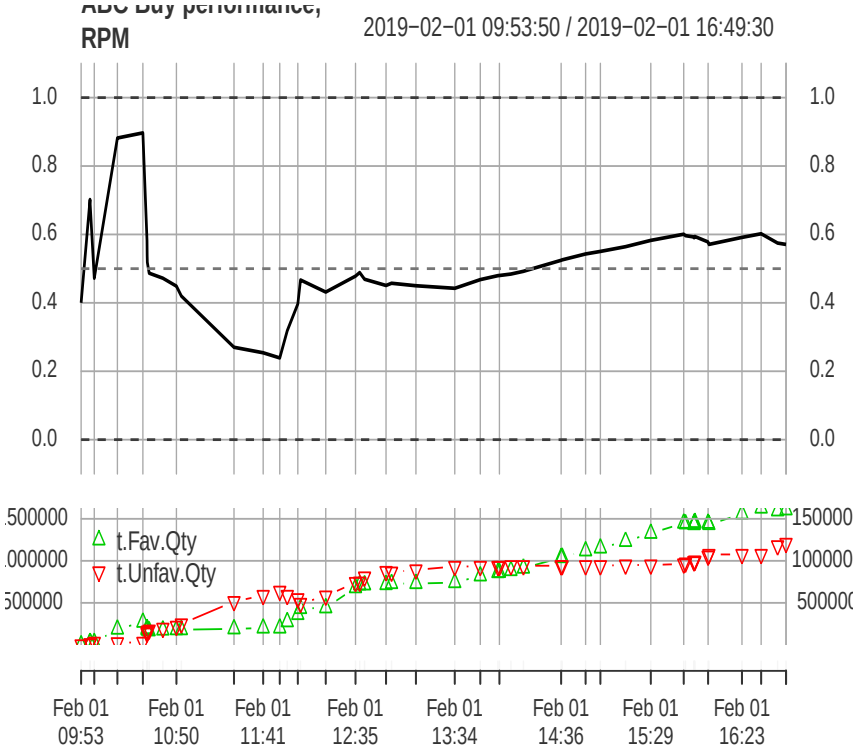


Figure 5: Execution RPM

```
benchRPM <- benchTradePerf('abc.port.day', 'ABC', side = 1, benchmark = 'RPM', MktData = ABC.day)
last(benchRPM$Trades.RPM.Perf)
```

```
#>                  Dates Symbol Side Avg.Exec.Price Mkt.Price t.Mkt.Qty
#> 50 2019-02-01 16:49:30    ABC  Buy       4991.218      5012   2814601
#>    t.Fav.Qty t.Unfav.Qty      RPM Quality
#> 50   1606594     1208007 0.570807 Average
```

```
plot(benchRPM, benchmark = 'RPM', legend.loc = 'topleft')
```

Statistical testing of benchamrked strategies

When testing trading strategies on Symbols, assessing whether there is a statistical significant difference in their performance is of interest. In other words, the goal is determining which given strategy outperformed the other or if they statistically bear the same results in terms of performance. Let us consider two execution strategies or equivalently two brokers using the "same" strategy, the baseline two-sided hypotesis we test is in their performance median:

$$H_0 : med(A) = med(B)$$
$$H_1 : maymed(A) \neq med(B)$$

All the statistical test included are *non-parametric tests*, that is distribution-free tests. These tests allow great flexibility, but in turn require that data verifies some assumptions in order for their results to be meaningful.
There exists a wide range of algorithmic trading strategies and even whithin the same category of strategies many sligthly different versions may exist as they often differ by brokerage firm. In a post-trade analysis framework, starting from trading strategies transactions prices we compare the overall performance of multiple orders, each executed under two different categories, to ultimately test whether data supports a difference in their median. In other words, the basic statistical hypothesis test setting is to test the null hypothesis of a same median against the alternative hypothesis of a different median.

