Geração de números uniformes

Importância e principais algorítmos

Prof. Walmes Zeviani
walmes@ufpr.br

Laboratório de Estatística e Geoinformação
Departamento de Estatística
Universidade Federal do Paraná

Atualizado em 2018-08-13

Justificativas

- Simulação computacional de processos estocásticos depende da geração de números aleatórios (GNA).
- TODO TODO TODO TODO TODO TODO TODO TODO TODO TODO TODO TODO TODO TODO TODO TODO TODO TODO TODO TODO TODO TODO TODO TODO TODO TODO TODO TODO TODO TODO

Objetivos

- Discutir a importância da geração de números aleatórios em Estatística.
- Descrever os principais algorítmos para GNA uniformes.

A distribuição uniforme

Se uma variável aleatória X tem distribuição uniforme contínua entre a e b, denotamos $X \sim \mathrm{U}(a, b)$, e sua função de densidade é

$$f(x, a, b) = \frac{1}{b-a} \cdot I(a \leq x < b), \quad -\infty < a < b < \infty. \tag{1}$$

A função de probabilidade é

$$F(x, a, b) = \Pr(X < x) = \int_{-\infty}^{x} f(x, a, b)\,\mathrm{d}x = \begin{cases} 0, & x < a \\ \dfrac{x-a}{b-a}, & a \leq x < b \\ 1, & x \geq b. \end{cases} \tag{2}$$

A média e variância são funções dos parâmetros

$$\mathrm{E}(X) = \frac{a+b}{2} \quad \text{e} \quad \mathrm{V}(X) = \frac{(b-a)^2}{12}. \tag{3}$$

Importância da distribuição uniforme

- A distribuição tem pouca importância do ponto de vista de modelagem.
- Porém, para simulação computacional, ela é central.
- Números aleatório da Uniforme são:
  - O principal ingrediente para GNA de outras distribuições.
  - Usados em várias aplicações de Monte Carlo (e.g. integração MC, inf. Bayesiana por MCMC, etc.)
  - Utilizados em otimização estocástica.
  - Empregados em jogos digitais (e.g. poker online).

Geração de números aleatórios

Números aleatórios reais

- São gerados por dispositivos via processos físicos.
  - Globo de sorteio da Mega-Sena e bingos.
  - Lançamento de um dado, moeda.
  - Roleta de cassino.
- Para uso em maior escala, são geralmente baseados em fenômenos que geram um nível baixo de rúido.
  - Ruído termal.
  - Ruído fotoelétrico.
  - Ruído acústico.
  - Movimento browniano de partículas.
- Os sinais precisam ser traduzidos para números (intensidade).
- A distribuição da v.a. pode não ter forma conhecida.
- A v.a. pode ter correlação serial.
- São usados em criptografia.

Geração de números aleatórios

Números pseudo-aleatórios reais

- São gerados por programas de computador, ou seja, algorítmos.
- Os número não são realmente aleatórios, pois, dado o algorítmo gerador, pode-se prever os números (tem reprodutibilidade).
- Do ponto de vista Estatístico, os números são imprevisíveis.
- Bons GNA terão propriedades interessantes para aplicações Estatística.

Propriedades desejáveis de um GNA

- Os valores gerados são completamente imprevisíveis e passam em testes de detecção de não aleatoriedade.
- A distribuição da v.a. é matematicamente conhecida (e.g. distribuição Uniforme).
- O GNA produz v.a. que não apresentarem correlação serial, ou seja, as realizações são independentes umas das outras.

Algorítmos para GNA Uniformes

- Existem vários algorítmos para GNA. No R, existem 7 opções para o algorítmo de GNA.
- Peça `help(Random)` para consultar a respectiva documentação.

Algorítmos disponíveis no R

- Wichmann-Hill.
- Marsaglia-Multicarry.
- Super-Duper.
- Mersenne-Twister (*default*).
- Knuth-TAOCP-2002.
- Knuth-TAOCP.
- L'Ecuyer-CMRG.

GNA feito por mim

Algorítmo

1. Definir um número $u_0 \in (0, 1)$ e atribuir $i = 0$.
2. Calcular $u_{i+1} = 0.5 + 0.5 \sin(2\pi u_i)$.
3. Incrementar i fazendo $i = i + 1$.
4. Repetir 2–3 até obter a sequência de tamanho desejado.

