The Analysis of High-Frequency Finance Data using ROOT

P Debie[1,2], M E Verhulst[1,2], J M E Pennings[1,3,4], B Tekinerdogan[5], C Catal[6], A Naumann[7], S Demirel[1,5], L Moneta[7], T Alskaif[5], J Rembser[7] and P van Leeuwen[2]

[1] Marketing and Consumer Behaviour Group, Wageningen University, Wageningen, the Netherlands
[2] Wageningen Economic Research, Wageningen, the Netherlands
[3] Department of Marketing and Supply Chain Management, Department of Finance, Maastricht University, Maastricht, the Netherlands
[4] Office for Futures and Options Research, University of Illinois at Urbana-Champaign, Urbana, Illinois, USA
[5] Information Technology Group, Wageningen University, Wageningen, the Netherlands
[6] Department of Computer Science and Engineering, Qatar University, Doha, Qatar
[7] European Organization for Nuclear Research (CERN), Geneva, Switzerland

E-mail: `philippe.debie@wur.nl`

Abstract.
High-frequency financial market data is conceptually distinct from high energy physics (HEP) data. Market data is a time series generated by market participants, while HEP data is a set of independent events generated by collisions between particles. However, there are similarities within the data structure and required tools for data analysis, and both fields share a similar set of problems facing the increasing size of data generated. This paper describes some of the core concepts of financial markets, discusses the data similarities and differences with HEP, and provides an implementation to use ROOT, an open-source data analysis framework in HEP, with financial market data. This implementation makes it possible to take advantage of the rich set of features available in ROOT and extends research in finance.

1. Introduction

Similar to research in high-energy physics (HEP), the size of datasets used in finance research has increased over time. For example, the New York Stock Exchange (NYSE) produces 2TB of data per day, consisting of 75 billion market events [1]. This is significantly smaller but still comparable with the LHC, which produces 250TB of compressed data per day [2].

However, while in HEP there is a clear dominant framework for data analysis, namely ROOT (an open-source data analysis framework) [3], this is not the case in finance. Datasets are often stored in CSV (Comma-Separate Value) files, and processed using different programming languages like Python, R, and MATLAB. This generates an obstacle when scaling existing research beyond the limitations of personal computers. Larger datasets, like high-frequency market data mentioned above, are often stored and analysed using custom solutions. While massively increasing the capabilities of such custom systems, it also slows down research. That is, a custom solution requires more time to develop, and more support and explanation to

	Level	Price	Volume
Ask	4	\$54.50	3
	3	\$54.00	5
	2	\$53.50	6
	1	\$53.00	12
		Spread	
Bid	1	\$49.00	14
	2	\$48.00	9
	3	\$47.50	38
	4	\$47.00	15

Table 1: A limit order book (LOB) example, showing the top 4 levels of the LOB.

reproduce. In addition, it makes reproducing existing work troublesome due to possible bugs or differences in implementation [4].

This paper describes the similarities and differences of data between HEP and finance and introduces an extension to ROOT, which can store and analyse high-frequency market data, with the goal of increasing reproducibility and facilitating the research on larger datasets. In addition, the collaboration between physics and finance research allows for the exchange of existing techniques in their respective fields, and yields cross-fertilisation and new insights for novel research. One of the results of this collaboration is shown in this paper; the extension of ROOT to process time-series data.

This study is structured as follows. Section 2 describes some of the core structures of a financial market, and what kind of data this generates. Section 3 discusses the similarities and overlapping problems between finance and HEP data. Section 4 describes the new extension built on top of the ROOT framework and shows some example use cases on how to analyse high-frequency data using ROOT. Section 5 further discusses some of the differences between finance and HEP data that can interfere with future research.

2. Financial markets and the limit order book

Nowadays, financial markets' trading platforms are predominantly electronic. These platforms showcase, among others, the limit order book (LOB): a computerised system of all available demand - the bid side - and supply - the ask side - for securities and financial instruments at a specific time. All market participants can submit orders to this platform with specifications, e.g. whether they wish to buy or sell, the associated price, and the quantity they want to buy or sell [5]. The following two main types of orders can be submitted to the LOB: market orders and limit orders. Market orders are immediately executed against the best bid or ask price. They consume volume from the LOB and are placed by traders who immediately accept the market price [6]. Limit orders rest in the LOB and are executed at a defined limit or at a better price. They provide volume to the LOB and are placed by traders who can wait for their order to be executed [6]. Generally, limit orders are much more numerous than market orders [7]. Although market and limit orders are the main type of orders, variations exist. For example, limit orders can be (partially) hidden (i.e. hidden or iceberg orders), and market orders can include special conditions such as whether to execute the volume completely or not at all (fill-or-kill) [8].

