Simulação de Eventos Discretos

Atividade III

Daniel dos Santos

30 de outubro de 18.

Funções Auxiliares

Gera uma v.a. $X \sim Exp(\lambda)$:

```
expr <- function(lambda){
  u <- runif(1)
  return(-1*log(u)/lambda)
}
```

Gera as v.a. $X_i \sim Exp(\lambda), i = 1, 2 \ldots, n$:

```
exprn <- function(n,lambda){
  exp <- NULL
  for(i in 1:n){
    exp[i] <- expr(lambda)
  }
  return(exp)
}
```

Gera uma v.a. $X \sim Poisson(\lambda)$:

```
poisson <- function(lambda){
  x <- 0
  u <- runif(1)
  i <- 0; p <- exp(-1*lambda); f <-p
  while(u>=f){
    i <- i+1;p <- p*lambda/i;f <- f+p
  }
  return(i)
}
```

Questão I

Gerando um processo de poisson não-homogêneo:

```
poisproc <- function(lambda, s){
  t <- s
  lambda_t <- (3+4/(t+1))
  t <- t + expr(lambda)
  while(runif(1) > lambda_t*1/7){
    t <- t + expr(lambda)
  }
  return(t)
}
```

Simulando o processo de fila:

λ = taxa de chegada do processo de poisson. No código: `lambda`;
λ_1 = parâmetro da primeira exponecial. No código: `lambda_1`;
λ_2 = parâmetro da primeira exponecial. No código: `lambda_2`;
N_A = Número de chegadas até o tempo t: No código: `Na`;
N_D = Número de saídas até o tempo t: No código: `Nd`;
$A_1(n)$ = Tempo de chegada do n-ésimo cliente ao servidor 1. No código: `A1`;
$D(n)$ = Tempo de saída do n-ésimo cliente do sistema. No código: `D`;
t_a = Tempo da próxima chegada. No código: `ta`;
t_1 = Tempo gasto do cliente atualmente no servidor 1, $t_1 \sim Exp(\lambda_1)$ No código: `t1`;
t_2=Tempo gasto do cliente atualmente no servidor 2, $t_2 \sim Exp(\lambda_2)$. No código: `t2`;
T = Tempo de análise. No código: `Time`;
$\lambda(t) = 3 + \frac{4}{t+1}$, se $t \geq 0$;

Código que simula a fila com dois servidores em série:

```
series_queue <- function(lambda, lambda_1, lambda_2, Time){
  t <- 0; Na <- 0; Nd <- 0; n1 <- 0; n2 <- 0; A1 <- 0; A2 <- 0; D <- 0
  ta <- poisproc(lambda, 0) ; t1 <- Inf ; t2 <- Inf
  while(ta <= Time){
    if(ta == min(ta, t1, t2)){
      t <- ta
      Na <- Na + 1
      n1 <- n1 + 1
      ta <- poisproc(lambda, t)
      if(n1 == 1) t1 <- t + expr(lambda_1)
      A1[Na] <- t
    }
    if(t1 < ta & t1 <= t2){
      t <-  t1
      n1 <- n1 - 1 ; n2 <- n2 + 1
      if(n1 == 0){
        t1 <- Inf
      } else {
        t1 <- t + expr(lambda_1)
      }
      if(n2 == 1) t2 <- t + expr(lambda_2)
      A2[Na - n1] <- t
    }
    if(t2 < ta & t2 < t1){
      t <- t2
      Nd <- Nd + 1
      n2 <- n2 - 1
      if(n2 == 0) t2 <- Inf
      if(n2 > 0) t2 <- t + expr(lambda_2)
      D[Nd] <- t
    }
  }
  while (n1!=0 || n2!=0){
    if (t1 <= t2){
      t <-  t1
      n1 <- n1 - 1
      n2 <- n2 + 1
      if(n1 == 0){
```

```r
        t1 <- Inf
      }
      else {
        t1 <- t + expr(lambda_1)
      }
      if(n2 == 1) t2 <- t + expr(lambda_2)
      A2[Na - n1] <- t
    } else {
      t <- t2
      Nd <- Nd + 1
      n2 <- n2 - 1
      if(n2 == 0) t2 <- Inf
      if(n2 > 0) t2 <- t + expr(lambda_2)
      D[Nd] <- t
    }
  }
  return(list(Chegada = A1, `Saída` = D, mean.perm = mean(D - A1)))
}
```

