

IME - USP

MAE – 312
Introdução aos Processos Estocásticos

Professor: Eduardo Jordão Neves

Alunos:

Edilene F. N. Gomes
Fernando Henrique Ferraz Pereira da Rosa
Guilherme
Karen Elisa do Vale Nogueira
Robson Lunardi
Mateus Moreira Costa
Vagner Aparecido Pedro Júnior

Índice

◆ **Parte I:** Problemas para “aquecimento”

Problema do prisioneiro

Problema da barra

Problema do jogo com números consecutivos

◆ **Parte II:** Paradoxo da inspeção

◆ **Parte III:** Cadeia de Markov a tempo contínuo

◆ **Parte IV:** Filas M/M/k

◆ **Parte V:** Funilaria e pintura

Introdução

O presente trabalho trata-se de uma extensão das aulas de introdução aos processos estocásticos. No início do curso, alguns problemas para “aquecimento”, envolvendo paradoxos probabilísticos conhecidos, foram propostos. Abordamos esses problemas através de simulações e, quando possível, através de análise teórica, comparando os resultados e discutindo-os.

A seguir, propomos uma modelagem computacional para a simulação de alguns tipos de cadeias de Markov a tempo contínuo, implementando e avaliando a qualidade dos resultados obtidos através da comparação com a modelação teórica das cadeias.

As simulações e algoritmos propostos foram implementados no pacote estatístico R.

Parte I: Problemas para “aquecimento”

1) Problema do prisioneiro

Descrição

Há três prisioneiros (eu e mais dois) e um de nós está condenado ‘a morte. O carcereiro (que nunca mente), sabe qual de nós irá morrer. Eu pergunto a ele qual dos outros dois prisioneiros não irá morrer. Ele se nega a responder alegando que, se ele disser qual dos dois não morre, a probabilidade de que eu morra passa de 1/3 para 1/2. O carcereiro está certo ao afirmar isso?

Resolução teórica

Sejam os três prisioneiros: A, B e C; I = indica o prisioneiro que vai morrer e II = indica o prisioneiro que foi apontado pelo carcereiro, ou seja, que não vai morrer.

$$P(I = A / II = B) = \frac{P(I = A, II = B)}{P(II = B)} = \frac{P(II = B / I = A) P(I = A)}{P(II = B / I = A) P(I = A) + P(II = B / I = B) P(I = B) + P(II = B / I = C) P(I = C)} = \frac{(1/3 * 1/2)}{[(1/3 * 1/2) + (1/3 * 0) + (1/3 * 1)]} = \frac{1}{3}$$

Portanto, o carcereiro está errado – a probabilidade de que eu morra continua sendo 1/3.

Implementação Computacional

Com a função abaixo no R, simulamos n vezes o problema descrito acima:

```
prison <- function(n) {  
  minha.morte <- numeric(n)  
  for (i in 1:n) {  
    morre <- sample(1:3,1)  
    decisao.carc <- sample((1:3)[c(-morre,-1)],1)  
    minha.morte[i] <- ifelse(morre==1,1,0)  
  }  
  mean(minha.morte)  
}
```

A idéia é criar um vetor de sucessos e fracassos (1 e 0, respectivamente), que indica se cada vez que repetimos o problema morremos ou não. Basta então tirar a média do vetor retornado para estimar a probabilidade de que morramos.

Simulação

```
> prison(10000)  
[1] 0.329  
> prison(10000)  
[1] 0.336  
> prison(100000)  
[1] 0.33149  
> prison(1000000)  
[1] 0.333443
```

Comentários

A simulação do problema mostra que, a probabilidade de que eu morra dado que o carcereiro me falou qual dos outros dois não morrerá, se aproxima de 1/3, confirmando o resultado visto em aula. Dessa forma o carcereiro está errado.

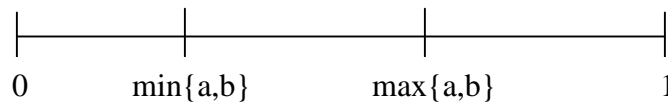
2) Problema da barra

Descrição

Quebra-se uma barra “uniformemente” em três pedaços. Qual a probabilidade de que seja possível formar um triângulo com esses três pedaços?

Implementação Computacional

Supomos que consigamos quebrar uma barra em qualquer ponto do seu comprimento, contrariando as leis da Física. Consideremos uma barra de tamanho 1 e sorteamos duas uniformes em $[0,1]$ que representarão os pontos de quebra na barra, ou seja, com probabilidades iguais, encontro dois pontos a e b entre 0 e 1. Tomamos o mínimo entre a e b , esse será o primeiro ponto de quebra da barra. O máximo entre a e b será o segundo ponto de quebra da barra. A ilustração abaixo mostra o procedimento:



Sabemos que para que três segmentos de reta formem um triângulo é preciso que qualquer um deles seja menor do que a soma dos demais. Notemos ainda que para a divisão da barra nos pontos acima, temos três segmentos, de tamanho: $\min\{a,b\}$, $\max\{a,b\} - \min\{a,b\}$ e $1 - \max\{a,b\}$ respectivamente. Basta então verificar se esses valores satisfazem a restrição imposta. Para isso usamos a função `is.triangle()`, dentro da função `bars()`, que simula o procedimento n vezes. No vetor `outcome` armazenamos para cada iteração se houve um sucesso(1) ou fracasso(0), então basta calcular a média para ver a probabilidade estimada.

```
bars <- function(n) {  
  is.triangle <- function(l) {  
    if ((l[1] + l[2] > l[3]) && (l[1] + l[3] > l[2]) && (l[3] + l[2] > l[1]))  
    { 1 }  
    else { 0 }  
  }  
  a <- runif(n)  
  b <- runif(n)  
  lados <- cbind(pmin(a,b),pmax(a,b)-pmin(a,b),1-pmax(a,b))  
  outcome <- apply(lados,1,is.triangle)  
  mean(outcome)  
}
```

Simulação

```
> bars(100)  
[1] 0.26
```

```
> bars(1000)
[1] 0.25
> bars(10000)
[1] 0.2476
> bars(100000)
[1] 0.24886
```

Comentários

Considerando as suposições impostas para a resolução do problema, chegamos a conclusão de que, a probabilidade de que seja possível formar um triângulo com os três pedaços da barra é aproximadamente 0,25.

3) Problema do jogo dos números consecutivos

Descrição

Estamos jogando eu e o meu oponente. O jogo consiste em sortear-se duas cartas com números consecutivos (n e $n+1$) e cada jogador escolher, ao acaso, uma delas. De posse da minha carta, ainda não sei quanto ela vale, porém, sei o quanto vale a carta do meu oponente. Sendo assim, a minha carta pode ser maior ou menor do que a dele. Se eu quiser jogar, mostro ao meu oponente a minha carta e, se ela for maior do que a dele, ou seja, se a minha for $(n+1)$, o meu oponente terá que me pagar $(n+1)$ dólares, caso a minha for menor, terei que pagar a ele $(n+1)$ dólares. Qual é o meu ganho médio no jogo?

Considerações Teóricas

O problema está mal formulado, no sentido de que não é possível sortear aleatoriamente um número qualquer entre os naturais com equiprobabilidade. Qualquer distribuição discreta uniforme num conjunto infinito tem função de probabilidade nula – ou seja, não pode ser definida. Para podermos simular o problema temos que fixar um ‘teto’, um N máximo, para que possamos sortear um número n de 0 a N com igual probabilidade para qualquer um dos números.

Implementação Computacional

Fixado um teto N , sorteamos um número n entre 0 e $(N-1)$. Com probabilidade meio, atribuímos as cartas n e $n+1$ ao jogador e oponente. Verificamos então qual é o ganhador e computamos o ganho médio para uma jogada com número de rodadas dado por rounds. A função `jogo(n, rounds, N)` simula esse procedimento n vezes. Supomos que o jogador sempre aceita jogar.

```
jogo <- function(n, rounds, N=100) {
game <- function(rounds, N=100) {
ganho <- 0
for (i in 1:rounds) {
carta.jogador <- sample(0:(N-1), 1)
if (runif(1) > 0.5) { carta.oponente <- carta.jogador + 1 }
else { carta.oponente <- carta.jogador - 1 }
if (carta.oponente > carta.jogador){
ganho <- ganho - carta.oponente
}
else {ganho <- ganho+ carta.oponente }
}
}
```

```
}  
ganho/rounds  
}  
ganho <- numeric(n)  
for(i in 1:n) {  
ganho[i] <- game(rounds,N)  
}  
ganho  
}
```

Simulação

```
> mean(jogo(10000,500,100))  
[1] -1.005809  
> mean(jogo(10000,100,500))  
[1] -0.513171  
> mean(jogo(10000,100,5000))  
[1] 0.487189  
> mean(jogo(10000,100,5000))  
[1] 0.634782
```

Comentários

Observamos que o o ganho médio por round fica ao redor do zero, variando aleatoriamente de simulação para simulação. Concluimos que o jogo não é vantajoso para o jogador.