Two statistical testing approach are contemplated, namely the *paired samples approach* and the *independent samples approach*. The suitability of the approach ultimately relies on analysts' specific research questions. Also, each of them critically depends on how transactional data was obtained in

the first place and on its distributional properties, therefore in turn one may be interested in studying the distribution of the overall performance/cost across the orders that is investigating the underlying data generating process. In R all the statistical tests we use are from the `stats` package.

```
# TODO: in 'benchTradeStats.R' docs I provided many usage examples
#
# However, these examples are based on "silly numbers" given only for sake of showing the function.
#
# If we are to add into the vignette examples on real (possibly anonymized) private orders,
# as I wish so that we can meaningfully interpret the tests, the provided code needs to refer
# to appropriate data that has to be adjusted accordingly.
#
# Then is simply a metter of chosing wich test to apply and show the different examples below
#
```

Paired samples approach

The *paired samples approach*, where: trades are two equal-length child orders belonging to the same parent order on a Symbol and each occurred in the same exact timeframe. Following Kissell, the preferred comparison metric against which trades have to be benchmarked is the VWAP benchmark. In this context, tests included are the *Wicolxon Signed Rank test* and the *Sign test*.

Independent samples approach

The *independent samples approach*, where trades can be on different symbols, that may have occured over different periods and possibly with different frequency; here Kissell suggests the Arrival Cost as the preferred trades benchmark metric. In this context, tests included are the *Median test* and the *Wicolxon-Mann-Withney test*.

Data generating process

In addition to the statistical tests above, it is possible to study the distribution of the overall performance/cost across the orders in order to support hypothesis on whether they come from the same distribution or not. Such analysis is said a *distribution analysis*, which for our purposes reduces to a *data generating process (DGP)* comparison. Statistical tests implemented in this framework are the *Chi-Square goodness-of-fit* test and the *Kolmogorov-Smirnoff goodness-of-fit* test.

Market impact models

I-Star model

The model provided by Kissell-Malamut, presented in Kissell (2013), is a cost allocation method to quantify the market impact of financial transactions, depending on an agent order size relative to the market volume. It is theoretically based on the supply-demand principle, although it may be rather difficult to express ourselves precisely in these terms and even so our interpretations may differ by the several possible scenarios that take place into the market in response to imbalances.

Theoretically the I-Star model can be estimated using private order data for which one intends to estimate the impact costs. The main limitations of this approach are: on one hand the lack of data and the effect of neglecting the effect of wider market movements than the ones of the single security on which the order was placed, on the other it may include potential opportunistic trading biases. Based on these considerations we follow Kissell's main discussion line, focusing on the use of market "tic data" and derived quantities that represent proxies of the corresponding order-related variables.

Market tick data and variables

In its most genearl setting, the model is based on market "tic data" only. It is difficult to relate Kissell's provided notion of "tick data" with respect to current data provision standards, which in turn may also vary by data vendors. Here should suffice to mention that an ideal market intraday dataset to input into the model includes trades prices and volumes, "bid" and "ask" prices in order to compute the spreads and possibly the so called "reason" (i.e, the classification of trades as "bid" or "ask"); for each security involved in the analysis.

All the historical variables needed the model are computed internally from market data and most of them are "rolling end-of-day quantities", meaning that they are based on previous variables over a specified *horizon* ($t = 1, ..., T$) that rolls one step ahead until data available allows. Some variables are annualized and hence need the total number of business days in a given market and within a given year (typically a factor of 252 days, in the US markets, or of 250 days), we denote it T_m. Below we provide a schematic view of these and other quantities involved are defined below:

