Summary

The paper, written by Yosihiko Ogata, titled, "On Lewis Simulation Method for Point Process", gives detailed a procedure to simulate univariate and multivariate point processes. In this document, I will list down the algo and necessary `R` code for simulating univariate Hawkes' self exciting process

Algorithm for simulating univariate Hawkes' process

Hawkes' model is a generalized point process model whose intensity function is given by

$$\Lambda(t) = \mu + \int_{-\infty}^{t} g(t-u)dN(u)$$

This paper deals with a response function of the type

$$g(t) = \alpha e^{-\beta t}$$

Consider an intensity function $\Lambda^*(t)$, a piece wise constant function such that

$$\lambda(t|t_1, \ldots, t_n) \leq \Lambda_i^*, \quad s_i \leq t < s_{i+1} \leq t_{n+1}$$

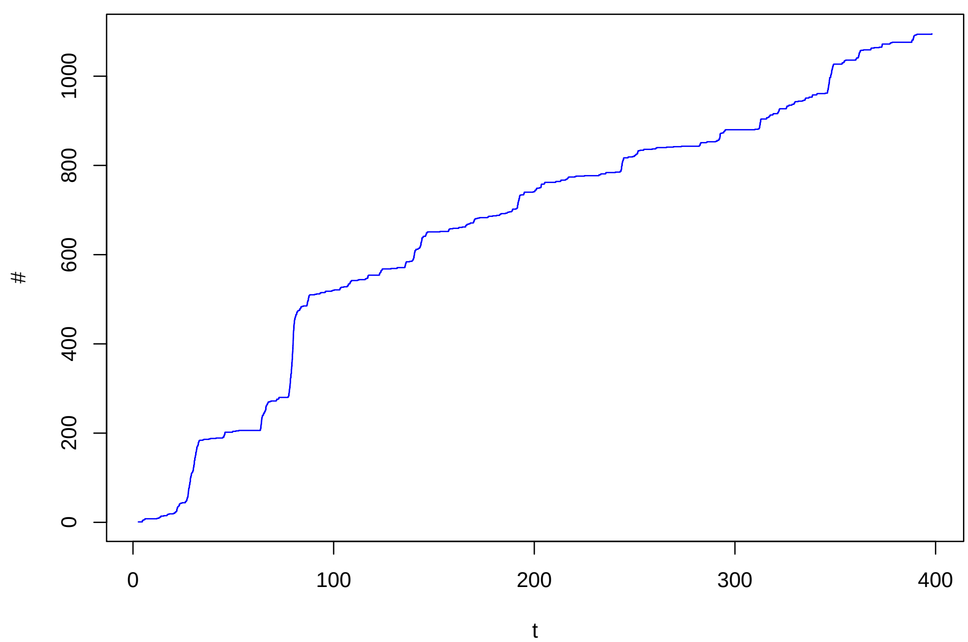
The following steps generate a Hawkes' process realization

1. Set $\Lambda_0^* = \mu$ and put $s_0 = 0$
2. Generate U_0 and put $u_0 = -\log(U_0/\Lambda_0^*)$
3. If $u_0 \leq T$, then put $t_1 = u_0$. Otherwise stop
4. Set $i = j = k = 0$ and $n = 1$
5. Set $k = k + 1$ and put $\Lambda_k^* = \lambda(t_n|t_1, t_2, \ldots t_{n-1}) + \alpha$
6. Set $j = j + 1$ and generate U_j
7. Set $i = i + 1$ and put $u_i = -\log(U_j/\Lambda_k^*)$
8. Put $s_i = s_{i-1} + u_i$. If $s_i > T$, stop
9. Set $j = j + 1$ and generate U_j
10. If $U_j \leq \lambda(s_i|t_1, \ldots, t_{n-1})/\Lambda_k^*$, set $n = n + 1$, put $t_n = s_i$, and go to step 5
11. Set $k = k + 1$, put $\Lambda_k^* = \lambda(t_n|t_1, t_2, \ldots t_{n-1})$ and go to step 6

The key to implementing this algorithm is to keep a running sum of the intensity function. Once one simulates the realization, we need a way to check whether the process indeed corresponds to the true process.

```
simulate_uni_hawkes <- function(mu, alpha, beta, t_max){
  arrivals          <- numeric()
  set.seed(1)
  s                 <- 0
  t                 <- 0
  lambda_star       <- mu
  s                 <- s -log(runif(1))/lambda_star
  t                 <- s
  dlambda           <- alpha
```

```
  arrivals           <- c(arrivals, t)
  while(s < t_max) {
    U                <- runif(1)
    s                <- s -log(U)/lambda_star
    u                <- runif(1)
    if(u <= (mu + dlambda*exp(-beta*(s-t)))/lambda_star){
      dlambda        <- alpha + dlambda*exp(-beta*(s-t))
      lambda_star    <- lambda_star + alpha
      t              <- s
      arrivals       <- c(arrivals, t)
    }
  }
  return(arrivals)
}
```



