Discrete Event Simulation

Ignacio Cascos

2019

Outline

3.1 **Poisson processes (R5.4, R5.5)**

3.2 **Gaussian processes**

3.3 **Discrete event simulation: queues, inventories, and collective risk (R7)**

Poisson processes

A **counting process** is a stochastic process $\{N(t) : t \geq 0\}$ that is continuous in the space of time and discrete in the space of states and such that

- $N(t) \geq 0$.
- $N(t)$ is an integer.
- If $t_2 \geq t_1$, then $N(t_2) \geq N(t_1)$.

If $t_2 > t_1$, the increment $N(t_2) - N(t_1)$ is the number of events during the interval $[t_1, t_2]$.

We can characterise the counting process either by means of the (increasing) sequence of rvs $T_1, T_2, \ldots$ where T_i is the time at which the i-th event occurs, or by the sequence of times between two consequtive events, $W_i = T_i - T_{i-1}$.

Homogeneous Poisson process

If the increments are independent and rv $N(t)$ follows a Poisson distribution, then $\{N(t) : t \geq 0\}$ is a **Poisson process**.

A counting propcess follows is a (homogeneous) **Poisson process** with rate (intensity) $\lambda > 0$ if

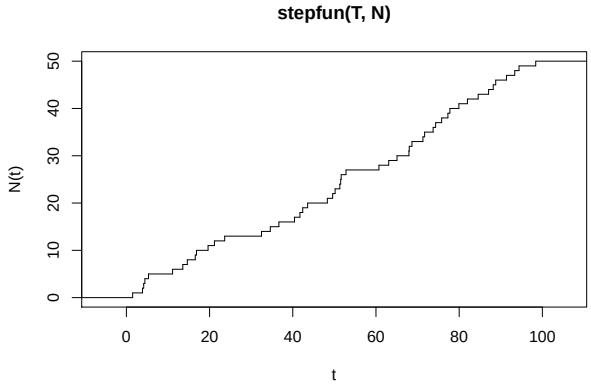
- $N(0) = 0$.
- $N(t_2) - N(t_1)$ is independent of $N(s_2) - N(s_1)$ for every $0 \leq t_1 < t_2 \leq s_1 < s_2$ (independent increments).
- $N(t_2) - N(t_1) \sim \mathcal{P}(\lambda(t_2 - t_1))$ (stationary increments with a Poisson distribution).

We have $W_i \sim \text{Exp}(\lambda)$ for every i and independent, and $T_i = \sum_{r=1}^{i} W_r \sim \text{Gamma}(i, \lambda)$.

Simulation of homogeneous Pois. process until k-th event

1. Set $T_0 = 0$.
2. Generate $W_1, W_2, \ldots, W_k$ Exponential rvs with parameter λ.
3. For $1 \leq i \leq k$, set $T_i = T_{i-1} + W_i$ and stop.

```
k <- 50 ; lmbd <- 0.5 ; set.seed(1)
T <- cumsum(rexp(k,rate=lmbd))
N <- 0:k
plot(stepfun(T,N),do.points=F,xlab="t",ylab="N(t)")
```



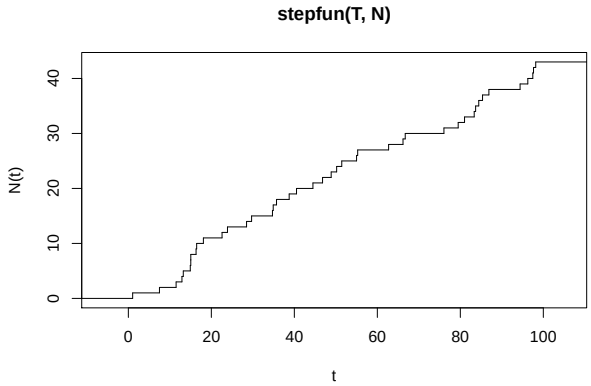
Simulation of homogeneous Poisson process in $[0, t_l]$

Conditional on the total number of events in $[t_1, t_2]$, the (unordered) event times are distributioned as uniform rvs in the interval (t_1, t_2).

