Formulário [ME]

* Regras de derivação

$(u+v)' = u' + v'$

$(u\,v)' = u'\,v + u\,v'$

$\left(\frac{u}{v}\right)' = \frac{u'\,v - u\,v'}{v^2}$

$(u^n)' = n\,u^{n-1}\,u' \quad (n \in \mathbb{R})$

$(\text{sen}\,u)' = u'\cos u$

$(\cos u)' = -u'\,\text{sen}\,u$

$(\text{tg}\,u)' = \frac{u'}{\cos^2 u} = \sec^2(u)\,u'$

$(\arctan(u))' = \frac{u'}{1+u^2}$

$(\sec u)' = \sec(u)\tan(u)\,u'$

$(\arcsin(u))' = \frac{u'}{\sqrt{1-u^2}}$

$(e^u)' = u'\,e^u$

$(a^u)' = u'\,a^u \ln a \quad (a \in \mathbb{R}^+ \setminus \{1\})$

$(\ln u)' = \frac{u'}{u}$

$(\log_a u)' = \frac{u'}{u \ln a} \quad (a \in \mathbb{R}^+ \setminus \{1\})$

$h = f \circ g \mid h'(a) = f'(g(a)) \times g'(a)$

* Primitivas

$\int \sin^2 x\,dx = \frac{1}{2}x - \frac{\sin 2x}{4}$ $\quad \int \cos^2 x\,dx = \frac{1}{2}x + \frac{\sin 2x}{4}$

f(x)	∫ f(x) dx
k	$kx + C, \quad k \in \mathbb{R}$
$u^n u'$	$\frac{u^{n+1}}{n+1} + C, \quad n \in \mathbb{R}\setminus\{-1\}$
$\frac{u'}{u}$	$\ln\lvert u\rvert + C$
$e^u u'$	$e^u + C$
$a^u u'$	$\frac{a^u}{\ln(a)} + C, \quad a \in \mathbb{R}^+\setminus\{1\}$
$\frac{u'}{1+u^2}$	$\arctan(u) + C$
$\frac{u'}{\sqrt{1-u^2}}$	$\arcsin(u) + C$

f(x)	∫ f(x) dx
$\sin(u)\,u'$	$-\cos(u) + C$
$\cos(u)\,u'$	$\sin(u) + C$
$\tan(u)\,u'$	$-\ln\lvert\cos(u)\rvert + C$
$\cot(u)\,u'$	$\ln\lvert\sin(u)\rvert + C$
$\sec^2(u)\,u'$	$\tan(u) + C$
$\csc^2(u)\,u'$	$-\cot(u) + C$
$\sec(u)\,u'$	$\ln\lvert\sec(u) + \tan(u)\rvert + C$

Parâmetros de Variáveis Aleatórias

Valor Esperado de X ($\boldsymbol{\mu_X}$) $E[X] = \mu_X = \sum_{i=1}^{n} x_i f(x_i)$ $\qquad E[X] = \int_{-\infty}^{+\infty} x f(x)dx$

Variância de X ($\boldsymbol{\sigma_X^2}$) → Desvio Padrão de X ($\boldsymbol{\sigma_X}$) $\qquad Var[X] = \int_{-\infty}^{+\infty} (x-\mu)^2 f(x)dx$

DISCRETAS (Suporte Finito ou Infinito Numerável)

♦ Função (Massa) de Probabilidade → $f(x) = P[X = x]$

◊ Função de Distribuição (de Probabilidade) → $F(x) = P[X \le x] = \sum_{x_k \le x} f(x_k)$

1. Distribuição **Binomial** $\quad X \sim Bi(n,p)$ (Nº de sucessos em n provas de Bern)

$f(x) = \binom{n}{x} p^x (1-p)^{n-x}, \quad x = 0, 1, \dots, n \qquad E[X] = np \qquad Var[X] = np(1-p)$

2. Distribuição **Bernoulli** (Binomial de $n = 1$) $\quad X \sim Bern(p)$ (Moeda)

Provas de Bern A - Sucesso (p) $\quad \bar{A}$ - Insucesso ($1-p$)

$f(x) = p^x (1-p)^{1-x}, \quad x = 0, 1 \qquad E[X] = p \qquad Var[X] = p(1-p)$

3. Distribuição **Bin Neg** $\quad X \sim BinNeg(k,p)$ (Nº de PdeB a realizar até obter k suce.)

$f(x) = \binom{x-1}{k-1} p^k (1-p)^{x-k}, \quad x = k, k+1, \dots \qquad E[X] = \frac{k}{p} \qquad Var[X] = \frac{k(1-p)}{p^2}$

Quando $\underline{k = 1}$, temos a Distribuição **Geométrica** - nº de PdeB até realizar o 1º sucesso

4. Distribuição de **Poisson** $\quad X \sim Po(\lambda)$ (Contagem em Intervalos de Natureza Contínua)

$f(x) = \frac{e^{-\lambda}\lambda^x}{x!}, \quad x = 0,1,2,\dots \qquad E[X] = Var[X] = \lambda$

5. Distribuição **Uniforme** $\quad X \sim U(n)$ (Todos os valores são equiprováveis) $\quad f(x) = \frac{1}{n}$

CONTÍNUAS (Suporte Contínuo)

~~Função (Massa) de Probabilidade~~ → Não faz sentido! $\quad P[X = x] = 0$

◊ Função de Densidade de Probabilidade $\quad f(x) \quad$ **p.d.f.** Probabilidade de x assumir um valor específico

◊ Função de Distribuição Comulativa $\quad F(x) \quad$ **c.d.f.** Probabilidade **Acumulada** de x ser ≤ a um valor esp.

$f(x) \ge 0 \qquad \int_{-\infty}^{+\infty} f(x)\,dx = 1 \qquad F(x) = P[X \le x] = \int_{-\infty}^{x} f(x)dx$

1. Distribuição **Uniforme** $X \sim U(a,b) \quad E[X] = \frac{a+b}{2} \quad Var[X] = \frac{(b-a)^2}{2}$

$f(x) = \frac{1}{b-a}$, $a \le x \le b$ (é constante entre a e b) ; e 0 c.c.

$F(x) = \frac{x-a}{b-a}$, $a \le x \le b$; 0 em $x < a$; e 1 em $x > b$

2. Distribuição **Normal** $X \sim N(\mu, \sigma) \qquad f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$

$Z = \frac{X-\mu}{\sigma} \sim N(0,1)$. (z-score)

TAN $(X \pm Y) \sim N\left(\mu_X \pm \mu_Y, \sqrt{\sigma_X^2 + \sigma_Y^2}\right)$

$\bar{X} = \frac{\sum_{i=1}^{n} X_i}{n} \sim N\left(\mu_{\bar{X}} = \mu, \sigma_{\bar{X}} = \frac{\sigma}{\sqrt{n}}\right)$

3. Distribuição **Exponencial** $\quad X \sim Exp\left(\frac{1}{\lambda}\right)$

(Intervalos de Tempo entre 2 ocorrências)

$f(x) = \lambda e^{-\lambda x}$ e $F(x) = 1 - e^{-\lambda x}$, $x \ge 0 \qquad \lambda$ é a **média da distribuição**

$E[X] = \frac{1}{\lambda} \quad Var[X] = \frac{1}{\lambda^2}$

Propriedades Valor Esperado $E[X]$

- Sendo X e Y duas v.a's e k, a e b constantes $\mathbb{R}$,
  - $E[k] = k$
  - $E[kX] = kE[X]$
  - $E[X \pm Y] = E[X] \pm E[Y]$
  - $E[aX + bY] = aE[X] + bE[Y]$
  - $E[XY] = E[X] \times E[Y]$, se X e Y forem independentes