- *Arrival Price*. Ideally is the first bid-ask spreads midpoint. When missing spread data, the first daily market price is used as a proxy.
- *Annualized volatility*. Is the standard deviation of the close-to-close security returns, scaled on the number of business days in a given year:

$$\sigma = \sqrt{\frac{T_m}{T-1}\sum_{t=2}^{T}(r_i - r_{avg})^2}$$

  expressed in decimal units.
- *Average Daily Volume* (ADV). Over the specified horizon:

$$ADV = \frac{1}{T}\sum_{t}^{T} V_t$$

- *Imbalance* (Q). It is calculated from "buy initiated trades" and "sell initiated trades". When trade 'Reason' is already available there is no need to explicitly infere trades direction. In cases such a 'Reason' is missing, the Lee-Ready *tick test* will be used to infere trading direction. In its essence, the test is based on determining the sign of price changes: uptick or zero-uptick trades are considered "buy initiated", whereas downtick or zero-downtick trades are counted as "sell initiated". We express it as

$$Q_t = |\sum V_{buy} - \sum V_{sell}|$$

  where V_{buy} and V_{sell} stands for total market volume of buy or sell initiated trades, respectively. To note is that, as the "reason" refers to each trade, "buy initiated trades" and "sell initiated trades" can only be deduced from intraday data and then taken to a daily scale.
- *Imbalance size*. It is defined as the ratio:

$$\frac{Q_t}{ADV}$$

  It is expressed on a daily basis and the values are in decimal units. In the I-Star modeling context it represents a proxy of a private agent order size.
- *Imbalance side*. It is the signed imbalance and it indicates which side of the market is prevailing. Either $+1$ or -1 indicating respectively prevailing buy or sell initiated trades, i.e $\sum V_{buy} > \sum V_{sell}$ or $\sum V_{buy} < \sum V_{sell}$, respectively.
- *Percentage of volume* (POV). The ratio between market imbalance and the market daily volume traded over a given day:

$$POV = \frac{Q_t}{V_t}$$

- *Volume Weighted Average Price* (VWAP). Expressed as:

$$VWAP = \sum \frac{P_t Q_t}{Q_t}$$

  it is commonly used as a proxy of fair market price. In the present context is specifically used as a proxy of the average execution price.
- *Arrival Cost*. The usual arrival cost benchmark metric. In a single security analysis framework it refers to the arrival cost of private order transactions, whereas with respect to the full model with market tick data only is an analogous metric based on the VWAP as proxy of a fair average execution price:

$$ArrivalCost = \log(\frac{VWAP}{P_0})ImbalanceSide10^4$$

  where P_0 is the arrival price.

I-Star model equations

We start from calculating the total cost of transacting the entire order and then distribute this quantity within single trade periods that took place.

Also, with respect to each trade period impact we can distinguish between a temporary and a permanent market impact (Lee-Ready, 1991).

The I-Star model is made of three main components, all expressed in basis points. First of all there is the *Instantaneous impact* (I), the theoretical impact of executing the entire order at once. We express it here in its "power" functional form, suggested by the author as the empirically most robust, stable and accurate over time one with respect to linear and non-linear alternatives:

$$I = a_1(\frac{Q}{ADV})^{a_2}\sigma_3^a$$

where the parameter a_1 is the *sensitivity to trade size*, a_2 is the *order shape parameter* and a_3 the *volatility shape parameter*.

Then we consider the *market impact* (MI), which represents the period-by-period impact cost due to a given trading strategy and is expressed as:

$$MI = b_1 POV^{a_4} I + (1 - b_1) I$$

where a_4 is said *POV shape parameter* and b_1 is the *percentage of total temporary market impact*.

Lastly, the *Timing risk measure*, a proxy for the uncertainty surrounding the cost estimate:

$$TR = \sigma\sqrt{\frac{S(1 - POV)}{3T_m ADV POV}}10^4$$

where S is the private order size.

The first two equations are part of the model estimation, whereas the last one is used as a measure of risk esposure for a given order.