Table 1 presents an example of the LOB. The LOB consists of the bid and ask side. The bid (ask) side contains all buy (sell) limit orders and is organised in descending (ascending) price order. The first, or best, level on either side is the highest price someone is willing to buy for (the bid side) and the lowest price someone is willing to sell for (the ask side). The difference between the best bid and best ask level is called the bid-ask spread. A narrower bid-ask spread indicates that traders are closer to agreeing on a price. A trade occurs when traders agree on a price, and volume is removed from the LOB.

Financial markets have different designs and rules - also called the market microstructure - depending on the type of market. For example, markets have different tick sizes (i.e. minimum allowable price change), position limits (i.e. maximum stocks/contracts you may own) and a daily maximum or minimum price fluctuation (e.g. circuit breakers and price limits). In the last decade, new market participants emerged due to the digitalisation of trading platforms. Apart from humans entering orders manually, it became possible for computers, or trading algorithms, to trade automatically in markets. The latter are often classified as high-frequency traders, as they trade in milliseconds [9]. In addition, regulation changed, as digitalisation also allows, for example, for different manipulation schemes.

LOBs create two main types of data, each containing messages that the exchange receives: LOB data and order data. LOB data consists of modifications to the LOB, often stamped at the millisecond or nanosecond time-resolution. Messages contain information on LOB levels that change, i.e. when a new price level is added to the LOB, when a price level is removed from the LOB and when volume on an existing price level changes. These messages can be used to reconstruct the state of the LOB at any time. LOB data messages show the aggregated effect of individual orders. In other words, when two traders add volume at the exact same time and price level, the exchange receives one message with the aggregated volume change in the LOB. Order data, on the other hand, consists of messages related to individual orders. Using the aforementioned example, the exchange would receive two messages: one for each trader with their specifics. Hence, each individual action of a trader results in a message. Aggregating all these individual orders also allows for the reconstruction of the LOB [10]. Both LOB and order data have specific trade messages when a trade occurs in the market, and both datasets have significantly increased over time. To illustrate, one day of message data was similar to 30 years of daily data at the start of this century [11], but the size of message data increased tenfold ten years later [12].

3. Similarities and Differences between HEP and Finance Data

Data from high-frequency finance, more precisely message-based market data as described in section 2, and data from HEP are generated by completely different processes and grounded in different domains. Market messages are a sequence of events forming a time series, while HEP consists of individual events, where each event is a collision. On the face of it, they could not be any different.

However, we draw similarities between the required tools for these two data types. Each entry (i.e. a collision of particles or a state of the LOB) is variable in size and shape, for example, the more limit orders are placed by traders, the more levels are in the LOB. Similarly, the more collisions occur in a beam of particles, the more information is recorded by the sensors. This inconsistency of the data size of an entry makes simple tabular storage solutions unpractical. ROOT, however, is not only accommodated to handle this kind of data but also built to maintain high performance while analysing a subset of the data.

Furthermore, there are other qualities of ROOT concerning market data. As described in the introduction, existing work in finance is often limited by the underlying data structures and algorithms of available libraries. ROOT, however, is scalable, can use parallelism implicitly, and does not require storing all data in memory at once while iterating or processing. It also avoids any of the existing aforementioned problems and thus paves the way for further research.

```
TimeFrame<LOB, Message> timeFrame;
timeFrame.add(messages);
timeFrame.setStateInitializer([&](int id)
{
    return LOB();
});
timeFrame.setStateUpdater([](
    int id, TimeNS time, LOB& lob, const Message& message)
{
    lob.update(time, message);
});
timeFrame.setForEachSnapshot(T_Second * 10, [](
    TimeNS time, const LOB& lob)
{
    // Use the limit order book object for the analysis
});
timeFrame.run();
```

Listing 1: A snippet of the C++ code showing how to use the extension to reconstruct the LOB and resample the time series into a fixed interval of 10 seconds, i.e. making snapshots.