Propriedades da $Var[X]$

- $Var(k) = 0$
- $Var(X + k) = Var(X)$
- $Var(kX) = k^2 Var(X)$
- $Var(X \pm Y) = Var(X) + Var(Y) \pm 2Cov(X,Y)$
- $Cov(aX, bY) = ab\,Cov(X,Y)$

$Var[X] = E[(X - E[X])^2] = E[(X - \mu_x)^2] = E[X^2] - (E[X])^2 = \sigma_x^2$

♦ Se X e Y forem **independentes**, então $Cov(X,Y) = 0$

♦ Se X é uma v.a. de média μ e variância σ^2 Então a v.a. $W = \frac{(X-\mu)}{\sigma}$ tem $E[W] = 0$ e $Var(W) = 1$

Intervalos de Confiança

$]I_{0.95}[_\mu = \bar{X} \pm erro = \bar{X} \pm \frac{1.96}{\sqrt{n}}\sigma$

$]I_{0.99}[_\mu = \bar{X} \pm erro = \bar{X} \pm \frac{2.576}{\sqrt{n}}\sigma$

```
# Preparar o espaço
conf<-0.95          # nível de confiança          q<- qnorm(conf+cauda/2)
cauda<-1-conf       # peso das caudas             # qnorm(0.975) = 1.96  = -qnorm(0.025)

# Cálculo do Intervalo Concreto
media<- mean(bd_oculos_sol$Pub.Mk)                LB  <- media-erro
dp   <- sd(bd_oculos_sol$Pub.Mk)                  UB  <- media+erro
erro <- q*dp/sqrt(nrow(bd_oculos_sol))            round(c(LB,UB),2)
```

1. Simulação em CD

Tipos de Simulação | Estocástica vs. Determinística

- Um modelo é determinístico se seu comportamento for totalmente previsível. Dado um conjunto de *inputs*, o modelo resultará num conjunto único de *outputs*.
- Um modelo é estocástico se tiver **variáveis aleatórias** como *input e outputs*.

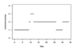
Tipos de Simulação | Estática vs. Dinâmica

- Os modelos estáticos representam o sistema num determinado momento | MC
- Os modelos dinâmicos representam os sistemas à medida que evoluem ao longo do tempo. (P.e., simular a Loja de *Croissants* durante o horário de funcionamento)

Simulação (Dinâmica) Discreta vs Contínua

- Na **Simulação** Discreta, as var. mudam apenas num conjunto discreto de pontos no tempo.

Ex: O nº de pessoas na fila da loja de croissants é um exemplo de simulação discreta. O nº de clientes muda apenas quando um novo cliente chega ou quando um cliente é atendido.



- Na **Simulação** Contínua, as variáveis de interesse mudam continuamente ao longo do tempo.

Ex: Viagem de carro foi criado onde o interesse está na velocidade do carro ao longo da viagem.

2. Geração de NPA

NAs → [1º Req.] Ser capazes de **simular de forma independente** nºs aleatórios $u_1, \dots, u_N$ de $U(0,1)$.
[2º Requisito] Os números $u_1, u_2, \dots, u_N$ necessitam de respeitar a **independência**.

∴ Devem ser realizações de **v.a. iid** e devem ser **não preditíveis**, i.e., não se deve conseguir prever o próximo nº a gerar a partir dos nºs já gerados.

NPAs → São sempre de tipo determinístico e com *output* aparentemente aleatório.
Devem ser: **Rápidos** de gerar | **Ciclos de nº longos** | **Reprodutibilidade** `set.seed(n)`
Os nºs aleatórios devem ser independentes e uniformemente distribuídos.

Qualidade dos NPAs

- Depende da semente
- Periodicidade
- Independência

Uma forma usual de testar NPA é aplicar testes estatísticos de **Uniformidade**

1. **Histograma** com a curva da função densidade de probabilidade
2. Teste de hipóteses - **Teste de Kolmogorov-Smirnov**

e de **Independência** às amostras geradas.

- Teste à autocorrelação de **Box-Pierce** com representação gráfica da autocorrelação.

- Os NPAs gerados pelos métodos seguintes são **não correlacionados**, ou seja, **independentes**. Apenas nos **MCMC** serão correlacionados.

Método Congruencial Linear de Lehmer (GNPA Uniformes)

◈ O algoritmo de Lehmer gera a sequência de inteiros $X_n \in 0,1,\cdots,m-1$ através de $X_n \neg (aX_{n-1} + c)\ mod\ m$, onde $\boldsymbol{mod}\ m$ (módulo m) **é o resto da divisão inteira por** m. [5mod2 = 1 e 4mod2 = 0].

◈ Gera-se inteiros entre 0 e $m - 1$. ◈ Nº aleatórios entre 0 e 1 podem ser obtidos através de $u_n = \frac{x_n}{m}$.

Algoritmo

Input: $m > 1$ (módulo),
$a \in \{1,2,\dots,m-1\}$ (constante multiplicativa)
$c \in \{0,1,\dots,m-1\}$ (incremento)
$X_0 \in 0,1,\dots,m-1$ (semente)

Output: $X_1, X_2, X_3 \dots$ uma sequência de NPA

1. for $n = 1,2,3,\dots$ do
2. $X_n \in (aX_{n-1} + c)$ mod m
3. output X_n
4. end for

Obs.: a expressão x congruente (y) módulo (m) significa que x é o resto da divisão inteira de y por m

No caso específico deste gerador, X_n congruente a $(aX_{n-1} + c)$ modulo m significa que X_n é o resto da divisão inteira de $(aX_{n-1} + c)$ por m.

◈ O output depende de 4 parâmetros: m (módulo), a (const. multiplicativa), c (incremento), X_0 (semente);

◈ Caso sejam escolhidos convenientemente a sequência resultante comporta-se de forma semelhante ao de uma sucessão de **v.a. iid uniformes**.

◈ Dado que X_n pode assumir apenas m valores distintos o *output* de um gerador congruencial começa a repetir-se depois de m passos, no máximo - é **periódico**. Para ultrapassar isso, $m = 2^{32}$ ou $m = 2^{64}$.

Ex.: `[1] 0.0625 0.8125 0.5625 0.3125 0.0625 0.8125 0.5625 0.3125` $\quad m = 64, a = 13, c = 0, X_0 = 4$

- Comprimento do Ciclo = **Período** = 4 → Não escolhia este a e c porque resulta num ciclo curto

Método da Transformação Inversa $F(X)^{-1}$ (GNPA Não Uniformes)

◎ Gera v.a. usando como *input* os valores NPAs uniformes em [0, 1].

◎ Transforma os NPAs de uma $U(0,1)$ noutra distribuição, tirando vantagem do facto da inversa da função distribuição ser tmb definida no intervalo [0,1].

Algoritmo

Input: F^{-1} (Inversa de F)
Output: $X \sim F$

1. $X = F^{-1}(U)$

◎ Aplicado para gerar observações de **v.a.** contínuas. Mas não pode ser usado diretamente para normais.

◎ **Não pode ser usado diretamente para simular v.a.** discretas. Só após uma adaptação simples.

Inverter uma Função

Se X é uma v.a. com distribuição Exponencial de parâmetro λ então a **cdf** é $F(x) = 1 - e^{-\lambda x}$, para $x \ge 0$.

$1 - e^{-\lambda x} = u$ em ordem a x. Ou seja:

$$1 - e^{-\lambda x} = u$$
$$e^{-\lambda x} = 1 - u$$
$$-\lambda x = log(1 - u)$$
$$x = -(1/\lambda) \times log(1 - u)$$

Então $\quad F^{-1}(u) = \frac{-log(1-u)}{\lambda}$.

1. Gerar $u \sim U(0,1)$ e substituir na função $F^{-1}(x)$

R: Pode-se aplicar sempre que $F(x)$ tem função analítica **conhecida** e **invertível**.
No caso de ser uma dist. contínua pode-se aplicar diretamente, mas no caso de ser uma dist. discreta tem de ser mediante artifícios (não é diretamente e é em apenas alguns casos).

