Générateur de nombres aléatoires

Générateur de nombres aléatoires

- Générateur de nombres aléatoires (random number generator, RNG) est un dispositif informatique ou physique conçu pour générer une séquence de nombres qui semblent aléatoires.
- « Aléatoire » signifie qu'ils ne présentent aucun motif perceptible, peu importe combien d'efforts nous avons mis en trouver un.
- Autrement dit, les nombres générés successivement ne peuvent pas être prédits.

Exemples

- Méthodes mécaniques : les méthodes génération de nombres aléatoires
  - Lancement de dés.
  - Renversement de pièces.
  - Mélange de cartes à jouer

  Problème d'efficacité: la génération de grandes quantités de nombres suffisamment aléatoires nécessite une grande quantité de travail et/ou de temps.
- Tables de nombres aléatoires: une méthode alternative est de collecter et transmettre les résultats de ces expériences

Génération par ordinateur

- Avec ordinateur, il existe plusieurs techniques pour générer rapidement des «nombres aléatoires».
- Cependant, les ordinateurs sont des machines déterministes et ne peuvent pas agir de manière aléatoire.
- Pourtant, ils peuvent générer des nombres d'une manière si complexe que, à toutes fins utiles, les nombres successifs sont imprévisibles sans motif discernable.

Nombres aléatoires standards

- Deux propriétés statistiques importantes :
  - Uniformité
  - Indépendance
- Un nombre aléatoire (standard) R doit être un échantillon d'une loi uniforme $U(0,1)$.

Génération de nombres pseudo-aléatoires

- « Pseudo » parce que la génération de nombres en utilisant un procédé connu élimine le potentiel de l'aléatoire véritable.
- Objectif: produire une séquence de nombres dans [0, 1] qui simule ou imite les propriétés idéales de nombres aléatoires $U(0,1)$.
- Facteurs importants dans les routines de nombres aléatoires:
  - Rapidité
  - Portabilité à différents ordinateurs
  - Avoir un cycle suffisamment long
  - Réplicable
  - Approche rapprochée des propriétés statistiques idéales d'uniformité
  - Indépendance

Techniques

- Méthode de congruence linéaire (Linear Congruential Method, LCM)
- Méthode de congruence linéaire combinée (Combined Linear Congruential Method, CLCM)
- Flux de nombres aléatoires (Random-Number Streams, RNS)

Générateur congruentiel linéaire

- Une séquence d'entiers $X_1, X_2 \ldots\ldots$ est obtenue par l'opération récursive

$$X_{i+1} = (aX_i)\ mod\ m, \qquad i = 0,1,2,\ldots$$

  - m est le module,
  - $0 < a < m$ est le multiplicateur
  - X_0 est le seed
- Expression équivalente

$$X_{i+1} = aX_i + mK_{i+1}, \qquad i = 0,1,2,\ldots$$

avec

$$K_{i+1} \coloneqq \lfloor aX_i/m \rfloor$$

Générateur congruentiel linéaire

- Reproduction :

$$X_{i+1} = aX_i + m\lfloor aX_i/m \rfloor$$

  - X_i est le seed de X_{i+1}
  - Il suffit de partir avec le même X_0
- Nombre des valeurs différentes $\leq m - 1$
- Période

$$P(X_0, a, m) \coloneqq \min(n \geq 1 : X_n = X_0)$$

Générateur congruentiel linéaire

- $X_0 = 1\ m = 11$

i\a	1	2	3	4	5	6	7	8	9	10
0	1	1	1	1	1	1	1	1	1	1
1	1	2	3	4	5	6	7	8	9	10
2		4	9	5	3	3	5	9	4	1
3		8	5	9	4	7	2	6	3	
4		5	4	3	9	9	3	4	5	
5		10	1	1	1	10	10	10	1	
6		9				5	3	3		
7		7				8	2	2		
8		3				4	5	5		
9		6				2	7	7		

Générateur congruentiel linéaire mixte

- Une séquence d'entiers $X_1, X_2, \ldots$ entre 0 et $m - 1$ est obtenue par l'opération récursive

$$X_{i+1} = (aX_i + c)\ mod\ m, \qquad i = 0,1,2,\ldots$$

  - m est le module,
  - $0 < a < m$ est le multiplicateur
  - $0 \leq c < m$ est l'incrément

- Le choix de a, c, m et X_0(seed) affecte considérablement les propriétés statistiques de la longueur du cycle.