Data pre-processing and parameters estimation

TODO: outliers analysis is still under discussion

Data pre-processing consists of data grouping and outlier analysis. The grouping may be carried before proceeding with the non-linear regression estimation. The grouping is based on buckets built with respect to three variables: the Imbalance size, the POV and the annualized volatility. It is irrespective of the security whose values fall into the buckets. A datapoints threshold in each bucket has to be reached in order to include the corresponding group in the estimation process.

Several aspects are worth empashizing. First of all, using Kissell's words "too fine increments [lead to] excessive groupings surface and we have found that a substantially large data grouping does not always uncover a statistical relationship between cost and our set of explanatory factors." This in turn points to an important consideration: also depending on the datapoints threshold specified, the data grouping may result in discarding data and this allows to exclude anomalous observations (outliers) with respect to the explanatory variables. On one hand is therefore understood how this step offers improvement margins to the nonlinear least squares estimation procedure, on the other it may cause convergence issues dependending on the effective shrinkage datapoints go through.

I-Star parameters estimation. The author suggests three methods to estimate model paramaters from the instantaneous and the market impact equations, namely the *two-step process*, the *guesstimate technique* and *nonlinear regression*. At present only this last one is implemented, through `stats::nls`.

The full model parameters are estimated by means of nonlinear least squares. There is a wide theory behind such approach, rich of pros and contra inherent to the specific iterative procedure used and their peculiarities in achieving converge. The interested reader may consult Venables and Ripley (2002). A general warning in estimating this model comes from the author himself: "Analysts choosing to solve the parameters of the model via non-linear regression of the full model need to thoroughly understand the repercussions of non-linear regression analysis as well as the sensitivity of the parameters, and potential solution ranges for the parameters."

In his modeling context the author sets a constrained problem providing bounds on parameters, in order to ensure feasible estimated values. The author's suggested bounds are implemented by default to follow his methodology, as reported in 'Details'. However, the opportunity to provide bounds is supported and left to the users. Likewise, initial parameters values to start the iterative constrained minimization problem resolution from is left to the user: to my knowledge at the time of writing, the author does not provide any specific clue in the estimation procedure used and especially there is no suggestion on particular starting values to begin with. It is valuable for a user to control starting values, as a way to check whether the estimated parameters come form a local optimum or if a global optimum may have been reasonably achieved.

```
# TODO: add 'iStarPostTrade' examples
#
# Before we can provide this example, the chosen dataset should be made publicly available.
#
# Also, whichever dataset will probably be in GB
# (our testing datasets had a minimum of around 1.5 GB in RStudio).
#
# This will slow down knitting, so I believe that it is better if we load results internally.
#
# Also, here we can show cost curves with the method implemented on 'iStarEst' class
#
```

Impact estimates, error and sensitivity analyses

Once the parameters have been estimated, the I-Star best fit equations provide impact costs estimates for a given market parent order specified by its size, POV, annualized volatility, side and arrival price. The instantaneous, market impacts (both temporary and permanent) and timing risk are described by the I-Star model equations explained above. The *cost error* is assessed as the difference between the arrival cost of the order and the market impact estimate. The *z-score* is a "risk-adjusted error" and is expressed as the ratio between the cost error and timining risk. The author reports that most accurate models possess z-scores distributions with mean zero and unit variance.

```
# TODO: add iStarSensitivity() example on estimation results or provided parameters,
#  we also have plotting methods
```

Almgren-Chriss (2000) model

In their proposed modeling framework Almgren and Chriss (2001) consider the following trade-off that ultimately determines the transaction cost: on one hand a trader may modulate the trading rate in order to decrease the *volatility risk* exposure to secutities prices fluctuations, on the other the rate of exection has the effect of increasing the transaction cost through the price impact these trades generate. This so called *trader dilemma* can be conxtualized from two equivalent points of view, maximize the expected trading revenues or minimize the expected transaction costs, for any given trader's *risk tolerance* as captured by the *risk-aversion* coefficient γ. The authors' fundamental idea of *optimal liquidation* is to find which trading trajectory minimizes the trade-off.