Atividades

- Implemente em R o algorítmo acima.
- Considere $x_0 = 0.3$ e gere 1000 números.
- Inspecione a amostra de números varificar existência de independência, periodicidade e forma da distribuição.

GNA de von Neumann

Artigo

von Neumann, J. (1951). Various Techniques Used in Connection with Random Digits. In the Monte Carlo Method (ed. A.S. Householder et al., 36–38). **Nat. Bur. Standards Appl.** Math Ser. no. 12.

Algorítmo

1. Definir um número u_0 de quadro digitos decimais e atribuir $i = 0$.
2. Calcular u_i^2. Agregar zeros à esquerda, quando necessário, para manter representação com 4 digitos: $u_i^2 = d_7 d_6 \ldots d_0$, onde cada d_j é um inteiro entre 0 e 9.
3. Fazer $u_{i+1} = d_5 d_4 d_3 d_2$.
4. Incrementar i fazendo $i = i + 1$.
5. Repetir 2–4 até obter a sequência de tamanho desejado.

Implemente e inspecione as propriedades deste GNA Uniformes.

O método congruencial linear

O método congruencial linear é baseado na seguinte expressão recursiva

$$u_{i+1} = (a u_i + c) \mod m, \tag{4}$$

em que

- $u_i \in \mathbb{N}$ no intervalo $[0, m-1]$,
- $a \in \mathbb{N}^+$ é chamado de multiplicador,
- $c \in \mathbb{N}^+$ é chamado de incremento,
- $m \in \mathbb{N}^+$ é chamado de módulo.

Se a, c e m forem adequadamente escolhidos, a cardinalidade do conjunto de valores gerados é no máximo m.

Implementação e avaliação

Baseado nas orientações em Ferreira 2013.

- Implemente e inspecione usando as configurações
  - $u_0 = 7$, $a = 7$, $c = 7$ e $m = 10$.
  - $u_0 = 7$, $a = 7^5$, $c = 0$ e $m = 2^{31} - 1$ (Park & Miller, 1988, apud Ferreira 2013).

Ferreira 2013 citando Schrage (1979), comenta que

$$au_i \mod m = \begin{cases} a(u_i \mod q) - r\lfloor u_i/q \rfloor & \text{se maior que 0,} \\ a(u_i \mod q) - r\lfloor u_i/q \rfloor + m & \text{caso contrário,} \end{cases} \tag{5}$$

é uma implementação mais portável para linguagens de baixo nível.

Em que $q = \lfloor m/a \rfloor$ e $r = m \mod a$, que resulta em $m \approx aq + r$.

Implementação em R

Método congruencial linear

```
# a = 7^5
# m = 2^31 - 1
rand_congr0 <- function(n = 1, x0, a = 16807, m = 2147483647) {
    if (missing(x0)) x0 <- as.integer(Sys.time())
    stopifnot(x0 < m)
    n <- n + 1
    x <- vector(mode = "integer", length = n)
    x[1] <- x0
    for (i in 2:n) {
        x[i] <- (a * x[i - 1]) %% m
    }
    return(x[-1])
}

rand_congr0(n = 10, x0 = 321)
```

Implementação em R

Método congruencial linear de Schrage

```
rand_congr1 <- function(n = 1, x0) {
    a <- 16807; m <- 2147483647; q <- 127773; r <- 2836
    # Verifique que: a * (321 %% q) == a * (321 - r * floor(321/q)).
    if (missing(x0)) x0 <- as.integer(Sys.time())
    stopifnot(x0 < m)
    n <- n + 1
    x <- vector(mode = "integer", length = n)
    x[1] <- x0
    for (i in 2:n) {
        k <- floor(x[i - 1]/q)
        x[i] <- a * (x[i - 1] - k * q) - k * r
        if (x[i] < 0) {
            x[i] <- x[i] + m
        }
    }
    return(x[-1])
}
```

Próxima aula

- Outros algorítmos para GNA Uniforme.
- GNA de variáveis aleatórias discretas.
- GNA de variáveis aleatórias contínuas.
- Pacote `purrr`.

Avisos

- Sabatina no Moodle do conteúdo da semana passada.
- Notas das sabatinas já estão disponíveis!

Referências bibliográficas

Ferreira, D. F. (2013). *Estatística Computational em Java*. Editora UFLA.