Parte II – Paradoxo da Inspeção

Descrição

Uma linha de ônibus entra em atividade logo pela manhã num dado horário fixado. Nesse horário parte o primeiro ônibus da garagem. Os intervalos entre a passagem de sucessivos ônibus é aleatório e obedece a uma distribuição de probabilidade arbitrária (Exponencial, Uniforme, Beta). Uma pessoa chega no ponto de ônibus em um determinado instante (muito tempo depois do horário de início da linha), qual é o tempo médio que essa pessoa espera até que o próximo ônibus passe?

Implementação Computacional

Criamos uma função `onib(N, chegada, intervalo, parâmetros` (da distribuição), que simula N vezes o problema, considerando o que a pessoa chega no instante `chegada` e o intervalo entre os ônibus tem distribuição `intervalo` com `parâmetros`

```
onib <- function(N, chegada=100, intervalo=rexp, ...) {
  intervalo <- match.fun(intervalo)
  espera <- numeric(N)
  for (i in 1:N) {
    tempo <- 0
    n <- 0
    while (tempo < chegada) {
      tempo <- tempo + intervalo(1, ...)
      n <- n+1
    }
    espera[i] <- tempo - chegada
  }
  espera
}
```

Simulação

```
mean(onib(10000)) # parametros padrões, dist exponencial, lambda=1
[1] 0.9788296
> mean(onib(10000, rate=2)) # dist exponencial, lambda=2
[1] 0.5004989
> mean(onib(10000, rate=1/2)) # dist exponencial, lambda = 1/2
[1] 1.990516
> mean(onib(10000, rate=1/3)) # dist exponencial, lambda = 1/3
[1] 2.979467
> mean(onib(10000, intervalo=runif, min=0, max=1))# uniforme [0,1]
[1] 0.3317195
> mean(onib(10000, intervalo=runif, min=0, max=2))# uniforme [0,2]
[1] 0.6653952
> mean(onib(10000, intervalo=runif, min=0, max=3))# uniforme [0,3]
[1] 0.9915383
```

Comentários

Quando consideramos exponencial com média 1h a distribuição do intervalo de tempo entre duas passagens do ônibus no ponto, a simulação do problema resulta em média aproximadamente 1h, o que quer dizer que, em média, independente do horário em que

chegamos ao ponto temos que esperar 1 hora até a passagem do ônibus. De modo análogo verificamos empiricamente que isso vale para aparentemente qualquer λ . De forma que se a distribuição dos intervalos é exponencial com taxa λ , teremos sempre que esperar em média $1/\lambda$ horas até a passagem do próximo ônibus.

Quando consideramos a distribuição uniforme de amplitude $(b-a)$, o tempo médio de espera foi de $(b-a)/3$. De forma que o tempo médio de espera que uma pessoa teria no caso da distribuição dos intervalos entre os ônibus uniforme em $(0,3)$, seria de 1 hora.

Parte III – Cadeia de Markov a tempo contínuo (M/M/1)

Seja $\{X_t\}$ uma cadeia de Markov a tempo contínuo, $t \in I$ em S , onde I é um conjunto de índices “contínuo”, por exemplo, $I = \mathbb{R}^+ = \{x \in \mathbb{R} / x \geq 0\}$ e $S =$ espaço de estados enumerável. $\{X_t\}$ satisfazendo a propriedade de Markov:

$$P(X_t = j / X_{s_0} = i_0; \dots; X_{s_{n-1}} = i_{n-1}; X_s = i) = P(X_t = j / X_s = i), 0 \leq s_0 < s_1 < \dots < s_{n-1} < t.$$

Um dos exemplos mais simples de cadeia de Markov a tempo contínuo é o Processo de Poisson. Seja $N(t) \sim PPP(t)$ um processo de contagem, onde $N(t)$ é o número de eventos no intervalo de 0 à t .

Por definição, temos: $N(s,t) = N(t) - N(s)$.

$N(t)$ satisfaz:

a) $N(0) = 0$ e se $s \leq t$, $N(s) \leq N(t)$.

b) $N(t,t+h)$

$$f(h) = P(N(t,t+h) = 1) = \lambda h + O(h).$$

c) Se $(S1,t1]$ e $(S2,t2]$ são intervalos disjuntos, então $N(S1,t1)$ e $N(S2,t2)$ são variáveis aleatórias independentes.

Teorema: Se $\{N(t)\} t \geq 0$ satisfaz a), b) e c), então $P(N(t) = k) = \frac{e^{-\lambda t} \cdot (\lambda t)^k}{k!}$.

Construção da cadeia de Markov

1) A cada estado $i \in S$, associe uma seqüência de variáveis aleatórias T_i^1, T_i^2, \dots independentes e identicamente distribuídas, com $T_i \sim$ exponencial (λ_i), que indica os sucessivos tempos de permanência em cada estado i .

2) Matriz de transição da “cadeia inversa”.

Q_{ij} = Probabilidade de saltar de i para j .

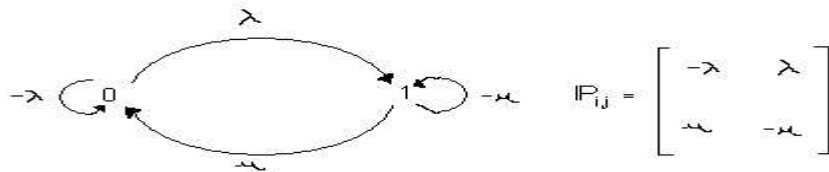
$$P_{ij}(t) = P(X(t) = j / X(0) = i)$$

$$\text{Taxa de transição do estado } i \text{ para o } j \Rightarrow q_{ij} = \frac{\partial P_{ij}(t)}{\partial t}$$

Sistema de dois estados

Segundo as definições acima, consideremos uma cadeia com espaço de estados $S = \{0,1\}$. $\{X_t\}$, $t \geq 0$. Podemos pensar por exemplo num consultório médico, com capacidade de atendimento para apenas um paciente. Ou o consultório está vazio (estado 0) ou está sendo atendido um paciente (estado 1). Se um paciente chega e vê o consultório ocupado, ele vai embora para não mais voltar. Supomos que os pacientes chegam conforme um processo pontual de Poisson com parâmetro (λ) e o tempo de atendimento segue uma distribuição exponencial com parâmetro (μ) . Supomos ainda que os atendimentos são independentes entre si.

A seguir, temos a representação gráfica dos estados da cadeia bem como a matriz de taxas.



Probabilidades de Transição

Condição para a medida estacionária:

$$\Pi(j)q_j = \sum_{k \neq j} \Pi(k)q_{kj}, \text{ qualquer } j \in S. \text{ Equação de balanceamento}$$

Resolvendo as equações de balanceamento e as equações avançadas e retrógradas de Kolmogorov obtemos:

$$P_{00}(t) = \frac{\mu}{(\mu + \lambda)} + \frac{\lambda}{(\mu + \lambda)} e^{-(\mu + \lambda)t}$$

$$P_{10}(t) = \frac{\mu}{(\mu + \lambda)} (1 - e^{-(\mu + \lambda)t})$$

$$P_{01}(t) = \frac{\lambda}{(\mu + \lambda)} (1 - e^{-(\mu + \lambda)t})$$

$$P_{11}(t) = \frac{\lambda}{(\mu + \lambda)} + \frac{\mu}{(\mu + \lambda)} e^{-(\mu + \lambda)t}$$

Distribuição Estacionária

Pelas equações de balanceamento obtemos que a distribuição estacionária é:

$$\Pi = \left(\frac{\mu}{(\mu + \lambda)} \quad \frac{\lambda}{(\mu + \lambda)} \right)$$

Implementação Computacional

A implementação computacional tem que impor algumas restrições ao problema, para que possamos modelá-lo. Isso ocorre por estarmos simulando um processo contínuo através de incrementos discretos, de forma que alguns cuidados devem ser tomados na hora de tirar conclusões sobre os resultados.

