Como eventos aleatórios acontecem?

Geração de um Processo de Poisson

ESTAT0090 – Estatística Computacional
Prof. Dr. Sadraque E. F. Lucena
sadraquelucena@academico.ufs.br

Cenário

Você é responsável por validar modelos estatísticos em um sistema onde eventos ocorrem aleatoriamente ao longo do tempo — como falhas em equipamentos, pagamentos de clientes ou chamadas em um call center. As bibliotecas prontas não oferecem suporte direto para essa simulação. Seu desafio é construir do zero um gerador de eventos que respeite o comportamento estatístico esperado. Como fazer isso com precisão, eficiência e controle?

Objetivos da aula

- Entender como eventos aleatórios se distribuem no tempo em um Processo de Poisson;
- Aprender duas formas de simular esse processo:
  - Pela geração dos tempos entre eventos com a distribuição Exponencial;
  - Pela simulação do número total de eventos com a distribuição de Poisson e alocação uniforme dos tempos.

O Que é um Processo de Poisson (Revisão Rápida)

- **O que é:** Contagem de eventos que ocorrem de forma aleatória e independente ao longo do tempo (ou espaço).
- **Exemplos:**
  - Chamadas telefônicas chegando em uma central.
  - E-mails chegando na sua caixa de entrada.
  - Defeitos que aparecem em um produto.
- **Parâmetro Principal:** λ (lambda) – taxa média de ocorrência de eventos por unidade de tempo.

A Propriedade Essencial para a Simulação

- Para conseguirmos simular a ocorrência desses eventos é crucial simularmos o tempo que decorre entre eles.
- **Exemplo:**
  - Se o primeiro e-mail chegou às 10:00 e o segundo às 10:05, o tempo entre eles foi de 5 minutos.
  - Se o terceiro chegou às 10:08, o tempo entre o segundo e o terceiro foi de 3 minutos.
  - São esses intervalos que nos interessam.
- Em um Processo de Poisson com taxa λ, os **tempos entre eventos sucessivos** (chamados de **tempos de interchegada**), denotados por X_i são:
  - Independentes uns dos outros;
  - Têm distribuição Exponencial(λ).

$$\overset{T_1}{\text{Início}} \mid \leftarrow \quad \rightarrow \mid \text{Evento 1} \mid \leftrightarrow \mid \overset{T_2}{\text{Evento 2}} \mid \longleftarrow \quad \longrightarrow \mid \overset{T_3}{\text{Evento 3}}$$

A Propriedade Essencial para a Simulação

- Como a distribuição Exponencial não tem memória, a duração do intervalo entre o primeiro e o segundo evento não influencia a duração do intervalo entre o segundo e o terceiro, e assim por diante.
- Cada “espera” até o próximo evento é uma nova “sorte” que segue uma distribuição exponencial.
- Essa taxa λ é a mesma taxa do processo de Poisson original.
  - Se o processo de Poisson tem uma taxa de $\lambda = 2$ eventos por minuto, o tempo médio entre os eventos será de $1/\lambda = 1/2 = 0{,}5$, ou seja, meio minuto (30 segundos).

Como Geramos os Tempos de Interchegada?

- Em geral as linguagens de programação geram ocorrências da distribuição exponencial.
- Se não houver uma função que gere os valores, use o método da inversão:

  1. Gere $u_i \sim$ Uniforme(0,1).
  2. Calcule o tempo de interchegada $T_i = -\frac{1}{\lambda}\log(u_i)$.

Obs.: Se a taxa de ocorrência dos eventos (λ) é grande, os tempos T_i tendem a ser pequenos.

Chegando aos Tempos dos Eventos

- A ocorrências T_i são os intervalos entre os eventos, mas o que queremos mesmo são os momentos no tempo em que cada evento acontece.
- Para isso, basta somarmos os tempos entre os eventos.
  - Se T_1 é o tempo até o primeiro evento (a partir do tempo 0), então o primeiro evento acontece em $S_1 = T_1$.
  - Se T_2 é o tempo depois do primeiro evento até o segundo, então o segundo evento acontece em $S_2 = T_1 + T_2$
  - E assim por diante! O tempo do j-ésimo evento, S_j, será a soma de todos os tempos de interchegada até aquele ponto, ou seja, $S_j = \sum_{i=1}^{j} T_i$.
- Essa é a forma de obter os tempos dos eventos em ordem crescente.

Gerando Eventos em um Intervalo de Tempo Fixo

- E se em vez de gerar um número fixo de n eventos, quisermos gerar todos os eventos que acontecem em um período de tempo específico, digamos, de 0 a T? Por exemplo, “gerar todas as chamadas que chegam entre 9:00 e 17:00”.