The trading model

Among the authors assumptions is there the security price dynamic is driven by a *discrete arithmetic random-walk* with independent increments, furthermore - following their extended discussion - a drift is permitted. As known it allows to incorporate directional views on the traded security price, however it may result in changed signs of the optimal trajectories computed (e.g. there may be buy trades in a selling program). Also, in light of the short-term trading horizons considered, the authors do not take into account other procecesses nor the carry or time-value of money. Another assumption is with respect to the equally spaced interval between the trading discrete times, $\tau = T/N$ where T is the time the execution program has to be completed and N represents the number of intervals the among the trading times. With respect to the impact functions, although not strictly necessary, we follow the authors' seminal work and assume linear forms. It is a simplification that as we shall specify shortly allows to provide explicit solutions to the optimization problem posed.

Permanent and temporary impact are functions of the *average trading rate*, $v_k = n_k/\eta$.

Permanent market impact. Sometimes described as "information leakeage" is an ebrilibrium-changing in the security price during the trading times, i.e an influence on its dynamic, that we write as:

$$S_k = S_{k-1} + \sigma\sqrt{\tau}\psi_k + \mu\tau - \tau g(v_k)$$

where ψ is a standard normal random variable, μ is the security price drift and σ is the security price volatility. The function $g(v_k)$ is the linear permanent impact function

$$g(v_k) = \gamma v_k$$

the parameter impact γ permanent impact parameter can be seen as a fixed cost independent of the trajectory (and is expressed in (currency/trade-unit)/trade-unit).

Temporary market impact. The execution of a sufficiently large n_k units between times $t_(k-1)$ and t_k may influence the price of the traded asset. The impact is always against the trader in that buying (selling) a too large number of units may increase (decrease) its price. Due to this impact the effective price per traded unit on the k-th transaction is:

$$\tilde{S}_k = S_{k-1} - h(v_k)$$

with $h(v_k)$, the linear temporary impact function, defined as:

$$h(v_k) = \epsilon sgn(n_k) + \eta v_k$$

the parameter ϵ is expressed in currency/trade-unit, authors report that a reasonable estimate for is the fixed costs of trading, such as half the bid-ask spread plus fees; the temporary impact parameter η is expressed in (currency/trade-unit)/(trade-unit/time) and is reportedly more difficult to estimate.

Associated with any given trading trajectory there is a *capture trajectory*, and a *total cost of trading*. The former is the full trading revenue upon completion of all trades:

$$\sum_{k=1}^{N} n_k \tilde{S}_k$$

Whereas the latter is the difference between the portfolio value at the beginning of trading (book value) and the capture trajectory:

$$XS_0 - \sum_{k=1}^{N} n_k \tilde{S}_k$$

which is Perold's (ex-post) *implementation shortfall*.

The optimization problem

A rational trader seeks to minimize the expectation of shortfall, for a given level of variance of shortfall. For each level of risk aversion there is a uniquely determined *optimal trading strategy*. The optimization problem Almgren-Chriss posed aims to find such a strategy, which has lower variance for the same or lower level of expected transaction costs, or, equivalently, no strategy which has a lower level of expected transaction costs for the same or lower level of variance. Mathematically the problem is posed as

$$\min[E(x) + \lambda V(x)]$$

where the *expected shortfall* $E(x)$ is the expectation of impact costs:

$$E(x) = \frac{1}{2}\gamma X^2 - \mu \sum_{k=1}^{N} \tau x_j + \epsilon \sum_{k=1}^{N} |n_k| + (\eta - \frac{\gamma\tau}{2}) \sum_{k=1}^{N} n_k^2$$

and the *variance of the shortfall* $V(x)$ is

$$V(x) = \sigma^2 \sum_{k=1}^{N} \tau x_j^2$$

The Lagrange multiplier λ is interpreted as a measure of risk aversion, that is how much the variance is penalized with respect to the to expected cost. As long as $\lambda \geq 0$ there exists a unique minimizer x_k^*.