Começemos notando que se as chegadas num intervalo $[0,t]$ seguem um processo pontual de Poisson com taxa λ , podemos dividir esse intervalo $[0,t]$ em i pedaços, de forma, que, por independência:

$$N(t) = N(t/i) + N(t/i) + \dots + N(t/i)$$

O que estamos fazendo é dividir o intervalo de 0 a t minutos, em $i=10$ intervalos sucessivos $[0,t/10]$ $[t/10,2t/10]$ $[2t/10],[3t/10]$, ..., $[9t/10,t]$. Em cada intervalo desses simulamos um processo pontual de poisson individual e independente dos outros com taxa λ/i . Se essa taxa λ/i for pequena em relação a $[0,t]$, de forma que a probabilidade de ocorrência de uma chegada em cada intervalo seja muito baixa, estamos propondo uma aproximação discreta razoável para o problema contínuo.

Portanto, nas simulações que se seguem, a variação dos parâmetros fica restrita a taxas de chegada menores do que 1, e a variação dos tempos de permanência nos estados recomenda-se que fique restrita também a valores menores do que 1. Dessa forma garantimos que a cada passo discreto que damos no sistema, a probabilidade da ocorrência de mais de uma chegada ou saída é baixa e portanto nossa aproximação razoável.

A primeira função criada é a que simula os estados do sistema acima durante $n/10$ minutos.

```
CTMC <- function(lambda,mi,n) {
  sistema <- numeric(n)
  for (i in 2:n) {
    tempo.atual <- i * 1/10
    chegada = rpois(1,lambda * 1/10)
    if (sistema[i-1] == 0) {
      if (chegada > 0) {
        sistema[i] = 1
        relógio <- rexp(1,mi)
        despertador <- tempo.atual + relógio
      }
    }
    else {
      if (tempo.atual >= despertador) {
        sistema[i] = 0
      }
      else {
        sistema[i] = 1
      }
    }
  }
  sistema
}
```

A idéia é dividir cada minuto em intervalos pequenos de 1/10 minuto, e simular as transições do sistema em passos de 1/10 minuto. Começando no 1 e indo até $n \cdot 1/10$ minutos. No vetor sistema armazenamos o estado do sistema a cada passo. Supomos começar no estado 0, e iteramos então os outros estados até terminar o número de minutos desejados. Cada vez que chega alguém e o sistema esta vazio, ativamos o despertador e o sistema fica nesse estado até que o despertador ‘toque’ (que ele seja maior que o tempo atual). A função retorna um vetor de 0’s e 1’s indicando os estados do sistema.

A função abaixo estima as probabilidades de transição $P_{ij}(t)$, para um instante t arbitrário (menor do que $n/10$, naturalmente).

```
estima.Pt <- function(t,sistema,inicial,final) {
  sucessos <- 0
  L <- length(sistema)
  if (t > L/10) { stop('nao é possivel estimar Pij(t) para t maior
que o tempo simulado.') }
  tamanho.janela <- t*10
  N <- L-tamanho.janela+1
  for (i in 1:N) {
    a <- sistema[i]
    b <- sistema[i+(tamanho.janela)-1]
    if (a != inicial) { N <- N -1 }
    else if (b == final) { sucessos <- sucessos + 1 }
  }
  sucessos/N
}
```

Ela funciona tomando um vetor de estados e tomando sucessivos intervalos de tamanho t , para estimar $P_{ij}(t)$. Suponhamos que desejemos estimar $P_{00}(1)$, e o vetor de estados simulados seja:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	0	1	0	1	0	1	1	1	0	0	1	0	0	1	0	1	0	1	0

Onde a primeira linha representa o tempo t em 1/10 minutos e a segunda linha o estado do sistema nesse tempo. Temos os estados do sistema de 1/10 minutos até 2 minutos. Queremos estimar $P_{00}(1)$. Pela definição $P_{00}(1)$ é a probabilidade de após um intervalo de tempo de 1 minuto, termos começado no estado 0 e tivermos continuado no estado 0. Para estimar esse valor, consideremos janelas sucessivas de tamanho 1 minuto, a partir do instante $t=1$, até o instante $t=2$. Verificamos então quais dessas janelas começaram no 0, e quais terminaram no 0. O número de janelas que começaram no 0 e terminaram no 0, dividido pelo número de janelas que começaram no 0, corresponde a uma estimativa da probabilidade de transição de 0 para 0 em 1 minuto ($P_{00}(1)$). Notemos que quanto maior for $P_{00}(t)$ em relação ao tempo total simulado, menos precisa será nossa estimativa, pois estaremos considerando um número menor de janelas. Ilustremos o algoritmo da função `estima.Pt()` para o caso da tabela de estados ilustrada acima:

Seja *sucessos* a variável contendo 0's e 1's de acordo com o sucesso ou fracasso do evento começar em 0 e terminar em 0. Seja *N* o número total de janelas nas quais começamos em 0.

A primeira janela que o algoritmo vai considerar será:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	0	1	0	1	0	1	1	1	0	0	1	0	0	1	0	1	0	1	0

sucessos = 1

N = 1

Começamos no 0, depois de 1 minuto continuamos no 0. É portanto um sucesso. Em seguida serão consideradas:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	0	1	0	1	0	1	1	1	0	0	1	0	0	1	0	1	0	1	0

sucessos = 2

N = 2

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	0	1	0	1	0	1	1	1	0	0	1	0	0	1	0	1	0	1	0

sucessos = 2

N = 2

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	0	1	0	1	0	1	1	1	0	0	1	0	0	1	0	1	0	1	0

sucessos = 3

N = 4

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	0	1	0	1	0	1	1	1	0	0	1	0	0	1	0	1	0	1	0

sucessos = 3

N = 4

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	0	1	0	1	0	1	1	1	0	0	1	0	0	1	0	1	0	1	0

sucessos = 4

N = 5

E assim por diante, obtendo no final *sucessos* = 5 e *N* = 6. Estimariamos nesse caso $P_{00}(t)$ por $5/6$. A função é genérica o suficiente entretanto para lidar com qualquer outro espaço de estados, não necessariamente $S=\{0,1\}$, e pode ser utilizada para estimar $P_{ij}(t)$ para qualquer cadeia de markov a tempo contínuo simulada de acordo com as restrições impostas aqui.

Simulação

Sejam os parâmetros:

$$\lambda = \frac{1}{30} \text{ e } \mu = \frac{1}{20}$$

Ou seja, chegam 1/30 pessoas por minuto em média, ou 2 pessoas por hora. E cada pessoa leva em média 20 minutos para ser atendida. Com o comando abaixo simulamos esse sistema por 60 mil minutos, começando no estado 0 (ninguém sendo atendido), armazenando o resultado no vetor problema2.

```
lambda <- 1/30
mi <- 1/20
problema2 <- CTMC(lambda,mi,600000)
```

Podemos agora, estimar a distribuição estacionária através desse vetor:

```
summary(factor(problema2))/length(problema2)
      0      1
0.5971033 0.4028967
```

Pelo desenvolvimento teórico do problema, sabemos que esses valores são:

$$\Pi = \left(\frac{\mu}{(\mu + \lambda)} \quad \frac{\lambda}{(\mu + \lambda)} \right) = (0.60 \quad 0.40)$$