Média do tempo de permanência: Para $\lambda = 7$, $\lambda_1 = 1$, $\lambda_2 = 3$ e $T = 100$.

```r
tm <- 0
for(i in 1:100){
  tm[i] <- series_queue(7,1,3,100)$mean.perm
}
mean(tm)
```

```
## [1] 114.1012
```

Questão II

Função que simula o lucro no dia:

```r
lucro_dia <- function(lambda){
  n <- poisson(lambda)
  custo_dia <- exprn(n, 1/1000)
  return(11000 - sum(custo_dia))
}
```

Função que simula o lucro médio dos dias em um ano:

```r
lucro_medio <- function(lambda){
  d <- NULL ; prob <- NULL
  d[1] <- 25000 + lucro_dia(lambda)
  prob[1] <- ifelse(d[1] >= 0, 1 ,0)
  for(i in 2:365){
    d[i] <- d[i-1] + lucro_dia(lambda)
    prob[i] <- ifelse(d[i] >= 0, 1, 0)
  }
  return(mean(prob))
}
```

Probabilidade de lucrar:

```
lm <- 0
for(i in 1:100){
  lm[i] <- lucro_medio(10)
}
mean(lm)
```

```
## [1] 0.9932877
```

Questão III

Algorítimo que utilizamos:

- Assumimos:
  1. Sistema precisa de n máquinas funcionando;
  2. Máquina falha independente depois de um tempo $X \sim F$, para algum F;
  3. Máquinas quebradas são imediatamente substituídas e mandadas para reparo;
  4. As máquinas são reparadas na ordem em que foram quebradas;
  5. Máquinas consertadas ficam como reservas;
  6. Um reparado dura um tempo $R \sim G$, para algum G;
  7. O sistema para se alguma máquina falha sem que haja outras máquinas na reserva;
  8. Assuma inicialmente $n + s$ o total de máquinas;
  9. O objetivo da simulação é determinar o valor esperado do tempo, $E[T]$;
- Variáveis:
  1. **Tempo** t;
  2. **Estado** do Sistema r, número de máquinas no reparo
  3. **Eventos** $t_1 \leq t_2 \leq \cdots \leq t_n$ tempos em que houveram falhas e t^* tempo para o término do próximo reparo;
- Algorítimo: dadas as constantes n, s, e $F(x)$, $G(x)$;
  Inicialize $t = r = 0, t^* = \infty$, gere $X_i \sim F, i = 1, 2, \ldots, n$ e ordene para conseguir os ti's;
  Enquanto $r < s + 1$, atualize o estado do sistema usando dois casos:

- **Se** $t_1 < t^*$ (uma nova falha)
  reinicie $t = t_1$, $r = r + 1$
  se $r < s + 1$ gere $X \sim F$,
  e ordene $t_2, t_3, \ldots, t_n, t + X$;
  se $r = 1$, gere $R \sim G$, e reinicie $t^* = t + R$;
- **Senão** (um reparo terminado)
  reinicie $t = t^*$, $r = r - 1$;
  caso contrário, $t^* = \infty$.
- **Saída** tempo de quebra $T = t$

Código que simula um modelo de reparos: Para $n = 4$, $s = 3$, $\lambda_1 = 1$ e $\lambda_2 = 2$.

```
repair_model <- function(n, s, lambda_1, lambda_2){
  t <- 0 ; r <- 0 ; t_ <- Inf
  x <- exprn(n,lambda_1) ; x <- sort(x)
  while(r < s + 1){
    if(x[1] < t_){
      t <- x[1] ; r <- r + 1
      if(r < s + 1) {
        X <- expr(lambda_1)
        x <- x[-1]
        x <- sort(c(x, t + X))
```

```r
      }
      if(r == 1){
        R <- expr(lambda_2)
        t_ <- t + R
      }
    } else {
      t <- t_ ; r <- r - 1
      if(r > 0){
        R <- expr(lambda_2)
        t_ <- t + R
      } else {
        t_ <- Inf
      }
    }
  }
  return(t)
}

repair_model(4, 3, 1 ,2)
```

```
## [1] 1.221662
```

Simulando a média:

```r
rm_m <- NULL
for(i in 1:100){
  rm_m[i] <- repair_model(4, 3, 1, 2)
}
mean(rm_m)
```

```
## [1] 1.473241
```