Método da Aceitação/Rejeição

⩓ É baseado na função **densidade de probabilidade** (pdf), em vez da função distribuição. Logo, se a pdf de X for conhecida, $f(x)$, então é possível gerar NPA de X pelo **método da aceitação-rejeição**.

⩓ Método que gera amostras de uma distribuição de interesse, $f(x)$,a partir da geração de candidatos de uma distribuição conhecida, $g(x)$, e rejeitando um subconjunto dos candidatos gerados.

⩓ O que se deve garantir é que a curva de g tem que estar sempre "acima" da curva de f.
Daí, multiplicamos g por uma constante, c, de forma a garantir que, $c \times g(x) \ge f(x)$
Logo, em média, cada valor amostral de X necessita de c iterações

⩓ Aceitamos x com probabilidade $\frac{f(x)}{c \times g(x)}$ $\quad$ Note-se que o quociente varia entre 0 e 1 ($c * g(x)$ é sempre maior ou igual a $f(x)$), o que permite que este quociente seja interpretado como uma frequência relativa (probabilidade).

Se $\frac{f(x)}{c \times g(x)}$ for **grande**: $\quad f(x)$ grande - nº muito provável na distribuição alvo ou de interesse; $g(x)$ pequeno - nº pouco provável na distribuição de candidatos

- Aceita-se esse número porque podemos não ter novamente proposta de um número tão raro (a sua P de ocorrência é baixa na dist. de candidatos) e ele tem P elevada na distribuição alvo.

Se $\frac{f(s)}{c \times g(s)}$ for **pequeno**: $\quad f(x)$ pequeno - nº pouco provável na distribuição alvo ou de interesse; $g(x)$ grande - nº muito provável na distribuição de candidatos

- Rejeita-se esse número porque este tem grande probabilidade de aparecer como candidato, mas não é comum (ou seja, tem probabilidade, P, baixa) na distribuição alvo.

Algoritmo - R

- O nº de gerações de $g(x)$ necessários para obter um valor gerado $f(x)$ tem **dist. Geométrica** com valor esperado c.
- c é designada por Constante de Rejeição; convém então escolher c através de $c = sup\ f(x)/g(x)$
- Prob. Total de Aceitação p/ qualquer iteração = $\sum_x P(\text{aceitar} \mid x) * P(X = x) = \sum_x \frac{f(x)}{c \times g(x)} * g(x) = \frac{1}{c}$

Outras Transformações

◈ Pretende-se gerar NPAs de uma dist., com base numa relação existente com outra dist. conhecida.
Ex.: Gerar valores de uma distribuição $\log N(\mu,\sigma)$ a partir de valores gerados de uma $N(\mu,\sigma)$. [**1.** ✓]

Transformações Importantes

1. Se $Y \sim N(\mu,\sigma)$, usando a transformação $X = e^Y$, X tem distribuição $\log N(\mu,\sigma)$, pois $\log X = Y$
2. Se $Z \sim N(0,1)$, então $V = Z^2 \sim \chi^2(1)$.
3. Se $U \sim \chi^2(m)$ e $V \sim \chi^2(n)$ são indp., então $F = \frac{U/m}{V/n}$ tem **dist.** F **com** $(\boldsymbol{m}, \boldsymbol{n})$ **graus de liberdade**.
4. Se $Z \sim N(0,1)$ e $V \sim \chi^2(n)$ são indp., então $T = \frac{Z}{\sqrt{V/n}}$ tem **dist.** t **de Student com** $\boldsymbol{n}$ **gdl**.
5. Se $U,V \sim U(0,1)$ são indp., então $Z_1 = \sqrt{-2\log U}\cos(2\pi V)$ e $Z_2 = \sqrt{-2\log U}\sin(2\pi V)$ são 2 v.a. indp.$\sim N(0,1)$.
6. Se $U \sim Gama(r,\lambda)$ e $V \sim Gama(s,\lambda)$ são indp., então $X = U/(U+V)$ tem **dist.** $\boldsymbol{Beta(r,s)}$.
7. Se $U,V \sim U(0,1)$ são indp., então $X = \lfloor 1 + log(V)/log(1-(1-q)^U) \rfloor$ tem **dist.** $\boldsymbol{Logarítmica(q)}$, onde $\lfloor x \rfloor$ denota a parte inteira (arredondada para baixo) de x. `floor()`
8. $U \sim U(0,1)$, então $X = -(1/\lambda)logU$ terá **distribuição** $\boldsymbol{Exp(\lambda)}$.

Somas/Convoluções e Misturas | Tipos Especiais de Transformações.

Somas Importantes

1. Se $Z_1, \dots, Z_n \sim N(0,1)$, então $V = \sum_{i=1}^{n} Z_i^2 \sim \chi^2(n)$
2. A dist. binomial negativa $BinNeg(r,p)$ pode ser definida como a convolução de r va iid $Geom(p)$.
3. A soma de $\boldsymbol{r}$ v.a. $Exp(\lambda)$ indp. tem dist. $Gama(r,\lambda)$ → A soma de grau r da $Exp(\lambda)$ é uma $Gama(r,\lambda)$.
4. A soma de n va iid $Ber(p)$ tem dist. $Bin(n,p)$.

\+ A soma de $X_1 \sim Pois(\lambda_1)$ e $X_2 \sim Pois(\lambda_2)$ dá $X \sim Pois(\lambda_1 + \lambda_2)$

- Uma v.a. X é uma **mistura discreta** se $F_X(x) = \sum_{i=1}^{k} \theta_i F_i(x)$, com $\theta_i > 0$ e $\sum_{i=1}^{k} \theta_i = 1$
  Os valores θ_i são designados por **pesos** ou **probabilidades da mistura**.

- E mistura **contínua** se $F_X(x) = \int_{-\infty}^{+\infty} F_{X|Y=y}(x) f_Y(y)dy$, com $f_y > 0$ e $\int_{-\infty}^{+\infty} f_Y(y)dy = 1$
  f_y é designada por **função de mistura**.

3. Método de Monte Carlo em Inferência Estatística

Estimação Pontual | Propriedades dos Estimadores

◊ Seja T, estimador para θ.

- T diz-se não enviesado se $E[T] = \theta$. $\quad$ Ou seja, o "- disperso" ou "+ concentrado" em torno do parâmetro que estima
- T diz-se eficiente se, entre todos os estimadores não enviesados para θ, for o que apresenta $\boldsymbol{min}(Var)$
- T diz-se suficiente se utiliza toda a informação disponível na amostra, relevante para a estimação de θ.

EP | Propriedades Assintóticas dos Estimadores

- T diz-se não enviesado assintoticamente (ou **assintoticamente centrado**) se $\lim_n E[T_n] = \theta$
- T diz-se consistente em média quadrática se, $\lim_n E[(T_n - \theta)^2] = 0 \Leftrightarrow \lim_n (Var(T_n) + (E[T_n - \theta])^2) = 0 \Leftrightarrow \lim_n (Var(T_n)) = 0 \wedge \lim_n (E[T_n - \theta]) = 0$
- T diz-se assintoticamente mais eficiente se for aquele que apresenta Var mínima, de entre todos os consistentes em média quadrática.