Générateur congruentiel linéaire mixte

- Les entiers aléatoires sont générés dans $[0, m - 1]$, de manière à convertir les nombres entiers de nombres aléatoires entre $[0,1]$:

$$R_i = \frac{X_i}{m}, \qquad i = 1,2,\ldots.$$

- $c = 0 \Rightarrow$ un générateur congruentiel linéaire.

Exemple

- $X_0 = 27, a = 17, c = 43, m = 100$
  - $X_1 = (17 * 27 + 43)\ mod\ 100 = 502\ mod\ 100$
  - $R_1 = 0{,}02$
  - $X_2 = (17 * 2 + 43)\ mod\ 100 = 77$
  - $R_2 = 0{,}77$
  - $X_3 = (17 * 77 + 43)\ mod\ 100 = 1352\ mod\ 100 = 52$
  - $R_3 = 0{,}52$
  - ……….

Critères d'un bon générateur

- Maximum de densité
  - Pas d'écart entre les values de R_i
  - Problème : R_i sont discrets
  - Solution: un très large m
- Maximum de Période
  - Éviter répétition
  - Solution: a, c, m, X_0 bien choisis selon la valeur de m
- Efficacité
  - Représentation de nombres en binaire => $m = ou \approx 2^n$

Méthode congruentiel linéaire combinée

- Combiner plusieurs générateurs à congruences linéaires
- Soient $X_{i,1}, X_{i,2}, \dots, X_{i,k}$ les $i$ème sorties des k générateurs à congruences linéaires (multiplicatifs):
- Le $j$ème générateur doit:
  - Avoir un modulo primaire m et un multiplieur a_j => période $m_j - 1$
  - $X_{i,j}$ approximativement uniformément distribuée en $[1, m-1]$

Méthode congruentiel linéaire combinée

- $X_i = \left(\sum_{j=1}^{k} (-1)^{j-1} X_{i,j}\right) mod\ (m_1 - 1)$
- $R_i = \begin{cases} \dfrac{X_i}{m_1} si\ X_i > 0 \\ \dfrac{m_1 - 1}{m_1} si\ X_i = 0 \end{cases}$
- Période maximale

$$p_{max} = \frac{(m_1 - 1)(m_2 - 1)\dots(m_k - 1)}{2^{k-1}}$$

Exemple

Ordinateur de 32-bits, $k = 2$ avec

$$m_1 = 2147483563$$
$$a_1 = 40014$$
$$m_2 = 2147\,483\,399$$
$$a_2 = 40692$$

Période combinée: $\frac{(m_1-1)(m_2-1)}{2} \approx 2.10^{18}$

- Choisir les seeds
  - $X_{1,0}$ dans [1, 2147483562]
  - $X_{2,0}$ dans [1, 2147483398]

Exemple

Boucle:

- Pour chaque générateur
  - $X_{1,j+1} = 40014\ X_{1,j}\ mod\ 2147483563$
  - $X_{2,j+1} = 40692\ X_{2,j}\ mod\ 2147483399$
- $X_{j+1} = (X_{1,j+1} + X_{2,j+1})\ mod\ 2147483562$
- $R_{j+1} = \begin{cases} \frac{X_{j+1}}{2147483563} si \quad X_{j+1} > 0 \\ \frac{2147483562}{2147483563} si \quad X_{j+1} = 0 \end{cases}$
- $j = j + 1$