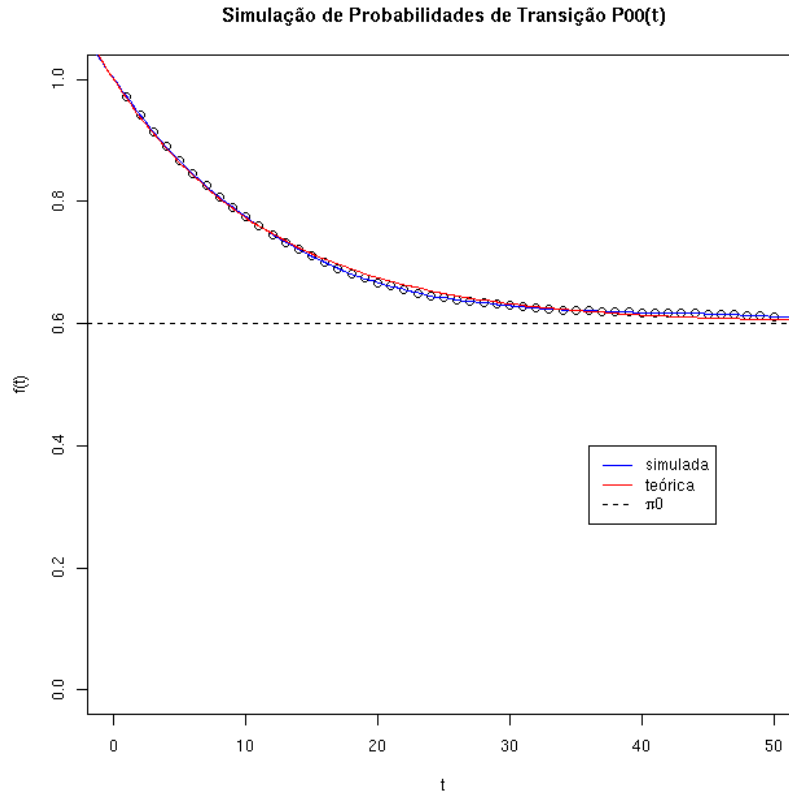
Logo notamos que a estimação da distribuição estacionária pela simulação foi bem sucedida. Outra forma de estimar a distribuição estacionária para o estado j , é estimar $P_{ij}(t)$ para t grande, logo que conforme t cresce essa probabilidade converge para a distribuição estacionária. Estimando então, nessa ordem, $P_{00}(10000)$, $P_{10}(10000)$, $P_{01}(10000)$, $P_{11}(10000)$ temos:

```
estima.Pt(10000,problema2,0,0)
[1] 0.5964387
estima.Pt(10000,problema2,1,0)
[1] 0.5978925
estima.Pt(10000,problema2,0,1)
[1] 0.4035613
estima.Pt(10000,problema2,1,1)
[1] 0.4021075
```

Novamente os resultados são consistentes com as considerações teóricas. Queremos agora estimar as funções $P_{ij}(t)$ e compará-las com as distribuições teóricas. Para isso, podemos utilizar a função já criada `estima.Pt()` aplicando ela em um range de valores de t , e calculando seu valor para cada um desses valores de t . Para isso usamos o comando:

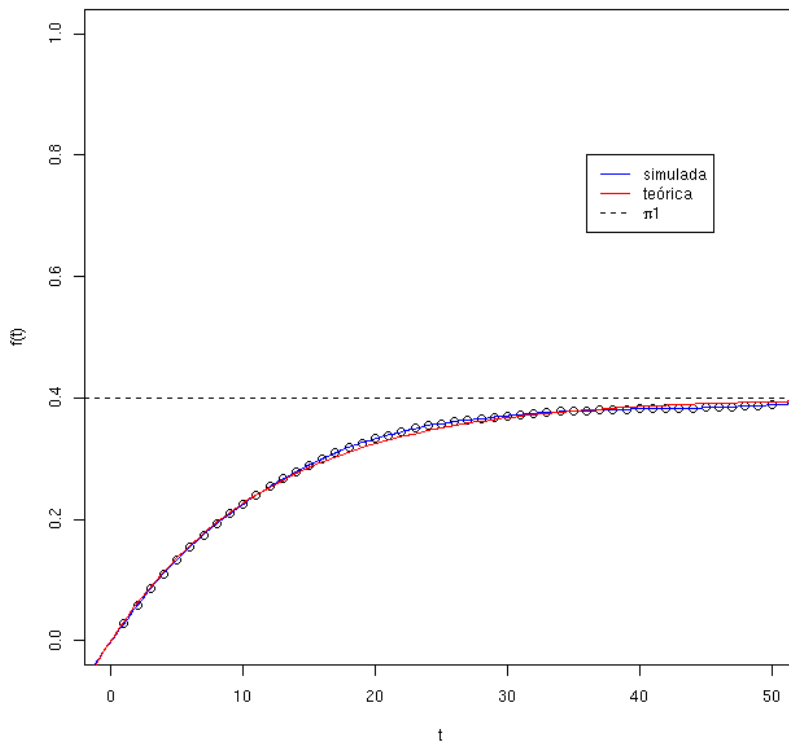
```
t <- 1:50
y <- sapply(t,estima.Pt,sistema=problema2,inicial=0,final=0)
```

Fazemos então o diagrama de dispersão de t por y , acrescentando como referência a curva teórica (em vermelho), a distribuição estacionária (linha tracejada) e o ajuste de smoothing splines de t por y (azul). Obtemos o gráfico:

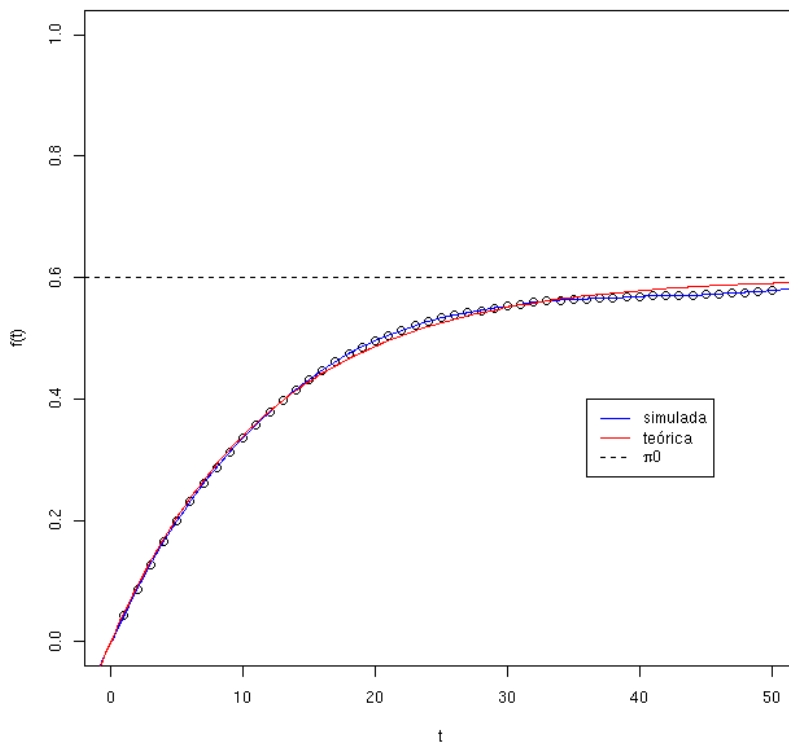


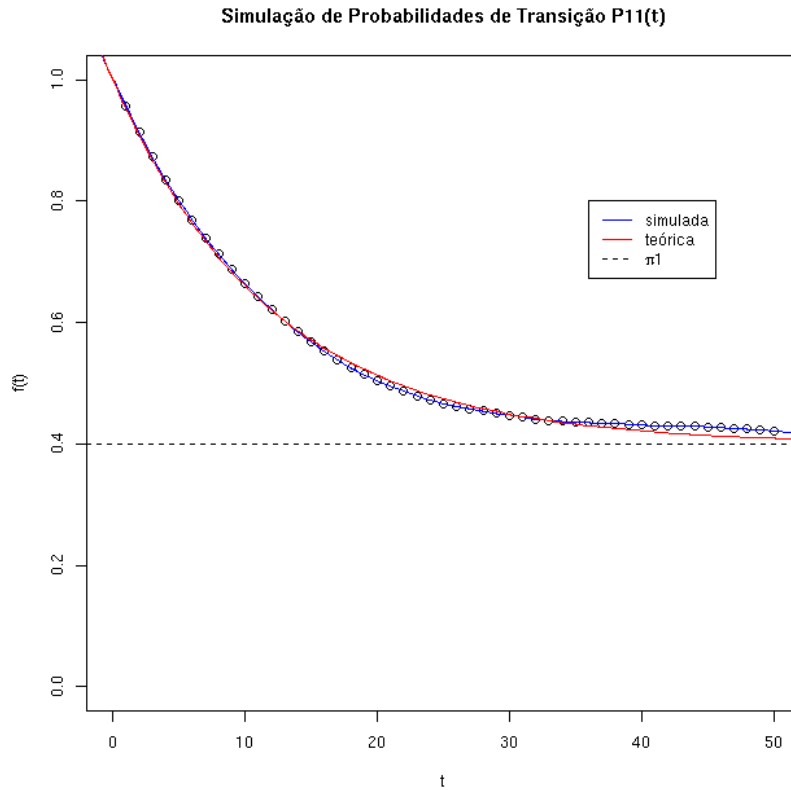
Que mostra uma aproximação muito boa. De forma análoga fazemos o mesmo para os outros $P_{ij}(t)$, obtendo os gráficos:

Simulação de Probabilidades de Transição P01(t)



Simulação de Probabilidades de Transição P10(t)





Comentários

Conforme observamos os resultados da simulação, os valores foram condizentes com os calculados teoricamente, mostrando que as suposições para modelagem foram adequadas e que o sistema foi simulado corretamente.