- SMC (= **MC Paramétrico** - necessita de um modelo - gera amostras da f.d.p. da dist. da pop.)
  Usa amostragem repetida para obter as propriedades estatísticas de algum fenómeno.
- Permite estimar a potencia de um teste de hipóteses
- Os métodos de Monte Carlo variam, mas tendem a seguir um padrão específico:

Algoritmo

1. Definir um domínio de possíveis *inputs*
2. Gerir os *inputs* aleatoriamente a partir de uma dist. de probabilidade
3. Executar um cálculo determinístico nos inputs
4. Agregar os resultados

A distribuição amostral de $\hat{\theta}^{(j)}$ deve ser próxima da distribuição amostral verdadeira para N grande. A média da distribuição será então uma **estimativa pontual** para θ.

$$\hat{\theta}_{MC} = \frac{1}{N}\sum_{j=1}^{N} \hat{\theta}^{(j)}$$

Estimação do Erro Padrão

⩓ O erro padrão de um estimador é o seu **desvio-padrão**. Sabemos que a distribuição de $\bar{X}$ tem média μ e variância σ^2/n. Então o erro padrão de $\bar{X}$ é $\quad SE(\bar{X}) = \sqrt{Var(\bar{X})} = \sqrt{\frac{\sigma^2}{n}} = \frac{\sigma}{\sqrt{n}}$

Estimação do MSE [Mean Squared Error = $\frac{\sum(y_i - \hat{y}_i)^2}{n}$]

⩓ Definir o **valor esperado**, y_i, e calcular a média dos erros quadrados

4. MCMC (Método de MC via Cadeias de Markov)

AS

- Todos os **Métodos MCMC** (**M-H**, **P.A.**, **A.I.**, ...) são baseados no **Método da Aceitação/Rejeição**.
- Nos **Métodos MCMC** pode-se retirar uma **amostra de um número** $\boldsymbol{m < n}$ dos valores gerados para obter uma **amostra independente**.

Cadeias de Markov

Diz-se **aperiódica** se todos os estados tiverem período 1. **(Não basta terem o msm período)**

- Constituem uma classe de algoritmos para simular **distribuições multivariadas** complexas.
- Baseiam-se na construção de uma **CM homogénea** que tem como **dist. lim a que se pretende simular.**

- Um Processo de Markov (PM) é um **sistema** que possui **estados**, e que os alterna com uma dinâmica.
- Todos os estados formam um **espaço de estados**, que geralmente, é finito (embora possa ser grande).
- Uma sequência de estados chama-se **cadeia**.
  Uma sequência de observações do sistema ao longo do tempo formam **cadeias de estados**, o que é chamado de **história**.

Propriedades de Markov

1. A **propriedade de Markov** afirma que a dinâmica futura do sistema de um qualquer estado depende apenas desse estado. → Cada estado é independente (Falta de Memória)
2. **Estacionaridade:** As probabilidades de transição entre estados não mudam com o tempo. $T = N \times N$

CM | É uma sequência de v.a. dependentes, onde a prob. condicional de X_t depende apenas de X_{t-1}.

$$P(X_t = x_j \mid X_{t-1} = x_{it-1}, \dots, X_2 = x_{i2}, X_1 = x_{i1}) = P(X_t = x_j \mid X_{t-1} = x_{it-1})$$

- **Kernel de Transição/Markov** (K) é a distribuição condicional de X_{t+1} dado X_t (é P condicional)
  A distribuição condicional de $X_t|X_{t-1}$ é o **Kernel de transição** $\boldsymbol{K}$ e tem-se $X_{t+1}|X_t, X_{t-1}, \cdots, X_2, X_1, X_0 \sim K(X_{t+1}|X_t)$
- **Distribuição Estacionária:** Geralmente têm uma distribuição estacionária.
  Se $X_t \sim f$ então $X_t + 1 \sim f$ tendo-se $\int \chi K(y \mid x) f(x)dx = f(y)$
- **Irredutibilidade:** significa que o *kernel* permite movimentos em todo o espaço de estados.
  Para qualquer X_0, a sucessão $\{X_t\}$ tem uma P positiva de visitar qualquer região do espaço de estados.
- **Recorrência:** implica que elas retornam infinitas vezes a qualquer estado.
- **Ergodicidade:** A distribuição estacionária é também uma distribuição limite, e a cadeia converge para o valor esperado, independentemente do valor inicial.
  Se f é a distribuição de probabilidade limite então $X_t \to X \sim f$, para qualquer valor inicial X_0
- Para funções integráveis h, e desde que exista o valor médio $E[h(X)]$,
  $\frac{1}{T}\sum_{t=1}^{T} h(X_t) \xrightarrow{P} E[h(X)]$, quando $T \to \infty$

Lei dos Grandes Números (por vezes chamado **Teorema Ergódico**)
Isto significa que uma CM converge para o valor esperado, se se repetir continuamente gerando independentemente series de CM.

Métodos MCMC

- Da existência de um processo estacionário sabe-se que, independentemente do valor inicial, x_0, uma amostra da distribuição estacionária é gerada, desde que o período seja longo o suficiente.
- Fase de burn-*in* (fase de aquecimento) - o MCMC precisa de um certo tempo para produzir valores de uma dist. estacionária, portanto, os primeiros valores pertencentes à fase de *burn-in* serão excluídos.
  NOTA: A dimensão adequada do período de *burn-in* **depende** da **variância da dist. proponente.**
- Cada "região" pode ser alcançada com P + para a região definida.
  E com uma infinita cadeia de Markov imaginária, pode-se voltar a qualquer ponto (recorrência).
- Enquanto uma cadeia for recorrente, o MCMC converge.
  Mas um MCMC muitas vezes leva muito tempo para convergir, por isso é sempre necessário avaliar a velocidade de convergência.

A ideia de base dos métodos MCMC para obter amostras de uma distribuição alvo f é

- Construir uma cadeia de Markov com distribuição estacionária f , e
- Simular essa cadeia por um longo período de tempo, até que ela convirja (aproximadamente) para a sua distribuição estacionária.
- A cadeia vai gerar a **amostra da sua distribuição estacionária, que é a distribuição alvo** f.

Desvantagem

− O pressuposto de i.i.d é violado dado que o próx. valor simulado depende sempre do valor anterior. → As amostras são **correlacionadas**.

Vantagens

\+ Uma amostra com valores correlacionados não é desejável, mas mesmo assim, os MMCMC são preferidos em situações mais complexas:

- Mesmo com valores correlacionados, é possível selecionar uma (sub)amostra de valores que não seja correlacionada.
- As **cadeias de Markov possuem diferentes propriedades de convergência**, que podem ser exploradas para se obter dist. propostas + fáceis de tratar numericamente.
- O conhecimento necessário da distribuição alvo f de onde se quer gerar os números é mínimo.
- Estes métodos facilitam a resolução de problemas de alta dimensão (contrariamente ao método de Aceitação-Rejeição), através de uma sequência de problemas menores que são mais fáceis de resolver (e.g. amostrador de Gibbs).

Algoritmo Metropolis-Hastings

- Gera variáveis **correlacionadas** a partir de uma **CM**. Note-se que até agora, temos gerado **variáveis iid**
- Existe um valor candidato Y, gerado a partir de uma **dist. candidata** $g(y|X_t)$ e se este valor candidato:
  - **é aceite**, a cadeia move-se para o estado Y no tempo $t+1$ e $X_{t+1} = Y$
  - **não é aceite**, a cadeia permanece no estado X_t e $X_{t+1} = X_t$. $\quad$ Logo, repete valores!
- As condições exigidas para a cadeia gerada são:
  - **Irredutibilidade**: uma $P \neq 0$ de transição (mesmo que ocorra em mais de 1 etapa) de qualquer estado para qualquer outro estado
  - **Recorrência positiva**: se o tempo esperado até o retorno for finito
  - **Aperiodicidade:** não deve haver estados absorventes (estados em que uma vez inseridos não podem mais ser deixados)
- O algoritmo GERAL de **M-H** gera uma cadeia de Markov $X_0, X_1, \dots$ de acordo com:

1. Defina uma distribuição candidata $g(.|X_t)$ $\quad$ (Tem de ter domínio/suporte = ao de f → D **Importa p/ eficiência**)
2. Defina um valor inicial X_0, dentro do domínio de g $\quad$ (De preferência um valor que esteja no domínio de f)
3. Repita os seguintes passos até convergir para uma dist. estacionária:
   - Gere um valor **candidato** $Y = X_{t+1}$ a partir de $g(\cdot|X_t)$ (note que o valor candidato é dependente do valor anterior)
   - Gere U de uma $U(0,1)$
   - Calcule a probabilidade de aceitação de M-H

$$\alpha(X_t, Y) = min\left(\frac{f(Y) \times g(X_t \mid Y)}{f(X_t) \times g(Y \mid X_t)}, 1\right)$$

Se $U \le \alpha(X_t, Y)$:
Aceitar Y e faça $X_{t+1} = Y$;
Caso Contrário $X_{t+1} = X_t$

Nota: O algoritmo foi desenvolvido de forma que a dist. estacionária da cadeia é a dist. alvo f

→ Na prática há necessidade de esperar que a cadeia atinja o seu estado de equi.: Deve "deitar-se fora" n inicial de realizações da cadeia (*burn-in sample*).

