# Markov-modulated (marked) Poisson processes

Jan-Ole Koslik

The functionalities of `LaMa` can also be called to fit so-called **Markov-modulated Poisson processes**. These are doubly stochastic Poisson point processes where the intensity is directed by an underlying continuous-time Markov chain. Such processes are useful for modeling arrival times, for example of calls in a call center, or patients in the hospital. The main difference compared to continuous-time HMMs is that we explicitely model the arrival times as random-variables, i.e. we assign probabilities to them. A homogeneous Poisson process is mainly characterized by the fact that the number of arrivals in a fixed time interval is Poisson distributed with a mean that is proporional to the length of the interval. The waiting times between arrivals are exponentially distributed. While the latter is not true for non-homogeneous Poisson processes in general, we can interpret a Markov modulated Poisson process as alternating between homogeneous Poisson processes, i.e. when the unobserved continuous-time Markov chain stays in a particular state for some interval, the associated Poisson rate in that interval is homogeneous and state-specific. To learn more about Poisson processes, see Dobrow (2016).

## Example 1: Markov-modulated Poisson processes

### Setting parameters

We choose to have a considerably higher rate and shorter stays of the underlying Markov chain in state 2, i.e. state 2 is **bursty**.

```
# state-dependent rates
lambda = c(2, 15)
# generator matrix of the underlying Markov chain
Q = matrix(c(-0.5,0.5,2,-2), nrow = 2, byrow = TRUE)
```

### Simulating an MMPP

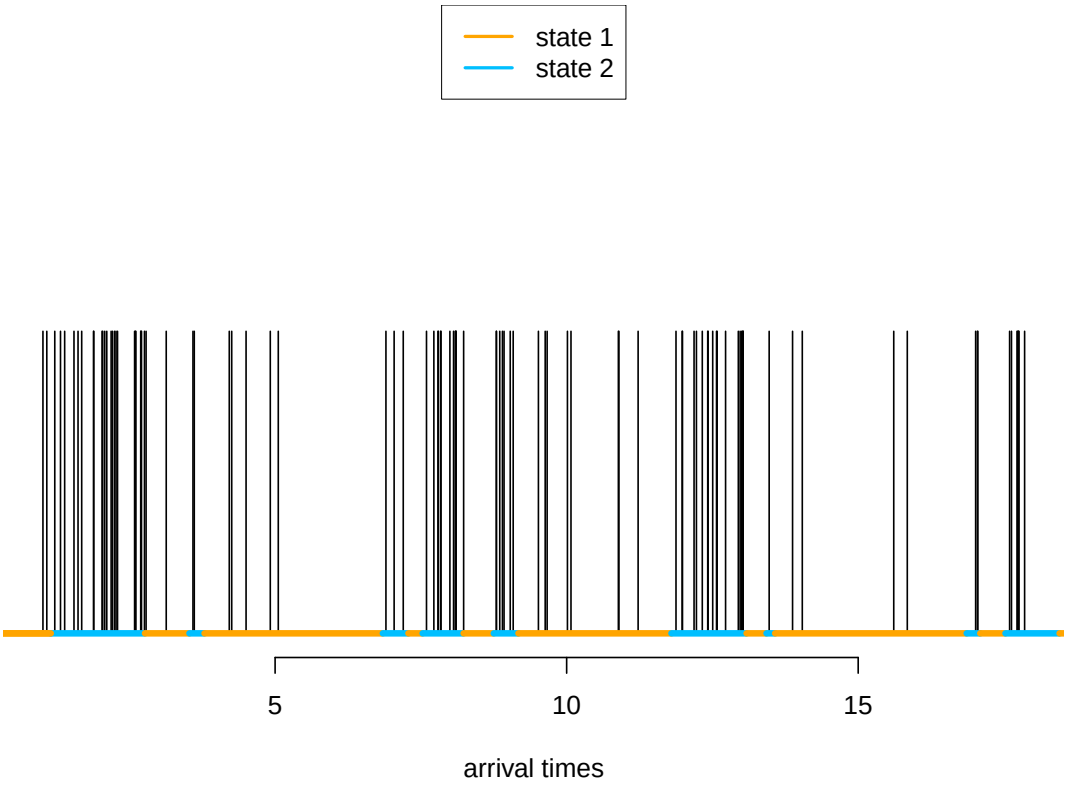
```
set.seed(123)

k = 200 # number of state switches
trans_times = s = rep(NA, k) # time points where the chain transitions
s[1] = sample(1:2, 1) # initial distribuion c(0.5, 0.5)
# exponentially distributed waiting times
trans_times[1] = rexp(1, -Q[s[1],s[1]])
# in a fixed interval, the number of arrivals is Pois(lambda * interval_length)
n_arrivals = rpois(1, lambda[s[1]]*trans_times[1])
# arrival times within fixed interval are uniformly distributed
arrival_times = runif(n_arrivals, 0, trans_times[1])
for(t in 2:k){
  s[t] = c(1,2)[-s[t-1]] # for 2-states, always a state swith when transitioning
```

```
  # exponentially distributed waiting times
  trans_times[t] = trans_times[t-1] + rexp(1, -Q[s[t], s[t]])
  # in a fixed interval, the number of arrivals is Pois(lambda * interval_length)
  n_arrivals = rpois(1, lambda[s[t]]*(trans_times[t]-trans_times[t-1]))
  # arrival times within fixed interval are uniformly distributed
  arrival_times = c(arrival_times,
                    runif(n_arrivals, trans_times[t-1], trans_times[t]))
}
arrival_times = sort(arrival_times)
```