1. Generate $k = N(t_l)$ a Poisson rv with rate λt_l.
2. Generate $U_1, U_2, \ldots, U_k$ random numbers in $(0, 1)$.
3. Sort the previous sample as $U_{(1)} < U_{(2)} < \cdots < U_{(k)}$.
4. For $1 \leq i \leq k$, set $T_i = t_l U_{(i)}$ and stop.

Simulation of homogeneous Poisson process in $[0, t_l]$

```
tl <- 100 ; lmbd <- 0.5 ; set.seed(2)
k <- rpois(1,lambda=lmbd*tl)
u <- runif(k) ; T <- tl*sort(u) ; N <- 0:k
plot(stepfun(T,N),do.points=F,xlab="t",ylab="N(t)")
```



Nonhomogeneous Poisson process

A counting process is a **nonhomogeneours Poisson process** with intensity function $\lambda : \mathbb{R}_+ \mapsto \mathbb{R}_+$ if

- $N(0) = 0$.
- $N(t_2) - N(t_1)$ is independent of $N(s_2) - N(s_1)$ for every $0 \leq t_1 < t_2 \leq s_1 < s_2$ (independent increments).
- $N(t_2) - N(t_1) \sim \mathcal{P}\left(\int_{t_1}^{t_2} \lambda(s)\,\mathrm{d}s\right)$ (nonstationary increments with a Poisson distribution).

We denote $m(t) = \int_0^t \lambda(s)\,\mathrm{d}s$ and $\lambda_m \geq \lambda(s)$ for every $0 \leq s \leq t_l$.

Simulation of nonhomogeneous Poisson process until k-th event

The cdf of rv time until next event from time t is

$$F_t(r) = P(W_i \leq r | T_{i-1} = t) = 1 - P(W_i > r | T_{i-1} = t)$$
$$= 1 - P(N(t+r) - N(t) = 0) = 1 - \exp\left(-\int_t^{t+r} \lambda(s)\,\mathrm{d}s\right)$$

We can use the inverse transform technique to simulate the first k events of a nonhomogenoeus Poisson process as:

1. Set $T_0 = 0$.
2. Generate $U_1, U_2, \ldots, U_k$ random numbers in $(0, 1)$.
3. For $1 \leq i \leq k$, set $T_i = T_{i-1} + F_{T_{i-1}}^{-1}(U_i)$ and stop.

Simulation of nonhomogeneous Poisson process until k-th event (thinning)

The previous algorithm requires an explicity expression for F_t^{-1} (for every t). An alternative approach is the thinning technique that consists on rejecting some of the simulated events.

1. Set $T_0 = 0$, $T^* = 0$, $i = 0$.
2. Generate X Exponential rv with rate λ_m.
3. Set $T^* = T^* + X$.
4. Generate U random number in $(0, 1)$.
5. If $U > \lambda(T^*)/\lambda_m$, set $i = i + 1$ and $T_i = T^*$
6. If $i = k$ Stop.
7. Go to Step 2.

Simulation of nonhomogeneous Poisson process in $[0, t_l]$

When simulating a nonhomogenous Poisson process on a fixed time interval, we build a dmf out of the intensity function by scaling it.

1. Generate $k = N(t_l)$ a Poisson rv with rate $m(t_l) = \int_0^{t_l} \lambda(s)\,\mathrm{d}s$
2. Generate $V_1, V_2, \ldots, V_k$ independent random numbers with density $\lambda/m(t_l)$.
3. Sort the previous sample as $V_{(1)} < V_{(2)} < \cdots < V_k$.
4. For every $1 \leq i \leq k$, set $T_i = V_{(i)}$ and stop.

More about Poisson and counting processes

- **Superposition principle** of Poisson processes. If $\{N_1(t)\}$ and $\{N_2(t)\}$ are two independent Poisson processes with respective intensity functions λ_1 and λ_2, then $\{N_1(t) + N_2(t)\}$ is a Poisson process with intensity function $\lambda_1 + \lambda_2$.
  - **Simulation**: Generate the path of a Poisson process with intensity function $\lambda_1 + \lambda_2$. If an event occurs at time t, it corresponds to the Poisson process $\{N_1(t)\}$ with probability $\lambda_1(t)/(\lambda_1(t)+\lambda_2(t))$.
- **Compound Poisson process**. We can associate an independent rv X_i to each event occurring in a Poisson process $N(t)$ and set the compount Poisson process $S(t) = \sum_{k=0}^{N(t)} X_k$.
  - **Simulation**: Generate the path of the Poisson process $N(t)$, at each event, generate a rv with the distribution of X and update $S(t)$.