Algoritmo

Passo 1: Comece do zero.

- Faça `t=0` (`t` é o tempo atual, começamos no tempo zero)
- Faça `I=0` (`I` é o contador de eventos, ainda não aconteceu nenhum)

Passo 2: Calcule o Próximo Salto. Use um gerador da exponencial, ou faça

- Gere $u \sim$ Uniforme$(0, 1)$
- Calcule o tempo para o próximo evento: `tempo_proximo = - (1/lambda) * log(U)`
- Atualize o tempo atual: `t = t + tempo_proximo`

Gerando Eventos em um Intervalo de Tempo Fixo

Continuação do algoritmo

Passo 3: Verifique se Passou do Limite.

- Se `t > T`, pare. Isso significa que o próximo evento (que calculamos no Passo 2) cairia depois do nosso tempo limite `T`. Não nos interessa mais nada que aconteça depois de `T`.

Passo 4: Registre o Evento.

- `I = I + 1` (incrementamos o contador de eventos porque um novo evento acabou de acontecer)
- `T(I) = t` (registramos o tempo desse evento na nossa lista de tempos de evento)

Passo 5: Repita o processo.

- Volte para o Passo 2 e calcule o próximo tempo de interchegada.

Exemplo 13.1

Implementação no R:

```r
ProcessoPoisson <- function(lambda = 1, T = 10){
  t <- 0   # tempo interchegadas
  I <- 0   # número de ocorrências
  S <- vector()   # tempo das ocorrências

  while(t <= T){
    u <- runif(1)
    t <- t - (1/lambda)*log(u)
    I <- I + 1
    S[I] <- t
  }
  cat("Número de eventos:", I, "\n")
  return(S)
}
```

Exemplo 13.1

```
set.seed(12345)
tempos1 <- ProcessoPoisson(lambda = .5)
```

```
Número de eventos: 8
```

```
tempos2 <- ProcessoPoisson(lambda = 1)
```

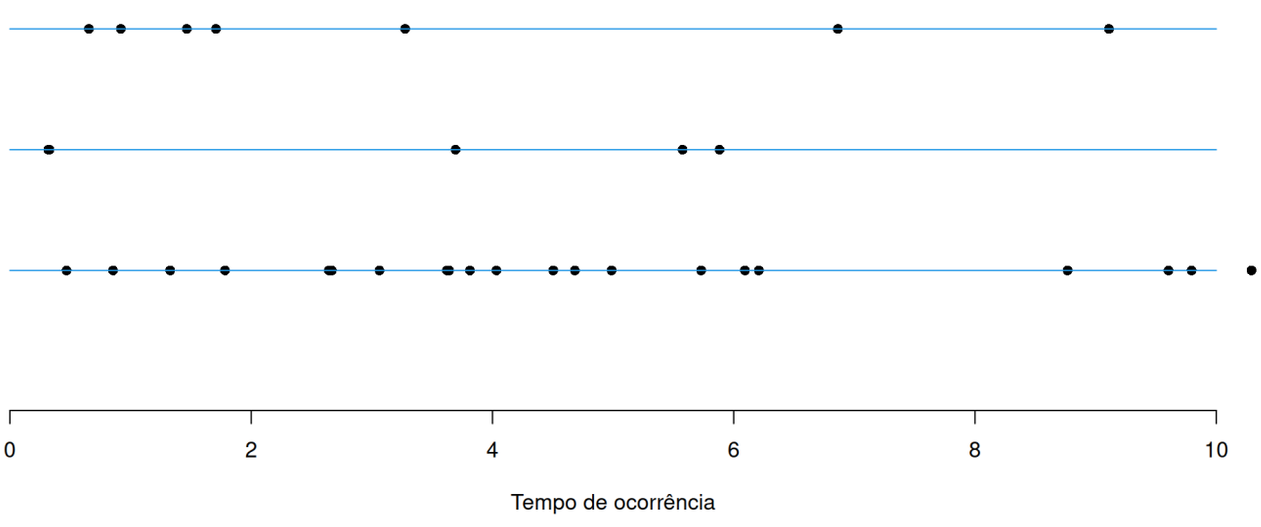
```
Número de eventos: 6
```

```
tempos3 <- ProcessoPoisson(lambda = 2)
```

```
Número de eventos: 21
```

