IFT 3245
Simulation et modèles

Fabian Bastin
DIRO
Université de Montréal

Automne 2016

Tests statistiques

L'étude des propriétés théoriques d'un générateur ne suffit; il est indispensable de recourir à des tests statistiques pour valider celui-ci.

Nous formulons l'hypothèse

$\mathcal{H}_0$: "$\{u_0, u_1, u_2, \ldots\}$ sont des variables aléatoires i.i.d. $U(0,1)$".

Nous savons que $\mathcal{H}_0$ est fausse, mais comment pouvons-nous le détecter?

Pour ce faire, définissons une statistique T, fonction de u_i, dont la distribution sous $\mathcal{H}_0$ est connue (ou approximativement). Nous rejetons $\mathcal{H}_0$ si la valeur de T est trop extrême.

Tests statistiques

La puissance et l'efficacité du test dépendent fortement de la classe d'alternatives. Différents tests détectent des problémes différents.

L'idéal serait de disposer d'une statistique T qui imite la variable aléatoire d'intérêt pratique. Ce n'est cependant guére réalisable en pratique.

Impossible de construire un générateur de variables aléatoires qui passerait tous les tests. Compromis: construire un générateur qui passe le plus de tests raisonnables.

La plupart des tests étudient la répartition uniforme d'un ensemble ou sous-ensemble quelconque de points produits par le générateur.

Tests statistiques

Avoir une bonne couverture uniforme ne suffit toutefois pas: les points générés doivent avoir l'apparence d'indépendance les uns par rapport aux autres.

Les tests d'uniformité, qui représente la majeure partie des tests à notre disposition, remplissent partiellement ce second objectif dans la mesure où ils sont validés pour des sous-ensembles de points très variés.

Certains tests cependant sont spécifiquement conçus pour tester les corrélations entre les points produits par un générateur.

TestU01

Le logiciel TestU01 (`http://www.iro.umontreal.ca/~simardr/testu01/tu01.html`), implémente un nombre très important de tels tests Nous nous limiterons à quelques exemples afin d'illustrer les idées sous-jacentes.

Test de collisions

Partitionnons l'hypercube $[0,1)^t$ en $k = d^t$ boîtes cubiques de tailles égales. Générons n points $\boldsymbol{u}_i = (u_{ti}, \ldots, u_{ti+t-1})$ dans $[0,1)^t$, et posons X_j le nombre de points dans la boîte j.

Le nombre de collisions est donné par

$$C = \sum_{j=0}^{k-1} \max(0,\, X_j - 1).$$

Sous $\mathcal{H}_0$, C suit approximativement une loi de Poisson de moyenne $\lambda = n^2/(2k)$, pour n suffisamment grand, et un petit λ.

Test de collisions

Si nous observons c collisions, nous calculons la p-valeur à droite comme

$$p^+(c) = P[X \geq c \mid X \sim \text{ Poisson}(\lambda)].$$

Nous rejetons $\mathcal{H}_0$ si $p^+(c)$ est de manière consistante très proche de 0 (trop de collisions) ou de 1 (pas assez de collisions).

Une interprétation possible est l'étude par simulation Monte Carlo du comportement probabiliste d'un algorithme de hachage.

Espacement entre anniversaires

Nous partitionnons à nouveau l'hypercube $[0,1)^t$ en $k = d^t$ boîtes cubiques, et générons n points.

Soit $I_{(1)} \leq I_{(2)} \leq \cdots \leq I_{(n)}$ les numéros de boîte où les points tombent, triés par ordre croissant.

Nous calculons les espacements

$$S_j = I_{(j+1)} - I_{(j)},\ 1 \leq j \leq n-1.$$

Le nombre de collisions entre les espacements est défini comme

$$Y = \sum_{j=1}^{n-1} I\left[S_{(j+1)} = S_{(j)}\right].$$

Espacement entre anniversaires

Pour k suffisamment grand, Y suit approximativement une loi de Poisson de moyenne $\lambda = n^3/(4k)$.

Si Y prend la valeur y, la p-valeur à droite est

$$p^+(y) = P[X \geq y \mid X \sim \text{ Poisson}(\lambda)].$$

Interprétation comme problème de simulation: nous pouvons vouloir étudier la distribution de Y par simulation Monte Carlo, pour des valeurs modérées de k et n.

Test d'autocorrélation

Générons n uniformes $u_1, \ldots, u_n$ et calculons l'autocorrélation empirique de distance k:

$$\hat{\rho}_k = \frac{1}{n-k} \sum_{j=1}^{n-k} \left(u_j u_{j+k} - \frac{1}{4} \right).$$

La distribution empirique de N valeurs de

$$\sqrt{12(n-k)}\hat{\rho}_k$$

est comparée avec la distribution normale standard, qui est la distribution théorique asymptotique quand $n \to +\infty$.

L'approximation n'est valide que pour n très grand, et avec k très petit en comparaison de n.