Let's visualize the simulated MMPP

```
n = length(arrival_times)
color = c("orange", "deepskyblue")
plot(arrival_times[1:100], rep(0.5,100), type = "h", bty = "n", ylim = c(0,1),
     yaxt = "n", xlab = "arrival times", ylab = "")
segments(x0 = c(0,trans_times[1:98]), x1 = trans_times[1:99],
         y0 = rep(0,100), y1 = rep(0,100), col = color[s[1:99]], lwd = 4)
legend("top", lwd = 2, col = color, legend = c("state 1", "state 2"), box.lwd = 0)
```




What makes the MMPP special compared to a regular Poisson point process is its **burstiness** when the Markov chain is in the second state.

### Writing the negative log-likelihood function

The likelihood of a stationary MMPP for waiting times $x_1, \ldots, x_n$ is (Meier-Hellstern (1987), Langrock, Borchers, and Skaug (2013))

$$L(\theta) = \delta\Big(\prod_{i=1}^{n} \exp((Q-\Lambda)x_i)\Lambda\Big)1,$$

where $Q$ is the generator matrix of the continuous-time Markov chain, $\Lambda$ is a diagonal matrix of state-dependent Poisson intensities, $\delta$ is the stationary distribution of the continuous-time Markov chain, and 1 is a column vector of ones. For more details on continuous-time Markov chains, see the vignette *continuous-time HMMs* or also Dobrow (2016). We can easily calculate the log of the above expression using the standard implementation of the general forward algorithm `forward_g()` when choosing the first matrix of state-dependent densities to be the identity (i.e.) the first row of the `allprobs` matrix to be one and all other matrices of state-dependent density matrices to be $\Lambda$.

```
mllk = function(theta.star, timediff, N=2){
  lambda = exp(theta.star[1:N]) # state specific rates
  Q = diag(N) # generator matrix
  Q[!Q] = exp(theta.star[N+1:(N*(N-1))])
  diag(Q) = 0
  diag(Q) = -rowSums(Q)
  Qube = LaMa::tpm_cont(Q-diag(lambda), timediff) # exp((Q-Lambda)*deltat)
  allprobs = matrix(lambda, nrow = length(timediff+1), ncol = N, byrow = T)
  allprobs[1,] = 1
  delta = solve(t(Q+1), rep(1,N), tol = 1e-20)
  -LaMa::forward_g(delta, Qube, allprobs)
}
```

### Fitting an MMPP to the data

```
theta.star = log(c(2, 15, # lambda
                   2, 0.5)) # off-diagonals of Q

timediff = diff(arrival_times)

t1 = Sys.time()
mod = nlm(mllk, theta.star, timediff=timediff, stepmax = 10)
# we often need the stepmax, as the matrix exponential can be numerically unstable
Sys.time()-t1
#> Time difference of 0.07216215 secs
```

