a03 Métodos de simulación

a03-02 - Números aleatorios

Generación de números aleatorios

Alumno pakistaní, que solo habla inglés
Problemas de horario con el de Visión

Grupos de prácticas

Se pueden cambiar,
Fechas de trabajo y test
Qué grupos presenten cada día

Generación de números aleatorios

En las simulaciones habrá que identificar los factores
Y ajustar la distribución de probabilidad

> Parámetros : variables aleatorias que pertenecen a alguna distribución

Generar valores en dos pasos:

1. Valores en una distribución en concreto: `Y~U[0,1]` Distribución uniforme entre 0 y 1.
   Todos los valores tienen la misma probabilidad

2. `X = f(Y)`
   f = operaciones matemáticas que generan nuestra distribución.

> Todavía se estudian contrastes y generadores de números

Hoy nos centraremos en la distribución uniforme.

Breve introducción a la estadística 15 mins

- **Fenómeno aleatorio**: ocurren en la naturaleza y no se conoce con exactitud

- Ejemplos
  - Juegos de azar: cara-cruz, cartas, poker, acertar la ruleta
- ¿Cómo se estudian?
  - asignamos una `variable aleatoria`
    - Identificamos dos cosas
      - Conjunto de valores que puede tomar este fenómeno aleatorio
      - Probabilidades asociadas: probabilidad que puede tomar cada uno de esos valores
  - Ejemplo moneda:
    - `{cara, cruz}`
    - `{0.5, 0.5}`
  - Ejemplo Ruleta color:
    - {éxito, fracaso}
    - {0.5, 0.5}
  - Ejemplo Ruleta número:
    - {éxito, fracaso}
    - {1/n°valores, 1-1/n°valores}
- **Soporte: conjunto de valores que puede tomar**
  - Variables *Continuas*: Soporte continuo
  - Variables *Discretas*: Soporte discreto
-

> Soporte: conjunto de valores que puede tomar

> No es cierto que infinito sean continuos.
> Ejemplo: Edad ➜ Enteros no negativos. Es discreto e infinito

Probabilidad descriptiva

- Medidas
  - centralización: media, mediana, moda. La media es la más usada.
  - dispersión: varianza , desviación típica. (son similares, porque una es el cuadrado de la otra), coeficiente de variación
  - forma: coeficiente de curtosis, apuntamiento, quartiles, percentiles

Usar una medida de centralización puede llevar a errores.
Por eso siempre debeos apoyarnos en una medida de dispersión.

Varianza grande == mayor dispersión

Estos 3 casos tienen la misma media, pero diferentes varianzas
––||||–– (varianza pequeña)
–|–|–|–| (varianza grande)
|||––||| (La media no está en la población)

Para distinguir situaciones atípicas, necesitamos un **histograma**.

Función de densidad

`f(X)` : densidad
`F(X)` : Distribución.

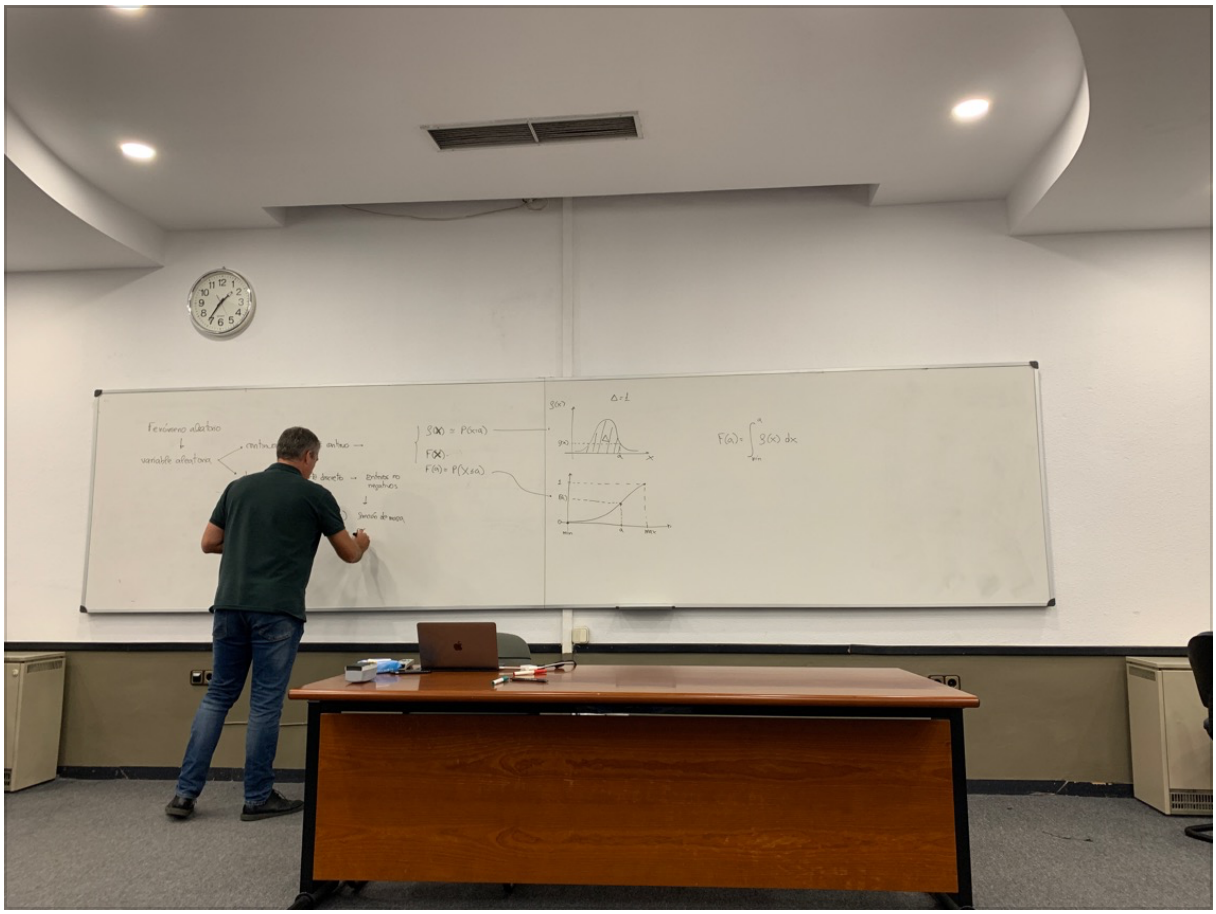
F(a) = PProbabilidad de que mi variable sea <= a
`f(X) = P(x=a)` **MAL** , en soporte continuo, será siempre 0, porque teneos infinitos posibles valores
Es un ~= en vez de solo un =

