Data Analysis for High-Frequency Trading

Assignment 3

Hawkes Process and Order Book Simulation.

Julian Albert
ALBJUL005

Department of Statistical Sciences
University of Cape Town
South Africa
October 5, 2019

Contents

Introduction **2**
Classification 2

Estimate a Hawkes Process for Order-Book Events **3**
Four Dimensional Hawkes Process 8

Simulate the Order-Book **12**

Conclusions **13**

Introduction

The Hawkes process is a class of point processes whose future depends on their own history. This class of processes is particularly useful for estimating and simulating order book events as they occur asynchronously in event-time and there exists relationships between the occurrence of order-book events. As seen in previous Assignments trade events exhibit auto-correlation in the event arrival times which leads to the observation that event arrival causes the conditional intensity function to rise. This event arrival process can thus be thought of as an example of a "self-exciting" process which can be modelled using the Hawkes process — a counting process that models a sequence of "arrivals" of some type over time. Each arrival excites the process in that the likelihood of a subsequent arrival is increased for some time period after the initial arrival. We will estimate the parameters of a multivariate Hawkes process using a single day of trading for a particular stock. The resulting non-homogeneous parameter estimates will be used to simulate the arrival of order-book events (classified into eight subcategories). Using these arrivals times, we then simulate the order-book by means of plotting the best-ask price, best-bid price and mid-price over a day of trading.

Classification

The first part of this assignment is the classification of order-book events into eight subcategories, this is done using Table (1). First we classify trades as either buyer or seller initiated and then we determine whether or not they have moved the order book. To determine whether the the trades are buyer (+1) or seller (-1) initiated market orders we use the quote data. We can further subdivide these into buyer and seller initiated market orders that do or do not move the best bid/ask price. For buyer (seller) initiated trades, if the best sell (buy) price after the trades are higher (lower) than the best sell (buy) price before the trade, then we classify the trade as "Aggressive buy (sell) moves the ask (bid)" otherwise the trade is classified as "Aggressive buy (sell) doesn't move the ask (bid)". The LO classification are self explanatory, where the bid (ask) is at or lower (higher) then the previous bid (ask) the classification is an order than does not move the best bid (ask). Similarly if the bid (ask) lies in the spread it is classified as between quotes.

No.	Buy/Sell	Order Type	Price Move?	Classification
1	Sell	(MO)	✓	Market Sell Moves Bid
2	Sell	(MO)	✗	Market Sell Doesn't Move Bid
3	Buy	(MO)	✓	Market Buy Moves Ask
4	Buy	(MO)	✗	Market Buy Doesn't Move Ask
5	Buy	(LO)	✓	Bid Between Quotes
6	Buy	(LO)	✗	Bid $\leq$ Best-Bid
7	Sell	(LO)	✓	Ask Between Quotes
8	Sell	(LO)	✗	Ask $\geq$ Best-Ask

Table 1: Order-Event Classification as Per [3], MO refers to Market-Order whilst LO refers to Limit-Order.

Estimate a Hawkes Process for Order-Book Events

Before we dive into Estimating the process for different order-book events we can introduce the process. A Hawkes process is a point-processes $N(t)$ given the number of events on the time interval $[0, t]$ for some $s \leq t$ with $N(s) \leq N(t)$

$$\begin{aligned}&\mathbb{P}[N(t+h) - N(t) = 1|N(s)] = \lambda(t)h + o(h)\\&\mathbb{P}[N(t+h) - N(t) > 1|N(s)] = o(h)\end{aligned} \tag{1}$$

where the intensity $\lambda(t)$ is self-exciting with event arrival-times t_m and kernel $\phi = \alpha\beta e^{-\beta(t)}$ we get

$$\lambda(t) = \mu + \int_{-\infty}^{t} \phi(t-s)dN(s) = \mu + \sum_{t_m < t} \phi\left(t - t_m\right) \tag{2}$$

a point-process with an intensity function that is linear in past events, where the baseline intensity μ are events not stimulated by prior events. Defining

$$d\lambda(t) = -\beta\lambda(t)dt + \alpha\beta dN(t) \tag{3}$$

and multiplying by the kernel $e^{-\beta(t)}$

$$g(\lambda, t) = \lambda e^{\beta t} \tag{4}$$

Such that

$$\frac{\partial g}{\partial t} = \lambda\beta e^{\beta t} \qquad \frac{\partial g}{\partial \lambda} = e^{\beta t} \qquad \frac{\partial^2 g}{\partial \lambda^2} = 0 \tag{5}$$

Applying *Ito's lemma* we get

$$dg = \lambda\beta e^{\beta t}dt + e^{\beta t}d\lambda \tag{6}$$

$$d\lambda e^{\beta t} = \lambda\beta e^{\beta t}dt + e^{\beta t}\left(-\beta\lambda dt + \alpha\beta dN\right) \tag{7}$$

$$= \left(\lambda\beta e^{\beta t} - \lambda\beta e^{\beta t}\right)dt + \alpha\beta e^{\beta t}dN \tag{8}$$

$$= \alpha\beta e^{\beta t}dN \tag{9}$$

we therefore obtain

$$\lambda(t)e^{\beta t} = \lambda(0) + \alpha\beta \int_0^t e^{-\beta(t)}dN(t) \tag{10}$$

$$\lambda(t) = \mu + \alpha\beta \int_0^s e^{-\beta(t-s)}dN(s) \tag{11}$$

where $\mu = e^{-\beta t}\lambda(0)$. In [2] we get an exact approximation of this integral which will be used for calculating the intensities and plotting them as in Figure (3), (4) and (5) expressed by Equation (12).

$$\lambda(t) = \mu + \sum_{t_i < t} \alpha\beta e^{-\beta(t-t_i)} \tag{12}$$

In a multivariate setting we have that the intensity function is

$$\lambda_m(t) = \mu_m + \alpha_{mn}\beta_m \int_0^t e^{-\beta_m(t-s)} dN^m(s) \tag{13}$$

where $\boldsymbol{\alpha}$, $\vec{\beta}$, $\vec{\mu}$ represent a matrix and vectors of $k \times k$ and $k \times 1$ respectively with k being the number of dimensions (events).

One challenge when modelling using self-exciting point processes is estimating parameters from observed data, to do this we can use a maximum likelihood estimate for our parameters where the likelihood function is give by [1] in Equation (14)

$$L(\boldsymbol{\Theta}|t_n) = \prod_{i=1}^{N} \lambda(t_i; \boldsymbol{\Theta}) \exp\left(-\int_{t_{i-1}}^{t_i} \lambda(s; \boldsymbol{\Theta}) ds\right) \tag{14}$$

To obtain the MLE we take the natural logarithms to get the log-likelihood in Equation (15)

$$\ell(\boldsymbol{\Theta}|t_n) = -\int_0^{t_n} \lambda(s; \boldsymbol{\Theta}) ds + \sum_{i=1}^{n} \ln(\lambda(t_i; \boldsymbol{\Theta})) \tag{15}$$

minimising the negative log-likelihood is equivalent to maximising the likelihood function, however, the optimisation is less expensive and more computationally stable optimisations. Note here that $\boldsymbol{\Theta}$ represent the parameter set containing $\boldsymbol{\alpha}$, $\vec{\beta}$, and $\vec{\mu}$.

To perform this optimisation we use optim() in R, the nragtive log-likelihood function to be minimised corresponds to the likelihoodHawkes() function from the Hawkes() package [6]. The function does require the following constraints;

$$\begin{aligned} \boldsymbol{\alpha}, \vec{\mu} &\geq 0 \\ eigenvalue(\text{diag}(\vec{\beta}) - \boldsymbol{\alpha}) &> 0 \end{aligned}$$

which are implemented in the code by creating a new function which either calls likelihoodHawkes() if the constraints are met or calls likelihoodHawkes() plus some very large penalty — making the solution infeasible as we are minimising the objective.

The first issue with trying to estimate Θ for all 8 event types presented previously is that of local maxima. As the dimensions increase the shape of the negative log-likelihood function can become complex and may not be globally convex. The lack of convexity in the objective function would result in the MLE being the local maximum rather than the global maximum [5]. A usual approach used in trying to identify the global maximum involves using several sets of different initial values for the maximum likelihood estimation, however, this proved unsuccessful.