4. Implementation

This section describes the implementation of the extension built on top of ROOT. The extension (1) contains the basic tools to reconstruct the LOB based on a time series of order messages; (2) makes it possible to resample the market messages, which are spaced irregularly in time, into a fixed interval time series (also called 'making snapshots', i.e. taking a snapshot of the state of the LOB at regular time intervals); and (3) contains a collection system to convert a single long time series in a set of smaller time series. These three components make it possible to apply the vast set of tools within ROOT, like RooFit and TMVA [3], to time series data. The implementation, including two examples, can be found found as a GitHub repository [13].

4.1. Reconstruction and resampling

Listing 1 shows how to reconstruct the LOB while iterating over the messages. The function `setStateInitializer` sets a lambda function that defines how to initiate the LOB upon processing the first message corresponding to that LOB, and `setStateUpdater` sets a lambda function that defines how each following message updates the state of the LOB. The `Message` and the `LOB` object are templated, thus allowing similar functionality in different domains. Next, the `setForEachSnapshot` function sets a lambda function that contains the main processing or analysis, this lambda is called for each LOB and each snapshot. In the example listing 1, a resampling period (also known as a snapshot length) of 10 seconds is chosen. The type `TimeNS` is used to represent a date-time object using nanoseconds since an epoch, with `T_Second` as a constant. Finally, the function `run` is called, which initiates the event loop, and starts the analysis. This component was recently used in related work to build a novel visualisation of the LOB over a period of time [14]. The use of ROOT and the proposed extension was instrumental in storing and reconstructing the large data volume received from the financial exchange. It made it possible to visualise unseen details in high-resolution market data.

4.2. The collection system

The collection system is designed to transform a single time series into a set of smaller time series. For example, consider a researcher interested in studying trades occurring in a specific market. This researcher does not need all LOB data, only subsets containing time windows

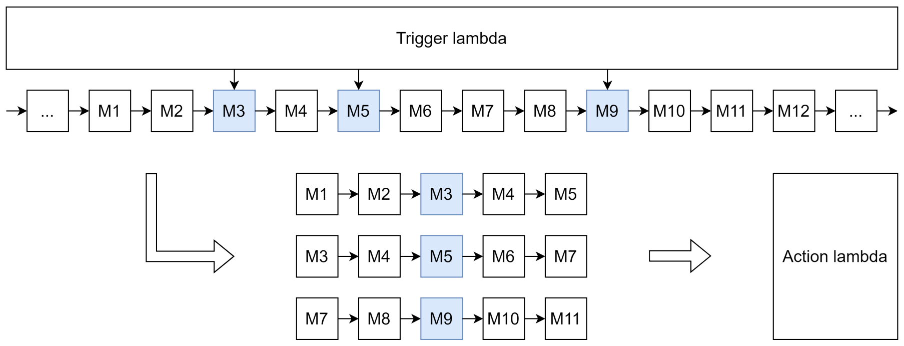

Figure 1: The conceptual diagram of the collection system. The trigger lambda selects messages in the time series, here shown from M1 (message 1) to M12. M3, M5, M9 are selected, and set as the centre of the three newly generated time series, yielding a dataset of size three.

```cpp
// Initialise timeFrame similar as listing 1.
timeFrame.setTrigger([&](int id, TimeNS time, const Message& message)
{
    // Return true if trigger message
});
timeFrame.setAction(-T_Minute, T_Minute, [&](
    int id, TimeNS time, const Message& triggerMessage,
    const std::list<std::pair<TimeNS, Message>>& msgs)
{
    // Process extracted time series
});
timeFrame.run();
```

Listing 2: A snippet of the C++ code showing how to use the collection system of the extension to transform a single time series into a set of time series, in line with the diagram in figure 1.

surrounding trades. Hence, the single long LOB time series can be transformed by the collection system into a set of smaller time series only containing the trades of interest. If each of these smaller time series is considered as an entry in the transformed dataset, i.e. a high dimensional encoding for a time series, then the standard methods from ROOT, which are designed to work with non-time series data, can be used. This increases the number of statistical tools available and makes it possible for new research. For example, one of our current working papers uses the collection system to extend the research from Degryse et al. [15].