Curva creciente

Función de Densidad

El área que deja una función de densidad debe ser la unidad =1
`Gaus` :

La relación entre
F(a) = integral min



Formación de masa

P(X)

P(a) = P(x=a)

En caso discreto también se habla de

P(a) = P(X=a) si el soporte es finito

P(a) = P(x<=a)

La relación entre, ya no es de una integral, sino entre un

F(a) = E(x<=a) P(X=x) **DISCRETO**

Integral **CONTINUO**

> OJO: mirar la Relación en la función de densidad y de probabilidad? en el caso discreto y en el caso continuo.

MÉTODOS DE SIMULACIÓN : Capítulo 2. Generación

de números aleatorios

En la práctica no necesitamos implementar los contrastes, solo utilizarlos

Se estudian a nivel teóricos
todos los métodos son imperfectos

Se siguen proponiendo generadores y contrastes
En la historia, han habido dos autores que tuvieron un rifirrafe en sus hipótesis propuestas de generadores sy contrastes, se tiraban el trabajo del uno al otro.

Antes de los ordenadores -> físicos

Tippet 1927 ➜ se utilizaban tablas del censo, 10.000 números
Royo y Ferrer ➜ 250M basados en lotería
Rand Corporation: 1 millón de números aleatorios basados en medición de ruido
Con eventos físicos, no se puede repetir la secuencia y por tanto tampoco el experimento

Propiedades

- Memoria: No nos interesa tener almacenados todos los números en memoria
- Reproducibilidad: repetir experimento sin tener que guardar la secuencia.

Idea de von Neumann : Procedimientos algorítmicos de generación de números ➜ La idea (von Neumann) es producir números que parezcan aleatorios, empleando las operaciones aritméticas del ordenador: partiendo de una semilla inicial (*u0*, *u1,…, u-p+1*), generar una sucesión mediante *ui= d*(*ui-1*,…, *ui-p*), para cierta función *d*.

El periodo de los generadores surge de la naturaleza recursiva de las funciones. Dado un mismo parámetro de entrada en un momento dado, la secuencia será la misma.

La semilla marca el punto de start
El uso de semillas nos permite Mutabilidad y Reproducibilidad

¿Cómo contrastamos y comprobamos que la secuencia es aleatoria?

- Kolmogorov (definición filosófica): Una sucesión de números es aleatoria si no puede producirse eficientemente mediante un programa más corto que la propia serie.
- Definición estadística: Una sucesión de números aleatorios (*ui*) es una sucesión de números en (0,1) con las propiedades de

- Propiedades estadísticas necesarias
  - **Uniformidad en (0,1)** y
  - **Aleatoriedad o independencia estadística.**

Otras propiedades :

1. Rapidez
2. Poco consumo de memoria
3. Portabilidad
4. Sencillez de implementación
5. Reproducibilidad y mutabilidad (semilla)
6. Periodo suficientemente largo. No podemos generar secuencias infinitas, siempre habrá un periodo

Contrastes empíricos

> Podríamos utilizar estos contrastes para estudiar cualquier distribución. Nosotros podemos centrarnos en Uniformidad