Alternatively, I look at a bivariate Hawkes process for two sets of cases, the first is aggressive buy (sell) market orders where the price is greater (less) than the best ask (bud). The second is passive limit orders where the new bid (ask) is higher (lower) than the previous bid (ask) thus updating the quote in the LOB. Initial parameters are chosen arbitrarily with the assumption that the optimisation algorithm will converge, the starting values for $\boldsymbol{\alpha}$, $\vec{\beta}$, and $\vec{\mu}$ are chosen to be a small diagonal matrix, a vector of values between 0.5 and 1, and the mean number of events divided by seconds in the day respectively. The starting parameters for aggressive and passive orders are

$$\vec{\lambda}_{agg} = \begin{bmatrix} 0.014 & 0.010 \end{bmatrix} \qquad \vec{\lambda}_{pass} = \begin{bmatrix} 0.133 & 0.090 \end{bmatrix}$$

with kernel starting parameters

$$\boldsymbol{\alpha}_{agg} = \begin{bmatrix} 0.01 & 0 \\ 0 & 0.01 \end{bmatrix} \qquad \vec{\beta}_{agg} = \begin{bmatrix} 0.05 & 0.05 \end{bmatrix}$$

$$\boldsymbol{\alpha}_{pass} = \begin{bmatrix} 0.01 & 0 \\ 0 & 0.01 \end{bmatrix} \qquad \vec{\beta}_{pass} = \begin{bmatrix} 0.05 & 0.05 \end{bmatrix}$$

And optimal solutions

$$\vec{\lambda}^*_{agg} = \begin{bmatrix} 0.006 & 0.006 \end{bmatrix} \qquad \boldsymbol{\alpha}^*_{agg} = \begin{bmatrix} 0.023 & 0.003 \\ 0.000 & 0.017 \end{bmatrix} \qquad \vec{\beta}^*_{agg} = \begin{bmatrix} 0.044 & 0.051 \end{bmatrix}$$

$$\vec{\lambda}^*_{pass} = \begin{bmatrix} 0.033 & 0.023 \end{bmatrix} \qquad \boldsymbol{\alpha}^*_{pass} = \begin{bmatrix} 0.066 & 0.002 \\ 0.000 & 0.075 \end{bmatrix} \qquad \vec{\beta}^*_{pass} = \begin{bmatrix} 0.088 & 0.100 \end{bmatrix}$$

We can see that the optimal parameters have moved off of the initial parameters, in both cases optimisation converged (likely to local optima). It is worth noting that changes in α impact the number of events that arise from simulating the Hawkes process with the optimal parameters. Table (2) shows the four events and how the number of events observed in the data compare to the number of events simulated using a bivariate Hawkes process. Although these results are not necessarily outstanding it was decided to leave them unaltered as any further tuning would simply introduce bias and likely lead to an extreme case of overfitting. We can see however that the 2-dimensional Hawkes process approximates relatively well.

No.	Classification	n_{true}	n_{sim}
1	Agg. Sell	375	449
3	Agg. Buy	262	223
5	Pass. Buy	3658	3825
7	Pass. Sell	2477	2339

Table 2: Number of Event Arrivals True vs Simulated

To evaluate the parameters influence on the likelihood function I simulate 10000 (100 × 100) combinations of $\vec{\beta}$, and $\vec{\mu}$ as well as 10000 $\boldsymbol{\alpha}$ diagonal matrices. Where the ranges are

$$\begin{array}{ccccc} 0.0010 & \leq & \mu & \leq & 0.1500 \\ 0.0500 & \leq & \beta & \leq & 5.0000 \\ 0.0001 & \leq & \alpha_{ii} & \leq & 0.0400 \end{array}$$

For aggressive and passive the range was tuned. In the univariate case we would vary the parameters as opposed to varying combinations of parameters making insightful inference much easier. We plot the contours for the various combinations of parameters in the multi-dimensional likelihoods. This is achieved by fixing one of the three parameters at their MLE and plotting the remaining two parameters against one another to create a profile for visualising the relationship between these parameters, the resulting objectives for the bivariate Hawkes process are illustrated in Figure (1) and (2).

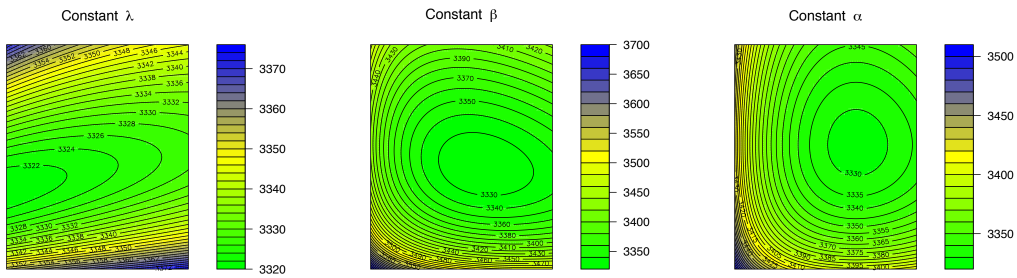

Figure 1: Objective Surfaces for Aggressive Order Types - Bivariate Hawkes

In both case we observe relatively "well-behaved" objective functions. Where we have smaller numbers of events it appears better results for simulation are achieved using smaller parameters for $\boldsymbol{\alpha}$ and $\vec{\beta}$, thus our initial starting point may somewhat be "directly" related to the number of event arrivals. However, the results for starting parameters on a mix of number of events is relatively robust (i.e. choosing different starting parameters for each event breaks down, maybe due to some iid. assumptions).

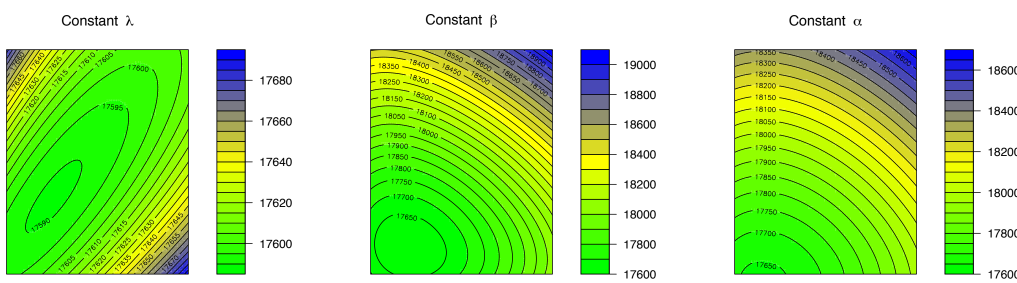

Figure 2: Objective Surfaces for Passive Order Types - Bivariate Hawkes

We can see that the objective surface is relatively stable in two-dimensions as the likelihood function (note $\lambda \equiv \mu$) is relatively simple. However, it is worth noting that the range for α is very small. Also unclear here is exactly how the dependencies and auto-correlations between paramters effect the process simulations. Recall that $N(t)$ is the number of "event arrivals" of the process by time t and that the sequence of event times is assumed to be observed within $[0, T]$ for finite T [5]. A property of the Hawkes process is that the events usually arrive clustered in time. Emprically what may occur is that the process might have started sometime in the past, prior to the moment when we start observing it $t = 0$. Hence, there may be unobserved event times which occurred before time zero, which could have generated "offspring" events during the interval $[0, T]$ [5]. It is possible that the unobserved event times could have had an impact during the observation period, i.e. some time $t > 0$, but because we are not aware of them, their contribution to the event intensity is not recorded. Such phenomenon are referred to as edge effects [5].

Of particular interest to us may be the intensities which can be recovered from our simulated Hawkes processes using Equation (10). We can plot the intensities in Figures (3) and (4), we see that in the aggressive case the events number of simulated events are similar. Also worth noting is the sporadic nature of both aggressive buys and sells appear similar with occasionally higher peaks being observe in the aggressive sells. With the passive order types we observe a higher baseline intensity and extreme jumps (LOB stuffing), in a South African context this may be due to the lack of market makers (liquidity providers). We also observe less aggressive buy and passive sell events relative to passive sell and aggressive buy events.

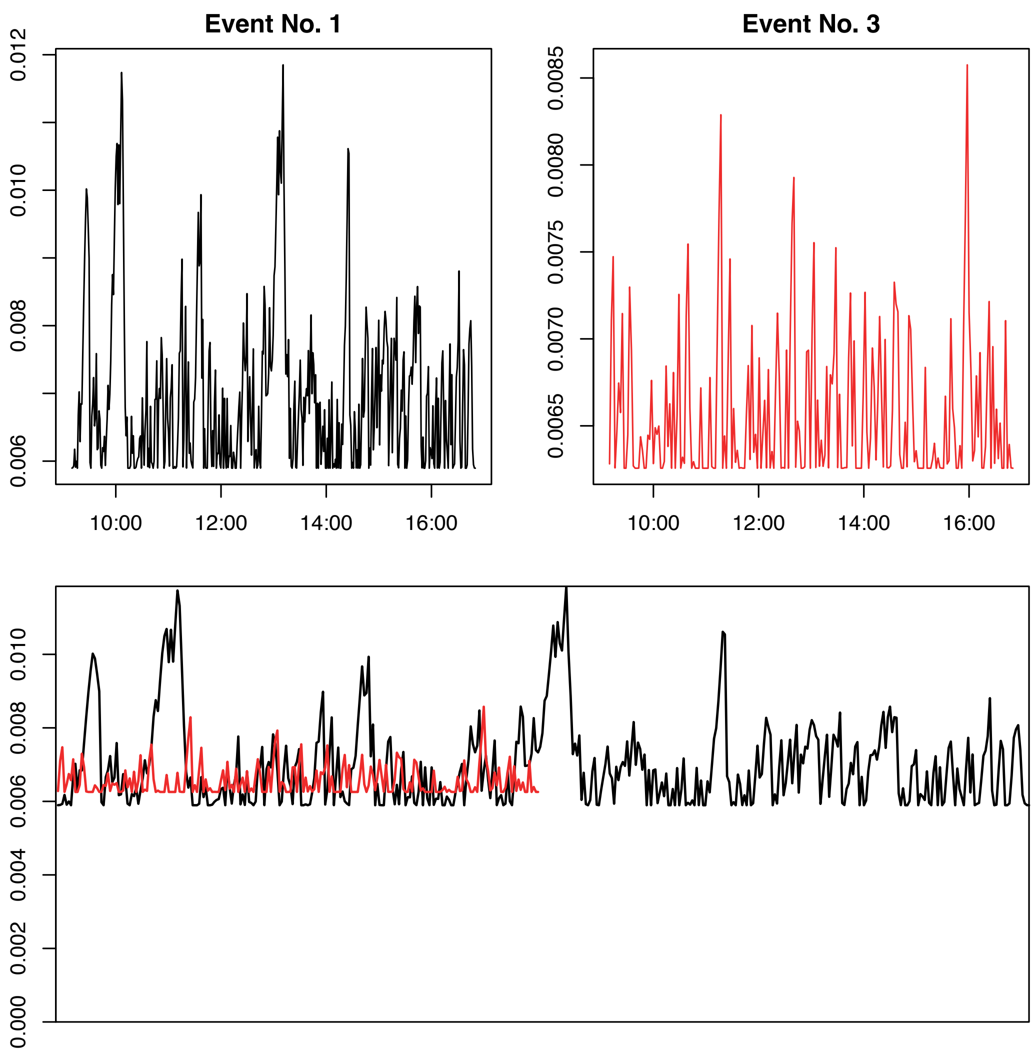

Figure 3: Intensities for Aggressive Order Types $k = 2$.