Estimating via MLE

The log likelihood given in the paper is coded below

```
loglik     <- function(params, arrivals){
  mu_i     <- params[1]
  alpha_i  <- params[2]
  beta_i   <- params[3]
```

```
  term_1    <- -mu_i*arrivals[n]
  term_2    <- sum(alpha_i/beta_i*(exp( -beta_i * (arrivals[n] - arrivals)) - 1))
  Ai        <- c(0, sapply(2:n, function(z) {
                 sum(exp( -beta_i * (arrivals[z]- arrivals[1:(z - 1)])))
                }))
  term_3   <- sum(log( mu_i + alpha_i * Ai))
  return(-term_1- term_2 -term_3)
}
```

Using `nlm` function

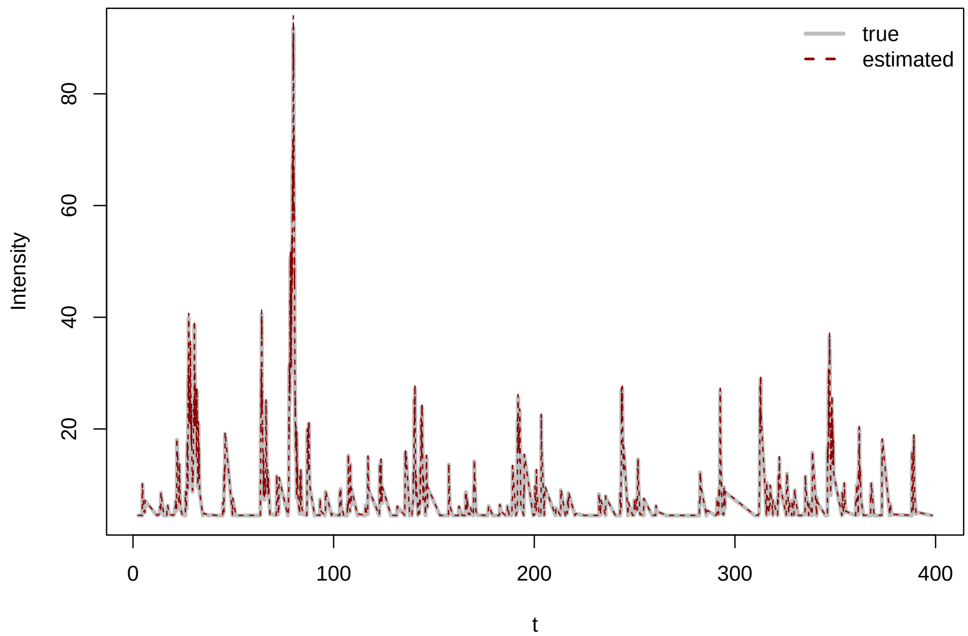
```
case1_solution2  <- nlm(loglik, c(0.1,1, 6), hessian = TRUE, arrivals = sample_f1)
paste( c("mu", "alpha", "beta" ), round(case1_solution2$estimate,2), sep=" = ")

## [1] "mu = 0.49"    "alpha = 4.05" "beta = 4.92"
```

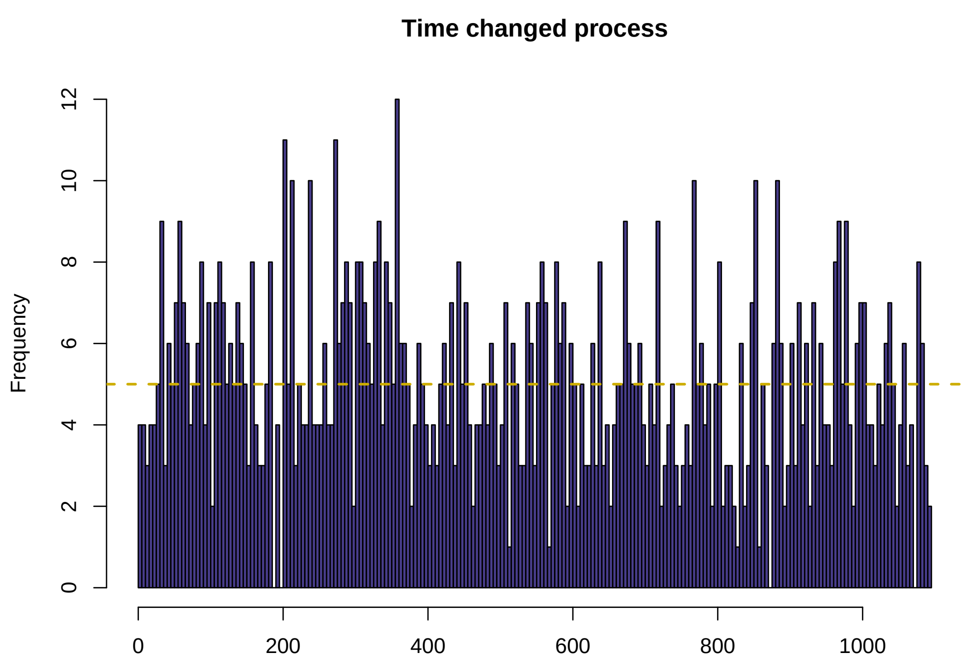
Estimated vs. Actual intensity function

```
estimated_intensity  <- function(params, arrivals){
  mu        <- params[1]
  alpha     <- params[2]
  beta      <- params[3]
  Ai        <- sapply(1:n, function(z) {
                 sum(exp( -beta * (arrivals[z]- arrivals[1:z])))
                })
  return(mu + alpha *Ai)
}
```

```
compensator       <- function(params, arrivals){
  mu     <- params[1]
  alpha  <- params[2]
  beta   <- params[3]
  result <- sapply(1:n, function(z){
    mu*arrivals[z] + sum(alpha/beta*(1-exp(-beta*(arrivals[z]-arrivals[1:z]))))
  })
  return(result)
}
```