Bondad de ajuste o uniformidad

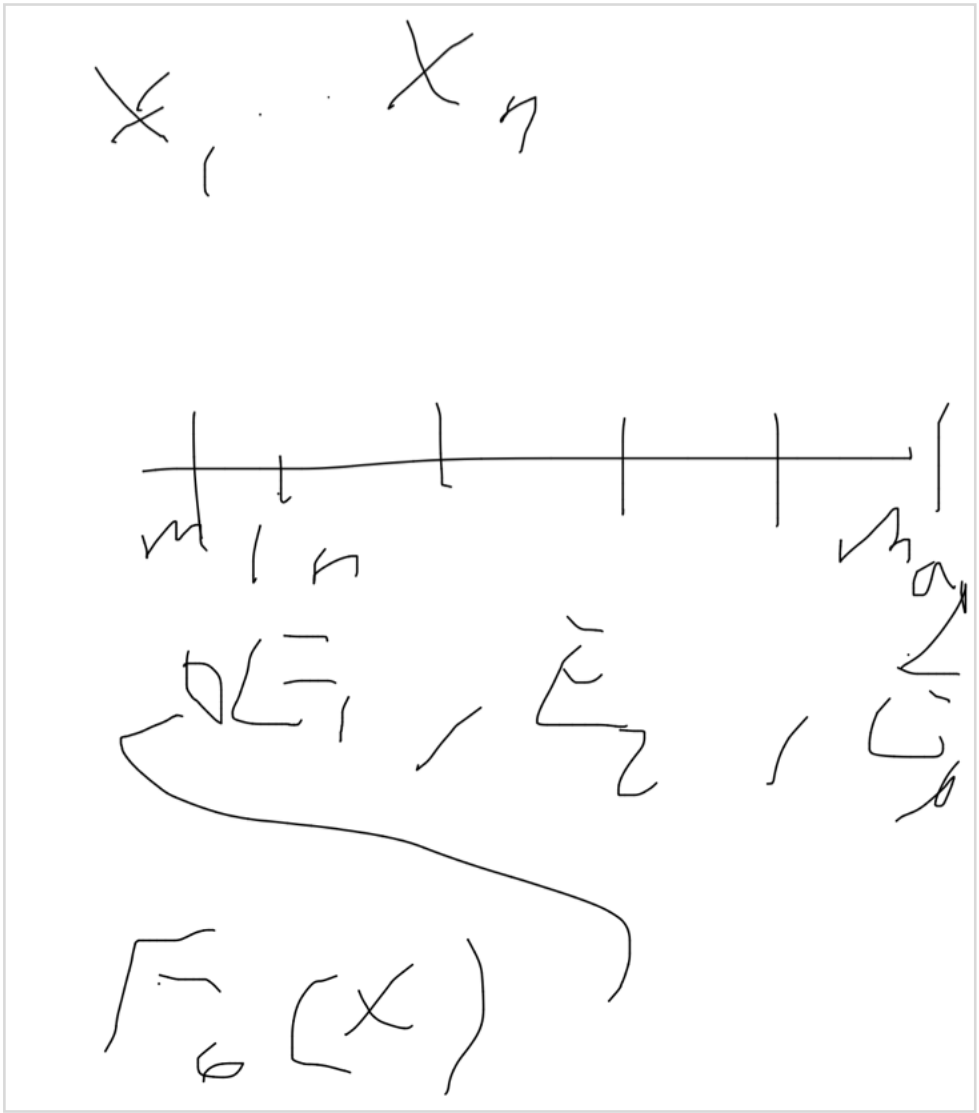
- Contraste `X2` (Chi cuadrado) - *Contraste de Pearson*:
  - De los más antiguos
    - Ventajas
      - Continuos y discrestas
    - Desventajas
      - No es muy potente:
        - Si me rechaza, entonces no es
        - Si me lo acepta, no es 100% seguro. Hay que usar otro contraste.
- Kolmogorov-Smirnov
  -

> Se suele usar una batería de contrastes para confirmar los distintos resultados.

> Usaremos los paquetes estadísticos, no es necesario implementarlos nosotros

Contraste Chi**2: Pasos de contraste de hipótesis

- Se parte de una muestra x1 .. xn
- Hipótesis: Nuestra distribución es una en concreto: Fn(X) = F0(X)

    - Donde n es cada distribución : Normal(media, varianza, máxima verosimilitud), Weibull, Uniforme , … etc
    - Seleccionas 15 y te dice cual sí es
- Cogemos el valor mínimo y máximo en la muestra
    - Para saber el recorrido de valores
- El contraste va a dividir K-Clases (mínimo 5). Recorridos de la misma longitud
- Frecuencias observadas: O1, O2
- Contrastamos con las frecuencias esperadas
    - P(xi € O3) = P(a<= xi <=b) = Integral(a,b) f(x)dx
    - Es decir calcular la función de distribución
    - 
- Nivel de discrepancia

$$X^2 = \sum_{i=1}^{k} \frac{(O_i - E_i)^2}{E_i}$$

- Comprobar si el valor que obtenemos tiene una distribución de X2
  - Comprobar en una tabla
  - Comprobando los grados de libertad
    - Número de clases (K) : pi
    - r = parámetros de nuestra distribución
    - **k-r-1**
    - En la uniforme r = 0
    - En la normal (N(M,desviación^2) ➜ r = 2 parámetros (media y desviación típica)
  - Buscamos el primer valor más pequeño que el mío

> En nuestro caso, para contrastar la uniformidad escogeremos *k*subintervalos de [0,1] de la misma longitud, siendo *pi*= 1/*k*y, por lo tanto, *Ei= n/k*y *r*= 0, ya que no ha sido necesario estimar ningún parámetro de la distribución para obtener *pi*.