CP | Passeio Aleatório

- Num passeio aleatório verifica-se $X_{t+1} = X_t + \varepsilon_t$ com $\varepsilon_t \sim N(0,\sigma)$.
  Se os ε_t são independentes então X_t é independente de $X_{t-2}, X_{t-3}, \cdots$, mas condicional a X_{t-1}
  - Num **PA**, em cada etapa, qualquer movimento para a **frente** tem probabilidade p de ocorrer e qualquer movimento para **trás** tem probabilidade $1-p$.
  - Um movimento ocorrido do estado t para o $t+1$ depende apenas do estado t. Logo, um passeio aleatório é uma Cadeia de Markov.

Se $g(\cdot|X_t)$ é **simétrica,** então $g(X_t|Y) = g(Y|X_t)$, e a P(aceitação) é dada por $\alpha(X_t, Y) = min\left(\frac{f(Y)}{f(X_t)}, 1\right)$

Se um valor candidato $Y = X_{t+1}$ é gerado a partir de uma dist. candidata simétrica, então a probabilidade da cadeia se mover de X_t para X_{t+1} depende apenas da distância entre eles: $g(X_{t+1}|X_t) = g(|X_{t+1} - X_t|)$.

Logo, a cada iteração, um incremento Z é gerado a partir de $g(\cdot)$, e Y é definido como $Y = X_t + Z$ (**definição de random walk**).

Algoritmo

O incremento aleatório Z pode ser $N(0,\sigma)$, de forma que o $Y|X_t \sim N(X_t,\sigma)$, para algum $\sigma > 0$ constante. Mas o incremento Z também pode ser $U(-\delta,\delta)$, por exemplo.

- Defina uma dist. proposta g **simétrica**
- Defina X_0, dentro do domínio de f
- Repita os seguintes passos até convergir:

1. Gere um valor **candidato** $Y = X_{t+1} = X_t + Z$
2. Gere U da $U(0,1)$
3. Calcule a taxa de aceitação $\alpha(X_t, Y)$
   Se $U \leq \alpha(X_t, Y)$ aceite Y e faça $X_{t+1} = Y$;
   Caso contrário faça $X_{t+1} = X_t$

CP | Amostrador Independente

Neste método, **a $g(x)$ não depende de valores anteriores da cadeia**, ou seja, $g(Y|X_t) = g(Y)$.

A probabilidade de aceitação é $\alpha(X_t, Y) = min\left(\frac{f(Y)}{f(X_t)} / \frac{g(Y)}{g(X_t)}, 1\right) = min\left(\frac{f(Y)\,g(X_t)}{f(X_t)\,g(Y)}, 1\right)$

Nota: Embora **os valores de $Y = X_{t+1}$ sejam gerados de forma independente, a cadeia resultante não será iid, já que a probabilidade de aceitação ainda depende de X_t.**

Algoritmo:

- Defina uma dist. candidata g similar à f
- Defina X_0, dentro do domínio de g
- Repita os seguintes passos até convergir:

1. Gere um valor **candidato** Y a partir de g
2. Gere U de uma $U(0,1)$
3. Calcule a taxa de aceitação $\alpha(Xt, Y)$
   Se $U \leq \alpha(X_t, Y)$ aceite Y e faça $X_{t+1} = Y$;
   Caso contrário faça $X_{t+1} = X_t$

5. Métodos de Reamostragem

Bootstrap (MC Não Paramétrico porque não precisa de saber nenhum modelo, apenas usa a amostra original c/ reposição)

- Uma amostra **bootstrap** é um conjunto de valores extraídos aleat. **com reposição** da amostra original.
- A estimativa de θ na b –ésima reamostra bootstrap é $\hat{\theta}^{(b)}$
- $F_n(x)$ é um estimador de $F_X(x)$ para todo o x.

A EP **Bootstrap** é o valor médio $\bar{\hat{\theta}}^* = \frac{1}{B}\sum_{b=1}^{B}\hat{\theta}^{(b)}$

Estimativa **Bootstrap** do viés: $\widehat{\text{viés}}^*(\hat{\theta}) = \bar{\hat{\theta}}^* - \hat{\theta}$

com $\bar{\theta}^* = \hookleftarrow$ e $\hat{\theta} = \hat{\theta}(x) = \hat{\theta}(x_1, \ldots, x_n)$

```
mean(rowMeans(S)) - mean(smpl)
```

Algoritmo:

Para cada estimativa de bootstrap indexada $b = 1, \ldots, B$:

1. Gere uma amostra $x^{*(b)} = (x^*, \ldots, x_n^*)$, através de amostragem **com reposição de** amostra observada $x_1, \ldots, x_n$.
   A dimensão da amostra bootstrap pode ser de dimensão diferente da dimensão da amostra inicial.
2. Calcule a b –ésima estimativa $\hat{\theta}^{(b)}$ da b –ésima amostra de bootstrap $x^{*(b)}$

Cross-Validation

- Conjunto de Validação
- Leave-One-Out (LOOCV $k = n$)
- k-Folds ($k = c$)

CV método é simples e fácil de implementar. No entanto o MSE de validação pode ser altamente variável e, apenas um subconjunto de obs. é usado para ajustar o modelo (dados de treino), sendo estas duas grandes desvantagens deste método.

LOOCV em comparação com a abordagem do CV tem menos enviesamento e o MSE é menos variável. Contudo, é computacionalmente intensivo (desvantagem).

K-fold é o melhor método dado que computacionalmente não é tão intensivo.

- A taxa de erro de treino pode subestimar dramaticamente a taxa de erro de teste. ✓
- O erro de teste pode ser facilmente calculado aplicando o método às observações usadas no seu ~~treino~~. ✗
- Pode-se avaliar a performance de um modelo pela estimativa das taxas de erro de ~~treino~~. ✗

6. Simulação de Eventos Discretos

Elementos de um Modelo de Simulação

1 | Objetos do Modelo

Entidades: elementos individuais do sistema que estão sendo simulados e cujo comportamento está sendo explicitamente rastreado. Cada entidade pode ser identificada individualmente;

Recursos: também elementos individuais do sistema, mas não são modelados individualmente, mas tratados como itens contáveis cujo comportamento não é rastreado.

Ex: Na nossa Loja de Croissants,
- Os clientes provavelmente serão recursos, pois não estamos interessados no que cada um deles faz.
- Os funcionários podem ser considerados entidades ou recursos:

No 1º caso, podemos rastrear o tempo total de trabalho de cada um deles; no 2º caso, o modelo só seria capaz de produzir uma visão geral de como os funcionários estão ocupados.

2 | Organização de Entidades e Recursos

Atributos: propriedades de entidades e recursos. Geralmente é usado para controlar o comportamento do objeto.

