Statistical Computing with R – MATH 6382[1,*]
Set 4 (Simulation – Stochastic Processes)

Tamer Oraby
UTRGV
tamer.oraby@utrgv.edu

[1] Based on textbook.

[*] Last updated December 2, 2016

Stochastic Processes

Stochastic Processes

- Fix a probability space $(\Omega, \mathcal{B}, P)$
- A stochastic process is a collection of random variables

$$\{X_t(\omega) \in \mathcal{R} : t \in T, \omega \in \Omega\}$$

- T is time set, $\mathcal{R}$ is the state space
- For each $t \in T$, X_t is a single random variable
- For each $\omega \in \Omega$, $X_\bullet(\omega) : T \to \mathbb{R}$ is called the sample path
- Let $0 \in T$, then X_0 is the initial state of the process
- There are four types of stochastic processes depending on whether T and $\mathcal{R}$ are countable or uncountable
- Let $\mathcal{R}$ be $\mathbb{N}$ from now on. (Sometimes it is $\mathbb{N} \cup \{0\}$ or even $\mathbb{Z}$ but it will be clear.)

Markov Chains and Processes

- A Markov Chain X_t (if discrete time) or Markov Process $X(t)$ (if continuous time)
- A Markov Chain X_t (for simplicity $\mathcal{R} = \mathbb{N}$) satisfies the *Markov Property*:

$$P(X_n = i_n | X_0 = i_0, ..., X_{n-2} = i_{n-2}, X_{n-1} = i_{n-1}) = P(X_n = i_n | X_{n-1} = i_{n-1})$$

  for all $i_j \in \mathcal{R}$ and $j = 0, 1, \ldots, n$
- The one-step transition probability $P_{i,j} = P(X_n = j | X_{n-1} = i)$ is independent of n (time homogeneous MC)
- The transition matrix is

$$\mathbf{P} = \begin{pmatrix} P_{11} & P_{12} & \cdots \\ P_{21} & P_{22} & \cdots \\ \vdots & \vdots & \ddots \end{pmatrix}$$

Markov Chains and Processes

- $P_i(n) = P(X_n = i)$ is the probability that the Markov Chain is at state i at time n
- The n-step transition probability is

$$P_{i,j}^{(n)} = P(X_n = j | X_0 = i)$$

- The Chapman-Kolmogorov equation

$$P_{i,j}^{(n)} = \sum_{k=1}^{\infty} P_{i,k}^{(s)} P_{k,j}^{(n-s)}$$

for any $0 \leq s \leq n$

- The n-step transition matrix is the n^{th} power of the one-step transition matrix $\mathbf{P}^{(n)} = \mathbf{P}^n$

Markov Chains

- $P_i(n+1) = \sum_{k=1}^{\infty} P_k(n)\, P_{k,i}$
- If $\mathbf{P}(n) = (P_1(n), P_2(n), \ldots)$, then

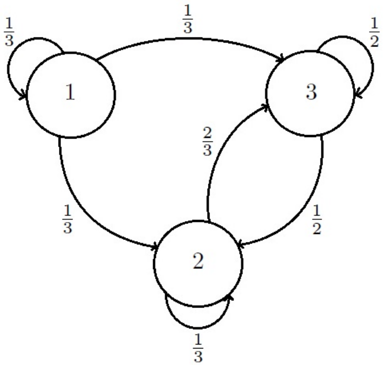
$$\mathbf{P}(n+1) = \mathbf{P}(n) \cdot \mathbf{P}$$

- Which implies $\mathbf{P}(n) = \mathbf{P}(0) \cdot \mathbf{P}^n$

Markov Chains

Example: Suppose a machine is in one of three states: State 1 is idle, State 2 is working, State 3 is broken down. The following digraph shows the transition probabilities and the states.

$$\mathbf{P} = \begin{bmatrix} P_{11} & P_{12} & P_{13} \\ P_{21} & P_{22} & P_{23} \\ P_{31} & P_{32} & P_{33} \end{bmatrix} = \begin{bmatrix} \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ 0 & \frac{1}{3} & \frac{2}{3} \\ 0 & \frac{1}{2} & \frac{1}{2} \end{bmatrix}$$