Contraste Kolmogorov-Smirnov: Teórico VS Empírico

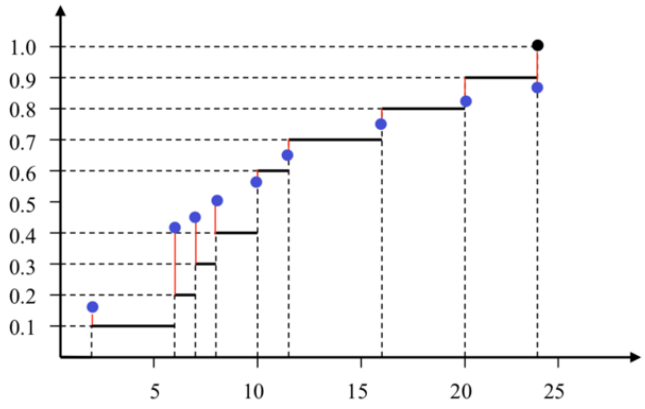
- Muestra: x1…xn
- Gráfico: Distribución escalonada
  - Valores más repetidos ➜ el salto es más grande
- Estamos contrastando que Fn(X) = F0(X)
  - Dibujaos la función de distribución
- Medimos la distancia entre el valor experimental y el valor teórico
  - Nos quedamos con el **Máximo de todas las diferencias**

$$D_n = \max_{1 \le i \le n} \left\{ \max \left[\left| \frac{i}{n} - x_{(i)} \right|, \left| x_{(i)} - \frac{i-1}{n} \right| \right] \right\}$$

Ejemplo. Contrastar si la siguiente muestra de duraciones de vida puede suponerse exponencial: 16, 8, 10, 12, 6, 10, 20, 7, 2, 24.

Ordenamos la muestra: 2, 6, 7, 8, 10, 10, 12, 16, 20, 24.

Representamos la función empírica:



x	$F_n(x)$	$F_\theta(x)$	$D_n(x)$
2	0.1	0.16	0.06
6	0.2	0.41	0.21
7	0.3	0.46	0.16
8	0.4	0.5	0.1
10	0.6	0.58	0.02
12	0.7	0.65	0.05
16	0.8	0.75	0.05
20	0.9	0.82	0.08
24	1	0.88	0.12

Función de distribución exponencial

$$F_0(x) = 1 - e^{-\lambda x}$$

Estimamos el parámetro a partir de la media muestral:

$$\frac{16+8+12+\ldots+24}{10} = 11.5 = 1/\lambda$$

$D_n = 0.21 < D(\alpha=0.2, n=10) = 0.322$

Aceptamos la hipótesis

Contraste específico para el uniforme

Por debajo utiliza el X**2 (Chi cuadrado)

En las prácticas se necesita este contraste

- Agrupamos la muestra de 40 en 40 (sin ordenar)
- Calculamos el nivel de discrepancia entre lo observado y lo teórico en cada grupo
- Si es uniforme, comprobar que estos valores
- Hacer el contraste (Chi Cuadrado o Kolgomorov-Smirnov) para comprobar que estos valores encajan con una Chi Cuadrado

Contrastes de Aleatoriedad o Independencia estadística

Contraste de rachas

Generamos una muestra de nuestro generador

Secuencia de números: x1 .. xn

Convertimos en una sucesión binaria comparando un elemento con el siguiente. 1 si es menor (crece la racha) y 0 si es mayor (decrece la racha)

0's ➜ Racha decreciente

1's Crecientes

Contamos cuantas rachas hemos tenido en nuestra muestra

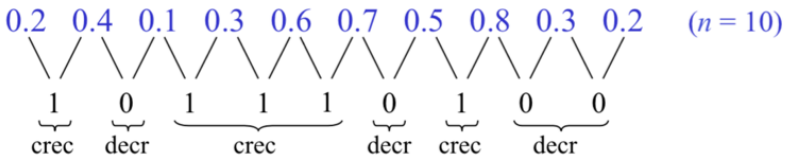
Calculamos varias muestras y varias rachas

El número de rachas de cada muestra, debe ser una normal con media y varianza estas

$$\mu = \frac{2n-1}{3} \qquad \sigma^2 = \frac{16n-29}{90}$$

Ejemplo. Contrastar la aleatoriedad de la siguiente secuencia de números: 0.2, 0.4, 0.1, 0.3, 0.6, 0.7, 0.5, 0.8, 0.3, 0.2.

Construimos la secuencia de símbolos binarios para contabilizar las rachas:



El número total de rachas es $n_r = 6$. La distribución del número de rachas en una muestra de tamaño 10 debería ser Normal(6.3, 1.45).

$$P\left(\left|z_{N(6.3,1.45)}\right| \geq 6\right) = P\left(z_{N(6.3,1.45)} \leq -6\right) + P\left(z_{N(6.3,1.45)} \geq 6\right) =$$

$$P\left(z_{N(0,1)} \leq \frac{-6-6.3}{\sqrt{1.45}}\right) + P\left(z_{N(0,1)} \geq \frac{6-6.3}{\sqrt{1.45}}\right) =$$

$$0 + P\left(z_{N(0,1)} \geq -0.25\right) = 1 - P\left(z_{N(0,1)} \leq -0.25\right) =$$

$$1 - 0.4030 = 0.5987$$

La probabilidad teórica de que el número de rachas en la secuencia sea mayor que 6 es 0.5987, por lo que rechazamos la hipótesis de aleatoriedad.