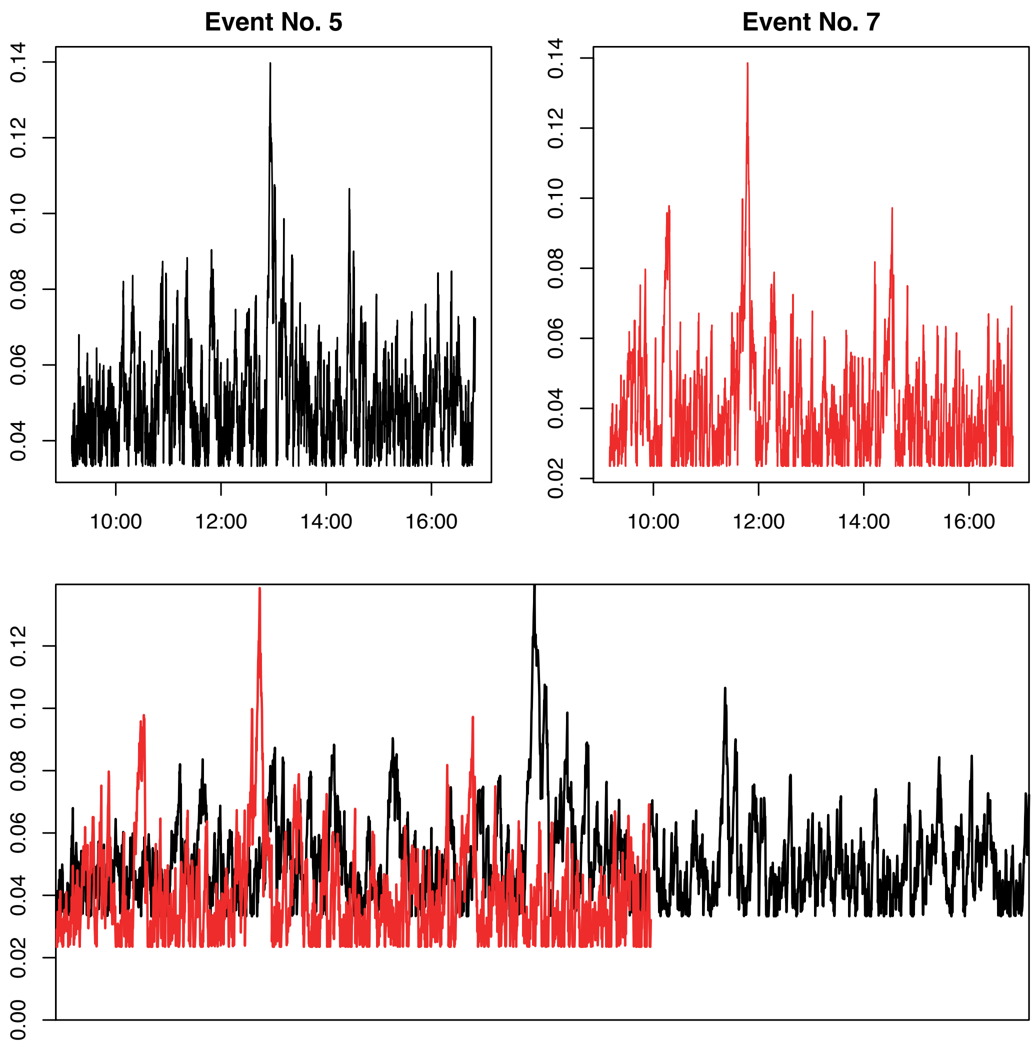

Figure 4: Intensities for Passive Order Types $k = 2$.

Four Dimensional Hawkes Process

Now we consider the case where we combine the two bivariate cases into a four-dimensional Hawkes process. Here the numerical instability is much higher, relatively many more iterations are used for convergence using the Nelder-Mead Simplex algorithm.

$$\vec{\lambda} = \begin{bmatrix} 0.014 & 0.009 & 0.133 & 0.090 \end{bmatrix} \qquad \boldsymbol{\alpha} = \begin{bmatrix} 0.01 & 0 & 0 & 0 \\ 0 & 0.01 & 0 & 0 \\ 0 & 0 & 0.01 & 0 \\ 0 & 0 & 0 & 0.01 \end{bmatrix} \qquad \vec{\beta} = \begin{bmatrix} 0.05 & 0.05 & 0.05 & 0.05 \end{bmatrix}$$

which are optimised to

$$\vec{\lambda}^* = \begin{bmatrix} 0.020 & 0.008 & 0.153 & 0.111 \end{bmatrix} \qquad \vec{\beta}^* = \begin{bmatrix} 0.005 & 0.000 & 0.163 & 0.092 \end{bmatrix}$$

$$\boldsymbol{\alpha}^* = \begin{bmatrix} 0.000 & 0.011 & 0.001 & 0.000 \\ 0.005 & 0.003 & 0.000 & 0.000 \\ 0.011 & 0.014 & 0.005 & 0.001 \\ 0.018 & 0.013 & 0.000 & 0.000 \end{bmatrix}$$

Using the new parameter set Θ^* we obtain the following estimates for number of event arrivals

No.	Classification	n_{true}	n_{sim}
1	Agg. Sell	375	188
3	Agg. Buy	262	82
5	Pass. Buy	3658	4895
7	Pass. Sell	2477	3478

Table 3: Number of Event Arrivals True vs Simulated - 4D Hawkes

Again, this proves less stable and the difference between empirical and simulated value is greater than in the bivariate case — intuitive as the number of events is highly variate the parameters should be less efficient. The process has difficulty finding the solution that can account for low MO activity and high LO activity and the Hawkes() package implicitly defines constraints that makes it difficult to initialise the parameters for each dimension in Θ^* at different points. Higher dimensional solutions would require advanced usage of non-liner constrained optimisations. Again we can plot the intensities, we see that the passive orders are much more sporadic with the jumps. The combined graph is very skewed as we observe many more passive orders and higher intensities hinder any visual inference from relative intensities, we can plot relative to time scale.