### Results

```
exp(mod$estimate)
#> [1]  1.9496891 15.0831215  1.8998849  0.4003955
```

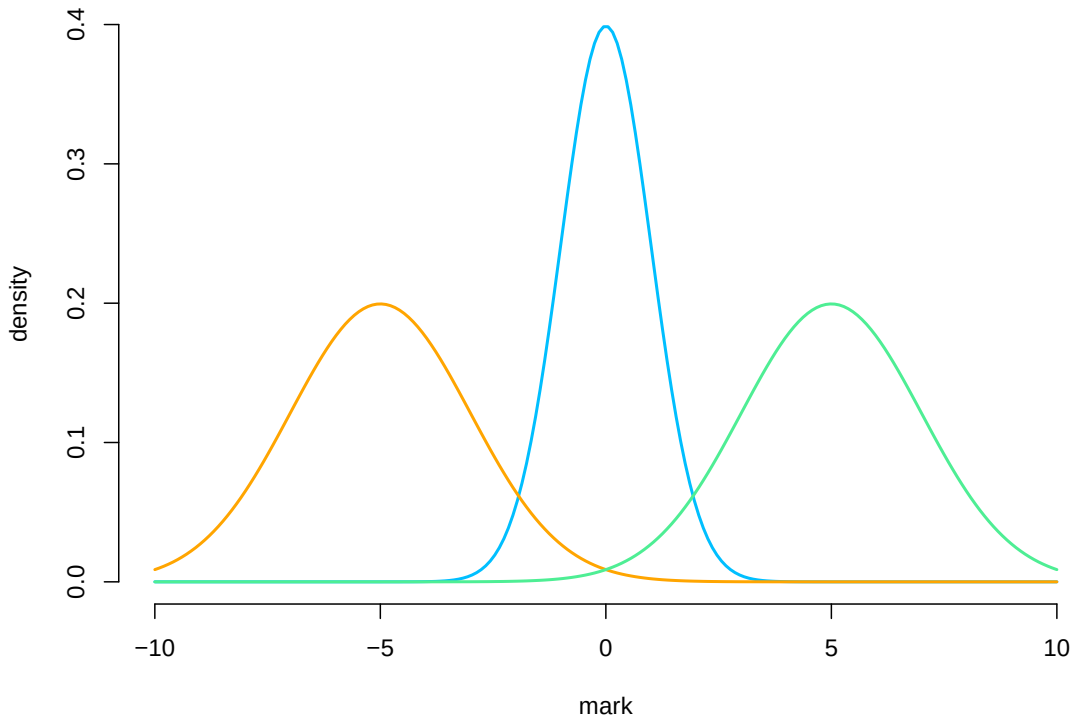
## Example 2: Markov-modulated marked Poisson processes

Such processes can also carry additional information, so called **marks**, at every arrival time when we also observe the realization of a different random variable that only depends on the underlying states of the

continuous-time Markov chain. For example for patient arrivals in the hospital we could observe a biomarker at every arrival time. Information on the underlying health status is then present in both, the arrival times (because sick patients visit more often) and the biomarkers.

```
# state-dependent rates
lambda = c(1, 5, 20)
# generator matrix of the underlying Markov chain
Q = matrix(c(-0.5,0.3,0.2,
             0.7, -1, 0.3,
             1 ,1,-2), nrow = 3, byrow = TRUE)
# parmeters for distributions of state-dependent marks
# (here normally distributed)
mu = c(-5, 0, 5)
sigma = c(2, 1, 2)

color = c("orange", "deepskyblue", "seagreen2")
curve(dnorm(x, 0, 1), xlim = c(-10,10), bty = "n", lwd = 2, col = color[2],
      n = 200, ylab = "density", xlab = "mark")
curve(dnorm(x, -5, 2), add = TRUE, lwd = 2, col = color[1], n = 200)
curve(dnorm(x, 5, 2), add = TRUE, lwd = 2, col = color[3], n = 200)
```




### Simulating an MMMPP

We now show how to simulate an MMMPP and additionally how to generalize to more than two hidden states.