Exemplo 13.1

```
par(mar=c(5,0,0,0))
plot(c(tempos1, tempos2, tempos3),
     c(rep(2, length(tempos1)), rep(1, length(tempos2)),
       rep(0, length(tempos3))), axes = F, ylim = c(-1, 3),
     xlim = c(0, 10), ylab = "", xlab = "Tempo de ocorrência",
     pch = 16)
segments(x0 = 0, y0 = 2, x1 = 10, y1 = 2, col = 4)
segments(x0 = 0, y0 = 1, x1 = 10, y1 = 1, col = 4)
segments(x0 = 0, y0 = 0, x1 = 10, y1 = 0, col = 4)
axis(1, at = c(0,2,4,6,8,10))
```



Outra Abordagem para simular o Processo de Poisson

- Em vez de simular os tempos entre eventos primeiro, vamos simular o número TOTAL de eventos que acontecem no intervalo de tempo T.
- Sabemos que o número de eventos N(T) em um processo de Poisson de taxa λ em um intervalo de tempo T segue uma distribuição de Poisson com média λT.
  - Ou seja, $N(T) \sim \text{Poisson}(\lambda T)$.

Passo 1: Simule esse N(T) para obter um valor, digamos, n. Agora sabemos que exatamente n eventos ocorreram entre 0 e T.

Onde os n Eventos Acontecem?

- Se sabemos que n eventos aconteceram em um intervalo [0,T], a localização desses n eventos dentro do intervalo [0,T] é uniformemente distribuída.
  - Pense assim: se 5 e-mails chegaram entre 9h e 10h, cada um deles tem a mesma chance de ter chegado em qualquer segundo desse intervalo.

Passo 2: 1. Como agora sabemos que n eventos aconteceram, gere n números aleatórios uniformes $u_1, u_2, \ldots, u_n$ (entre 0 e 1). 2. Para cada u_i, multiplique por T: $\{Tu_1, Tu_2, \ldots, Tu_n\}$. 3. Esses Tu_i são os tempos brutos em que os n eventos ocorreram dentro do intervalo $(0, T)$.

Dtealhe final: Ordenação

- Os tempos $\{Tu_1, Tu_2, \ldots, Tu_n\}$ que geramos no Passo 2 estão em ordem aleatória.
- Para ter a sequência dos eventos como normalmente desejamos (primeiro evento, segundo evento, etc.), precisamos ordená-los do menor para o maior.
  - **Exemplo:** Se você gerou [0.5T, 0.2T, 0.8T], a ordem correta seria [0.2T, 0.5T, 0.8T].

Passo 3: Ordene os valores $Tu_i, i = 1, \ldots, n$ para obter os tempos de cocorência dos eventos.

Eficiência dessa Abordagem

- Na primeira abordagem (Exponencial),fazíamos um laço que gerava exponenciais e somava até ultrapassar T. Se T fosse muito grande, ou λ muito grande, poderíamos gerar MUITOS X_i's.
- Na segunda abordagem, primeiro geramos apenas um número de Poisson para o total de eventos.
- Depois, geramos n uniformes e uma multiplicação simples.

- Evitamos cálulos complexos (logaritmos) e laços longos.

Exemplo 13.2

Implementação no R da segunda abordagem:

```r
ProcessoPoisson_abordagem2 <- function(lambda = 1, T = 10){
  # Passo 1: Simular o número total de eventos N(T)
  # N(T) segue uma distribuição de Poisson com média (lambda * T)
  num_eventos <- rpois(1, lambda * T)

  # Se nenhum evento ocorreu, retorna uma lista vazia
  if (num_eventos == 0) {
    cat("Número de eventos:", 0, "\n")
    return(numeric(0)) # Retorna um vetor numérico vazio
  }

  # Passo 2: Gerar os tempos dos eventos
  # Geramos 'num_eventos' números aleatórios uniformes entre 0 e 1
  U <- runif(num_eventos)

  # Multiplicamos por T para escalá-los para o intervalo (0, T)
  tempos_nao_ordenados <- T * U

  # Ordenamos os tempos para obter a sequência cronológica dos eventos
  tempos_eventos <- sort(tempos_nao_ordenados)
```

```
  cat("Número de eventos:", num_eventos, "\n")
  return(tempos_eventos)
}
```

Exemplo 13.2

```
set.seed(12345)
eventos_simulados <- ProcessoPoisson_abordagem2(lambda = 1, T = 10)
```

```
Número de eventos: 11
```

```
eventos_grandes <- ProcessoPoisson_abordagem2(lambda = 5, T = 20)
```