Ex: Estado de um funcionário (ocupado ou disponível).
Numa simulação + ampla, um atributo pode ser o tipo de croissant que o cliente comprará (p.e., chocolate, ...).

Estado: conjunto de var. necessárias para descrever o sistema em qualquer momento.

Ex: No caso mais simples, as variáveis são o número de clientes na fila e o número de funcionários ocupados.

Lista: coleção de entidades ou recursos ordenados de alguma forma lógica.

Ex: Os clientes que esperam em nossa loja podem ser ordenados no esquema "primeiro a chegar primeiro a ser servido" *(FIFO)*

3 | Operações dos Objetos

Durante um estudo de simulação, entidades e recursos irão cooperar e, portanto, mudar de estado.

Evento: instante de tempo em que o estado do sistema muda.

Ex: Quando um cliente termina de ser atendido: o nº de funcionários ocupados diminui em um e há um cliente a menos na fila.

Atividade: um período de tempo de duração especificada do qual se conhece o início (embora a sua duração possa ser aleatória).

Ex: O tempo que um funcionário leva a atender 1 cliente: isso pode ser especificado em termos de uma distribuição aleatória.

Delay: duração de tempo não especificada, que não é conhecida até o término.
Isso não é especificado pelo modelador com antecedência, mas é determinado pelas condições do sistema. Muitas vezes, este é um dos resultados desejados de uma simulação.

Ex: Tempo de espera de um cliente na fila de nossa loja de croissants.

Clock: variável que representa o tempo simulado.