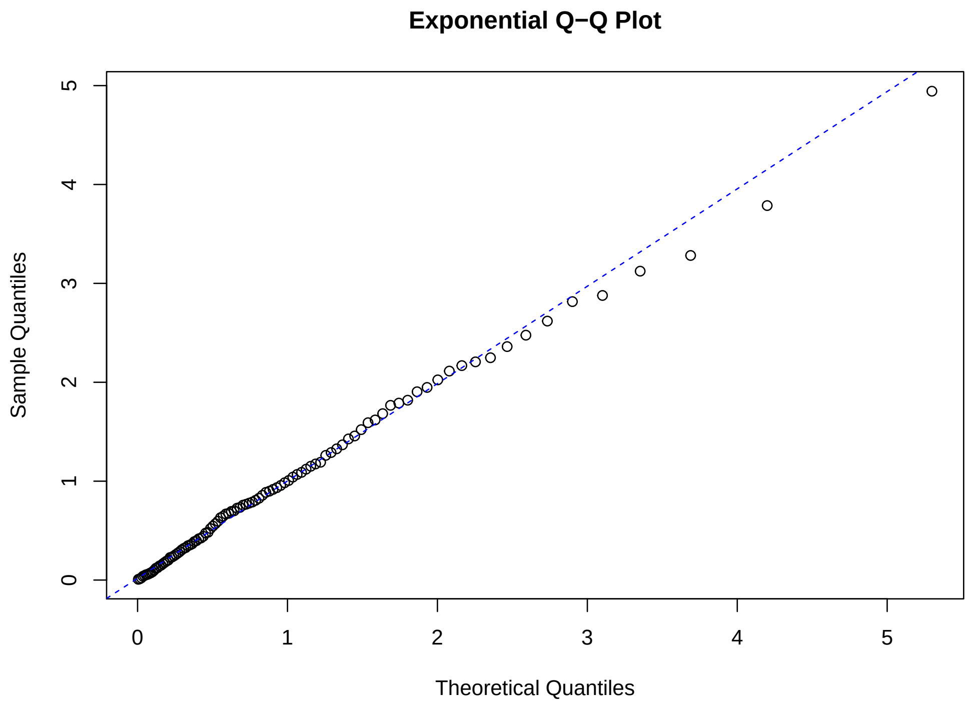


Another way to test the parameters is to apply time change to the original process and check whether the resulting process is a standard Poisson.



Splitting the time interval in to steps of 5 units should result in a frequency count of 5. The above histogram confirms that the parameters estimation via MLE has not gone wrong

QQ PLOT



Using hawkes package

```
negloglik_hawkes     <- function(params, history){
  mu      <- params[1]
  alpha   <- params[2]
  beta    <- params[3]
  return(likelihoodHawkes(mu, alpha, beta, history))
}
params_hawkes <- round(optim(c(0.1,2,6), negloglik_hawkes, history = sample_f1)$par,2)
paste( c("alpha", "beta","mu"), params_hawkes, sep=" = ")

## [1] "alpha = 0.49" "beta = 4.05"  "mu = 4.92"
```

Simulating and Estimating multivariate hawkes

```
set.seed(1)
lambda0   <- c(0.2,0.2)
alpha     <- matrix(c(0.5,0,0,0.5),byrow=TRUE,nrow=2)
beta      <- c(0.7,0.7)
history   <- simulateHawkes(lambda0, alpha, beta, 3600)
```

```r
nloglik_bi_hawkes  <- function(params, history){
  mu       <- c(params[1],params[2])
  alpha    <- matrix(c(params[3],params[4],params[5],params[6]),byrow=TRUE,nrow=2)
  beta     <- c(params[7], params[8])
  return(likelihoodHawkes(mu, alpha, beta, history))
}
params_hawkes    <- round(optim(c(rep(1,2), rep(0.2,4),rep(2,2)),
                              nloglik_bi_hawkes, history = history)$par, 2)
params           <- data.frame( estimates = params_hawkes)
rownames(params) <- c("$\\mu_1$", "$\\mu_2$", "$\\alpha_{11}$",
                      "$\\alpha_{12}$","$\\alpha_{21}$",
                      "$\\alpha_{22}$", "$\\beta_1$", "$\\beta_2$")
```

	estimates
μ_1	0.24
μ_2	0.34
α_{11}	0.74
α_{12}	-0.02
α_{21}	-0.07
α_{22}	1.09
β_1	1.11
β_2	1.75