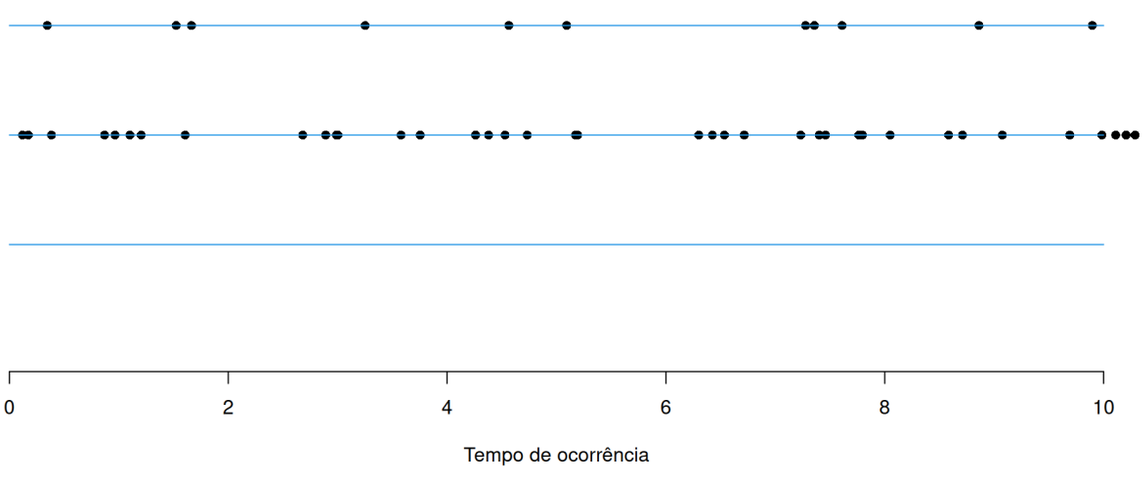
```
Número de eventos: 69
```

```
eventos_raros <- ProcessoPoisson_abordagem2(lambda = .1, T = 1)
```

```
Número de eventos: 0
```

Exemplo 13.3

```
par(mar=c(5,0,0,0))
plot(c(eventos_simulados, eventos_grandes, eventos_raros),
     c(rep(2, length(eventos_simulados)),
       rep(1, length(eventos_grandes)),
       rep(0, length(eventos_raros))), axes = F, ylim = c(-1, 3),
     xlim = c(0, 10), ylab = "", xlab = "Tempo de ocorrência",
     pch = 16)
segments(x0 = 0, y0 = 2, x1 = 10, y1 = 2, col = 4)
segments(x0 = 0, y0 = 1, x1 = 10, y1 = 1, col = 4)
segments(x0 = 0, y0 = 0, x1 = 10, y1 = 0, col = 4)
axis(1, at = c(0,2,4,6,8,10))
```



Exemplo 13.3

Como cientista de dados em uma empresa de desenvolvimento de software, você está testando um novo módulo crítico. A experiência anterior mostra que bugs sérios (que causam falhas no sistema) aparecem a uma taxa de 0.02 bugs por hora de uso contínuo. Para certificar a confiabilidade do módulo antes do lançamento, a equipe de engenharia precisa saber quantas horas de teste contínuo são esperadas até que os 5 primeiros bugs sérios sejam descobertos. Essa informação é crucial para planejar a duração da fase de testes.

Sua Tarefa:

- Simule o tempo necessário para a descoberta dos 5 primeiros bugs sérios.
- Utilize a Abordagem 1 (Tempos de Interchegada).
- A taxa (λ) é de 0.02 bugs por hora.

Exemplo 13.4

Você trabalha em uma instituição de microcrédito. A instituição concede pequenos empréstimos e observa que os pagamentos de clientes (que são eventos de entrada de caixa) ocorrem a uma taxa média de 8 pagamentos por dia útil. A diretoria está planejando o fluxo de caixa para a próxima semana de 5 dias úteis e precisa de uma projeção dos momentos (dias e frações de dia) em que todos os pagamentos esperados ocorrerão nesse período. Essa projeção ajudará a otimizar a alocação de fundos.

Sua Tarefa:

- Simule os horários de chegada de todos os pagamentos de clientes durante a próxima semana de 5 dias úteis.
- Utilize a Abordagem 2 (Total de Eventos + Uniforme).
- A taxa (λ) é de 8 pagamentos por dia.
- O “tempo” total (T) é de 5 dias úteis.
- Apresente os horários de chegada dos pagamentos (em dias, a partir do início da semana), em ordem cronológica. Por exemplo, 0.5 para meio-dia do primeiro dia, 1.25 para um quarto do segundo dia, etc.

Ganho da aula

- Capacidade de simular um Processo de Poisson do zero em qualquer linguagem de programação;
- Compreensão intuitiva da relação entre tempo, taxa de ocorrência (λ) e aleatoriedade;
- Aplicação prática de simulação para resolver problemas reais.

Fim

Aula baseada no material “Métodos Computacionais Aplicados à Estatística Implementação no Software R” de Cristiano de Carvalho Santos.