The solution of this problem depends on the specified forms of the impact functions. In the linear case assumed explicit solutions of the optimal holding trajectory and as a consequence of the associated optimal trade trajectory exist:

$$x_j = \frac{\sinh[\kappa(T - t_j)]}{\sinh(\kappa T)} X + (1 - \frac{\sinh[\kappa(T - t_j)] + \sinh(\kappa t_j)}{\sinh(\kappa T)})\bar{x}$$

$$n_j = \frac{2\sinh(\frac{1}{2}\kappa\tau)}{\sinh(\kappa T)} \cosh[\kappa(T - t_{j-1/2})]X + \frac{2\sinh(\frac{1}{2}\kappa\tau)}{\sinh(\kappa T)}\{\cosh(\kappa t_{j-1/2}) - \cosh[\kappa(T - t_{j-1/2})]\}\bar{x}$$

where $t_{j-1/2} = (j - 1/2)\tau$, $\bar{x} = \mu/2\lambda\sigma^2$ is the optimal level of security holding for a time-independent porfolio and allows to incorporate a drift correction into the optimal trajectories dynamics, when the hypothesized drift term is not null. The parameter κ, sometimes called the *urgency parameter*,

expresses the curvature of the optimal trajectory, approximated for small equally spaced trade periods, that is:

$$\kappa \sim \sqrt{\frac{\lambda\sigma^2}{\eta}} + O(\tau)$$

as $\tau \to 0$. Also, κ provides an interesting interpretation in terms of the time needed to complete the order. Rewriting $\theta = 1/\kappa$, the so called *half-life* of a trade, is evident that the larger the value of κ and the smaller the time θ needed deplete the execution, that is the rapidly the trade schedule will be completely executed. In other terms it can be seen as a measure of the liquidity of the traded security. Resolving the optimization problems with respect to different such values of λ leads to the *efficient frontier*, that is the locus of the optimal trading strategies where each strategy has the minimum transaction cost for a given level of volatility or the minimun volatility for a given amount of transaction costs (i.e. cost-volatility optimal combinations). The frontier is a smooth convex function, differentiable at its minimal point which is what the authors call the *naive strategy*, corresponding to trading at a constant rate.

It is important to stress how under the above context the optimal execution trajectories can be statically determined. This is not always exactly true, as there could be serial correlation effects and paramters shifts due to regime changes (news, events, etc.). However the author shows how even in that case optimal trajectory are piecewise static and notwithstanding that gains from introducing these components into the dynamics is argued to be negligible for large portfolios with respect to the impact costs.

Almgren-Chriss example

We can proceed illustrating the example authors provide in their original contribution (Almgren and Chriss 2001, see section 4.4).

```
set.seed(333)
Securities <- data.frame()
Securities[1, 'Init.Price'] <- 50
Securities[1, 'Units'] <- 10^6
Securities[1, 'Complete.By'] <- 5
Securities[1, 'Trade.Periods'] <- 20 # 5 # 20 better gives the idea of what it should look like (but we shoul
mu <- 0.02
sigma <- 0.95
gamma <- 2.5 * 10^-7
epsilon <- 0.0625
eta <- 2.5 * 10^-6
```

Optimal trajectories associated with different risk tolerance, i.e. different values of the risk-aversion coefficient, are given for example assuming in turn the following coefficients:

```
lambda_averse <- 2 * 10^-6; lambda_neutral <- 0; lambda_propense <- (-2) * 10^-7
ac_averse <- acOptTxns(Securities = Securities, mu = 0, sigma = sigma,
                       gamma = gamma, epsilon = epsilon, eta = eta,
                       lambda = lambda_averse)
ac_neutral <- acOptTxns(Securities = Securities, mu = 0, sigma = sigma,
                        gamma = gamma, epsilon = epsilon, eta = eta,
                        lambda = lambda_neutral)
ac_propense <- acOptTxns(Securities = Securities, mu = 0, sigma = sigma,
                         gamma = gamma, epsilon = epsilon, eta = eta,
                         lambda = lambda_propense)
```