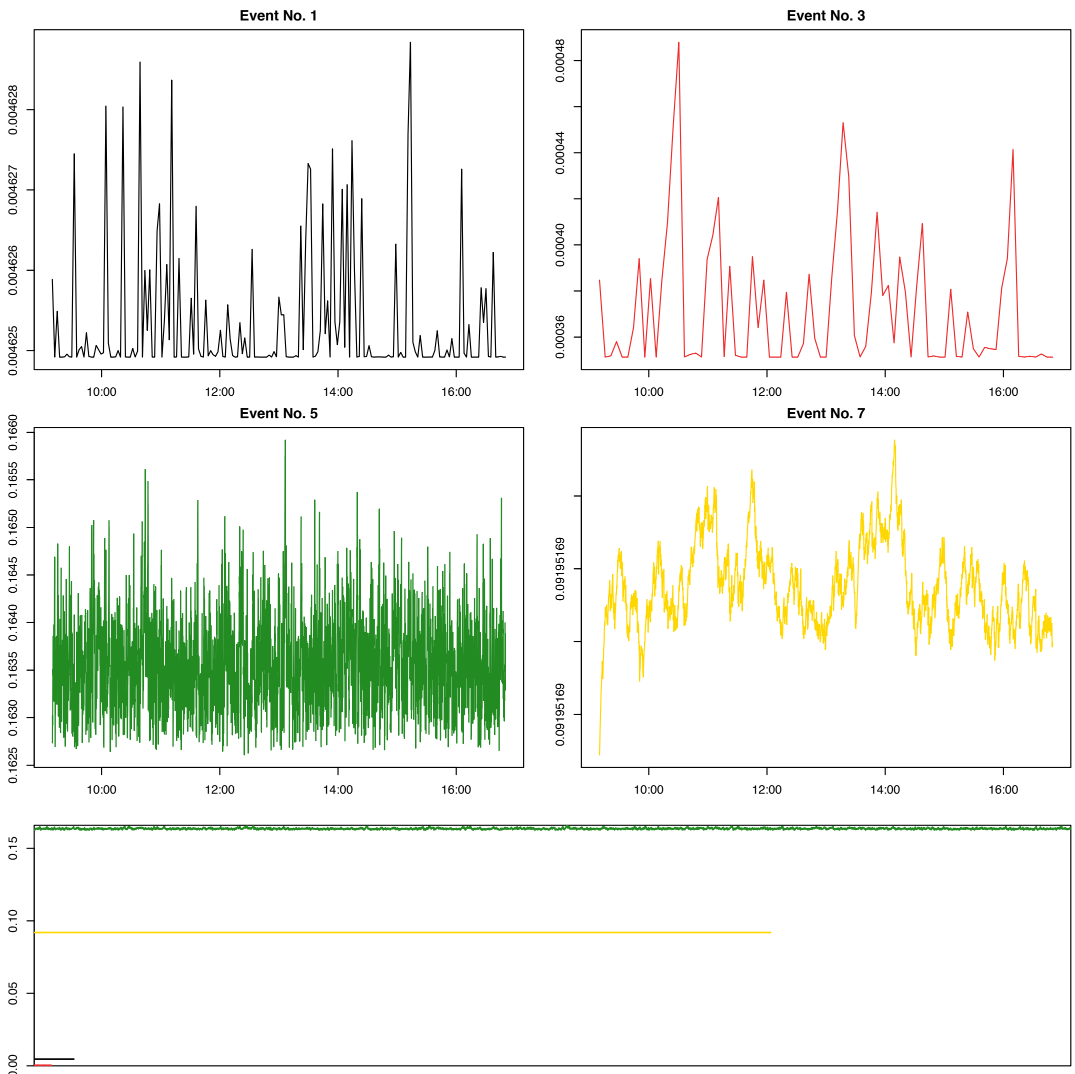

Figure 5: Comparison of Intensities $\lambda(t)$ for Different Order Types and Order 4 Hawkes Process.

Having applied an exponential kernel we can check whether the resulting inter-arrival times are exponentially distributed. To do this we assume no cross-excitation between different order types (i.e. set the cross diagonal values of the $\boldsymbol{\alpha}$ matrix to 0). We can define a recursive duration [4] (time-change) as

$$\Lambda\left(t_{j-1}, t_j\right) = \left[t_j - t_{j-1}\right]\mu + \alpha\left[1 - e^{-\beta(t_j - t_{j-1})}\right]E(j-1) \tag{16}$$

Where,

$$E(j-1) = 1 + e^{-\beta(t_{j-1} - t_{j-2})}A(j-2)$$

From Equation (16) we can generate empirical quantile plots for the distribution of the durations of the simulated Hawkes process with a particular choice of estimated parameters — optimal parameter set Θ^*. The results of checking exponential results are illustrated in Figure (6) which shows the durations of the simulated results as a QQ - plot compared to an exponential distribution with rate parameter $1/\bar{\Lambda}$. The plots show that the exponential assumption holds well for lower quantiles and for the aggressive orders seems relatively well-fitted, however, deviate about the exponential quantile line for larger quantiles especially for passive orders.

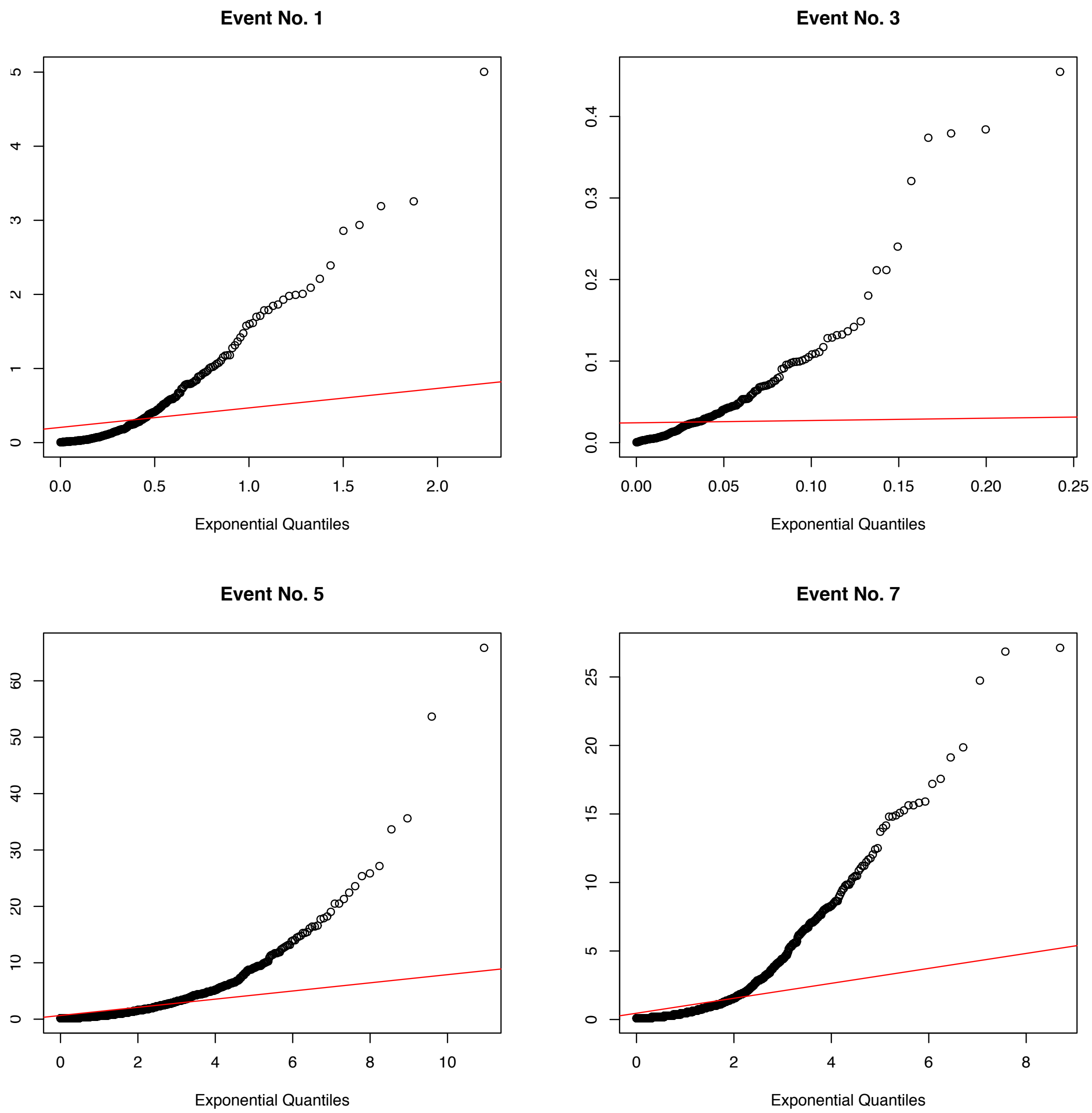

Figure 6: Exponential QQ-Plots for Different Events.

Simulate the Order-Book

Simulating the Order book is a unique problem, after having simulated the event arrivals using a calibrated Hawkes process we know when the events arrive. What is unknown is the value and quantity of the events. To generate an accurate reflection of the order-book is to make many assumptions about the order type dynamics, in a real-world attempt we would simulate order arrival volumes and use the volumes to determine a relative price impact. For the purposes of this Assignment we simulate prices only, noting that in order for the simulation to conform to empirical results we are almost surely overfitting. In both the simulated and empirical LOB the ask is represented in blue whilst the bid is in black. Figure (7) shows the results of 3 runs for a simulated order-book. We know the event arrivals from calibration in the previous steps, what is unknown is the dynamics of the orders which arrive. I assume that for an aggressive buy (sell) order the price impact follows an exponential distribution causing an uptick (downtick) on the best ask (bid) of $\exp(\lambda)$ where λ is a rate of one. For a passive buy (sell) I assume a truncated normal distribution with mean zero and standard deviation one hundred, the order will move the bid (ask) up (down) with this distribution. Note here the truncated distribution is used to ensure the spread is not crossed.

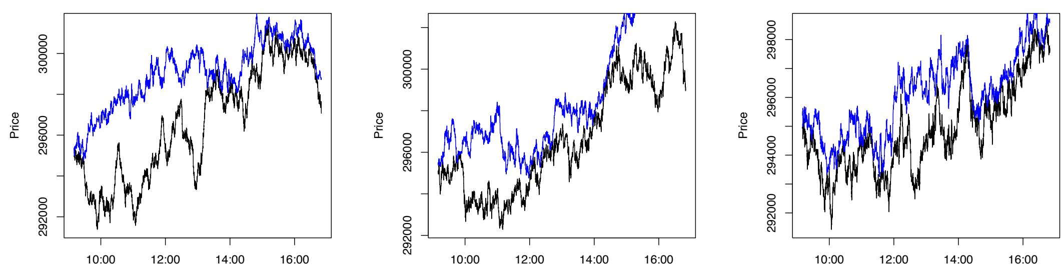

Figure 7: Simulated Order-Book for Four Event Types using Hawkes Process

Figure (7) is clearly not very robust in capturing the market dynamics, this is likely due to a rather "guess-like" justification for the distributions of prices. As mentioned previously a better representation would be to simulate volumes and place assumptions on the price impact. It is worth mentioning that if we simulate the process many times and take the mean of the paths with the above mentioned distributions I actually find that bids decrease whilst asks increase (the spread increases causing a big divergence). Thus restrictions for simulating the order book would likely need to capture the spread contracting and relaxing based on market pressures. When the simulated LOB is compared to Figure (8) we can loosely create a justification for the simulation being a success, but it is rather chosen to be like this and the power of the simulation is particularly weak.