Parte IV – Filas M/M/k

Descrição

Consideremos um sistema M/M/k, ou seja: chegadas de acordo com processo pontual de Poisson, tempo de atendimento independente exponencial e k atendentes. Pergunta-se: qual sistema é melhor, um com n atendentes cada um com taxa de atendimento μ ou um sistema com 1 atendente com taxa de atendimento $n\mu$?

Organização do atendimento e Definição do Espaço de Estados

Para podermos escrever um modelo computacional para simular o problema acima, precisamos torná-lo um pouco mais preciso. Em primeiro lugar, vamos estipular que a fila pode crescer indefinidamente, ou seja, não há limite para o número de pessoas que possam ficar na fila. As pessoas chegam e se não há nenhum atendente disponível elas aguardam em fila. Em segundo, vamos estabelecer que no caso que haja mais de um atendente, as pessoas aguardam em fila única.

Seja o espaço de estados $S = \{0,1,2,\dots\}$, que representa o número de cliente no sistema num dado instante. Há n atendentes disponíveis, cada um atende com taxa μ . Dado que estamos no estado i, se $i > n$, temos i-n pessoas na fila. Se $n \leq i$, temos n-i pessoas sendo atendidas e nenhuma na fila. Os clientes chegam de acordo com um processo pontual de Poisson com taxa λ .

Implementação Computacional

Para simular o problema atual, utilizamos a mesma idéia básica utilizada para a simulação do sistema de dois estados: dividimos cada minuto em pedaços de 1/10 minuto e simulamos a evolução do sistema a cada 1/10 minutos. A função criada para tal fim é a `filanat()`, que recebe como parâmetros a taxa de chegada de clientes λ , a taxa do tempo de atendimento de cada atendente μ , o número de atendentes `nat`, e o tempo em 1/10 minutos `n` que queremos simular o sistema. As opções `verbose` e `file` permitem salvar a evolução do sistema passo a passo.

```
filanat <- function(lambda,mi,nat,n,verbose=FALSE,file="") {
  clientes.chegaram <- 0
  clientes.atendidos <- 0
  sistema <- numeric(n)
  despertadores <- rep(-1,nat)
  for (i in 2:n) {
    tempo.atual <- i * 1/10
    chegada = rpois(1,lambda * 1/10)
    clientes.chegaram <- clientes.chegaram + chegada
    sistema[i] <- sistema[i-1]+chegada
    if (sistema[i-1] < nat & chegada > 0) { # verifica se
      tinha alguem livre em i-1 e aloca chegadas se sim
      vagas <- nat - sistema[i-1]
      if (vagas != sum(despertadores == -1)) {
stop('non conforming supositions!')
      }
      for (j in 1:min(vagas,chegada)) {
        relógio <- rexp(1,mi)
        despertadores[which(despertadores == -
1)][1]] <- tempo.atual + relógio
```

```

                                clientes.atendidos <- clientes.atendidos
+ 1                                }
                                }
                                if (sum(tempo.atual >= despertadores & despertadores != -
1) > 0) { # verifica se tem algum atendente que liberou algum de i-1 pra
i, se sim, libera o caixa e aloca se tiver algum na fila
                                for(j in which(tempo.atual >= despertadores &
despertadores != -1)) {
                                    sistema[i] <- sistema[i]-1
                                    despertadores[j] <- -1
                                }
                                if (sum(despertadores == -1) - max(nat-
sistema[i],0) > 0) { # se essa condicao é verdadeira, há despertadores
que precisam ser disparados ainda
                                    for(j in 1:(sum(despertadores == -1) -
max(nat-sistema[i],0))) {
                                        relógio <- rexp(1,mi)
                                        despertadores[which(despertadores
== -1)[1]] <- tempo.atual + relógio
                                        clientes.atendidos <-
clientes.atendidos + 1
                                    }
                                }
                                }
                                if (verbose) {
cat(tempo.atual,despertadores,sistema[i],'\n',file=file,append=TRUE) }
                                }

list(sistema=sistema,clientes.atendidos=clientes.atendidos,clientes.chegam
ram=clientes.chegaram)
}

```

A implementação desse sistema é mais complicada pois estamos permitindo um número arbitrário de atendentes e um espaço de estados potencialmente infinito. Basicamente, agora precisamos manter um vetor de despertadores, sendo que cada despertador é ativado quando um cliente entra no sistema e desativado quando ele sai. As diversas avaliações lógicas são para se alocar corretamente os clientes conforme eles vão chegando, e para ajustar adequadamente o valor dos despertadores para cada atendimento. Também mantemos controle de quantos clientes chegaram no sistema e quantos foram atendidos até o final da simulação. Para ilustrar as estruturas de dados utilizadas para a modelação do sistema, consideremos a simulação de um sistema com 2 atendentes com taxa de atendimento $\mu = 1/5$ por min e chegadas $\lambda = 1/2$ por min, durante 20 minutos. Utilizamos o comando:

```
fila.exemplo <- filanat(1/2,1/5,2,200,verbose=T,file="relatorio.txt")
```

Lemos o resultado então no R novamente:

```
relat <- read.table('relatorio.txt',col.names=c("tempo","despertador
1","despertador 2","estado do sistema"))
```

Obtemos dessa forma a tabela abaixo. A primeira coluna indica em que tempo estamos, as duas colunas seguintes representam o despertador de atendimento para os atendentes 1 e 2. O estado do sistema indica em que estado o sistema se encontra no dado instante de tempo. Nesse caso se o estado é 0, não há nenhum cliente sendo atendido. Se o estado é 1 há um cliente sendo atendido, 2 há dois clientes sendo atendidos, se 3 há dois clientes sendo atendidos e 1 na fila, e assim por diante. Se o despertador de atendimento é -1, quer dizer que o dado caixa não está atendendo ninguém. Assim que ele começar a atender alguém, o valor desse campo diz até quando o cliente vai ficar no sistema.

tempo (min.)	despertador 1	despertador 2	estado do sistema	Comentários
0.2	-1	-1	0	O sistema começa vazio.
0.3	-1	-1	0	
0.4	-1	-1	0	
0.5	-1	-1	0	
0.6	-1	-1	0	
0.7	-1	-1	0	
0.8	-1	-1	0	
0.9	1.416755	-1	1	No instante 0.9 chega alguém no sistema, e é atendido pelo
1	1.416755	-1	1	atendente 1. Ele só vai acabar o
1.1	1.416755	-1	1	atendimento no instante 1.41.
1.2	1.416755	-1	1	
1.3	1.416755	-1	1	No instante 1.4 ele ainda
1.4	1.416755	-1	1	continua em atendimento. No
1.5	-1	-1	0	instante 1.5 já não há mais
1.6	-1	-1	0	ninguém no sistema.
(...)				
7.4	-1	-1	0	
7.5	-1	-1	0	
7.6	9.81985	-1	1	Chega alguém no instante 7.6, e
7.7	9.81985	-1	1	vai ficar sendo atendido até o
7.8	9.81985	-1	1	instante 9.81.
7.9	9.81985	-1	1	
8	9.81985	-1	1	
8.1	9.81985	-1	1	
8.2	9.81985	-1	1	
8.3	9.81985	-1	1	
8.4	9.81985	18.25512	2	No instante 8.4 chega mais
8.5	9.81985	18.25512	2	alguém. O atendimento vai durar
8.6	9.81985	18.25512	2	até o instante 18.25. Como só há
8.7	9.81985	18.25512	2	dois atendentes, ninguém mais
8.8	9.81985	18.25512	2	pode ser atendido.
8.9	9.81985	18.25512	2	
9	9.81985	18.25512	2	
9.1	9.81985	18.25512	2	
9.2	9.81985	18.25512	2	
9.3	9.81985	18.25512	2	
9.4	9.81985	18.25512	2	
9.5	9.81985	18.25512	2	
9.6	9.81985	18.25512	2	
9.7	9.81985	18.25512	2	