The *efficient trading frontier* can then be shown as follows:

```
lambdas <- seq(-5e-07, 5e-06, 1e-07)
opt.pts <- matrix(NA, nrow = length(lambdas), ncol = 2)
for (l in 1:length(lambdas)) {
  opt.pts[l, ] <- acOptTxns(Securities = Securities, mu = 0, sigma = sigma, gamma = gamma, epsilon = epsilon,
                            lambda = lambdas[l])$Optimal.Comb
  e <- opt.pts[, 1]
  v <- opt.pts[, 2]
}
```

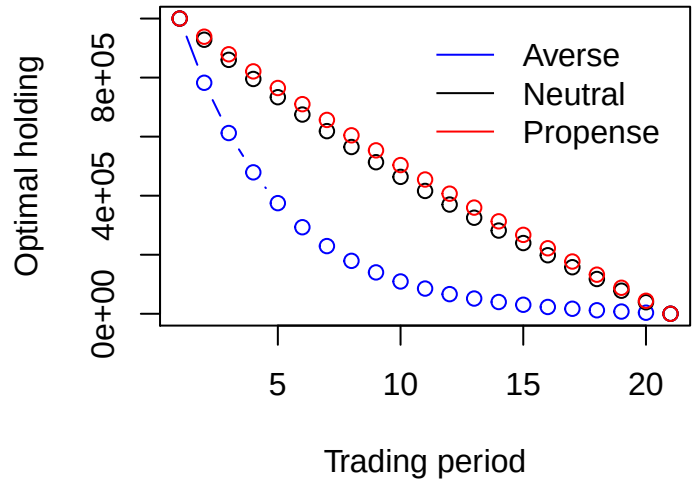


Figure 6: Optimal holding trajectories, risk profiles comparison.

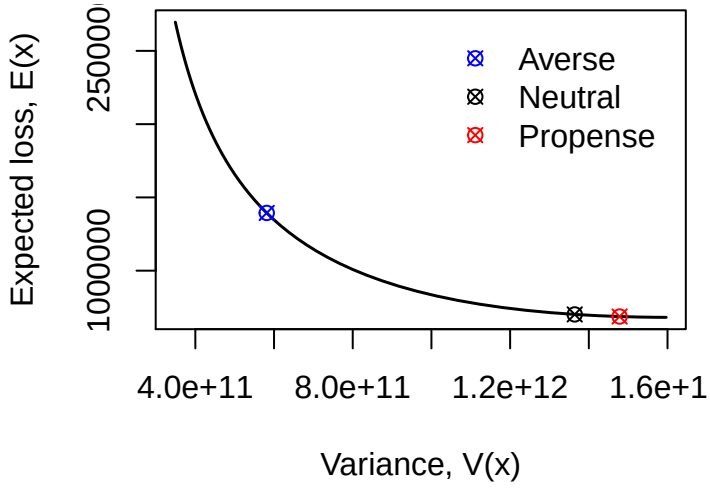


Figure 7: The efficient trading frontier.

References

Almgren, Robert, and Neil Chriss. 2001. "Optimal Execution of Portfolio Transactions." *Journal of Risk* 3: 5–39.

Kissell, Robert. 2013. *The Science of Algorithmic Trading and Portfolio Management*. 1st ed. Elsevier Science.

Venables, W. N., and B. D. Ripley. 2002. *Modern Applied Statistics with S*. Springer.

Vito Lestingi
MSc. Quantitative Finance Student - Sapienza University of Rome

lestingiv@gmail.com

Jasen K Mackie
Algorithmic Trading Services - Iress

jasen.mackie@iress.com

Brian G Peterson
TODO

brian@braverock.com