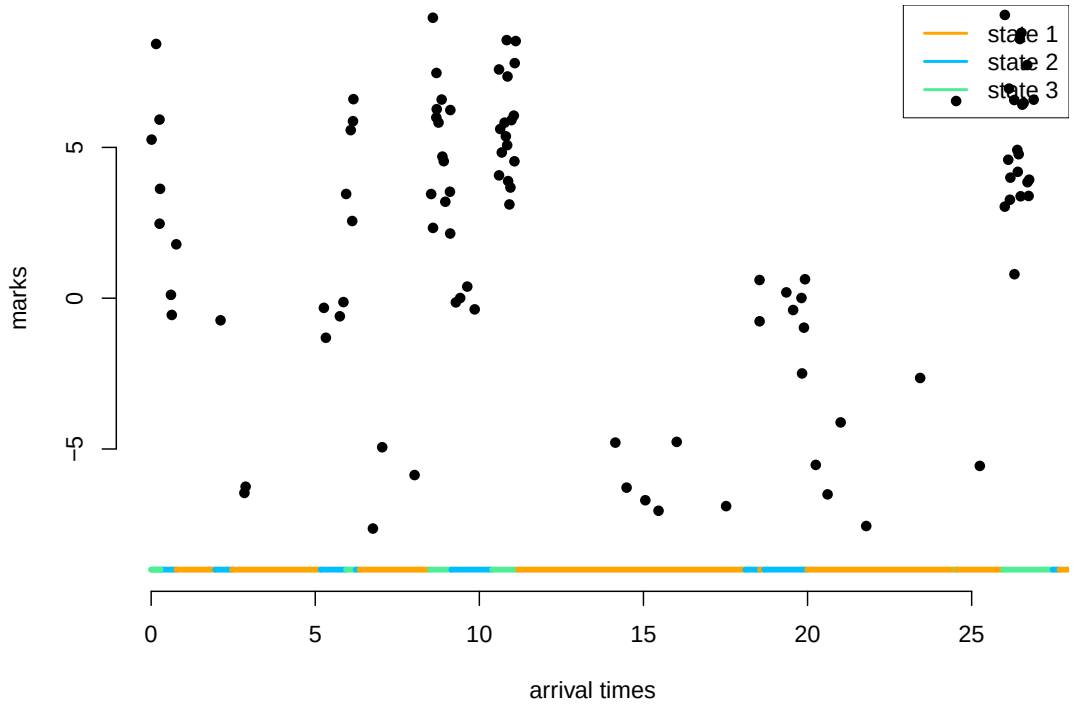
```
set.seed(123)
k = 200 # number of state switches
trans_times = s = rep(NA, k) # time points where the chain transitions
```

```
s[1] = sample(1:3, 1) # initial distribuion uniformly
# exponentially distributed waiting times
trans_times[1] = rexp(1, -Q[s[1],s[1]])
# in a fixed interval, the number of arrivals is Pois(lambda * interval_length)
n_arrivals = rpois(1, lambda[s[1]]*trans_times[1])
# arrival times within fixed interval are uniformly distributed
arrival_times = runif(n_arrivals, 0, trans_times[1])
# marks are iid in interval, given underlying state
marks = rnorm(n_arrivals, mu[s[1]], sigma[s[1]])

for(t in 2:k){
  # off-diagonal elements of the s[t-1] row of Q divided by the diagonal element
  # give the probabilites of the next state
  s[t] = sample(c(1:3)[-s[t-1]], 1, prob = Q[s[t-1],-s[t-1]]/-Q[s[t-1],s[t-1]])
  # exponentially distributed waiting times
  trans_times[t] = trans_times[t-1] + rexp(1, -Q[s[t],s[t]])
  # in a fixed interval, the number of arrivals is Pois(lambda * interval_length)
  n_arrivals = rpois(1, lambda[s[t]]*(trans_times[t]-trans_times[t-1]))
  # arrival times within fixed interval are uniformly distributed
  arrival_times = c(arrival_times,
                    runif(n_arrivals, trans_times[t-1], trans_times[t]))
  # marks are iid in interval, given underlying state
  marks = c(marks, rnorm(n_arrivals, mu[s[t]], sigma[s[t]]))
}
arrival_times = sort(arrival_times)
```

Let's visualize the simulated MMMPP

```
n = length(arrival_times)
plot(arrival_times[1:100], marks[1:100], pch = 16, bty = "n",
     ylim = c(-9,9), xlab = "arrival times", ylab = "marks")
segments(x0 = c(0,trans_times[1:98]), x1 = trans_times[1:99],
         y0 = rep(-9,100), y1 = rep(-9,100), col = color[s[1:99]], lwd = 4)
legend("topright", lwd = 2, col = color,
       legend = c("state 1", "state 2", "state 3"), box.lwd = 0)
```