More about Poisson and counting processes

- **Mixed Poisson process**. The intensity (rate) is not a function of time, but a structural rv instead, so $N(t) \sim \mathcal{P}(\Lambda t)$ with Λ rv.
  - **Simulation**: Generate an observation of Λ and simulate the homogeneous Poisson process.
- **Doubly stochastic Cox process**. The intensity function is a stochastic process itself.
  - **Simulation**: Generate a path of the stochastic process $\Lambda(t)$ and simulate the nonhomogeneous Poisson process.
- **Renewall process**. The times between events are not exponentially distributed.
  - **Simulation**: As the simulation of the homogeneous Poisson process until k-th event, but not the time between two consecutive events is not exponentially distributed.

3.2 Gaussian processes

A stochastic process $X(t)$ is **Gaussian** if all its marginal distributions are normal.

- Every random vector $(X(t_1), X(t_2), \ldots, X(t_n))^t$ is normaly distributed.
- Completely determined by *mean function* $\mu_X(t) = \mathbb{E}[X(t)]$ and *autocovariance* $C_X(t_1, t_2) = \mathrm{Cov}(X(t_1), X(t_2))$.

Brownian motion

A Gaussian process $B(t)$ is a **brownian motion** (Wiener process) if

- $B(0) = 0$.
- Given $0 \leq t_1 < t_2 < \ldots < t_n$, the increments $(B(t_2) - B(t_1)), (B(t_3) - B(t_2)), \ldots, (B(t_n) - B(t_{n-1}))$ are *indepenent*.
- If $s > 0$, $B(t + s) - B(t) \sim \mathrm{N}(0, \sqrt{s})$
- Has continuous paths, that is, $B(t)$ is a continuous function of t.

A brownian motion with *drift* μ and *volatility* σ is given by

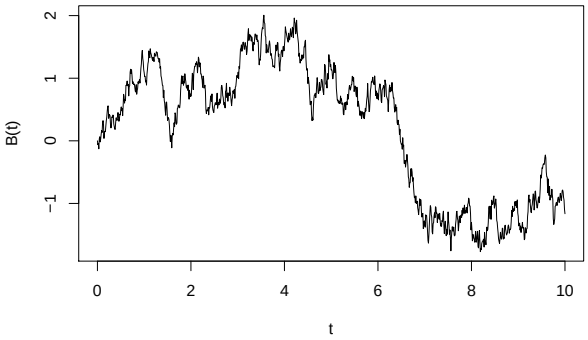
$$X(t) = \mu t + \sigma B(t),$$

so $X(t) \sim \mathrm{N}(\mu t, \sigma\sqrt{t})$.

Simulating a Brownian motion

1. Set $B_0 = 0$.
2. Generate n normals $X_i \sim \mathrm{N}(0, \sqrt{t_i - t_{i-1}})$.
3. Set $B_{t_i} = B_{t_{i-1}} + X_i$.

```
n <- 100 ; tl <- 10 ; t <- seq(0,tl,by=1/n) ; set.seed(1)
B <- append(0,cumsum(rnorm(n*tl,mean=0,sd=1/sqrt(n))))
plot(t,B,type="l",ylab="B(t)")
```



Geometric Brownian motion

The **geometric** (or exponential) **brownian motion** is a continuous-time stochastic process that is used to model stock prices.

$$S(t) = S(0) \exp\left((\mu - \sigma^2/2)t + \sigma B(t)\right)$$

where $S(0)$ is the stock price at time 0. The univariate marginals of a geometric brownian motion follow a log-normal distribution.

If $X \sim \mathrm{N}(\mu, \sigma)$, then $e^X \sim$ `lnorm(meanlog` $= \mu$`,sdlog` $= \sigma$`)`,

$$\mathbb{E}[e^X] = \exp\{\mu + \sigma^2/2\} \quad , \quad \mathrm{Var}[e^X] = (\exp\{\sigma^2\} - 1)\exp\{2\mu + \sigma^2\}.$$

3.3 Discrete event simulation

Discrete event simulation models a system as a discrete sequence of events in time.