Markov Chains

Example:

$$\mathbf{P}^n = \begin{bmatrix} \frac{1}{3^n} & \frac{3}{7} - \frac{1}{3^{n+1}} - \frac{2}{21(-6)^n} & \frac{4}{7} - \frac{2}{3^{n+1}} + \frac{2}{21(-6)^n} \\ 0 & \frac{3}{7} + \frac{4}{7(-6)^n} & \frac{4}{7} - \frac{4}{7(-6)^n} \\ 0 & \frac{3}{7} - \frac{3}{7(-6)^n} & \frac{4}{7} + \frac{3}{7(-6)^n} \end{bmatrix}$$

and

$$\lim_{n\to\infty} \mathbf{P}^n = \begin{bmatrix} 0 & \frac{3}{7} & \frac{4}{7} \\ 0 & \frac{3}{7} & \frac{4}{7} \\ 0 & \frac{3}{7} & \frac{4}{7} \end{bmatrix}$$

Thus,

$$\lim_{n\to\infty} P(n) = (0, \frac{3}{7}, \frac{4}{7})$$

Markov Chains

- A stationary distribution $\pi = (\pi_1, \pi_2, \ldots)$ of a Markov Chain is a probability distribution such that $\pi = \pi\mathbf{P}$ and so if $P(0) = \pi$ then $P(n) = \pi$ for all $n \geq 1$.

- If the Markov Chain is strongly ergodic (irreducible + aperiodic + positive recurrent) then there exists a unique stationary distribution π such that for any i

$$\lim_{n\to\infty} P_{i,j}^{(n)} = \pi_j$$

- In the previous example

$$\pi = (0, \frac{3}{7}, \frac{4}{7})$$

Markov Chains

Theorem

Let $\mathbf{P}$ *be a transition matrix such that* $\nu = (\nu_1, \nu_2, \ldots)$ *is a probability distribution that satisfies the detailed balance equation*

$$\nu_i P_{i,j} = \nu_j P_{j,i} \text{ reversibility}$$

for all i *and* j *then* ν *is a stationary distribution.*

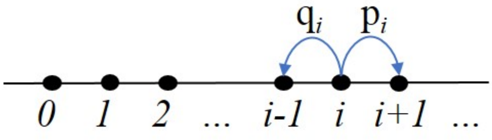
Proof.

Fix $j \geq 1$.

$$(\nu \mathbf{P})_j = \sum_{i=1}^{\infty} \nu_i P_{i,j} = \sum_{i=1}^{\infty} \nu_j P_{j,i} = \nu_j \sum_{i=1}^{\infty} P_{j,i} = \nu_j$$

Thus, $\nu \mathbf{P} = \nu$. □

Simple Random Walk on the Line

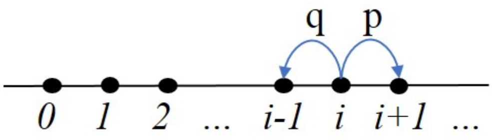


$$P_{i,j} = \begin{cases} p_i & \text{if } j = i+1, \\ q_i & \text{if } j = i-1, \\ 0 & \text{if } j \neq i \pm 1. \end{cases}$$

- It is site-dependent transitions with $p_i + q_i = 1$
- If $p_i = \frac{1}{2}$ then it is symmetric random walk
- The line may be unbounded and if bounded then there must be boundary conditions like reflections or absorption.

Simple Random Walk on the Line

A Random Walk can be represented as a partial sum $S_n = \sum_{k=1}^{n} X_k$



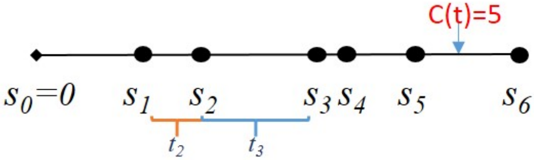
$$X_k = \begin{cases} 1 & \text{with probability } p, \\ -1 & \text{with probability } q, \end{cases}$$

Let T_{ii} be be the first return time to state i (starting from i).
Then the pmf of T_{00} is

$$P(T_{00} = 2n) = C_n^{2n} p^n q^n$$

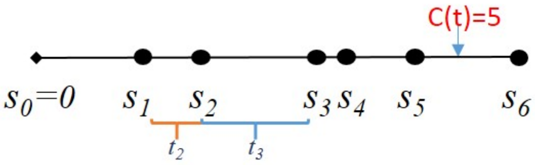
with $n = 1, 2, \ldots$ and zero otherwise.

Counting Process



A counting process is a continuous-time Markov chain (CTMC) in which $C(t)$ or $C([0, t])$ is the number of events – like birth, death, infections, or generically arrivals – occurring by time t. The time between arrivals $t_i = s_i - s_{i-1}$ is called the inter-arrival time.

Counting Process



- The counting process has independent increments if $C(s_{j+1}) - C(s_j)$ and $C(s_{i+1}) - C(s_i)$ are independent for all i and j such that $i \neq j$, or generally $C((t, s])$ and $C((u, v])$ are independent whenever $(t, s]$ and $(u, v]$ are disjoint.
- The counting process has stationary increments if $C((s, s + h])$ depends only on h.