R

```
# ==== (Re)Amostragem Aleatória Simples ====
sample(x,          – Vetor de onde escolher os números discretos
       size,       – Nº de elementos a amostrar;
                     Se não for especificado é igual à dimensão x
                     (Permutação aleatória dos elementos)
       replace = T/F,   – Com/Sem Reposição
       prob = c())      – Probabilidade dos elementos

# ==== Geradores de Dist. Do R ======
rxx(n, *)   – Gerar n NPAs
matrix(rX(n*m), nrow=n, ncol=m)  – Gerar n × m NPAs
dxx(x, *)   – Função Densidade num certo x
pxx(p, *)   – Função de Distribuição F(x) = P[X ≤ x]
qxx(q, *)   – Função Quantil (Inversa da fd)
```

```
# =============================== Qualidade dos NPAs ================================
# Histograma
ks.test
hist(u, prob = TRUE)

# Teste de Kolmogorov-Smirnov (de ajustamento)
ks.test(u,"punif") # Se p-value < 0.05 -> Não rejeitamos
                   a H0 de que a seq. tem dist. U.

# f.d.p. da U(0,1)
abline(h=1, col="red")
text(0.85,.95, "fdp U(0,1)", col="red");

# Gráfico QQ
# O QQ-plot representa a sequência x para verificar se os dados podem seguir uma dist.
# Se os pontos não diferem muito da linha reta e, neste caso, podemos assumir que os
# dados são de uma certa dist. (p.e. U(0,1)).
n <- 1000
x <- runif(n)
q <- qunif(ppoints(n)) # ppoints gera uma sequência de pontos de probabilidade
qqplot(q, x,
       cex=0.25,       # cex diminui o tamanho dos pontos para 25%
       xlab="U(0,1)",
       ylab="sample")
abline(0, 1, col = 2)

# Teste à Independência - Teste de Box-Pierce ou lags
Box.test(u, lag = 5)
acf(u) # Estima a função de autocorrelação e elabora por defeito o gráfico da função
```

```
# ============================ Gerador Congruencial Linear ============================
alcm <- function(N, x0, a, c, m){
   x <- rep(0,N)                          # Replica os elementos do vetor entre 0 e N
   x[1] <- x0
   for (i in 2:N) x[i] <- (a*x[i-1]+c)%%m  # %% implementa o módulo (mod)
   return(x)
}                                 # Se quisermos que os NPAs ~ U(0,1) então (x/m)
```

```
# ========================== Método da Transformação Inversa ==========================
# Gerar 1000 valores do modelo f(x) = 3x^2 , x ∈ [0,1]
n <- 1000
u <- runif(n)
x <- u^(1/3)
hist(x, prob = TRUE, main = bquote(f(x)==3*x^2)) # bquote - incluir a expressão da função no título
curve(3*x^2,0,1, add=T ,col="red")

# Obter 5 observações de uma v.a. Exponencial com o parâmetro λ = 2.
 invF <- function(u,lambda) -log(1-u)/lambda
 u <-runif(5)
 invF(u, lambda = 2)

# Gerar pela inversa 20 valores do modelo Bernoulli com p = 0.4
 p <- 0.4;  n <- 20;
 u <- runif(n)
 x <- as.integer(u > 0.6) # Se fizermos mean(x) e var(x) podemos comparar com a pop.
```

```
# ========================== Método da Aceitação-Rejeição ============================
# Encontrar o valor máximo
foo <- function(x) fx(x)/gx(x)
opt <- optimize(foo, c(0,4), maximum = TRUE)
c <- opt$objective
paste("Valor Ótimo de c =", round(c,2)) # Nº Valores Aceites = n x c

# Algoritmo
k = 0           # Estabelece um contador para o número de observações aceites
count = 0       # Estabelece um contador para o número de iterações percorridas
accept = c()    # Cria um vector vazio onde iremos guardar as simulações, x, aceites

# Percorrer o algoritmo até atingir os 10 000 NPAs gerados
while(k < 10000){
  # Gerar x da distribuição g(x) proposta - N(0,1)
  X <- rnorm(1)

  # Gerar um u da distribuição Uniforme(0,1)
  U_sim <- runif(1, 0, 1)

  # Calcular o valor de f(x)/(c × g(x))
  cg_x <- fx(X)/(c*dnorm(X))

  # Comparar os dois valores - Aceita-se ou Rejeita-se
  if(U_sim <= cg_x){

    # Se o valor de U(0,1) for inferior ou igual a cg(x), então:
    # Aceitar a simulação acrescentando uma linha ao vetor 'accept'
    accept = rbind(accept, X)
    k = k + 1
  }
  # Caso contrário, descartar a simulação, aumentar o contador em 1, e
  #                 seguir para a próxima iteração
  count = count + 1
}
paste("Nº de Iterações",count)
paste("Nº de Valores 'y' Aceites",length(accept))
```

```
# ================== Simulação de Eventos Discretos [1] | Simmer =====================
library(simmer)
env <- simmer("Croissant shop")              # Cria um Ambiente de Simulação

customer <- trajectory("Customer") %>%       # Cria uma Trajetória (Cliente)
   log_("Here I am") %>%                     # Produz texto q é mostrado durante a simu.
   seize("employee", 1) %>%                  # Bloqueia o recurso employee
   timeout(function() runif(1,1,5)) %>%      # Define a dist. da atividade do recurso
   release("employee", 1)                    # Liberta o recurso employee

env %>%
   add_resource("employee", capacity = 2) %>%  # Gera-se 2 funcionários
   add_generator("customer", customer,         # Gera-se a chegada dos clientes
                 function() rexp(1,1/3)      # dist. OU at(vetor de tempos) OU
                                               from_to(start_time, stop_time, dist,
                                                 arrive = T/F,  # Se T ent o 1º é gerado no star_time
                                                                # Se F ent é gerado depois da dist
                                                 every = NULL   # Repete neste ciclo de tempo)
                 mon = 2)
# Se o simulador deve monitorar as chegadas geradas:
#   0 = sem monitoramento,    1 = monitoramento simples,
#   2 = nível 1 + monitoramento de atributo de chegada
                                             # Executa a simulação por 2h
# ----------------------------------------------------------------------------------
env %>% get_mon_arrivals(ongoing=TRUE) # Monitorização dos clientes (= T até os n atendidos)
        ("name", "start_time", "end_time", "activity_time", "finished", "replication")
   %>% transform(waiting_time = end_time - start_time - activity_time)

env %>% get_mon_resources()     # Estatísticas dos recursos ("resource", "time", "server", "queue"
                                  "capacity", "queue_size", "system", "limit", "replication")

env %>% get_mon_attributes()    # Atributos ("time", "name", "key", "value", "replication")
```

```
# ============================ Métodos de Transformação =============================
## Usando a relação existente entre as distribuições uniforme contínua e logaritmica,
   obtenha uma amostra aleatória de 1000 números com distribuição Logaritmica(0.5).
n <- 1000;       theta <- 0.5;
u <- runif(n);  v <- runif(n);
x <- floor(1 + log(v) / log(1 - (1 - theta)^u)) # floor dá a parte inteira de um número

## Calcula as probabilidades teóricas (exactas) usando a definição da distribuição
k <- 1:max(x)
p <- -1/log(1 - theta) * theta^k/k

## Comparação gráfica dos valores gerados com a prob. teórica
par(mfrow = c(1, 2))                                   # Gráficos lado a lado
plot(prop.table(table(x)), xlab = "X", ylab = "P[X]")  # Freq. Relativa Empíricas
points(p ~ I(k + 0.1), type = "h", col = 2, lwd = 2)
legend("topright", legend = c("Empírico", "Teórico"),
       lty = 1, col = c(1, 2))
plot(ecdf(x))                  # Gráfico da Função de Distribuição Cumulativa Empírica
lines(cumsum(p), type = "s", col = 2)
```

```
# ====================================== Somas =======================================
Ex.:  Distribuição Qui-Quadrado
      Gerar n = 1000 NPA χ²(df) através da convolução de df quadrados de Normal Padrão;
      ie, se Z_1,…,Z_v são iid N(0,1) então V = Σ_1^n Z_i² tem distribuição χ²(df).

      Algoritmo:
      Para gerar m valores de uma χ²(df)
      1. Criar uma Matriz Z (n × df) com n × df v.a. N(0,1)
      2. Obter uma Matriz X de elementos dados pelo quadrado cada elemento da Matriz Z
      3. Calcular as somas por linha, da matriz anterior (X).
      4. Sair como vetor das somas por linha.

Vamos considerar df = graus de liberdade = 2 = v.
n <- 1000;    v <- 2;
X <- matrix(rnorm(n*v), nrow=n, ncol=v)^2 # Matriz dos quadrados das N(0,1)
# Soma por linhas da matriz anterior
y <- rowSums(X)
# Alternativa | y <- apply(X, MARGIN=1, FUN=sum) # vector de dimensão n
mean(y) # deverá ser próximo de v = 2 |  var(y)  # deverá ser próximo de 2v = 4

Ex.:  Seja X_1 ~ Gama(2,2) e X_2 ~ Gama(2,4) independentes.
      Gerar 1000 NPAs pela convolução S = X_1 + X_2 e pela mistura f(X) = 0.5f(X_1) + 0.5f(X_2).

    Algoritmo para Soma/Convulção:
      1. Gerar X_1 NPA Gama(2,2)
      2. Gerar X_2 NPA Gama(2,4)
      3. Obter S = X_1 + X_2

    Algoritmo para Mistura:
      1. Gerar um inteiro k ∈ {1,2}, onde P(1) = 0.5 e P(2) = 0.5
      2. Se k = 1       Output: x NPA do modelo Gama(2,2)
         Se k = 2       Output: x NPA do modelo Gama(2,4)
n <- 1000;          x1 <- rgamma(n, 2, 2);      x2 <- rgamma(n, 2, 4)
s <- x1 + x2          # Define a convolução
u <- runif(n)
k <- as.integer(u > 0.5) # Vetor de 0 e 1
x <- k * x1 + (1-k) * x2 # Define a mistura
par(mfcol=c(1,2))
hist(s, prob=TRUE, xlim=c(0,5), ylim=c(0,1), main = "Soma de Gammas")
hist(x, prob=TRUE, xlim=c(0,5), ylim=c(0,1), main = "Mistura de Gammas")

Ex.:  Gerar 1000 NPAs com dist. com distribuição Binomial(n = 6, p = 0.5) pela soma de distribuições de Bernoulli.
x <- numeric(1000)
for (i in 1:1000) {                                  OU