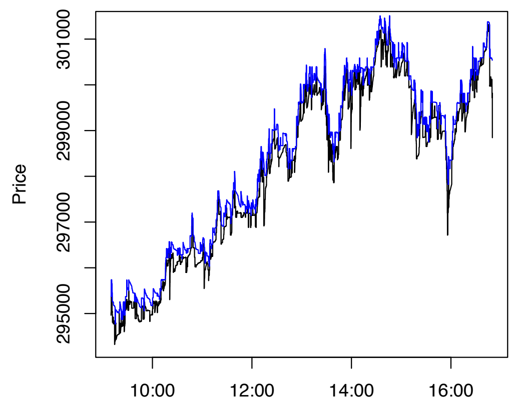

Figure 8: True LOB.

Conclusions

This assignment was particularly challenging, the whole premise of simulating order-book dynamics is full of biases due to lack of formal definition. Simply classifying trades — the first step in the process — is subjective and in reality there are many different order types that would make calibrating a Hawkes process infeasible. The process provides a good baseline for event arrival simulation, however, knowing when an event occurs is not enough to model market dynamics. The problem is synonymous with knowing I have an exam on the 22^{nd}. I know when the exam (event) is occurring but I can not precisely model the content (volume and price impact) in the exam. Sure, I have an outline based on my knowledge of the examiner and content load (history and domain knowledge) which allows me to make assumptions about the event dynamics. But there is almost surely no way of providing exact simulation in the presence of noise, this begs the question as to whether we are better off making simplistic model assumptions or whether complexity trumps all.

- END -

References

[1] Daley, D., and Vere-Jones, D. Basic properties of the poisson process. *An Introduction to the Theory of Point Processes: Volume I: Elementary Theory and Methods* (2003), 19–40.

[2] Dassios, A., Zhao, H., et al. Exact simulation of hawkes process with exponentially decaying intensity. *Electronic Communications in Probability 18* (2013).

[3] Large, J. Measuring the resiliency of an electronic limit order book. *Journal of Financial Markets 10*, 1 (2007), 1–25.

[4] Ozaki, T. Maximum likelihood estimation of hawkes' self-exciting point processes. *Annals of the Institute of Statistical Mathematics 31*, 1 (1979), 145–155.

[5] Rizoiu, M.-A., Lee, Y., Mishra, S., and Xie, L. A tutorial on hawkes processes for events in social media. *arXiv preprint arXiv:1708.06401* (2017).

[6] Zaatour, R. *hawkes: Hawkes process simulation and calibration toolkit*, 2014. R package version 0.0-4.