Homogeneous Poisson Process

A *homogeneous Poisson Process* $N(t)$ with rate (intensity) $\lambda > 0$

$$P(N(t+s) - N(s) = k) = \frac{(\lambda t)^k}{k!} e^{-\lambda t}$$

for $k = 0, 1, \ldots$ and $N(0) = 0$.
The mean value function $m(t) = \mathbf{E}(N(t)) = \lambda t$
Then, if T_1 is the time of the first arrival

$$P(T_1 > t) = P(N(t) = 0) = e^{-\lambda t}$$

which means $T_1 \sim exp(\lambda)$
Also, the inter-arrival times $T_1, T_2, \ldots$ are i.i.d. $\sim exp(\lambda)$

Homogeneous Poisson Process

Let $S_n = T_1 + T_2 + \cdots + T_n$, then

$$[N(t) \geq n] \Leftrightarrow [S_n \leq t]$$

since

$$\begin{aligned}
P(t < S_n \leq t + \Delta t) &= P(S_n \leq t + \Delta t) - P(S_n \leq t) \\
&= P(N(t + \Delta t) \geq n) - P(N(t) \geq n) \\
&= \sum_{k=n}^{\infty} \left[\frac{(\lambda(t + \Delta t))^k}{k!} e^{-\lambda(t+\Delta t)} - \frac{(\lambda t)^k}{k!} e^{-\lambda t} \right] \\
&= (\lambda(\Delta t)) \frac{(\lambda t)^{n-1}}{n-1!} e^{-\lambda t} + o(\Delta t)
\end{aligned}$$

and the latter which means $T_i \sim exp(\lambda)$ by Memoryless property.

Non-homogeneous Poisson Process

A *non-homogeneous Poisson Process* $N(t)$ with intensity $\lambda(t) > 0$

$$P(N(t+s) - N(s) = k) = \frac{(\int_s^{t+s} \lambda(u)du)^k}{k!} e^{-\int_s^{t+s} \lambda(u)du}$$

for $k = 0, 1, \ldots,$ $N(0) = 0$ and independent but not stationary increments.
The mean value function $m(t) = \mathbf{E}(N(t)) = \int_0^t \lambda(u)du$
Homogeneous Poisson Process is the special case when $\lambda(t) = \lambda$ (a constant).

Renewal Process

A *Renewal Process* $R(t)$ is a counting process in which the inter-arrival times are i.i.d but not necessarily exponentially distributed. They could be lognormal or Weibull, for examples.
The mean value function $m(t) = \mathbf{E}(R(t))$ determines the inter-arrival times.

Birth and Death Process

A *Birth and Death Process* $X(t)$ is a continuous-time Markov chain (CTMC) and is a counting process in which

$$P_{ij}(\Delta t) = \begin{cases} \lambda i \Delta t + o(\Delta t) & \text{if } j = i+1, \\ \mu i \Delta t + o(\Delta t) & \text{if } j = i-1, \\ 1 - (\lambda + \mu) i \Delta t + o(\Delta t) & \text{if } j = i, \\ o(\Delta t) & \text{if } j \neq i-1, i, i+1, \end{cases}$$

where λ is the per-capita birth rate and μ is the per-capita death rate.

Brownian motion or Wiener Process

A Wiener process $W(t)$ on $[0, \infty)$ is a continuous time - continuous state process with independent stationary increments such that

1. $W(0) = 0$
2. (Independent increments) $W(s_2) - W(s_1)$ and $W(t_2) - W(t_1)$ are independent for all $0 < s_1 < s_2 < t_1 < t_2$
3. (Stationary normal increments) $W(t) - W(s) \sim N(0, t - s)$ for all $0 < s < t$

Geometric Brownian motion

A Geometric Brownian motion $S(t)$ with drift μ and volatility σ on $[0, \infty)$ is a continuous time - continuous state process with

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

where $W(t)$ is Brownian motion

Brownian bridge

A Brownian bridge $W_{t_0,x}^{t_1,y}(t)$ is a Brownian motion starting at x at time t_0 and going through y at time t_1. Thus,

$$W_{t_0,x}^{t_1,y}(t) = x + W(t - t_0) - \frac{t - t_0}{t_1 - t_0} \cdot (x + W(t_1 - t_0) - y)$$

for $t_0 < t < t_1$.