Contraste2: Contraste de rachas por encima y por debajo de la mediana

- Convertimos en sucesión binaria mirando la mediana. 1 si es menor y 0 si es mayor

$$\mu = k + 1 \qquad \sigma^2 = \frac{k(k-1)}{2k-1}$$

TEST DE PÓKER

> Test de póker es el más potente y es el que usaremos
> Contraste muy específico para contrastar generadores de números aleatorios.

- Transofrmamos nuestra secuencia original en enteros (del 1 al 10)
- Agrupamos en grupos de 5. Para tener frecuencias acumuladas
  - cada clase son las jugadas de póker
  - La probabilidad de cada jugada es conocida por Teoría de la probabilidad. Probabilidad de póker, de full, de parejas, tríos … etc
- Para que funcione muy bien , tenemos que tener al menos 5 repeticiones en cada clase
  - Si el repoker, tenemos menos que juntar esta clase con la del póker
  - Esto es habitual para muestras menores de 250K
-

Ejemplo. Dada la siguiente muestra de 1500 números, realizar el contraste de Póker de aleatoriedad:

0.12 0.23 0.78 0.84 0.16 0.22 0.29 0.47 0.95 0.13 0.33 0.27 0.42 0.32 0.69 0.72 0.17 0.22 0.09 0.81 …

2 3 8 9 2 | 3 3 5 10 2 | 4 3 5 4 7 | 8 2 3 1 9 …

AABCD AABCD AABCD ABCDE …

Clases	Frec. observadas	Frec. Esperadas
AAAAA	3	$n \times p_{AAAAA} = 300 \times 0.0001 = 0.3$
AAAAB	5	$n \times p_{AAAAB} = 300 \times 0.0045 = 1.35$
AAABB	5	$n \times p_{AAABB} = 300 \times 0.0090 = 2.7$
AAABC	25	$n \times p_{AAABC} = 300 \times 0.0720 = 21.6$
AABBC	27	$n \times p_{AABBC} = 300 \times 0.1080 = 32.4$
AABCD	170	$n \times p_{AABCD} = 300 \times 0.5040 = 151.2$
ABCDE	65	$n \times p_{ABCDE} = 300 \times 0.3024 = 90.72$

$$X^2 = \frac{(8-1.65)^2}{1.65} + \frac{(5-2.7)^2}{2.7} + \frac{(25-21.6)^2}{21.6} + .. + \frac{(65-90.72)^2}{90.72} = 24.43 + 1.959 + 0.5351 + ... + 7.314 = 37.46$$

El número de grados de libertad es k-r-1 = 6-0-1 = 5 y de la tabla de la distribución de la χ^2 tenemos $\chi^2_{0.995}(5) = 16.7 < X^2 = 37.46$, por lo tanto, **aceptamos la hipótesis**.

a03-03 - Números aleatorios 2

Antonio estudió en el plan 92

Recapitulación

Bondad de ajuste con dos

Aleatoriedad (Rachas, Test de póker)

Generadores congruenciales

Generadores algorítimicos recursos.

Punto de partida de números aleatorios.

Los siguientes generadores intentan solucionar las 2 carencias:

- Longitud de secuencia finita ➜ Periodo
- Secuencia, no es totalmente aleatoria

Fórmula recursiva generadores congruenciales

- a: multiplicador
- b: sesgo
- m: módulo
- xn:

> xn+1 = (a*xn +b) mod m
> [0, m)

- nunca podremos tener una secuencia mayor al módulo `m`
- Para transformar estos valores al intervalo 0,1 , tenemos que dividir por m

- Si b = 0 ➜ generadores multiplicativos
  - **Cuando son multiplicativos, el periodo máximo será** `m-1`