Appendix

R Code

```
##
#
# Author: Julian Albert
# Date: 22 September 2019
#
# Description:
# HFT Assignment 3 - Basically want to work with TAQ data and try simulate the
# order book activity using hawkes processes
#
#------------------------------------------------------------------------------#

# 0. Clean Workspace, Directory and Library ----

## Clean Workspace
rm(list=ls())
dev.off() # Close Plots
setwd("~") # Clear Path to User

## Locations
project_folder <- "/Documents/UCT/Coursework/HFT"
loc_script <- "/Assignment_3/UCT_Assignment/Code/HFT_Ass3"
loc_figs <- "/Assignment_3/UCT_Assignment/Figs"
loc_data <- "/Data"

## Directories
dir_script <- paste("~", project_folder, loc_script, sep = '')
dir_figs <- paste("~", project_folder, loc_figs, sep = '')
dir_data <- paste("~", project_folder, loc_data, sep = '')

## Filenames
filen_dat <- "/dat_TAQ.rds"

## Set Working Directory to Script Location
setwd(dir_script)

## Libraries - Lazyload
if (!require("pacman")) install.packages("pacman")
p_load(tidyverse, lubridate, zoo, data.table, Cairo, hawkes, truncdist)
```

```r

## Import Clean Data
dat_TAQ <- readRDS(file = paste(dir_data, filen_dat, sep = ""))
#setwd(dir_data)
#load(file = "dat_SimHawkes.RData")

## Options
options(digits.secs = 3)

# 0.1 Simulate Accept/Reject for Inhomogenous Poisson Process ----

## Plot an Inhomogeneous Poisson process by thinning
set.seed(123)
func.lambda <- function(t){2 + sin(t)}

M = 4; Time = 4*pi
p = numeric(); py = numeric()
r = numeric(); ry = numeric()
t = 0

while(t < Time)
{
  t = t + rexp(1, M)
  if(t < Time){u = runif(1)}
  if(u <= func.lambda(t)/M){
    p = c(p, t)
    py = c(py, M*u)}
  else{
    r = c(r, t)
    ry = c(ry, M*u)}
}

t <- seq(0, Time, by = 0.01)
plot(c(0, Time), c(0, M), type = 'n', xlab = "Time", ylab = "",
     xaxs = 'i', yaxs ='i')
lines(t, func.lambda(t), type = 'l')
lines(c(0, Time), c(M, M), lwd = 4)
points(p, py, pch = 16, col = 'darkgreen')
points(r, ry, pch = 3, col = 'red')
segments(p, rep(0, length(p)), p, py, lty = 'dashed', col = 'darkgreen')
dev.off()

# 1. Data Classification ----

## Let's work with Naspers
dat_NPN <- dat_TAQ$NPN

# Select one day of trading
random_date <- sample(unique(as.Date(dat_NPN$DateTimeL)), 1)
dat_NPN_oneday <- dat_NPN %>%
  filter(as.Date(DateTimeL) == random_date)

# Classify Market Orders... >> There is a bug here, if sell follows buy?
dat_NPN_oneday_classified <- dat_NPN_oneday %>%
  mutate(Test.Sign = Trade.Sign, After_Bid = lead(L1.Bid),
         Before_Bid = lag(L1.Bid), After_Ask = lead(L1.Ask),
         Before_Ask = lag(L1.Ask)) %>%
  mutate(MO_Class =
           ifelse(Trade.Sign == -1 & After_Bid < Before_Bid, "Sell Moves Bid",
            ifelse(Trade.Sign == -1, "Sell Doesn't Move Bid",
              ifelse(Trade.Sign == 1 & After_Ask > Before_Ask, "Buy Moves Ask",
                ifelse(Trade.Sign == 1, "Buy Doesn't Move Ask", NA)))))

# Classify Limit Orders... >> Bids and Asks need to be seperate

## Bid Classification
tmp_bids_class <- dat_NPN_oneday %>%
  select(DateTimeL, L1.Bid, L1.Ask) %>%
  mutate(Before_Bid = lag(L1.Bid),
         Before_Ask = lag(L1.Ask)) %>%
  mutate(LO_Bid_Class =
           ifelse(L1.Bid > Before_Bid & L1.Bid < Before_Ask, "Bid Btwn Quotes",
            ifelse(L1.Bid <= Before_Bid, "Bid At/Below Best Bid", NA)))
```

```r

## Ask Classification
tmp_asks_class <- dat_NPN_oneday %>%
  select(DateTimeL, L1.Bid, L1.Ask) %>%
  mutate(Before_Bid = lag(L1.Bid),
         Before_Ask = lag(L1.Ask)) %>%
  mutate(LO_Ask_Class =
           ifelse(L1.Ask > Before_Bid & L1.Ask < Before_Ask, "Ask Btwn Quotes",
                  ifelse(L1.Ask >= Before_Ask, "Ask At/Above Best Ask", NA)))

## Final DataFrame with Classified Events
dat_NPN_oneday_classified <- dat_NPN_oneday_classified %>%
  mutate(LO_Bid_Class = tmp_bids_class$LO_Bid_Class,
         LO_Ask_Class = tmp_asks_class$LO_Ask_Class)

## We need event arrival-times... subset data into events, take arrival times

## All of the events
Event_vector <- c("Sell Moves Bid", "Sell Doesn't Move Bid",
                  "Buy Moves Ask", "Buy Doesn't Move Ask",
                  "Bid Btwn Quotes", "Bid At/Below Best Bid",
                  "Ask Btwn Quotes", "Ask At/Above Best Ask")
## Column names of data
Column_name_vector <- c("MO_Class", "LO_Bid_Class", "LO_Ask_Class")

## function to subset data to get arrival times
func.get_events <- function(data, Event, Column_name, Type)
{
  ## Subset data into event
  dat_filtered <- data %>%
    filter(get(Column_name) == Event)

  if(Type == "Quote"){
    ## Get Quote relevant columns
    dat_filtered <- dat_filtered %>%
      select(DateTimeL, Type, L1.Ask, L1.Bid, MidPrice, Volume.Ask, Volume.Bid,
             MicroPrice)
  } else{
    ## Get Trade relevant columns
    dat_filtered <- dat_filtered %>%
      select(DateTimeL, Type, Price, Trade.Sign, Volume.Trade)
  }
  return(dat_filtered)
}

## Initialise lists
MO_Classes = LO_Bid_Classes = LO_Ask_Classes = list()

## Market Order Events are the first 4
for(i in 1:4){
  MO_Classes[[i]] <- func.get_events(dat_NPN_oneday_classified,
                                     Event_vector[i], Column_name_vector[1],
                                     "Trade")
}
names(MO_Classes) <- Event_vector[1:4]

## Bid Limit Order Events are 5th and 6th
for(i in 1:2){
  LO_Bid_Classes[[i]] <- func.get_events(dat_NPN_oneday_classified,
                                         Event_vector[4+i], Column_name_vector[2],
                                         "Quote")
}
names(LO_Bid_Classes) <- Event_vector[5:6]

## Ask Limit Order Events are 7th and 8th
for(i in 1:2){
  LO_Ask_Classes[[i]] <- func.get_events(dat_NPN_oneday_classified,
                                         Event_vector[6+i], Column_name_vector[3],
                                         "Quote")
}
names(LO_Ask_Classes) <- Event_vector[7:8]

## Event Arrivals Relative to Day Start
```

```r
DayStartTime <- as.POSIXct(paste(random_date, "09:10:00", sep = " "),
                           timezone = "Africa/Johannesburg")
DayEndTime <- dat_NPN_oneday[NROW(dat_NPN_oneday), ]$DateTimeL

## Get Arrival times for each Event Type relative to day starting time
func.calc_arrival <- function(data)
{
  dat_new <- data %>%
    mutate(arrival_times = difftime(data$DateTimeL, DayStartTime, units = "secs"))

  return(dat_new)
}

MO_Classes <- lapply(MO_Classes, func.calc_arrival)
LO_Bid_Classes <- lapply(LO_Bid_Classes, func.calc_arrival)
LO_Ask_Classes <- lapply(LO_Ask_Classes, func.calc_arrival)

## Now We Want just arrival times
func.select_arrival <- function(data)
{
  dat_new <- data %>% select(arrival_times)
  return(dat_new)
}

MO_Arrivals <- lapply(MO_Classes, func.select_arrival)
LO_Bid_Classes <- lapply(LO_Bid_Classes, func.select_arrival)
LO_Ask_Classes <- lapply(LO_Ask_Classes, func.select_arrival)

# All arrival times in a single list
All_Arrivals <- list(MO_Arrivals[[1]], MO_Arrivals[[2]],
                     MO_Arrivals[[3]], MO_Arrivals[[4]],
                     LO_Bid_Classes[[1]], LO_Bid_Classes[[2]],
                     LO_Ask_Classes[[1]], LO_Ask_Classes[[2]])

Event_No_Names_vec <- c()
for(i in 1:8) {Event_No_Names_vec[i] =(paste("Event_No", i, sep = ""))}
names(All_Arrivals) <- Event_No_Names_vec
# lapply(All_Arrivals, function(x) any(is.na(x))) # No NA values

## Pull removes tibble
All_Arrivals_list_of_vectors <- lapply(All_Arrivals, function(x) as.vector(pull(x)))

# 2. Estimate Hawkes Process ----

k = 2
## Likelihood function with constraints
func_likelihoodHawkes = function(params, k, columns)
{
  lambda = matrix(params[1:k], nrow = k)
  alpha = matrix(params[(k+1):(k^2 + k)], nrow = k, byrow = TRUE)
  beta = matrix(params[(k^2 + k + 1):(k^2 + 2*k)], nrow = k)
  eigenval = Re(eigen(diag(c(beta)) - alpha)$values)
  if(min(eigenval) <= 0 | min(alpha) < 0 | min(lambda) < 0){
    pen <- 1000000
    ll <- likelihoodHawkes(lambda, alpha, beta,
                           All_Arrivals_list_of_vectors[columns])
    return(ll + pen)
   }else{
     pen <- 0
     ll <- likelihoodHawkes(lambda, alpha, beta,
                            All_Arrivals_list_of_vectors[columns])
   return(ll + pen)
   }
}

## convert vector of pars into matrix
func.get_pars <- function(params, k)
{
  lambda0 = matrix(params[1:k], nrow = k)
  alpha = matrix(params[(k+1):(k^2 + k)], nrow = k, byrow = TRUE)
  beta = matrix(params[(k^2 + k + 1):(k^2 + 2*k)], nrow = k)
  return(list(lambda0, alpha, beta))
}
```

```r

## Horizon and true number of events
secs_in_day <- as.numeric(difftime(DayEndTime, DayStartTime, units = "secs"))
n_true <- lapply(All_Arrivals, function(x) NROW(x)) %>% unlist()

# lambda0 will be 8x1 | alpha will be 8x8 | beta will be 8x1
## Market Sell and Buy (aggressive)
lambda_init <- lapply(All_Arrivals_list_of_vectors[c(1, 3)],
                      function(x) NROW(x)/secs_in_day) %>% unlist()
lambda_init_3 <- lapply(All_Arrivals_list_of_vectors[c(5, 7)],
                        function(x) NROW(x)/secs_in_day) %>% unlist()
lambda_init_4 <- lapply(All_Arrivals_list_of_vectors[c(1, 3, 5, 7)],
                        function(x) NROW(x)/secs_in_day) %>% unlist()
## Random starting parameters >> satisfy constraints
set.seed(123)
alpha_init <- diag(0.01, k) %>% as.vector()
beta_init <- rep(0.05, k)
params_init <- as.numeric(c(lambda_init, alpha_init, beta_init))

alpha_init_3 <- diag(0.01, k) %>% as.vector()
beta_init_3 <- rep(0.05, k)
params_init_3 <- as.numeric(c(lambda_init_3, alpha_init_3, beta_init_3))

alpha_init_4 <- diag(c(0.01, 0.01, 0.01, 0.01), 4) %>% as.vector()
beta_init_4 <- c(0.05, 0.05, 0.05, 0.05)
params_init_4 <- as.numeric(c(lambda_init_4, alpha_init_4, beta_init_4))
# func_likelihoodHawkes(params_init, k = 2)
## Market Sell and Buy (aggressive)
params_hawkes_optim_NM <- optim(params_init, func_likelihoodHawkes, k = 2,
                                columns = c(1, 3), method = "Nelder-Mead",
                                control = list(maxit = 5000))
## Market Sell and Buy (passive)
params_hawkes_optim_NM_3 <- optim(params_init_3, func_likelihoodHawkes, k = 2,
                                  columns = c(5, 7), method = "Nelder-Mead",
                                  control = list(maxit = 5000))
## Market Sell and Buy (passive) and aggressive
params_hawkes_optim_NM_4 <- optim(params_init_4, func_likelihoodHawkes, k = 4,
                                  columns = c(1, 3, 5, 7), method = "Nelder-Mead",
                                  control = list(maxit = 10000))

## Converges
params_hawkes_optim_NM$convergence
params_hawkes_optim_NM_3$convergence
params_hawkes_optim_NM_4$convergence
## Optimal Pars
optim_pars <- params_hawkes_optim_NM$par
optim_pars_3 <- params_hawkes_optim_NM_3$par
optim_pars_4 <- params_hawkes_optim_NM_4$par
## Get parameter matrices
params_mat_hawkes <- func.get_pars(optim_pars, k)
params_mat_hawkes_3 <- func.get_pars(optim_pars_3, k)
params_mat_hawkes_4 <- func.get_pars(optim_pars_4, 4)
## Clean names
names(params_mat_hawkes) <- c("lambda0_opt", "alpha_opt", "beta_opt")
names(params_mat_hawkes_3) <- c("lambda0_opt", "alpha_opt", "beta_opt")
names(params_mat_hawkes_4) <- c("lambda0_opt", "alpha_opt", "beta_opt")

## Simulate(estimate) hawkes processes
res.sim_hawkes <- simulateHawkes(lambda0 = params_mat_hawkes$lambda0_opt,
                                 alpha = params_mat_hawkes$alpha_opt,