Ornstein–Uhlenbeck process

An Ornstein–Uhlenbeck process $V(t)$ models the velocity of a moving particle in fluid (performing a Brownian motion) with friction to mass given by α and mass $1/\beta$. It is defined by scaling and change of time as

$$V(t) = e^{-\alpha t} B(\frac{\beta^2}{2\alpha} e^{-\alpha t})$$

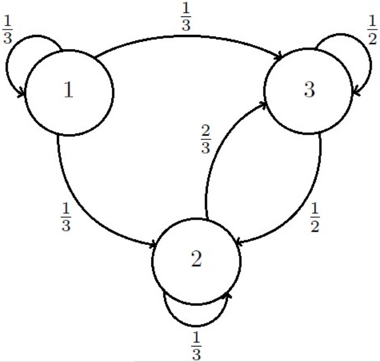
where $B(t)$ Brownian motion. It is also called Brownian motion-driven OU.

Simulating Stochastic Processes

Simulating a Finite Markov Chain

Example: Suppose a machine is in one of three states: State 1 is idle, State 2 is working, State 3 is broken down. The following digraph shows the transition probabilities and the states. Simulate the process n=10 sample paths of length 100 if the initial probability distribution is $P(0) = (.5, .5, 0)$

$$\mathbf{P} = \begin{bmatrix} \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ 0 & \frac{1}{3} & \frac{2}{3} \\ 0 & \frac{1}{2} & \frac{1}{2} \end{bmatrix}$$



Simulating a Finite Markov Chain

$$\mathbf{P} = \begin{bmatrix} \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ 0 & \frac{1}{3} & \frac{2}{3} \\ 0 & \frac{1}{2} & \frac{1}{2} \end{bmatrix}$$

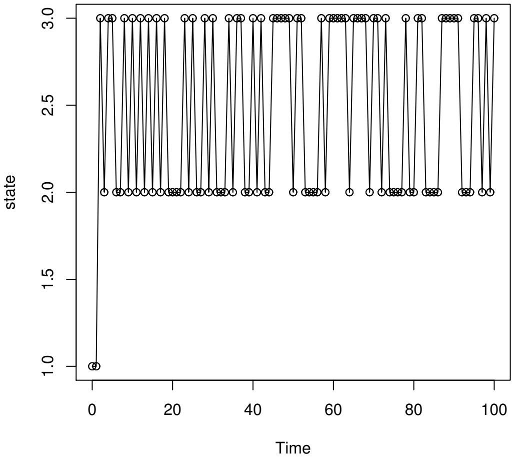
Algorithm:

1. Generate a number 1, 2 or 3 using $P(0)$, say j
2. Generate a number 1, 2 or 3 using $c(P_{j,1}, P_{j,2}, P_{j,3})$
3. Update j and return to step 2
4. Repeat steps 2 and 3 for T=1000 time steps.
5. Repeat the previous steps N=100 times

Simulating a Finite Markov Chain

```
N<-100
T<-1000
P<-matrix(c(1/3,1/3,1/3,0,1/3,2/3,0,1/2,1/2),3,byrow=T
init<-sample(1:3,N,prob=c(.5,.5,0),rep=T)
simu<-c()
state<-c()
for (n in 1:N){
state[1]<-init[n]
for (t in 2:(T+1)){
state[t]<-sample(1:3,1,prob=P[state[t-1],],rep=F)}
simu<-rbind(simu,state)}
plot(0:T,state,xlab="Time")
lines(0:T,state,type="o")
```

Simulating a Finite Markov Chain



Simulating a Finite Markov Chain

- Distribution of the end points of the N=100 possible paths

```
endpaths<-simu[,T+1]
table(endpaths)/sum(table(endpaths))
endpaths
   2    3
0.44 0.56
```

- Distribution of the last 100 states visited by any of the paths (say number 1) after running for a long time

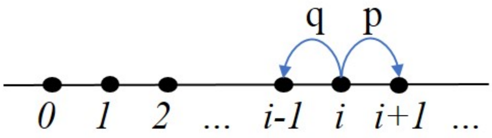
```
pend<-simu[1,(T-100):T]
table(pend)/sum(table(pend))
pend
        2         3
0.4356436 0.5643564
```

Simulating a Finite Markov Chain

- Distribution of the last 100 states visited by all the 100 paths after running for a long time

```
endpend<-simu[,(T-100):T]
table(endpend)/sum(table(endpend))
endpend
        2         3
0.4294059 0.5705941
```