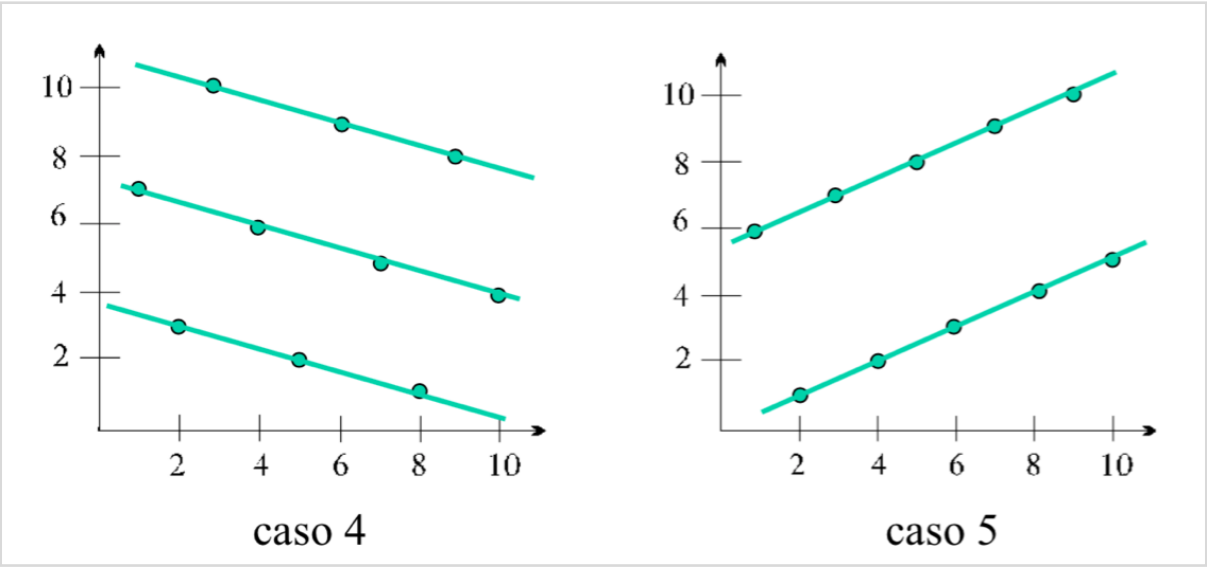
El generador debe ser rápido, por ello las operaciones deberían ser sencillas.

La operación módulo se puede realizar de forma eficiente

```
def generadorCongruencial(x0=0, a=3939, m=2**31-1, b=0):
  return (a*x0+b) % m
```

Estas semirectas nos indican cuál es más aleatorio
Más rectas, más aleatorio.

Número finito de planos ➜ Estructura reticular; En más dimensiones se llama hiperplanos.



Observaciones:

1. Un generador congruencial tiene ciclos.
2. La longitud del ciclo depende de la selección de los parámetros (comparar casos 1 y 3).
3. Dentro de selecciones de parámetros que conducen a una misma longitud de ciclo, algunas salidas parecen más aleatorias que otras (comparar casos 1 y 2).
4. La representación de los pares (*xi*, *xi*+1) sugiere que éstos se disponen en un número finito de rectas (representación de tuplas ➜ hiperplanos).

Propiedades que se deben cumplir para tener un buen generador

Comprobar para cualquier generador si tiene periodo máximo.

Esta proposición valida la longitud de periodo, no nos permite elegir buenos valores.

> **Proposición**: Un generador congruencial tiene periodo máximo *m*si y sólo si:
> 1) mcd (*b*, *m*) = 1;
> 2) *a*= 1 mod *p*para cada factor primo *p*de *m*;
> `a −1 es múltiplo de p`
> 3) *a*= 1 mod 4 si 4 divide a *m*.

Para conseguir una *Eficiencia computacional* el módulo debería ser `∗m∗= 2β` . La operación sería retener una cantidad de bits.

En la práctica, se utiliza `∗m∗= 2β−1` o `∗m∗= 2β +1` para evitar que sea div 4.

> **Proposición**. Un generador multiplicativo con módulo *m*= 2β ≥ 16 tiene periodo máximo *m*/4 si y sólo si *a*mod 8 = 3 ó *a*mod 8 = 5 y *x*0 es impar.

b=0 , es lo mismo, pero nos reduce el

m debe ser primo y lo más grande posible

m debería ser el número primo más grande que se pueda representar en nuestra aritmética de CPU (16, 32 o 64 bits)

Los más frecuentes serán: `2^31−1` y `2^16+1` que además permite la eficiencia computacional

a debe ser una raíz primitiva de m-1

Proposición. Un generador multiplicativo tiene periodo *m*-1 sólo si *m*es primo. El periodo divide a *m*-1 y es *m*-1 si y sólo si *a*es una raíz primitiva de *m*-1, es decir, $a \neq 0$, $a(m\text{-}1)/p \neq 1 \bmod m$, para todos los factores primos *p*de *m*-1.

Si encontramos una raíz primitiva, calcular las siguientes raíces es trivial.

Proposición. Si *a*es una raíz primitiva de *m*, *ak*mod *m*lo es siempre que mcd(*k*, *m*-1) = 1.

Pueden haber muchas raíces.

Distintos autores han analizado las distintas raíces primitivas para calcular su aleatoriedad

Se estudia por distancia espectral de los hiperplanos

Marsaglia, calculó una cuota superior

Número de hiperplanos aleatorios

	$n=3$	$n=5$	$n=7$	$n=9$	$n=10$
$m=2^{16}$	73	23	16	14	13
$m=2^{32}$	2953	220	80	48	41

RANDU de IBM era muy mal generador

Fishman y Moore (1986)

Analizaron todas las raíces primitivas de m 23093, identificaron 410 multiplicadores

Fishman posteriormente encontró a=16807 (generador congruencial estándar), a=48271 y a=69621 (estos dos eran mejores que el primero, pero fueron menos utilizados)

Generador congruencial estándar

Arquitecturas de 32 bits, Park y Miller (1988) → mínimo estándar:

1. Ser de periodo máximo;

2. Que su salida parezca aleatoria;
3. Que se pueda implementar de forma eficiente en aritmética de 32 bits.

```
function aleator (isemilla)
  isemilla= mod (isemilla × 16807.d0, 2147483647.d0)
  aleator = isemilla / 2147483647
  return
end
```

isemilla*a puede dar un error de desboradamiento

Método de Schrage (1983) (Evitando desbordamiento)

se pueden reordenar las operaciones para evitar el desbordamiento

```
function aleato2 (isemilla)
  k = isemilla/127773
  isemilla = 16807 ×(isemilla - k × 127773) - 2836 × k
  if (isemilla.lt0)    isemilla = isemilla + 2147483647
  aleato2 = 1./2147483647 × isemilla
  return
end
```

Si nuestra simulación es muy compleja, y necesitamos secuencias mayores de (2147483647 -1), necesitaremos otro generador. para el resto se considera un buen generador.

Otros generadores

Generadores de registro de desplazamiento (aumenta el nivel de recursividad) - L'ecuyer

Para proponer generadores, es fundamental expresar el periodo de forma matemática. Normalmente basados en polinomios característicos. (Algebra finita de n elementos)

(Paper de referencia para hacer el trabajo y la presentación)

Si aumento el nivel de recursividad

Si el polinomio es primitivo ->El periodo de generador es `m^k −1` (muchísimo más grande que m-1). Lo malo es que no aporta solución par aletoriedad

Nos dará secuencia de número binarios aleatorios
k = **número de retardos tomados**

Parece ser que estos generadores tienen buenas propiedades de aleatoriedad, sin embargo, producen estructuras reticulares, como los congruenciales, lo que ha llevado a una cierta polémica y disuasión sobre su calidad.

Ejemplo. Tomando $m = 2$ tenemos el generador $x_n = (a_1 x_{n-1} + \ldots + a_k x_{n-k}) \bmod 2$, donde $a_1, \ldots, a_k \in \{0,1\}$, que es equivalente a reescribir

$$x_n = x_{n-j1} \text{ XOR } x_{n-j2} \text{ XOR } \ldots \text{ XOR } x_{n-jk},$$

cuando $a_{n-j1} = a_{n-j2} = \ldots = a_{n-jk}, = 1$ y $a_j = 0$ para los otros j.

Una elección habitual es escoger polinomios de la forma $P(z) = z^k - z^{k-r} - 1$ o equivalentemente $x_n = x_{n-r} \text{ XOR } x_{n-k}$

Valores habituales para k y n (garantizan un periodo máximo):

$$\left\{ \begin{array}{l} k = 607 \quad r = 273 \\ x_n = x_{n-273} \text{ XOR } x_{n-607} \end{array} \right\} \text{ o } \left\{ \begin{array}{l} k = 521 \quad r = 32 \\ x_n = x_{n-32} \text{ XOR } x_{n-521} \end{array} \right\}$$

NO NOS GARANTIZA ALEATORIEDAD

Generador de Tausworthe

Coge secuencias binarias en valores entre 0 y 1 , distribuidos uniformemente.

$$u_i = \sum_{s=1}^{l} 2^{-s} b_{it+s}$$

girador de Mersenne: (práctica)

Generadores de Fibonacci Retardados

La operación binaria suele ser +,-, * , XOR
El periodo máximo está relacionado al análisis de sucesiones lineales repulsivas de enteros.

Marsaglia

Da un periodo , fácil de implementar
Problema de estructura reticular.

Son operaciones lineales : `axn+b` son generadores recursivos lineales

Generadores no lineales

La no linealidad se puede introducir en dos puntos:

1. Usar un generador con función de transición lineal, produciendo la salida mediante una transformación no lineal del estado. (función de transición)
2. Usar un generador con función de transición no lineal. (transformación)

Es difícil demostrar su periodo.

Método congruencial de inversión explícita

Es muy bueno en aleatoriedad (totalmente aleatorio), pero no mejora el periodo que sigue siendo m.

Método congruencial de inversión explícita

Sea $m \geq 5$ un número primo y $F_m = \{0,1,\ldots,m\text{-}1\}$ el álgebra finita de orden m.

Para un entero z, se define $\overline{z} \in F_m$,

$$\overline{z} \equiv z^{m-2} \bmod m$$

que es la inversa de z para la multiplicación en F_m, si $\overline{z} \neq 0 \bmod m$.

Existen algunos algoritmos eficientes para su cálculo.

Dados $a, b \in F_m$, $a \neq 0$, la sucesión es

$$y_n = \overline{a\,n + b} \qquad n \geq 0$$

Su periodo máximo es m.

Secuencia totalmente aleatoria, pero solo de longitud m

Determinar bajo que circunstancias se alcanza el periodo máximo, es muy complicado

AES , es mejor que el anterior y se usa mucho para encriptación

Combinación de generadores

Combinar secuencias para aumentar la aleatoriedad y el periodo.
Lo difícil es demostrar matemáticamente el periodo.