9.8	9.81985	18.25512	2	Toca o despertador do primeiro
9.9	-1	18.25512	1	atendimento, o primeiro cliente
10	-1	18.25512	1	vai embora. O segundo
10.1	-1	18.25512	1	continua. O sistema volta pro
(...)				estado 1.
14.3	-1	18.25512	1	
14.4	-1	18.25512	1	
14.5	15.48404	18.25512	2	Chega mais alguém no instante
14.6	15.48404	18.25512	2	14.5. Ocupando o caixa 1.
14.7	15.48404	18.25512	3	Chega outra pessoa logo em
14.8	15.48404	18.25512	3	seguida, ficando na fila.
14.9	15.48404	18.25512	3	
15	15.48404	18.25512	3	
15.1	15.48404	18.25512	3	
15.2	15.48404	18.25512	3	
15.3	15.48404	18.25512	3	No instante 15.5 toca o
15.4	15.48404	18.25512	3	despertador do caixa 1. O cliente
15.5	20.99808	18.25512	2	que estava la vai embora e o que
15.6	20.99808	18.25512	2	estava na fila vai ser atendido
15.7	20.99808	18.25512	2	até o instante 20.99. Não há
15.8	20.99808	18.25512	2	ninguém na fila.
(...)				
19.6	20.99808	23.64792	4	O sistema termina no instante 20
19.7	20.99808	23.64792	4	min com dois clientes sendo
19.8	20.99808	23.64792	4	atendidos e dois na fila.
19.9	20.99808	23.64792	4	
20	20.99808	23.64792	4	

Simulação

Vamos primeiro simular uma fila com 4 atendentes com taxa de tempo de atendimento $1/20$ e taxa de chegada $1/10$.

```
fila.quad <- filanat(1/10,1/20,4,600000)
```

Podemos agora obter a distribuição estacionária estimada:

```
> t(t(summary(factor(fila.quad$systema))/length(fila.unic$systema)))
  [,1]
0 0.1233900000
1 0.2504516667
2 0.2624133333
3 0.1793750000
4 0.0919616667
5 0.0479350000
6 0.0236216667
7 0.0128533333
8 0.0043100000
9 0.0013400000
10 0.0006366667
11 0.0004450000
12 0.0008816667
```

```
13 0.0002416667
14 0.0001433333
```

Podemos por exemplo observar que o sistema ficou 12% do tempo vazio e pouco mais de 51% do tempo com os dois caixas atendendo e ninguém na fila. No resto do tempo (37%) haviam pessoas na fila. Temos também que:

```
> fila.quad$clientes.atendidos
[1] 6062
> fila.quad$clientes.chegaram
[1] 6062
```

Ou seja, todos os clientes que chegaram foram atendidos.

Façamos agora o processo análogo para uma fila com 1 atendente e taxa 4/20.

```
fila.unic <- filanat(1/10,4/20,1,600000)
```

Obtendo a distribuição estacionária estimada:

```
> t(t(summary(factor(fila.unic$sistema))/length(fila.unic$sistema)))
  [,1]
0 5.022167e-01
1 2.541750e-01
2 1.227717e-01
3 5.512000e-02
4 2.895333e-02
5 1.411167e-02
6 8.905000e-03
7 5.438333e-03
8 4.060000e-03
9 2.495000e-03
10 9.766667e-04
11 5.066667e-04
12 2.283333e-04
13 4.166667e-05
```

Donde vemos que o sistema ficou 50% do tempo vazio e 25% do tempo ocupado sem fila. Logo houveram filas 25% do tempo. O número de clientes que chegaram/foram atendidos foi de:

```
> fila.unic$clientes.atendidos
[1] 5984
> fila.unic$clientes.chegaram
[1] 5989
```

Podemos ainda estimar o tamanho médio da fila nos dois casos:

```
> mean(fila.quad$sistema[which(fila.quad$sistema>4)]-4)
[1] 1.896745
> mean(fila.unic$sistema[which(fila.unic$sistema>1)]-1)
[1] 2.129326
```

Comentários

A partir da comparação do resultado das simulações acima, não é possível concluir taxativamente que um dos dois sistemas é razoavelmente melhor do que o outro. O sistema com 1 atendente ficou uma menor parte do tempo (25%) com pessoas na fila, enquanto o sistema com 4 atendentes ficou 37% do tempo com pessoas na fila. Entretanto o tamanho médio da fila no sistema com 1 atendente foi um pouco maior que o tamanho médio no sistema com 4 atendentes. O número de clientes atendidos foi quase 100% nos dois casos, não servindo como critério de desempate. Logo pelos critérios estudados, os dois sistemas aparentam ser equivalentes.

Parte V – Funilaria e Pintura

Descrição

Uma oficina mecânica presta serviços de funilaria e de pintura, ambos com capacidade para atender um carro por vez, ao mesmo tempo. Todos os carros dos clientes que chegam, necessariamente, passam pelo serviço de funilaria e em seguida pelo serviço de pintura. O sistema é composto por cinco estados, $S = \{00, 10, 01, 11, E1\}$, abaixo descritos:

00 : não há carro na funilaria e na pintura.

10 : há um carro na funilaria.

01 : há um carro na pintura.

11 : há um carro na funilaria e um na pintura.

E1 : há um carro que já passou pela funilaria e está aguardando a saída do carro que está na pintura para fazer o processo de pintura.

Organização do atendimento

Os clientes chegam conforme um Processo Pontual de Poisson com taxa (λ), o tempo necessário para o serviço de funilaria em um carro segue uma exponencial com parâmetro (μ_1) e o tempo necessário para o serviço de pintura em um carro segue uma exponencial com parâmetro (μ_2).

Os clientes que chegam e encontram o serviço de funilaria ocupado, vão embora “para não mais voltar”, ou seja, o cliente foi “perdido”.

Distribuição estacionária

Para encontrarmos a distribuição estacionária, basta resolvermos as equações de balanceamento, e a restrição da soma das probabilidades:

$$(00) \lambda P_{00} = \mu_2 P_{01}$$

$$(10) \mu_1 P_{10} = \lambda P_{00} + \mu_2 P_{11}$$

$$(01) (\lambda + \mu_2) P_{01} = \mu_1 P_{10} + \mu_2 P_{E1}$$

$$(11) (\mu_1 + \mu_2) P_{11} = \lambda P_{01}$$

$$(E1) \mu_2 P_{E1} = \mu_1 P_{11}$$

$$P_{00} + P_{10} + P_{01} + P_{11} + P_{E1} = 1$$

Temos 6 equações e 5 incógnitas (as taxas são constantes, no caso). Descartamos a última equação (E1), pois ela é redundante dadas as outras 5. Montamos então esse sistema na forma matricial:

$$A_{5 \times 5} \times \tilde{X}_{5 \times 1} = \tilde{B}_{5 \times 1} \quad \text{ou seja} \quad \begin{bmatrix} \lambda & \mu_2 & 0 & 0 & 0 \\ -\lambda & 0 & \mu_1 & -\mu_2 & 0 \\ 0 & (\lambda + \mu_1) & -\mu_1 & 0 & -\mu_2 \\ 0 & -\lambda & 0 & (\mu_1 + \mu_2) & 0 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} \times \begin{bmatrix} P_{00} \\ P_{01} \\ P_{10} \\ P_{11} \\ P_{E1} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

Assim dados valores fixados para μ_1, μ_2 e λ , basta resolver essa equação matricial para obtermos a distribuição estacionária.

Implementação Computacional

A implementação computacional desse sistema foi bem similar aos outros dois já implementados. Utilizamos a mesma idéia básica da subdivisão do intervalo de tempo em unidades menores e da evolução discreta do sistema por essas unidades.

```
funilaria <- function(lambda,mi1,mi2,n,verbose=FALSE,file="") {
  clientes.perdidos <- clientes.atendidos <- 0
  sistema <- rep("00",n)
  desp.mi1 <- desp.mi2 <- -1

  for(i in 2:n) {
    tempo.atual <- i * 1/10
    chegada = rpois(1,lambda * 1/10)
    sistema[i] <- sistema[i-1]
    if (tempo.atual >= desp.mi1 & tempo.atual >= desp.mi2 &
desp.mi1 != -1 & desp.mi2 != -1) { # tocam mi1 e mi2
      sistema[i] <- "01"
      desp.mi1 <- -1
      desp.mi2 <- tempo.atual+rexp(1,mi2)
    }
    else if (tempo.atual >= desp.mi2 & desp.mi2 != -1) { #
toca só mi2
      if (sistema[i-1] == "E1") { sistema[i] <- "01";
desp.mi2 <- tempo.atual+rexp(1,mi2) }
      else if (sistema[i-1] == "11") { sistema[i] <-
"10"; desp.mi2 <- -1}
      else { sistema[i] <- "00"; desp.mi2 <- -1 }
    }
    else if (tempo.atual >= desp.mi1 & desp.mi1 != -1) { #
toca só mi1
      if (sistema[i-1] == "11") { sistema[i] <- "E1" }
      else { sistema[i] <- "01"; desp.mi2 <-
tempo.atual+rexp(1,mi2) }
      desp.mi1 <- -1
    }
    if (tempo.atual < desp.mi1) { # nao toca mi1 (mi1 ligado)
      substr(sistema[i],1,1) <- "1"
    }
    if (tempo.atual < desp.mi2) { # nao toca mi2 (mi2 ligado)
      substr(sistema[i],2,2) <- "1"
    }
  }

  # qq modificacao q ocorreu no sistema por despertadores
tocando

  # acabou. agora resta ver se chegou alguem
  if (chegada > 0 & substr(sistema[i],1,1) == "0") {
    clientes.atendidos <- clientes.atendidos + 1
    substr(sistema[i],1,1) <- "1"
  }
}
```

```

        relógio <- rexp(1,mil)
        desp.mil <- tempo.atual + relógio
    }
    else if (chegada > 0 & substr(sistema[i],1,1) == "1") {
        clientes.perdidos <- clientes.perdidos + chegada
    }
    if (verbose) {
cat(tempo.atual,desp.mil,desp.mi2,sistema[i],'\n',file=file,append=TRUE)
    }

}

list(sistema=sistema,clientes.perdidos=clientes.perdidos,clientes.atendidos=clientes.atendidos,lambda=lambda,mil=mil,mi2=mi2,n=n)
}