Simulating Simple Random Walk on $\mathbb{Z}$



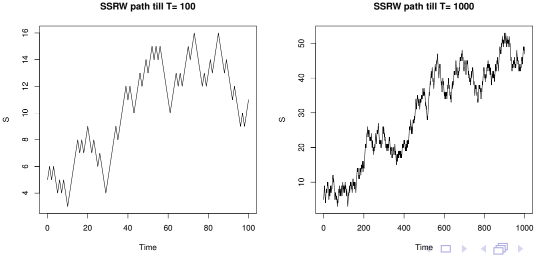
Algorithm:

1. Generate a number steps, of ± 1 with probabilities p and $1 - p$, equal to the length of total time T
2. Calculate the cumulative number of those steps to get the states visited by the SRW at the times between 1 and T
3. Add the initial state X_0 to the result of the previous step of the algorithm

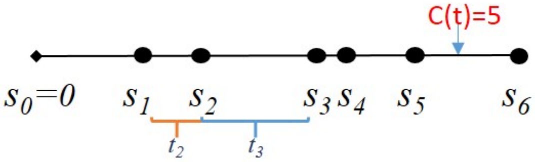
Simulating Simple Symmetric Random Walk on $\mathbb{Z}$

Example: Simulate a SSRW on $\mathbb{Z}$ starting from $X_0 = 5$ for $T = 100$ steps.

```
T<-100
X0<-5
steps<-sample(c(-1,1),size=T,rep=T)
S<-cumsum(c(X0,steps))
head<-paste("SSRW path till T=",T)
plot(0:T,S,type="l",xlab="Time",main=head)
```



Simulating a Poisson Process



To simulate the PP up to time T

Algorithm 1:

1. Generate a number random exponentially distributed inter-arrival times $T_j \sim exp(\lambda)$ (or any other distribution with a positive support for a renewable process)
2. Calculate the cumulative sum S_n of those random times till T
3. Then $N(T) = \min\{n : S_n > T\} - 1$

Simulating a Poisson Process

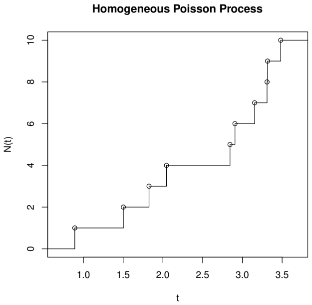
Example: Simulate a HPP with rate $\lambda = 2$ at time $T = 3.4$

```
T<-3.4
lambda<-2
Tj<-rexp(100,lambda)
Sn<-cumsum(Tj)
NT<-min(which(Sn>T))-1
NT
[1] 9
Sn[1:(NT+1)]
[1] 0.8913949 1.5022377 1.8253882 2.0450156
2.8440251 2.9066200 3.1539422
 [8] 3.3100125 3.3164891 3.4819056
```

Simulating a Poisson Process

Example: Simulate a HPP with rate $\lambda = 2$ at time $T = 3.4$

```
sfun <- stepfun(Sn[1:(NT+1)],
0:length(Sn[1:(NT+1)]), f = 0)
plot.stepfun(sfun,main="Homogeneous Poisson
Process",xlab="t",ylab="N(t)")
```



Simulating a Poisson Process

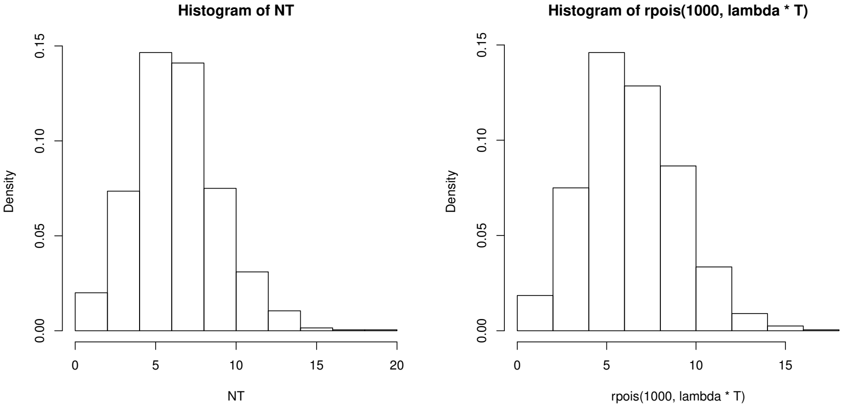
Example: Using simulation of a HPP with rate $\lambda = 2$ estimate the pmf of $N(3.4)$

```
T<-3.4
lambda<-2
rN<-1000
NT<-replicate(rN,expr={Tj<-rexp(100,lambda)
Sn<-cumsum(Tj)
min(which(Sn>T))-1})
table(NT)/sum(table(NT))
NT
 0 1 2 3 4 5 6 7
0.001 0.008 0.031 0.053 0.094 0.133 0.160 0.163
8 9 10 11 12
 0.119 0.091 0.059 0.037 0.025
 13 14 15 16 18 19
0.017 0.004 0.002 0.001 0.001 0.001
```