Quelques générateurs largement utilisés

Avant 2003, le générateur dans MS Excel était

$$u_i = (9821.0\, u_{i-1} + 0.211327) \mod 1.$$

Le générateur est à présent une combinaison de trois GCLs simplistes. A la même période, le générateur dans MS VisualBasic était

$$\begin{aligned} x_i &= (1140671485\, x_{i-1} + 12820163) \mod 2^{24}, \\ u_i &= x_i / 2^{24}. \end{aligned}$$

Le générateur dans `java.util.Random` de Java est quant à lui

$$\begin{aligned} x_{i+1} &= (25214903917\, x_i + 11) \mod 2^{48} \\ u_i &= [2^{27} \left(x_{2i} \mod 2^{26} \right) + x_{2i+1} \bmod 2^{27}]/2^{53} \end{aligned}$$

Résultats de tests: test de colisions

Considérons $t = 2$, $d = n/16$, $\lambda = 128$.

n	Java		VisualBasic		Excel	
	c	$p^+(c)$	c	$p^+(c)$	c	$p^+(c)$
2^{14}					132	
2^{15}			75	$1 - 3.1 \times 10^{-7}$	128	
2^{16}			38	$> 1 - 10^{-15}$	121	
2^{17}			0	$> 1 - 10^{-15}$	170	2.2×10^{-4}
2^{18}			0	$> 1 - 10^{-15}$	202	9.5×10^{-10}
2^{19}			0	$> 1 - 10^{-15}$	429	$< 10^{-15}$

Résultats de tests: test de colisions

Si on rejette les dix premiers bits de chaque u_i: on prend $u_i := 1024u_i \mod 1$.

n	Java		VisualBasic		Excel	
	c	$p^+(c)$	c	$p^+(c)$	c	$p^+(c)$
2^{14}			8192	$< 10^{-15}$		
2^{15}			24576	$< 10^{-15}$		
2^{16}			57344	$< 10^{-15}$		
2^{17}			122880	$< 10^{-15}$		
2^{18}			253952	$< 10^{-15}$	224	1.0×10^{-14}
2^{19}	160	3.5×10^{-3}	516096	$< 10^{-15}$	425	$< 10^{-15}$

Espacements d'anniversaire

Considérons $t = 2$, $\lambda = 1$.

n	d	Java		VisualBasic		Excel	
		y	$p^+(y)$	y	$p^+(y)$	y	$p^+(y)$
2^8	2^{11}						
2^{10}	2^{14}			10	1.1×10^{-7}		
2^{12}	2^{17}			592	$< 10^{-15}$	5	3.7×10^{-3}
2^{14}	2^{20}			11129	$< 10^{-15}$	71	$< 10^{-15}$
2^{16}	2^{23}			64063	$< 10^{-15}$	558	$< 10^{-15}$
2^{18}	2^{26}	14	4.5×10^{-12}	261604	$< 10^{-15}$	4432	$< 10^{-15}$

Espacements d'anniversaire

n	d	LCG-16807	
		y	$p^+(y)$
2^{10}	2^{14}		
2^{12}	2^{17}	2	
2^{14}	2^{20}	170	$< 10^{-15}$
2^{16}	2^{23}	10060	$< 10^{-15}$

Espacements d'anniversaire

Et en rejetant les 10 premiers bits:

n	Java		VisualBasic		Excel	
	y	$p^+(y)$	y	$p^+(y)$	y	$p^+(y)$
2^8			52	$< 10^{-15}$		
2^9			199	$< 10^{-15}$		
2^{10}			672	$< 10^{-15}$		
2^{11}			1754	$< 10^{-15}$		
2^{12}			3901	$< 10^{-15}$		
2^{13}			8102	$< 10^{-15}$		
2^{14}			16374	$< 10^{-15}$		
2^{15}	21	6.0×10^{-15}	32763	$< 10^{-15}$		
2^{16}	99	$< 10^{-15}$	65531	$< 10^{-15}$	7	4.5×10^{-3}
2^{17}	697	$< 10^{-15}$	—	—	34	$< 10^{-15}$
2^{18}	639	$< 10^{-15}$	—	—	186	$< 10^{-15}$

Tests systématiques pour des familles de générateurs de nombres aléatoires

Pour une famille de générateurs de nombres aléatoires, on cherche une relation su type [$n_0 \approx K\rho^{\gamma}$,] où n_0 est la taille d'échantillon minimum pour obtenir un fort rejet, ρ est la longueur de période, et K et γ sont des constantes.
Pour des GCLs, on obtient (grossièrement) pour le test de collisions:

$$n_0 \approx 16\rho^{1/2},$$

et pour le test d'espacement entre anniversaires:

$$n_0 \approx 16\rho^{1/3}.$$

Nous souhaitons dès lors construire le générateur de nombres aléatoires avec ρ suffisamment grand de sorte que générer n_0 nombres est infaisable en pratique.

Librairie de tests

Tests effectués sur la plupart des générateurs existants.

Pratiquement aucun des générateurs présent dans les logiciels commerciaux ne passe tous les tests. Vrai en particulier pour les générateurs modulo-2.

Les vainqueurs sont des MRG avec une bonne période et une bonne structure, des générateurs multiplicatifs “lagged-Fibonacci”, des générateurs non-linéaires conçus pour la cryptologie, et certains générateurs avec des composantes venant de différentes familles.

Librairie de tests

Il est possible de combiner efficacité empirique et théorique, comme l'illustre le MRG32k3a.

Note: générateur multiplicatifs "lagged-Fibonacci". Forme de base

$$x_n = x_{n-r} * x_{n-k} \bmod m.$$

Il a moins de propriétés mathématiques connues, difficultés du choix de r et de k, initialisation complexe. m est souvent pris comme une puissance de 2.