                                 beta = params_mat_hawkes$beta_opt,
                                 horizon = secs_in_day)

res.sim_hawkes_3 <- simulateHawkes(lambda0 = params_mat_hawkes_3$lambda0_opt,
                                   alpha = params_mat_hawkes_3$alpha_opt,
                                   beta = params_mat_hawkes_3$beta_opt,
                                   horizon = secs_in_day)

res.sim_hawkes_4 <- simulateHawkes(lambda0 = params_mat_hawkes_4$lambda0_opt,
                                   alpha = params_mat_hawkes_4$alpha_opt,
                                   beta = params_mat_hawkes_4$beta_opt,
                                   horizon = secs_in_day)
## Clean names
```

```r
names(res.sim_hawkes) <- Event_No_Names_vec[c(1, 3)]
names(res.sim_hawkes_3) <- Event_No_Names_vec[c(5, 7)]
names(res.sim_hawkes_4) <- Event_No_Names_vec[c(1, 3, 5, 7)]

## Check number of simulated vs true
n_sim <- lapply(res.sim_hawkes, function(x) NROW(x)) %>% unlist()
n_sim_3 <- lapply(res.sim_hawkes_3, function(x) NROW(x)) %>% unlist()
n_sim_4 <- lapply(res.sim_hawkes_4, function(x) NROW(x)) %>% unlist()

n_true[c(1, 3)]; n_sim
n_true[c(5, 7)]; n_sim_3
n_true[c(1, 3, 5, 7)]; n_sim_4

## Calculate Intensities of events
func.intensity <- function(lambda_opt, alpha_opt, beta_opt, res, tend)
{

  mu <- lambda_opt
  mult_constant <- alpha_opt*beta_opt
  integral <- sum(exp(-beta_opt*(res[tend] - res[c(1:(tend-1))])))
  lambda_t <- mu + mult_constant*integral

  return(lambda_t)
}

## Objectives checking... Lots of local minimum in optim()
func.obj_calc <- function(par_mat, k, n_tests, alpha_range, beta_range,
                          lambda_range, columns)
{

  alpha_diags <- lapply(seq(alpha_range[1], alpha_range[2],
                            length.out = n_tests^2), diag, nrow = 2)
  beta_comb <- expand.grid(seq(beta_range[1], beta_range[2], length.out = n_tests),
                           seq(beta_range[1], beta_range[2], length.out = n_tests))
  lambda_comb <- expand.grid(seq(lambda_range[1], lambda_range[2], length.out = n_tests),
                             seq(lambda_range[1], lambda_range[2], length.out = n_tests))

  z_lambda <- numeric(dim(lambda_comb)[1])
  z_alpha <- numeric(dim(lambda_comb)[1])
  z_beta <- numeric(dim(lambda_comb)[1])

  for(i in 1:dim(lambda_comb)[1])
  {
    ## Constant Lambda
    tmp.pars <- c(as.numeric(par_mat$lambda0_opt),
                  as.numeric(alpha_diags[[i]]),
                  as.numeric(beta_comb[i, ]))
    z_lambda[i] <- func_likelihoodHawkes(tmp.pars, k, columns = columns)
    ## Constant Alpha
    tmp.pars <- c(as.numeric(lambda_comb[i, ]),
                  as.numeric(par_mat$alpha_opt),
                  as.numeric(beta_comb[i, ]))
    z_alpha[i] <- func_likelihoodHawkes(tmp.pars, k, columns = columns)
    ## Constant Beta
    tmp.pars <- c(as.numeric(as.numeric(lambda_comb[i, ])),
                  as.numeric(alpha_diags[[i]]),
                  as.numeric(par_mat$beta_opt))
    z_beta[i] <- func_likelihoodHawkes(tmp.pars, k, columns = columns)
  }

  Z_lambdas <- matrix(z_lambda, nrow = n_tests)
  Z_betas <- matrix(z_beta, nrow = n_tests)
  Z_alphas <- matrix(z_alpha, nrow = n_tests)

  return(list(Z_lambdas = Z_lambdas, Z_betas = Z_betas, Z_alphas = Z_alphas))

}

alpha_range_1 <- c(0.01, 0.04)
beta_range_1 <- c(0.041, 0.06)
lambda_range_1 <- c(0.001, 0.01)
test1 <- func.obj_calc(params_mat_hawkes, 2, 100,
                       alpha_range_1, beta_range_1, lambda_range_1, c(1, 3))
```

```
alpha_range_3 <- c(0.06, 0.08)
beta_range_3 <- c(0.081, 0.11)
lambda_range_3 <- c(0.02, 0.1)
test3 <- func.obj_calc(params_mat_hawkes_3, 2, 100,
                       alpha_range_3, beta_range_3, lambda_range_3, c(5, 7))

func.plot_obj <- function(n_tests, z, levels, title)
{
  cols = c('green','yellow','blue')

  filled.contour(seq(0, 1, length.out = n_tests), seq(0, 1, length.out = n_tests), z,
                 plot.axes = {contour(seq(0, 1, length.out = n_tests),
                                      seq(0, 1, length.out = n_tests),
                                      z, add = T, nlevels = 30)}, main = title,
                 color.palette = colorRampPalette(cols))

}

setwd(dir_figs)
cairo_pdf("HFT_Ass3_fig_objs_agg_E13_lam.pdf", height = 5, width = 5)
func.plot_obj(100, test1$Z_lambdas, 30, expression("Constant "~lambda))
dev.off()

cairo_pdf("HFT_Ass3_fig_objs_agg_E13_beta.pdf", height = 5, width = 5)
func.plot_obj(100, test1$Z_betas, 30, expression("Constant "~beta))
dev.off()

cairo_pdf("HFT_Ass3_fig_objs_agg_E13_alpha.pdf", height = 5, width = 5)
func.plot_obj(100, test1$Z_alphas, 30, expression("Constant "~alpha))
dev.off()

cairo_pdf("HFT_Ass3_fig_objs_agg_E57_lam.pdf", height = 5, width = 5)
func.plot_obj(100, test3$Z_lambdas, 30, expression("Constant "~lambda))
dev.off()

cairo_pdf("HFT_Ass3_fig_objs_agg_E57_beta.pdf", height = 5, width = 5)
func.plot_obj(100, test3$Z_betas, 30, expression("Constant "~beta))
dev.off()

cairo_pdf("HFT_Ass3_fig_objs_agg_E57_alpha.pdf", height = 5, width = 5)
func.plot_obj(100, test3$Z_alphas, 30, expression("Constant "~alpha))
dev.off()

func.res_sim_hawkes <- function(res, k_events, param_mat, event_no)
{

  colour_vec <- c("black", "firebrick2", "forestgreen", "gold",
                  "darkorange", "darkorchid3", "violetred1", "dodgerblue3")

  ## Optimal Parameters per Event
  tmp.mu <- as.numeric(param_mat$lambda0_opt)
  tmp.alpha <- diag(param_mat$alpha_opt)
  tmp.beta <-  as.numeric(param_mat$beta_opt)
  lambda_t <- numeric()
  lambda_t_list <- list()

  ## Calculate for each event at each arrival
  for(event in 1:k_events){
    for(i in 2:NROW(res[[event]])){
      lambda_t[i] <- func.intensity(tmp.mu[event], tmp.alpha[event],
                                    tmp.beta[event], res[[event]], i)
    }
    lambda_t_list[[event]] <- na.omit(lambda_t)
    lambda_t <- numeric()
  }

  ## find max/min for plots
  tmp_max_intensity <- max(unlist(lapply(lambda_t_list, max)))
  tmp_min_lambdas <- min(unlist(lapply(lambda_t_list, NROW)))
  tmp_max_lambdas <- max(unlist(lapply(lambda_t_list, NROW)))

  if(k_events > 2){
```

```
    m <- matrix(c(1,2,3,4,5,5), nrow = 3, ncol = 2, byrow = TRUE)
    layout(mat = m, heights = c(2,2,1.5))
    }else{
  m <- matrix(c(1,2,3,3), nrow = 2, ncol = 2,byrow = TRUE)
  layout(mat = m, heights = c(0.4,0.4,0.2))}

  par(mar = c(2, 2, 2, 2))
  #par(mfrow = c(1, 2))
  for(i in 1:k_events){
    plot(y = lambda_t_list[[i]],
         x = seq(DayStartTime, DayEndTime, length.out = NROW(lambda_t_list[[i]])),
         type = 'l', col = colour_vec[i],
         ylab = expression(lambda~"(t)"), xlab = "Time",
         main = paste("Event No.", event_no[i], sep = " "))
  }
  #par(mfrow = c(1, 1))
  plot(y = c(0, tmp_max_intensity), x = c(0, tmp_max_lambdas),
       ylab = expression(lambda~"(t)"), type = 'n',
       xlab = '', yaxs = 'i', xaxs = 'i', xaxt="n")
  for(i in 1:k_events){
    lines(lambda_t_list[[i]], col = colour_vec[i], lwd = 1.5) }
}

setwd(dir_figs)
cairo_pdf("HFT_Ass3_fig_intensities_agg.pdf", height = 7, width = 7)
func.res_sim_hawkes(res.sim_hawkes, 2, params_mat_hawkes, c(1, 3))
dev.off()

cairo_pdf("HFT_Ass3_fig_intensities_pass.pdf", height = 7, width = 7)
func.res_sim_hawkes(res.sim_hawkes_3, 2, params_mat_hawkes_3, c(5, 7))
dev.off()

cairo_pdf("HFT_Ass3_fig_intensities_agg_and_pass.pdf", height = 10, width = 10)
func.res_sim_hawkes(res.sim_hawkes_4, 4, params_mat_hawkes_4, c(1, 3, 5, 7))
dev.off()

colour_vec <- c("black", "firebrick2", "forestgreen", "gold",
                "darkorange", "darkorchid3", "violetred1", "dodgerblue3")

## Optimal Parameters per Event
tmp.mu <- as.numeric(params_mat_hawkes_4$lambda0_opt)
tmp.alpha <- diag(params_mat_hawkes_4$alpha_opt)
tmp.beta <-  as.numeric(params_mat_hawkes_4$beta_opt)
lambda_t <- numeric()

lambda_t_list <- list()
## Calculate for each event at each arrival
for(event in 1:length(res.sim_hawkes_4)){
  for(i in 2:NROW(res.sim_hawkes_4[[event]])){
    lambda_t[i] <- func.intensity(tmp.mu[event], tmp.alpha[event],
                                  tmp.beta[event], res.sim_hawkes_4[[event]], i)
  }
  lambda_t_list[[event]] <- na.omit(lambda_t)
  lambda_t <- numeric()
}

list_intensity <- list()
types_vec <- c('AggSell', 'AggBuy', 'PassBuy', 'PassSell')

for(i in 1:length(res.sim_hawkes_4)){
df_intensity <- data.frame(Intensity = lambda_t_list[[i]],
                           Time = DayStartTime + res.sim_hawkes_4[[i]][-1],
                           Count = 1:(NROW(res.sim_hawkes_4[[i]])-1),
                           Event = types_vec[i])

list_intensity[[i]] <- df_intensity
}


setwd(dir_figs)
cairo_pdf("HFT_Ass3_fig_intensities_agg_wcount.pdf", height = 7, width = 7)
dat_plot_intensities <- do.call(rbind, list_intensity[c(1,2)]) %>% arrange(Time)
scaleFactor <- max(dat_plot_intensities$Intensity) / max(dat_plot_intensities$Count)
```

```
ggplot(dat_plot_intensities, aes(x=Time, color = Event)) +
  geom_line(aes(y=Intensity)) +
  geom_line(aes(y=Count * scaleFactor)) +
  scale_y_continuous(name="Intensity",
                     sec.axis=sec_axis(~.*1/scaleFactor, name="Count")) +
  theme_bw()
dev.off()

cairo_pdf("HFT_Ass3_fig_intensities_pass_wcount.pdf", height = 7, width = 7)
dat_plot_intensities <- do.call(rbind, list_intensity[c(3,4)]) %>% arrange(Time)
scaleFactor <- max(dat_plot_intensities$Intensity) / max(dat_plot_intensities$Count)
ggplot(dat_plot_intensities, aes(x=Time, color = Event)) +
  geom_line(aes(y=Intensity)) +
  geom_line(aes(y=Count * scaleFactor)) +
  scale_y_continuous(name="Intensity",
                     sec.axis=sec_axis(~.*1/scaleFactor, name="Count")) +
  theme_bw()
dev.off()

cairo_pdf("HFT_Ass3_fig_intensities_agg_and_pass_wcount.pdf", height = 7, width = 7)
dat_plot_intensities <- do.call(rbind, list_intensity) %>% arrange(Time)
scaleFactor <- max(dat_plot_intensities$Intensity) / max(dat_plot_intensities$Count)
ggplot(dat_plot_intensities, aes(x=Time, color = Event)) +
  geom_line(aes(y=Intensity)) +
  geom_line(aes(y=Count * scaleFactor)) +
  scale_y_continuous(name="Intensity",
                     sec.axis=sec_axis(~.*1/scaleFactor, name="Count")) +
  theme_bw()
dev.off()

# 3. Check Exponential results of data ----

# make list of dfs..
test_interarrivals <- lapply(All_Arrivals_list_of_vectors[c(1, 3, 5, 7)],
                             diff, differences = 1)

df <- list(); k_events = 4
for(j in 1:k_events){
df[[j]] <- list(inter_arrivals = test_interarrivals[[j]],
               alpha_opt = diag(params_mat_hawkes_4$alpha_opt)[j],
               beta_opt = params_mat_hawkes_4$beta_opt[j],
               lambda_opt = params_mat_hawkes_4$lambda0_opt[j])
}

names(df) <- Event_No_Names_vec[c(1, 3, 5, 7)]

func.exponential_check <-  function(df)
{

 A = c()
 E = c()
 A0 = 1
 E0 = 1
 A[1] = E0*exp(-df$beta_opt * df$inter_arrivals[1])
 E[1] = exp (- df$beta_opt*df$inter_arrivals[1])*A[1]

 for( i in 2:(length(df$inter_arrivals)) ){
   E[i] = 1 + exp(-df$beta_opt*df$inter_arrivals[i-1])*A[i -1]
   A[i] = E[i - 1]*exp(-df$beta_opt*df$inter_arrivals[i])
 }

 tmp = c()
 tmp[1] = df$inter_arrivals[1]*df$lambda_opt +
   df$alpha_opt*(1- exp(- df$beta_opt*df$inter_arrivals[1]))*E0

 for( i in 2:(length(df$inter_arrivals)) ){
   tmp[i] = df$inter_arrivals[i]*df$lambda_opt +
     df$alpha_opt*(1- exp(- df$beta_opt*df$inter_arrivals[i-1]))*E[i -1]
 }

 Total_duration = tmp
 return(Total_duration)