Simulating a Poisson Process

Example: Using simulation of a HPP with rate $\lambda = 2$ estimate the pmf of $N(3.4)$

```
hist(NT,prob=T)   hist(rpois(1000,lambda*T),prob=T)
```



Simulating a Poisson Process

Example: Using simulation of a HPP with rate $\lambda = 2$ estimate the pmf of $N(3.4)$

```
c(mean(NT),var(NT))
[1] 6.829000 6.346105
ks.test(NT,rpois(1000,lambda*T))$p
[1] 0.6852314
```

Simulating a Renewal Process

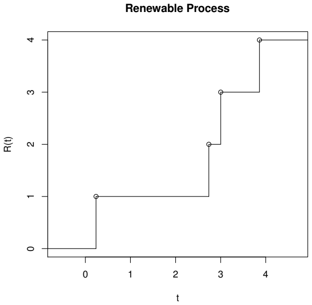
Example: Simulate a RP at time $T = 3.4$ with standard lognormally distributed inter-arrival time

```
T<-3.4
Tj<-rlnorm(100,0,1)
Sn<-cumsum(Tj)
RT<-min(which(Sn>T))-1
RT
[1] 3
Sn[1:(RT+1)]
[1] 0.2364835 2.7382336 3.0016010 3.8601179
```

Simulating a Renewal Process

Example: Simulate a RP at time $T = 3.4$ with standard lognormally distributed inter-arrival time

```
sfun <- stepfun(Sn[1:(RT+1)],
0:length(Sn[1:(RT+1)]), f = 0)
plot.stepfun(sfun,main="Renewable
Process",xlab="t",ylab="R(t)")
```



Simulating a Renewal Process

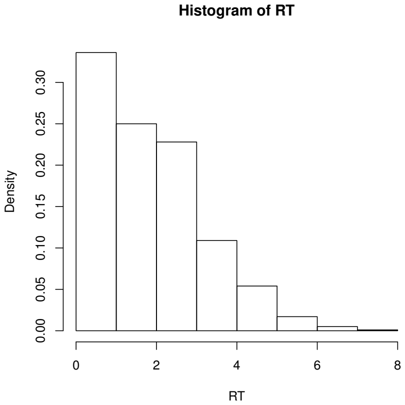
Example: Using simulation of a RP at time $T = 3.4$ with standard lognormally distributed inter-arrival time estimate the pmf of $R(3.4)$

```
T<-3.4
rN<-1000
RT<-replicate(rN,expr={Tj<-rlnorm(100,0,1)
Sn<-cumsum(Tj)
min(which(Sn>T))-1})
table(RT)/sum(table(RT))
RT
 0 1 2 3 4 5 6 7
0.123 0.213 0.250 0.228 0.109 0.054 0.017 0.005
8
0.001
```

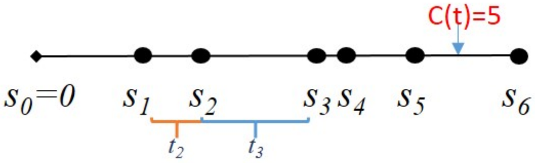
Simulating a Renewal Process

Example: Using simulation of a RP at time $T = 3.4$ with standard lognormally distributed inter-arrival time estimate the pmf of $R(3.4)$

```
hist(RT,prob=T)
```

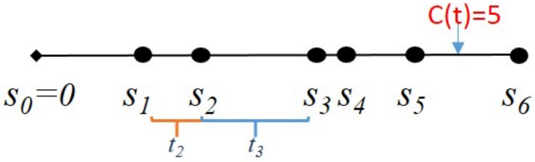


Simulating a non-homogeneous Poisson Process



To simulate the nHPP with intensity $\lambda(t)$ up to time T, find a λ_0 such that $\lambda(t) \leq \lambda_0$ for all $0 \leq t \leq T$. Simulate a HPP with rate λ_0 and accept an arrival at time t with probability $p_0 = \dfrac{\lambda(t)}{\lambda_0}$. This gives an arrival for the nHPP since the new rate is a thinning of the rate λ_0 by probability p_0 giving a new rate of $p_0\lambda_0 = \dfrac{\lambda(t)}{\lambda_0}\lambda_0 = \lambda(t)$

Simulating a non-homogeneous Poisson Process



Algorithm:

1. Generate a number random exponentially distributed inter-arrival times $T_j \sim exp(\lambda_0)$
2. Calculate the cumulative sum S_n of those random times till T
3. Retain S_n with probability $\dfrac{\lambda(S_n)}{\lambda_0}$ making a new sequence PS_n
4. Then $nHN(T) = \min\{n : PS_n > T\} - 1$

Simulating a non-homogeneous Poisson Process

Example: Simulate a nHPP with intensity $\lambda(t) = 3t^2$ at time $T = 3.4$.
Notice that the mean function is $m(t) = \mathbf{E}(N(t)) = \int_0^t \lambda(s)ds = t^3$.

```
T<-3.4
lambda<-function(t)3*(t)^2
lambda0<-lambda(T)
#lambda0<-(-1)*optim(1,function(t){-1*lambda(t)}
,method ="L-BFGS-B",lower=0,upper=T)$value
Tj<-rexp(lambda0*100,lambda0)
Sn<-cumsum(Tj)
p0<-lambda(Sn)/(lambda0)
thinby<-rbinom(n=1+0*p0,size=1,prob=p0)
PSn<-Sn[which(thinby==1)]
nNT<-min(which(PSn>T))-1
nNT
[1] Inf
```

Simulating a non-homogeneous Poisson Process

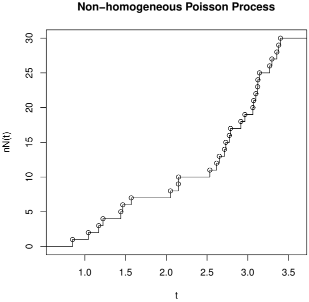
Example: Simulate a nHPP with intensity $\lambda(t) = 3t^2$ at time $T = 3.4$. Notice that the mean function is $m(t) = \mathbf{E}(N(t)) = \int_0^t \lambda(s)ds = t^3$. Try $\lambda_0 + 1 > \lambda_0 \geq \lambda(t)$.

```
T<-3.4
lambda<-function(t)3*(t)^2
lambda0<-lambda(T)
#lambda0<-(-1)*optim(1,function(t){-1*lambda(t)}
,method ="L-BFGS-B",lower=0,upper=T)$value
Tj<-rexp(lambda0*10,lambda0+1)
Sn<-cumsum(Tj)
p0<-lambda(Sn)/(lambda0+1)
thinby<-rbinom(n=1+0*p0,size=1,prob=p0)
PSn<-Sn[which(thinby==1)]
nNT<-min(which(Sn>T))-1
nNT
[1] 29
```

Simulating a non-homogeneous Poisson Process

Example: Simulate a nHPP with intensity $\lambda(t) = 3t^2$ at time $T = 3.4$. Notice that the mean function is $m(t) = \mathbf{E}(N(t)) = \int_0^t \lambda(s)ds = t^3$.

```
sfun <- stepfun(Sn[1:(nNT+1)],
0:length(Sn[1:(nNT+1)]), f = 0)
plot.stepfun(sfun,main="Non-homogeneous Poisson
Process",xlab="t",ylab="nN(t)")
```



Simulating Birth and Death Process

It is simulated using *Gillespie algorithm* with inter-arrival times that are exponentially distributed with rate $(\lambda+\mu)i$ and the event is counted as

- birth with probability $\frac{\lambda}{\lambda+\mu}$, and set $i = i + 1$
- death otherwise, i.e., with probability $\frac{\mu}{\lambda+\mu}$, and set $i = i - 1$

Recall: If $T_i \sim exp(\lambda_i)$ for $i = 1, \ldots, n$, then

$$\min_{1\leq i\leq n} T_i \sim exp(\sum_{i=1}^{n} \lambda_i)$$

and

$$P(T_j < \min_{\{i:1\leq i\leq n, i\neq j\}} T_i) = \frac{\lambda_j}{\sum_{i=1}^{n} \lambda_i}$$