We must simulate the *times* at wich the events ocurr and some random amounts associated with the *events* themselves.

The relevant **variables** are:

- **Time variable**, t, elapsed amount of (simulated) time.
- **Counter variables**, number of times certain events have occurred by time t.
- **Systems statate variable**, SS, *state of the system* at time t.

Whenever an **event** occurs the variables are updated.

Single-server queuing system

Customers arrive to a service station in accordance with a nonhomogeneous Poisson process with intensity function λ.

Upon arrival each customer either enters service is server is free or joins the waiting queue if server is busy.

When the server completes serving a customer, either begins serving new customer (the one that has been waiting for longer if FIFO queue) if there a customers in the queue or remains free until next customer arrives if there are no waiting customers.

The amount of time used in the service of a customer is a random variable (independent of the other service times) with some known probability distribution.

There is a final time t_l after which no additional arrivals are allowed in the system.

Single-server queuing system

Relevant variables

- **Time variable**, t.
- **Counter variables**, N_A and N_D number of arrivals and departures by time t.
- **Systems statate variable**, n, number of customers in the system by time t.

Events: arrivals and departures, occurring at times t_A and t_D.

The *output variables* to consider here are: A_i arrival time of i-th customer, D_i departure time of i-th customer, and T_p the time past the final time t_l that the last customer departs.

Inventory model

A shop stocks a particular type of good that sells at price r per unit.

Customers demanding the good appear in accordance with a Poisson process with rate λ, and the amount of good demanded by each of them is a rv with distribution G.

The shopkeeper keeps an amount of stock S on hand, and whenever the stock becomes low orders new units from the distributor. Using a (s_1, s_2) ordering policy, where $s_1 < s_2$, if the stock S goes below s_1, the shopkeeper orders $s_2 - S$ units.

The cost of ordering y units $c(y)$ is specified, it takes L units of time until the order is delivered. The payment is made upon delivery and the shop pays an inventory holding cost of h per unit of item in stock.

If a customer demands more of the product than is presently available, the amound on hand is sold, and the customer buys the rest somewhere else (the shopkeeper misses the chance to sell that amount of product).

Inventory model

Relevant variables

- **Time variable**, t.
- **Counter variables**, C, H, R total amount of ordering costs, of inventory holding costs, and revenue earned by time t.
- **Systems statate variable**, x amount of inventory on hand, and y amount on order by time t.

Events: arrivals of either customers (next customer arrives at time t_c) or orders (next order arrives at time t_o). The updating is performed at the smallest value between t_c and t_o.

The *output variable* to consider here is the shop's revenue.

Collective risk model

The policyholders of an insurance company generate claims according to independent Poisson processes with common rate λ ($N_C(t)$). These claims are independent and their amount follows distribution $X \sim F$. Clients enrol the company according to a Poisson process with rate ν ($N_A(t)$). Clients stay in the company for an exponentially distributed time with rate μ ($N_D(t)$ for the process of departures).

Each policyholder pays a fixed amoung c per unit of time in the company, the company starts with n_0 clients, and initial capital $a_0 \geq 0$.

The company's capital at time t is given by

$$A(t) = a_0 + ct(n_0 + N_A(t) - N_D(t)) - \sum_{i=1}^{N_C(t)} X_i,$$

and $P(\inf\{A(t) : 0 \leq t \leq t_l\} < 0)$ is the *ruin probability* over period $[0, t_l]$.

Collective risk model

Relevant variables

- **Time variable**, t.
- **System state variables**, n number of policy holders and a firm's capital by time t.

Events: new policyholder, lost policyholder, and new claim.

At time t with n policy holders, next event with occur in an exponential time with rate $\nu + n\lambda + n\mu$ and it will be a new client (with probability $\nu/(\nu + n\lambda + n\mu)$), a claim, or a lost client with each of the corresponding probabilities. The (system state variables), firm's capital and number of policy holders is to be updated.

The *output variable* to consider here is and indicator variable that the firm's capital is nonnegative throughout $[0, t_l]$