- BARAJEO
  - permutación fija (barajeo): composición, no es muy buena
  - permutación aleatoria (barajeo): desordenar aleatoriamente la secuencia
    - primer secuencia nos da resultados
    - segunda secuencia es el índice para identificar qué valor es el que se devuelve

Ejemplo (permutación aleatoria, barajeo)

Partimos de dos sucesiones aleatorias:

(u_n) → 0.7 0.2 0.32 0.84 0.25 0.12 0.33 0.47 0.84 0.72 … **Valores**

(v_n) → 0.1 0.7 0.23 0.42 0.35 0.21 0.47 0.72 0.68 0.12 … **Índices**

Tomamos $k = 5$ y hacemos

$T(0) = u_1 = 0.7$ $T(1) = u_2 = 0.2$ $T(3) = 0.32$ $T(4) = 0.84$ $T(5) = 0.25$

Tomamos valores de la segunda secuencia aleatoria y construimos los índices que me señalarán los elemento de T que debemos devolver, a la vez que actualizo T:

v_1=0.1→ j = ent(6×0.1)=0.6 → j = 0→ el valor que debemos devolver es $T(0)$ = 0.7. Hago $T(0) = u_6 = 0.12$

v_2=0.7→ j = ent(6×0.7)=4.2 → j = 4→ el valor que debemos devolver es $T(4)$ = 0.84. Hago $T(1) = u_7 = 0.33$

Composición

Tengo N generadores, cojo el resultado y realizo una operación entre ellos.

MRG32k3a ➜ está en una práctica para hacer en memoria

Generadores paralelos

En simulaciones que ha falta paralización:

Distintos hilos de ejecución necesitan números aleatorios, ¿Cómo los distribuyo?

Va en contra de la propiedad de **reproducibilidad**

En la práctica se utiliza un mismo generador con semillas distintas por cada hilo de ejecución.

Generadores comerciales

> Artículo de Park y Miller (1988) como advertencia sobre la mala calidad de algunos generadores comerciales.

ISML implementa generadores multiplicativos de módulo m= 231-1 y multiplicadores a= 16807, 397204094 y 950706376.

> Libro muy bueno: Numerical recipes 1992, 2007
>
> Bueno para ingenieros e informáticos

Hay algoritmos implementados y eficientes

en la mayoría de lenguajes y herramientas estadísticas se utilizan congruenciales

Contrastes de aleatoriedad modernos

Test de póker, Huecos, rachas, permutaciones. .Disponibles en cualquier entorno estadísticos. Son muy básicos para validar a un generador.
Estos contrastes Siempre dirían que los generadores son “buenos”

Contraste espectral

Analiza la estructura reticular de los generadores, calculando la distancia de los hiperplanos paralelos en diferentes dimensiones.
Para validar contrastes hace falta un problema de optimización.
Como es una fórmula cuadrática y no lineal, es difícil calcular el óptimo
Hay heurísticas para aproximar la solución.

Baterías de test de aleatoriedad

15 o 20 contrastes , con que uno diga que no sea aleatorio, el generador no es aleatorio.
Para la práctica
www.stat.fsu.edu/pub/diehard

DIE HARD Randomness Test Suite : programa de estudio de aleatoriedad

NIST (National institute of standard and technology)

Se usa más en criptografía en vez de en simulación.

Tuftest

- Prueba
  -
  - cumpleaños
  - de gorila

TESTU01

simul.iro.umontreal.ca/testu01/tu01.html
Otras baterías

La calidad hay que revisar

Los tests permiten 12MB , y los estudios suelen necesitar 40 muestras.

Información adicional

Dos sitios excelentes y actualizados con abundante información sobre números aleatorios son las páginas de Hellekalek http://random.mat.sbg.ac.at y de L'Ecuyer http://www.iro.umontreal.ca/~lecuyer.

Hay varios sitios que proporcionan generadores verdaderos de números aleatorios.

- http://www.random.org, **que usa ruido atmosférico;**
- http://random.hd.org/index.html (Java EntropyPool), que emplea ruido proveniente de entradas en distintos sitios web y otros dispositivos físicos.

Las revistas *Communications of ACM*y *Applied Statistics*publican a menudo nuevos algoritmos de generación de números aleatorios. En particular, los de esta última, pueden obtenerse de Statlib en http://lib.stat.cmu.edu/.

> Para repetir un experimento, necesito almacenar toda la secuencia aleatoria completa de

> ACM y Applied statics, cada 4 años saca estados de arte de generación de números aleatorios, y demás técnicas

La librería científica GNU tiene una sección entera dedicada a números aleatorios http://www.gnu.org/software/gsl/manual/html$_$node/Random-Number- Generation.html.

Referencias

hay papers que habrá que mirar 1 o dos papers para realizar el paper

Siguiente sesión veremos

Parámetros aleatorios.
Determinaremos la distribución mediante el contraste de hipótesis

Generaremos valores de esa distribución

X~Distrib(param)

> Clave: operaciones que tenga que realizar, no disminuya mi periodo

Para cada distribución que quiera generar, necesitaré varios valores aleatorios, por lo que el periodo ya no será `m-1`

> **Teorema central del límite**
> Si tenemos un buen generador de la normal, podremos generar muchas distribuciones.

índice