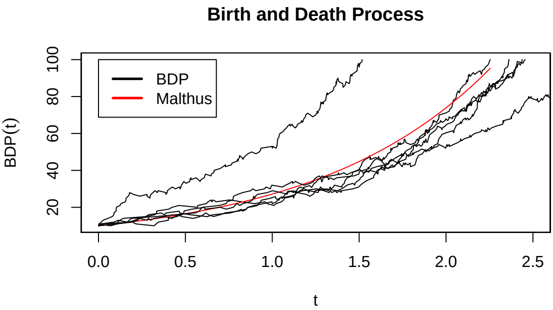
Simulating Birth and Death Process

Example: Simulate a birth and death process that starts with $X(0) = 10$ and till it either get extinct or it become abundant (say $X(t) = 100$) in which the per-capita birth rate $\lambda = 1.5$ and the per-capita death rate $\mu = .5$ compare that simulation to the Malthus Solution $X(t) = X(0)e^{(\lambda-\mu)t}$

```
X<-c(10);x<-X
t<-c(0);j<-1
lambda<-1.5;mu<-.5
while(x>0 & x<100){
 j<-j+1
 t[j]<-t[j-1]+rexp(1,(lambda+mu)*x)
 x<-x+(2*rbinom(1,1,lambda/(lambda+mu))-1)
 X<-c(X,x)}
plot(t,X,type="l",main="Birth and Death Process"
 ,xlab="t",ylab=expression(BDP(t)),xlim=c(0,2.5))
```

Simulating Birth and Death Process

```
lines(t,X[1]*exp((lambda-mu)*t),col="red")
legend(0,100,c("BDP","Malthus"),
 lty=c(1,1),lwd=c(2.5,2.5),col=c("black","red"))
```



Simulating Brownian motion or Wiener Process

How to generate a trajectory (sample path) of the Wiener process $W(t)$ on $[0, T]$

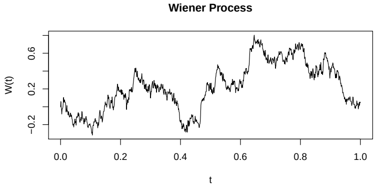
Algorithm:

1. Divide the time interval $[0, T]$ into n subintervals of length $\Delta t = T/n$
2. Set $W(0) = 0$
3. Generate next states according to $W(t + \Delta t) = W(t) + Z * \sqrt{\Delta t}$ where $Z \sim rnorm(0, 1)$
4. Or find cumulative sum of $W(n\Delta t) = \sum_{k=1}^{n} Z_k \sqrt{\Delta t}$ and $Z_1, Z_2, \ldots \sim N(0, 1)$

Simulating Brownian motion or Wiener Process

Example: Simulate a Wiener Process for $T = 1$

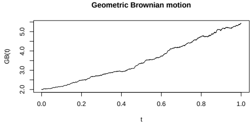
```
n<-1000; T<-1
delta<-T/n
W<-cumsum(c(0,rnorm(n,0,1)*sqrt(delta)))
plot(seq(0,T,delta),W,type="l",main="Wiener
Process",xlab="t",ylab="W(t)",xlim=c(0,T+delta))
```



Simulating Geometric Brownian motion

Example: Simulate a Geometric Brownian motion for $T = 1$ with drift $\mu = 1$ and volatility $\sigma = .1$ and starting at $S(0) = 2$

```
n<-1000; T<-1;mu<-1;sigma<-.1
delta<-T/n
W<-cumsum(c(0,rnorm(n,0,1)*sqrt(delta)))
S<-2*exp((mu-sigma^2/2)*seq(0,T,delta)+sigma*W)
plot(seq(0,T,delta),S,type="l",main="Geometric
Brownian motion",xlab="t",ylab="GB(t)"
,xlim=c(0,T+delta))
```



Simulating Brownian Bridge

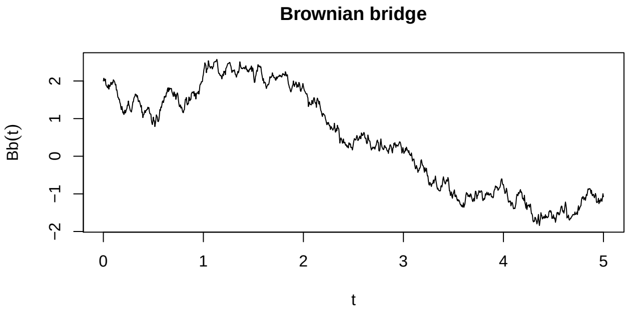
Example: Simulate a Brownian bridge $W_{0,2}^{5,-1}(t)$ starting at $t_0 = 0$ from $x = 2$ and going through $y = -1$ at time $t_1 = 5$

$$W_{0,2}^{5,-1}(t) = 2 + W(t) - \frac{t}{5} \cdot (3 + W(5))$$

for $0 < t < 5$.

```
n<-1000
t0<-0;t1<-5;x<-2;y<-1
delta<-(t1-t0)/n
W<-cumsum(c(0,rnorm(n,0,1)*sqrt(delta)))
t<-seq(t0,t1,delta)
Bb<-x+W-(t/t1)*(x-y+W[n])
plot(t,Bb,type="l",main="Brownian bridge",xlab="t"
,ylab=expression(Bb(t)),xlim=c(0,t1+delta))
```

Simulating Brownian Bridge



End of Set 4