**Writing the negative log-likelihood function**

The likelihood of a stationary MM**M**PP for waiting times $x_1, \ldots, x_n$ between marks $y_0, y_1, \ldots, y_n$ only changes slightly from the MMPP likelihood, as we include the matrix of state-specific densities (Lu (2012), Mews et al. (2023)):

$$L(\theta) = \delta P(y_0)\Big(\prod_{i=1}^{n} \exp((Q-\Lambda)x_i)\Lambda P(y_i)\Big)1,$$

where $Q$, $\Lambda$ and $\delta$ are as above and $P(y_i)$ is a diagonal matrix with state-dependent densites for the observation at time $t_i$. We can again easily calculate the log of the above expression using the standard implementation of the general forward algorithm `forward_g()` when first calculating the `allprobs` matrix with state-dependent densities for the marks (as usual for HMMs) and then multiplying each row except the first one element-wise with the state-dependent rates.

```
mllk = function(theta.star, y, timediff, N){
  lambda = exp(theta.star[1:N]) # state specific rates
  mu = theta.star[N+1:N]
  sigma = exp(theta.star[2*N+1:N])
  Q = diag(N) # generator matrix
  Q[!Q] = exp(theta.star[3*N+1:(N*(N-1))])
  diag(Q) = 0
  diag(Q) = -rowSums(Q)
  delta = solve(t(Q+1), rep(1,N), tol = 1e-20)
  Qube = LaMa::tpm_cont(Q-diag(lambda), timediff) # exp((Q-Lambda)*deltat)
  allprobs = matrix(1, length(y), N)
  for(j in 1:N){
    allprobs[,j] = dnorm(y, mu[j], sigma[j])
  }
  allprobs[-1,] = allprobs[-1,] * matrix(lambda, length(y)-1, N, byrow = T)
```

```
  -LaMa::forward_g(delta, Qube, allprobs)
}
```

### Fitting an MMPP to the data

```
theta.star = c(log(c(1, 5, 20)), # lambda
                    -5, 0, 5, # mu
               log(c(2, 1, 2)), # sigma
               log(c(0.7, 1, 0.3, 1, 0.2, 0.3))) # Q
timediff = diff(arrival_times)
t1 = Sys.time()
mod2 = nlm(mllk, theta.star, y = marks, timediff=timediff, N=3, stepmax = 5)
Sys.time()-t1
#> Time difference of 0.5065761 secs
```

### Results

```
N = 3
round(exp(mod2$estimate[1:N]),2)
#> [1]  0.96  4.86 19.50
# mu
round(mod2$estimate[N+1:N], 2)
#> [1] -5.19 -0.09  4.81
# sigma
round(exp(mod2$estimate[2*N+1:N]), 2)
#> [1] 1.79 0.96 2.01
Q = diag(N)
Q[!Q] = exp(mod2$estimate[3*N+1:(N*(N-1))])
diag(Q) = 0
diag(Q) = -rowSums(Q)
round(Q, 3)
#>        [,1]   [,2]   [,3]
#> [1,] -0.591  0.279  0.312
#> [2,]  0.926 -1.180  0.254
#> [3,]  1.193  1.210 -2.403
```

## References


Dobrow, Robert P. 2016. *Introduction to Stochastic Processes with r*. John Wiley & Sons.

Langrock, Roland, David L Borchers, and Hans J Skaug. 2013. "Markov-Modulated Nonhomogeneous Poisson Processes for Modeling Detections in Surveys of Marine Mammal Abundance." *Journal of the American Statistical Association* 108 (503): 840–51.

Lu, Shaochuan. 2012. "Markov Modulated Poisson Process Associated with State-Dependent Marks and Its Applications to the Deep Earthquakes." *Annals of the Institute of Statistical Mathematics* 64: 87–106.

Meier-Hellstern, Kathleen S. 1987. "A Fitting Algorithm for Markov-Modulated Poisson Processes Having Two Arrival Rates." *European Journal of Operational Research* 29 (3): 370–77.

Mews, Sina, Bastian Surmann, Lena Hasemann, and Svenja Elkenkamp. 2023. "Markov-Modulated Marked Poisson Processes for Modeling Disease Dynamics Based on Medical Claims Data." *Statistics in Medicine*.