The collection system iterates over a time series and selects events that pass a filter (e.g. a trade), hereafter named the trigger, or the trigger lambda if referring to the C++ implementation. If the trigger selects an event, a small time series is extracted, starting at a predefined number of time steps before the trigger and ending at a predefined number of time steps after the trigger. This smaller time series is then considered an entry in the new dataset. Figure 1 shows a conceptual diagram of this system.

Listing 2 shows how to run the collection system using the extension on ROOT. First, the `timeFrame` object should be initialised with `setStateInitializer` and `setStateUpdater`, similarly as in listing 1; this is omitted in this example. Next, the `setTrigger` function sets a lambda function, which is called for each entry in the time series, and should return `true` if that entry is a trigger (i.e. selected). In addition, the `setAction` function sets a lambda function, which is called for each sub-time-series generated. Lastly, the `run` function is called to run the event loop.

5. Discussion

As shown in subsections 4.1 and 4.2, the implementation is modular and reusable, more details can be accessed from the previously mentioned GitHub repository [13]. In addition, this study shows the flexibility of ROOT by being adaptable to other fields of study.

However, time series analysis still creates challenges, independent of the data analysis framework used. The statistical properties of data points are different. In HEP, every entry is i.i.d. (independent and identically distributed), while in time series data, every data point is dependent on every foregoing data point. This generates challenges when applying the standard tools (e.g. the statistical methods in RooFit) to finance data.

This challenge surfaces in the collection system presented in this work; it can convert a single time series into a set of individual smaller time series, thus generating a new dataset where each entry is a time series. However, the entries in this generated dataset are not i.i.d., since they are based on the underlying time series, and thus not all statistical methods of ROOT are applicable. This means that the researcher has to carefully consider the chosen methods and verify their suitability.

6. Conclusion

The proposed extension built on top of ROOT makes it possible to analyse high-frequency market data and other time-series data in general. It leverages the performance and optimisation of ROOT to process large volumes of data, while also decreasing the development time and the required analysis time. These advancements are required to extend the statistical power of future research.

The core components of this extension are a general-purpose method to reconstruct an object (e.g. a limit order book) while iterating over a time series (e.g. a sequence of messages), a resampling method to convert irregularly spaced time steps into regularly spaced time steps, and a collection system to convert a single time series into a set of individual data points.

References

[1] FINRA How do we process up to 75 billion market events daily? `https://technology.finra.org/articles/how-finra-processes-75-billion-market-events-daily.html` [Accessed February 7, 2022]
[2] CERN Storage `https://home.cern/science/computing/storage` [Accessed February 7, 2022]
[3] Brun R and Rademakers F 1997 *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* **389** 81–86
[4] Burman L E, Reed W R and Alm J 2010 *Public Finance Review* **38** 787–793
[5] Arzandeh M and Frank J 2019 *American J. of Agricultural Economics* **101** 1482–1498
[6] Hachmeister A 2007 *Informed traders as liquidity providers: evidence from the German equity market* vol 66 (Springer Science & Business Media)
[7] Brogaard J, Hendershott T and Riordan R 2019 *The J. of Finance* **74** 1621–1658
[8] Ranaldo A 2004 *J. of Financial Markets* **7** 53–74
[9] Hasbrouck J and Saar G 2013 *J. of Financial Markets* **16** 646–679
[10] Mendonça L and De Genaro A 2020 *J. of Financial Regulation and Compliance* **28** 369–408
[11] Gençay R, Dacorogna M, Muller U A, Pictet O and Olsen R 2001 *An introduction to high-frequency finance* (Elsevier)
[12] Fabozzi F, Focardi S M and Jonas C 2011 *Review of Futures Markets* **9** 7–38
[13] HighLO GitHub repository: TimeFrame `https://github.com/HighLO/TimeFrame` [Accessed January 24, 2022]
[14] Verhulst M E, Debie P, Hageboeck S, Pennings J M E, Gardebroek C, Naumann A, van Leeuwen P, Trujillo-Barrera A A and Moneta L 2021 *J. of Futures Markets* **41** 1715–1734
[15] Degryse H, Jong F D, Ravenswaaij M V and Wuyts G 2005 *Review of Finance* **9** 201–242