```

A implementação é mais longa que as dos outros dois sistemas e envolve muito mais avaliações lógicas por causa de certas peculiaridades desse sistema em específico, como por exemplo, cada cliente ter que passar por um estágio, depois ir para outro obrigatoriamente. Isso tornou o algoritmo praticamente uma exaustão de todas as possibilidades de transições de estado para estado supondo que os despertadores só tocam no máximo uma vez por incremento de tempo (suposição razoável para incrementos de tempo razoavelmente pequenos).

Criamos também uma função para facilitar a análise do resultado proveniente dessa simulação. A função recebe como parâmetros uma lista retornada pela função `funilaria`, o custo por minuto para funilaria, o custo por minuto para pintura e o preço cobrado de cada cliente, e retorna um objeto da classe `'analise.desempenho'`, com informações como lucro médio, número de clientes perdidos etc.

```

analise.desempenho <- function(funil.list,custo.amass,custo.pint,preco) {
    # custo por minuto para desamassar: custo.amass
    # custo por minuto para pintar: custo.pint
    # preco por cliente atendido: preco
    tempo.amass <- sum(funil.list$sistema == "10" |
funil.list$sistema == "11")/10
    tempo.pint <- sum(funil.list$sistema == "E1" | funil.list$sistema
== "11" | funil.list$sistema == "01")/10

    custo.total <- custo.amass * tempo.amass + custo.pint *
tempo.pint
    lucro.total <- funil.list$clientes.atendidos*preco - custo.total
    lucro.medio <- (10*lucro.total/funil.list$n)

    x <-
list(sistema=funil.list$sistema,mil=funil.list$mil,mi2=funil.list$mi2,cli
entes.perdidos=funil.list$clientes.perdidos,clientes.atendidos=funil.list
$clientes.atendidos,lambda=funil.list$lambda,n=funil.list$n,tempo.amass=t
empo.amass,tempo.pint=tempo.pint,custo.total=custo.total,lucro.total=lucr
o.total,lucro.medio=lucro.medio,custo.amass=custo.amass,custo.pint=custo.
pint,preco=preco)
    class(x) <- c("analise.desempenho", "list")
    return(x)
}

```

```
}
```

Por fim, criamos um método `print` para a classe `'analise.desempenho'`, da seguinte forma:

```
print.analise.desempenho <- function(x) {
  cat("\n      Analise de Desempenho do Sistema - Funilaria e
Pintura\n\n")
  cat("  Modelo teórico \n")
  cat("  Chegadas: PPP(lambda = ",x$lambda,")\n",sep='')
  cat("  Tempo para funilaria: exp(mi1=",x$mi1,")\n",sep='')
  cat("  Tempo para pintura: exp(mi2=",x$mi2,")\n",sep='')
  cat("  Minutos simulados: ",x$n/10,"\n\n")
  cat("  Simulação\n")
  cat("  Tempo médio \t Funilaria \t Pintura\n")
  cat("  esperado \t ",1/x$mi1,"\t\t",1/x$mi2,"\n")
  cat("  observado \t
",x$tempo.amass/x$clientes.atendidos,"\t",x$tempo.pint/x$clientes.atendid
os,"\n\n")
  cat("  Balanço financeiro\n")
  cat("  Custo para desamassar: \tR$
",format(x$custo.amass,nsml=2),"por min.\n")
  cat("  Custo para pintar:
\tR$",format(x$custo.pint,nsml=2),"por min.\n")
  cat("  Preço por cliente: \tR$",format(x$preco,nsml=2),"\n")
  cat("  Clientes atendidos:",x$clientes.atendidos," Clientes
perdidos:",x$clientes.perdidos,"\n")
  cat("  Custo total:
R$",format(x$custo.total,nsml=2),"Recebimento total: R$
",format(x$clientes.atendidos*x$preco,nsml=2),"\n")
  cat("  Lucro total: R$",format(x$lucro.total,nsml=2),"\n")
  cat("  Lucro médio por cliente:
\tR$",format(x$lucro.total/x$clientes.atendidos),"\n")
  cat("  Lucro médio por minuto: \tR$",format(x$lucro.medio),"\n")
  return(invisible(x))
}
```

Dessa forma a análise fica muito mais simples de ser realizada, concentrando-se nos aspectos práticos do problema e não na parte computacional. Como exemplo, seja um sistema com taxa de chegada $1/5$, taxa para funilaria $1/40$, taxa para pintura $1/10$, custo para funilaria de R\$ 0.50 por minuto, custo para pintura de R\$ 0.70 centavos por minuto e preço por cliente de R\$ 50,00. Para simular e analisar o desempenho desse sistema depois de 600 minutos fazemos:

```
> funil <- funilaria(1/5,1/40,1/10,6000)
> funil.anal <- analise.desempenho(funil,0.5,0.7,15)
> funil.anal
```

```
      Analise de Desempenho do Sistema - Funilaria e Pintura
```

```
Modelo teórico
Chegadas: PPP(lambda = 0.2)
Tempo para funilaria: exp(mi1=0.025)
Tempo para pintura: exp(mi2=0.1)
```

```
Minutos simulados: 600
```

Simulação

Tempo médio esperado	Funilaria	Pintura
40	29.16875	10
observado	29.16875	7.30625

Balanco financeiro

```
Custo para desamassar: R$ 0.50 por min.  
Custo para pintar: R$ 0.70 por min.  
Preco por cliente: R$ 50.00  
Clientes atendidos: 16 Clientes perdidos: 85  
Custo total: R$ 315.18 Recebimento total: R$ 800.00  
Lucro total: R$ 484.82  
Lucro médio por cliente: R$ 30.30125  
Lucro médio por minuto: R$ 0.8080333
```

Por fim, implementamos o cálculo da distribuição estacionária teórica, através da equação matricial mencionada na descrição do problema.

```
estacionaria.Teorica <- function(lambda,mi1,mi2,estado) {  
  A <- matrix(c(lambda,-lambda,0,0,1,-mi2,0,lambda+mi2,-  
lambda,1,0,mi1,-mi1,0,1,0,-mi2,0,mi1+mi2,1,0,0,-mi2,0,1),ncol=5)  
  colnames(A) <- c("P00","P01","P10","P11","PE1")  
  b <- matrix(c(0,0,0,0,1),ncol=1)  
  colnames(b) <- "P estacionaria"  
  resultado <- solve(A,b)  
  if (missing(estado)) { resultado }  
  else { resultado[paste("P",estado,sep=""),] }  
}
```

Simulação

Vamos simular o exemplo já citado acima mas durante mais tempo (60 mil minutos):

```
> funil2 <- funilaria(1/5,1/40,1/10,600000)
```

Verifiquemos em primeiro lugar, a distribuição estacionária estimada:

```
> t(t(summary(factor(funil2$sistema))/funil2$n))  
[,1]  
00 0.03581833  
01 0.07231333  
10 0.75654667  
11 0.10434333  
E1 0.03097833
```

A distribuição estacionária teórica será dada por:

```
> estacionaria.Teorica(1/5,1/40,1/10)  
P estacionaria
```

P00	0.03597122
P01	0.07194245
P10	0.74820144
P11	0.11510791
PE1	0.02877698

Os valores estimados pela simulação estão bem próximos dos teóricos, indicando a adequação do algoritmo proposto.