```

```
}

check_exps <- lapply(df, func.exponential_check)
tmp_exp_names <- c(1, 3, 5, 7)

setwd(dir_figs)
cairo_pdf("HFT_Ass3_fig_check_exps.pdf", height = 10, width = 10)
par(mfrow = c(2, 2))
for(i in 1:k_events){
  main = paste("Event No.", tmp_exp_names[i], sep = " ")
  qqplot(qexp(ppoints(length(check_exps[[i]])),
              rate = 1/mean(check_exps[[i]]) ), check_exps[[i]],
         xlab = "Exponential Quantiles", ylab = '', main = main)
  qqline(check_exps[[i]], col = 'red')
}
dev.off()

# 4. Simluate the Order-Book ----

## Save workspace so we can use same data...
setwd(dir_data)
save.image(file = "dat_SimHawkes.RData")
setwd(dir_script)

## Initialise Midprice and spread
MP_init <- 100
spread_init <- 4
## Sample initial prices and volumes using distributions???
sample_init_prices <- rnorm(1, 100, 20)
sample_init_vols <- rtrunc(1, "norm", mean = 200 , sd = 300)

##
simulated_arrivals <- res.sim_hawkes_4

## Update the Order-Book
types_vec <- c('AggSell', 'AggBuy', 'PassBuy', 'PassSell')
class(simulated_arrivals$Event_No1)
sim_dfs <- list()

for(i in 1:length(simulated_arrivals)){
  sim_dfs[[i]] <- data.frame(time = simulated_arrivals[[i]],
                             type = types_vec[i])
}

dat_ALL_sim_arrivals <- do.call(rbind, sim_dfs) %>%
  arrange(time)

DayLength <- NROW(dat_ALL_sim_arrivals)

sim_bid_prices <- c()
sim_ask_prices <- c()
spread_t <- c()
dat_true_init <- dat_NPN_oneday_classified[1, ]
sim_bid_prices[1] <- dat_true_init$L1.Bid
sim_ask_prices[1] <- dat_true_init$L1.Ask

func.sim_LOB <- function(daylength)
{
  for(t in 2:daylength){
    order <- dat_ALL_sim_arrivals[(t-1), ]
    spread <- max(sim_ask_prices[t-1] - sim_bid_prices[t-1], 0.00001)
    spread_t[t] <- spread
    if(order$type == 'AggSell'){
      tmp_neg_move = rexp(1, rate = 1)  # some distribution?
      sim_bid_prices[t] = sim_bid_prices[t-1] - tmp_neg_move
      sim_ask_prices[t] = sim_ask_prices[t-1]
    }else if(order$type == 'AggBuy'){
      tmp_pos_move = rexp(1, rate = 1) # some distribution?
      sim_bid_prices[t] = sim_bid_prices[t-1]
      sim_ask_prices[t] = sim_ask_prices[t-1] + tmp_pos_move
    }else if(order$type == 'PassBuy'){
      tmp_pos_move_pass_buy = rtrunc(1, "norm", b = spread, mean = 0, sd = 100) # some
          distribution?
```

```
      sim_bid_prices[t] = sim_bid_prices[t-1] + tmp_pos_move_pass_buy
      sim_ask_prices[t] = sim_ask_prices[t-1]
    }else if(order$type == 'PassSell'){
      tmp_neg_move_pass_sell = rtrunc(1, "norm", b = spread, mean = 0, sd = 100) # some
           distribution?
      sim_bid_prices[t] = sim_bid_prices[t-1]
      sim_ask_prices[t] = sim_ask_prices[t-1] - tmp_neg_move_pass_sell
    }
  }
  return(data.frame(Sim_L1.Bid = sim_bid_prices, Sim_L1.Ask = sim_ask_prices,
                    spread = spread_t))
}

test_OB <- func.sim_LOB(DayLength)

plot(y=test_OB$Sim_L1.Bid, x = (DayStartTime + dat_ALL_sim_arrivals$time),
     type = 'l', xlab = "Time", ylab = 'Price')
lines(y=test_OB$Sim_L1.Ask, x = (DayStartTime + dat_ALL_sim_arrivals$time), col ='blue')

setwd(dir_figs)
cairo_pdf('HFT_Ass3_fig_simulatedLOB3.pdf', height = 5, width = 5)
plot(y=test_OB$Sim_L1.Bid, x = (DayStartTime + dat_ALL_sim_arrivals$time),
     type = 'l', xlab = "Time", ylab = 'Price')
lines(y=test_OB$Sim_L1.Ask, x = (DayStartTime + dat_ALL_sim_arrivals$time), col ='blue')
dev.off()

cairo_pdf('HFT_Ass3_fig_trueLOB.pdf', height = 5, width = 5)
plot(na.omit(dat_NPN_oneday_classified$L1.Bid), type = 'l',
     xlab = "Time", ylab = 'Price', x = seq(DayStartTime, DayEndTime,
             length.out = length(na.omit(dat_NPN_oneday_classified$L1.Bid))))
lines(na.omit(dat_NPN_oneday_classified$L1.Ask), col = 'blue',
      x = seq(DayStartTime, DayEndTime,
              length.out = length(na.omit(dat_NPN_oneday_classified$L1.Ask))))
dev.off()
```

Plagiarism Declaration Form

A copy of this form, completed and signed, to be attached to all coursework submissions to the Statistical Sciences Department.

COURSE CODE:	**STA5091Z**
COURSE NAME:	**Data Analysis for High-Frequency Trading**
STUDENT NAME:	**Julian Albert**
STUDENT NUMBER:	**ALBJUL005**
GROUP NUMBER:	**1**

PLAGIARISM DECLARATION

- I know that plagiarism is wrong. Plagiarism is to use another's work and pretend that it is one's own.
- I have used a generally accepted citation and referencing style. Each contribution to, and quotation in, this tutorial/report/project from the work(s) of other people has been attributed, and has been cited and referenced.
- This tutorial/report/project is my own work.
- I have not allowed, and will not allow, anyone to copy my work with the intention of passing it off as his or her own work.
- I acknowledge that copying someone else's assignment or essay, or part of it, is wrong, and declare that this is my own work.
- Agreement to this statement does not exonerate me from the University's plagiarism rules.

Signature: Date: October 5, 2019