  # Gerar 6 números de Bernoulli e somar             X <- matrix(rbinom(n = 1000*6, size = 1, prob = 0.5),
  bern_sample <- rbinom(6, size=1, prob=0.5)                  nrow=1000, ncol=6)
  soma <- sum(bern_sample)                           y <- rowSums(X)
  x[i] <- soma
}
# Para confirmar os resultados mean()= E[X] = np = 6 × 0.5 = 3  e var() = np(1 − p) = 1.5
```

```
# =================================== Misturas =======================================
Ex.: Suponha que X_1~N(0,1) e X_2~N(3,1) e são independentes.
     Gere 1000 NPA da mistura f_X(X) = 1/3 f(X_1) + 2/3 f(X_2)
      Algoritmo:
      Para gerar NPA da mistura de duas normais é
      1. Gerar um inteiro k ∈ {1,2}, onde P(1) = 1/3 e P(2) = 2/3
      2. Se k = 1         Output: x NPA do modelo N(0,1)
         Se k = 2         Output: x NPA do modelo N(3,1)

n <- 1000;    mu<-c(0,3)
k <- sample(1:2, size=n, replace=TRUE, prob=c(1/3,2/3))
m <- mu[k]             # vector dim=n (mu_k1,...,mu_kn), elementos mu[1]=0 ou mu[2]=3
x <- rnorm(n, m, 1)
```

```
# ================================ MC | Estimadores ==================================
Ex.: Comparar 2 estimadores para o coef. de assimetria de dist. de probabilidade.
      100 amostras de dimensão n=20 | 100 amostras de dimensão n=100 | 100 com n=1000
      Obtenha, para cada amostra gerada e para cada estimador, a assimetria amostral (estimativa) correspondente.
      Recorrendo ao cálculo do erro-padrão e do erro quadrático médio, conclua pela sua preferência…

# Gerar as Amostras - matriz n linhas x 100 colunas
sample_20 <- replicate(100, rt(20, df = 20))
sample_100 <- replicate(100, rt(100, df = 20))
sample_1000 <- replicate(100, rt(1000, df = 20))

# Calcular s1 e s2 para cada amostra de 20 observações
s1_values_20 <- apply(sample_20,                   1                     Linhas
                      MARGIN = 2,        # MARGIN = 2 | Aplicar função nas Colunas
                      FUN = s1_estimator) # FUN = função que queremos aplicar

# Calcular Erro-Padrão e Erro Quadrático Médio para s1 e s2
se_s1_20 <- sd(s1_values_20) / sqrt(100)
mse_s1_20 <- mean((s1_values_20 - 0)^2) # Neste caso Valor Esperado = 0 …
```

```
# ================== Simulação de Eventos Discretos [2] | Simmer =====================
# Gráfico que representa os recursos
plot(resources, metric = "usage",steps=T)
• A linha verde     = nº de recursos ocupados (p.e., funcionários);
• A linha vermelha  = nº de pessoas na fila que estão à espera para serem atendidas.
• A linha azul      = nº total de clientes no sistema: fila + os que estão a ser atendidos.

[EXTRA]

# Clientes Prioritários (Só depois dos outros serem atendidos)
env %>%
  ... add_generator(.env, name_prefix, trajectory, distribution, mon, priority = 1)

# Clientes Prioritários com urgência (Com atendimento primeiro)
env %>%
  ... add_resource(name, preemptive = TRUE) %>%
  ... add_generator(…, priority = 1)

# Desistência do Serviço pelos Clientes
customer <- trajectory("Customer") %>%
   ... seize("counter", continue = FALSE, reject = trajectory("Balked customer")) %>%

env %>%
   ... add_resource("counter", queue_size = 1) %>% ... # queue_size = capacidade da fila

# Nº Clientes que Desistiram   sum(get_mon_arrivals(env)$activity_time == 0)
# Taxa Horária de Desistência  sum(get_mon_arrivals(env)$activity_time == 0)/now(env)*60
```

```
# ================================ MCMC M-H | Geral ==================================
Usar o MMH para gerar uma amostra da dist. de Rayleigh com σ = 4, usando como g a Qui-Quadrado.
## pdf da Rayleigh
f <- function(x, sigma) {(x / sigma^2) * exp(-x^2 / (2*sigma^2)) #x>0, sigma>0}

T <- 10000  # Nº de runs ;    x <- numeric(T);
x[1] <- rchisq(1, df=1)   # Valor inicial X_0 - Não é um valor inteiro e pode não o ser
k <- 0                    # Regista o número de rejeições
u <- runif(T)             # Gera os T valores NPA U(0,1)
for (i in 2:T) {
  xt <- x[i-1]
  y <- rchisq(1, df = xt)                       # Valor candidato – g(x)
  num <- f(y, sigma) * dchisq(xt, df = y);
  den <- f(xt, sigma) * dchisq(y, df = xt);
  alpha <- min(num/den,1)
  if (u[i] <= alpha) {x[i] <- y}
  else {x[i] <- xt;                             # O X_t é o x[i-1] = Valor Anterior
        k <- k+1}                               # Incrementa k porque y foi rejeitado
}

## Comparação da dist. teórica e empírica | fdp da Rayleigh sobreposta ao histograma das obs. geradas,
deitando fora as primeiras 2000 observações da cadeia de Markov (burn in sample). Comparar ainda através de um QQplot
os quantis amostrais e os quantis teóricos.
b <- 2001 # burn in sample fora; y <- x[b:T]; a <- ppoints(100)
QR <- sigma * sqrt(-2 * log(1 - a)) # quantis da Rayleigh

par(mfrow=c(1,2))
qqplot(QR, y, main="", xlab="Quantis Rayleigh", ylab="Quantis Amostrais")
abline(0, 1,col="red")
hist(y, main="", xlab="", freq=FALSE)
lines(QR, f(QR, 4))
```

```
# ========================= MCMC M-H | Passeio Aleatório ============================
Gerar valores de uma normal padrão, usando como distribuição candidata a U(−0.5, 0.5).
f <-function(x) dnorm(x, 0, 1);   delta <- 0.5;   N <- 500;  x <- numeric(N);
x[1] <- runif(1, -delta, delta)
for(i in 2:N) {
    z <- runif(1, -delta, delta)
    y <- x[i - 1] + z
    alpha <- min(f(y)/f(x[i - 1]), 1)
    u <- runif(1)
    if(u <= alpha) {x[i] <- y}
    else {x[i] <- x[i - 1]}
}
plot(x, type = "l")

# Comparar a dist. das amostras com a dist. teórica.
par(mfrow = c(1, 2))
plot(ecdf(x), main = expression(delta == 0.5))
curve(pnorm(x), add = TRUE, col = 2)
```

```
# ======================= MCMC M-H | Amostrador Independente =========================
Gere 500 NPA de uma distribuição Beta(2,3)  usando como candidata a N(0.5, 0.25)
f <- function(x) dbeta(x, shape1 = 2, shape2 = 3) # Dist. alvo: X ~ Beta(2, 3)
g <- function(x) dnorm(x, 0.5, 0.25)              # Dist. candidata: X ~ N(0.5, 0.25)

# Gráfico com as densidades sobrepostas
curve(f, 0, 1)
curve(g, add=TRUE, col=2)
legend("topright", legend=c("Alvo", "Candidata"), lty=1, col=1:2, bty="n")

# Algoritmo M-H | Amostrador Independente
N <- 500;     x2 <- numeric(N);   x2[1] <- 0.5;    k2 <- 0;
for(i in 2:N) {
    y <- rnorm(1, 0.5, 0.25)  # Distribuição proposta
    alpha <- min((f(y) * g(x2[i - 1])) / (f(x2[i - 1]) * g(y)), 1)
    u <- runif(1)
    if(u <= alpha) {
        x2[i] <- y
        k2 <- k2 + 1
    } else {x2[i] <- x2[i - 1]}
}

# Cadeia                  # Taxa de Aceitação      # Comparar com teórica
plot(x2, type = "l")      k2/N                     plot(ecdf(x2))
acf(x) # Função de Autocorrelação (Am. Ind.)       curve(pbeta(x, 2, 3), add = T, col = 2)
```

```
# =================================== Bootstrap ======================================
Seja x = (x_1,···,x_10) = (2,2,1,1,5,4,4,3,1,2). Obter a amostra bootstrap de x.
Qual é a função dist. empirica, F_n(x), a função dist. da amostra bootstrap, F_n^*(x), e a função dist. teórica da
população, F_X(x), de onde foi retirada a amostra?

# Amostra de uma Poisson(2)        x <- c(2, 2, 1, 1, 5, 4, 4, 3, 1, 2);
prop.table(table(x))               # Distribuição empírica
cumsum(prop.table(table(x)))       # Distribuição empírica acumulada

Repare-se que sendo X ~ Poisson(0) tem probabilidade ≠ 0. Reamostrar a partir desta amostra um grande nº de x produz uma
boa estimativa de F_n, mas não uma boa estimativa de F_X, dado que as amostras bootstrap nunca incluirão o 0.
am <- sample(x, replace = TRUE)     # Amostra via bootstrap | Um passo
prop.table(table(am))
cumsum(prop.table(table(am)))

plot(0:5, c(0, prop.table(table(amB))), type = "h")     # Representa a dist. empírica
points((0:5) + .1, dpois(0:5, 2), type = "h", col = 2) # Representa a dist. teórica

# Intervalo de Confiança de Wald (Intervalo normal padrão de Bootstrap) [EXTRA]
                              θ̂ ± Z_{α/2} × se(θ̂)

(lower95.limit <- mean(R) - qnorm(0.975) * se.R)
(upper95.limit <- mean(R) + qnorm(0.975) * se.R)
```

```
# ================== Simulação de Eventos Discretos [3] | Simmer =====================
# Abandono da Fila
customer <- trajectory("Customer") %>%
    ... renege_in(tempo, out = trajectory("Reneging customer") %>% log_("I am off")) %>%
    ... renege_abort() # Se o recurso ficar livre antes do tempo, o abandono deve ser cancelado

# Vários Balcões com Filas Individuais
customer <- trajectory("Customer") %>%
  # O cliente decide qual fila entrar – Regra do Exemplo: entrar na fila + curta
  ... select(c("counter1", "counter2"), policy = "shortest-queue") %>%
  seize_selected() %>%   # Entrar na fila escolhida – Bloqueia-a
  …
  release_selected() %>% # Libertar a fila escolhida – Já foi atendido

# Horário de Funcionamento (Aberto: 8h às 24h (com ≠ nº de recursos) | Fechado: 24h às 8h)
door_schedule <- schedule(timetable = c(8, 16, 24), values = c(Inf,2, 0), period = 24)

env %>%  ... add_resource("door", capacity = door_schedule) %>% ...

# Lotes de Clientes
customer <- trajectory() … %>% batch(n = Inf, timeout = 30) %>%  separate() %>%
  * n: o tamanho do lote;
  * timeout: define um temporizador que "dispara" lotes a cada x tempo, mesmo que o n não tenha sido preenchido.

# Bifurcações na Trajetória (85% segue a trajetória 1)
customer <- trajectory() … %>% branch(function() (runif(1) > 0.85) + 1, # Atenção ao >
                                      continue = c(FALSE, FALSE),
                                      trajectory("1"), trajectory("2"))
```