Temos ainda, interpretando esses valores que:

- 3% do tempo não tem ninguém no sistema;
- 7% do tempo somente a pintura está ocupada;
- 75% do tempo a funilaria está ocupada;
- 10% do tempo tanto a funilaria quanto a pintura estão ocupadas;
- 3% do tempo alguém está na espera para a pintura;

Suponhamos agora que estejamos interessados em estudar o desempenho dessa funilaria. Fixamos o preço de custo para a funilaria, para a pintura e para o cliente, neste caso sendo:

- R\$0.30 por minuto o custo da funilaria;
- R\$0.20 por minuto o custo da pintura;
- R\$30.00 é o preço cobrado do cliente atendido;

```
> obj2 <- analise.desempenho(funil2,0.3,0.2,30)
> obj2
  Analise de Desempenho do Sistema - Funilaria e Pintura

Modelo teórico
Chegadas: PPP(lambda = 0.2)
Tempo para funilaria: exp(mi1=0.025)
Tempo para pintura: exp(mi2=0.1)
Minutos simulados: 60000
Simulação
Tempo médio   Funilaria   Pintura
esperado      40          10
observado     39.67235    9.568433

Balanço financeiro
Custo para desamassar:      R$ 0.30 por min.
Custo para pintar:         R$ 0.20 por min.
Preço por cliente:         R$ 30.00
Clientes atendidos: 1302   Clientes perdidos: 10269
Custo total: R$ 17987.64  Recebimento total: R$ 39060.00
Lucro total: R$ 21072.36
Lucro médio por cliente:   R$ 16.18461
Lucro médio por minuto:    R$ 0.351206
```

Para estes valores fixados a funilaria & pintura atendeu 1302 clientes e perdeu 10269 clientes; o lucro total foi de R\$21072,36. Vamos supor agora que houve um aumento de preço no custo da pintura devido a alta no dólar. O preço passou de 0.2 por minuto para 0.4 por minuto. Com isso espera-se ocorrer uma diminuição do lucro:

```

> obj3 <- analise.desempenho(funil2,0.3,0.4,30)
> obj3
      Analise de Desempenho do Sistema - Funilaria e Pintura

Modelo teórico
Chegadas: PPP(lambda = 0.2)
Tempo para funilaria: exp(mi1=0.025)
Tempo para pintura: exp(mi2=0.1)
Minutos simulados: 60000

Simulação
Tempo médio   Funilaria       Pintura
esperado      40                10
observado     39.67235         9.568433

Balanço financeiro
Custo para desamassar:      R$ 0.30 por min.
Custo para pintar:         R$ 0.40 por min.
Preço por cliente:         R$ 30.00
Clientes atendidos: 1302   Clientes perdidos: 10269
Custo total: R$ 20479.26  Recebimento total: R$ 39060.00
Lucro total: R$ 18580.74
Lucro médio por cliente:   R$ 14.27092
Lucro médio por minuto:    R$ 0.309679

```

Como tivemos aproximadamente menos R\$2500,00 de lucro queremos repassar este “prejuízo” para os clientes. Como atendemos em média 1302 clientes vamos aumentar em R\$2,00 o preço por cliente para que o lucro volte ao que era antes, ficando assim:

```

> obj4 <- analise.desempenho(funil.list2,0.3,0.4,32)
> obj4
      Analise de Desempenho do Sistema - Funilaria e Pintura

Modelo teórico
Chegadas: PPP(lambda = 0.2)
Tempo para funilaria: exp(mi1=0.025)
Tempo para pintura: exp(mi2=0.1)
Minutos simulados: 60000

Simulação
Tempo médio   Funilaria       Pintura
esperado      40                10
observado     39.67235         9.568433

Balanço financeiro
Custo para desamassar:      R$ 0.30 por min.
Custo para pintar:         R$ 0.40 por min.
Preço por cliente:         R$ 32.00
Clientes atendidos: 1302   Clientes perdidos: 10269
Custo total: R$ 20479.26  Recebimento total: R$ 41664.00
Lucro total: R$ 21184.74
Lucro médio por cliente:   R$ 16.27092

```

Lucro médio por minuto: R\$ 0.353079

Recuperamos dessa forma o lucro que perdemos com a alta do dólar.

Suponhamos agora que queremos melhorar o atendimento procurando tentar atender o máximo de clientes em relação ao número de clientes antedidos + perdidos. Queremos também que o lucro não caia tanto. Após alguns testes foram encontrados os seguintes valores para os parâmetros:

```
> funil3 <- funilaria(1/15,1/30,1/10,600000)
> obj5 <- analise.desempenho(funil3,0.3,0.2,30)
> t(t(summary(factor(funil.list2$sistema))/funil.list2$n))

      [,1]
00 0.19317500
01 0.13108667
10 0.58578167
11 0.06938500
E1 0.02057167
```

Os parâmetros são:

- Chegada de clientes em média de 15 em 15 minutos (isso poderia ser obtido através de uma organização do atendimento para que fossem marcado 4 clientes por hora);
- Reduzimos o tempo de atendimento da funilaria para 1 cliente em média a cada meia hora;
- Mantemos o atendimento de 10 minutos para a pintura;
- Mantivemos todos os preços e custos iniciais.

Nota-se que como esperado, com a menor frequência de clientes e a melhoria do atendimento da funilaria houve uma diminuição do tempo de ocupação do sistema na parte da funilaria sendo de 58%. Houve, é claro então um aumento do tempo na pintura para 13%, e um aumento do tempo em que a Funilaria & Pintura fica vazia (19%). Mas houve ainda uma diminuição no tempo de alguém esperando na fila para a pintura (2%).

A saída da análise de desempenho foi a seguinte:

```
Analise de Desempenho do Sistema - Funilaria e Pintura

Modelo teórico
Chegadas: PPP(lambda = 0.06666667)
Tempo para funilaria: exp(mi1=0.03333333)
Tempo para pintura: exp(mi2=0.1)
Minutos simulados: 60000

Simulação
Tempo médio   Funilaria   Pintura
esperado      30          10
observado     29.96189   10.10869

Balanço financeiro
Custo para desamassar: R$ 0.30 por min.
```

```
Custo para pintar: R$ 0.20 por min.
Preco por cliente: R$ 30.00
Clientes atendidos: 1312 Clientes perdidos: 2654
Custo total: R$ 14445.52 Recebimento total: R$ 39360.00
Lucro total: R$ 24914.48
Lucro médio por cliente: R$ 18.98970
Lucro médio por minuto: R$ 0.4152413
```

Nota-se um aumento no lucro de quase 20% apenas pela melhoria no atendimento da funilaria e na organização de chegada de clientes. Não tivemos perda no número de clientes atendidos (1302 antes contra 1312 agora) e perdemos apenas 20% de clientes do que perdíamos com as velhas características, o que se ocorresse na vida real seria algo útil, pois com o tempo um número muito grande de perda de cliente poderia influenciar no sistema negativamente e uma redução da procura da funilaria ocorreria! Para concluir simulamos o mesmo aumento da pintura:

Análise de Desempenho do Sistema - Funilaria e Pintura

Modelo teórico

```
Chegadas: PPP(lambda = 0.06666667)
Tempo para funilaria: exp(mi1=0.03333333)
Tempo para pintura: exp(mi2=0.1)
Minutos simulados: 60000
```

Simulação

Tempo médio	Funilaria	Pintura
esperado	30	10
observado	29.96189	10.10869

Balanco financeiro

```
Custo para desamassar: R$ 0.30 por min.
Custo para pintar: R$ 0.40 por min.
Preco por cliente: R$ 30.00
Clientes atendidos: 1312 Clientes perdidos: 2654
Custo total: R$ 17098.04 Recebimento total: R$ 39360.00
Lucro total: R$ 22261.96
Lucro médio por cliente: R$ 16.96796
Lucro médio por minuto: R$ 0.3710327
```

Com isso novamente teríamos que aumentar o preço passado para o cliente que sendo bem atendido não recusaria em pagar.

Comentários

Não esgotamos todas as possibilidades de variação dos parâmetros, otimização de custos e possibilidades de modificação do sistema, mas pudemos verificar que o modelo computacional proposto foi adequado para a simulação do problema, visto que os resultados da simulação foram condizentes com as considerações teóricas estudadas.