Flux de nombres aléatoires

- La seed d'un générateur de nombres aléatoires à congruences linéaires :
  - Est la valeur entière X_0 qui initialise la séquence de nombres aléatoires.
  - Toute valeur dans la séquence peut être utilisée pour 'commencer' le générateur.
- Un flux de nombres aléatoires:
  - Un seed de départ prise dans la séquence X0, X1,. . . , Xp.
  - Si les flux sont séparés par des valeurs b, alors le flux i pourrait être défini par commençant à la seed

$$S_i = X_{b(i-1)}.$$

  - Les générateurs plus anciens avaient $b = 10^5$, les générateurs plus récents ont $b = 10^{37}$.

Flux de nombres aléatoires

- Un seul générateur de nombres aléatoires avec k flux peut agir comme k distinct générateurs de nombres aléatoires virtuels.

Test de nombres aléatoires

- Test de uniformité

$$H_0: R_i \sim U[0,1]$$
$$H_1: R_i \nsim U[0,1]$$

Le fait de ne pas rejeter l'hypothèse nulle H_0 signifie que la preuve de la non-uniformité n'a pas été détectée.

- Test d'indépendance

$$H_0: R_i \sim indépendante$$
$$H_1: R_i \nsim indépendante$$

Le fait de ne pas rejeter l'hypothèse nulle H_0 signifie qu'aucune preuve de dépendance n'a été détectée.

- Degré de signification α

$$\alpha = \Pr(rejet\ de\ H_0 | H_0\ vrai)$$

Test de nombres aléatoires

- Quand utiliser ces tests:
  - Si un générateur de nombres aléatoires bien connu est utilisé, il est probablement inutile de le tester.
  - Si le générateur n'est pas explicitement connu ou documenté, par exemple des tableurs, des calculatrices symboliques / numériques, des tests doivent être appliqués à de nombreux numéros d'échantillons.
- Types de tests:
  - Tests théoriques: évaluez les choix de m, a et c sans générer de nombres.
  - Tests empiriques: appliqués aux séquences réelles de nombres produits (ce que nous considérerons).

Test de Kolmogorov-Smirnov

- Comparer la cdf d'une loi uniforme théorique $F(x)$ avec une cdf empirique $\widehat{F_n}(x)$
- Statistique: $D = \max |F(x) - \widehat{F_n}(x)|$ dont la distribution est connue (tableaux de référence)

Exemple

1. Ordonner les $R_{(i)}$ en ordre croissante
2. Calculer $\frac{i}{n}$
3. Calculer $\frac{i}{N} - R_{(i)}$
4. Calculer $R_{(i)} - \frac{i-1}{n}$
5. $D^+ = \max\left\{\frac{i}{n} - R_{(i)}\right\}$
6. $D^- = \max\left\{R_{(i)} - \frac{i-1}{n}\right\}$
7. $D = \max\{D^-, D^+\} = 0{,}26$
8. Pour $\alpha = 0{,}05$, $D_\alpha = 0{,}565 > D$, H_0 non-rejeté

Test de Chi carré

- Statistique

$$X_0^2 = \sum_{i=1}^{n} \frac{(O_i - E_i)^2}{E_i}$$

  - n nombre de classes
  - E_i le nombre théorique dans la $i$ème classe
  - O_i le nombre observé dans la $i$ème classe
- X_0^2 suit une loi de Chi-carré de degré de liberté $n-1$ avec n observations ($n > 50$)

Test d'auto-corrélation

- Tester la corrélation entre tous les m nombres à partir du $i$ème nombre
  - L'auto-corrélation $\rho_{i,m}$ entre $R_i, R_{i+m}, R_{i+2m}, \dots, R_{i+(M+1)m}$
  - M est le plus grand entier tel que $i + (M+1)m < n$
- Hypothèses:
  - $H_0: \rho_{i,m} = 0 \quad si\ les\ nombres\ sont\ indépendents$
  - $H_1: \rho_{i,m} \neq 0 \quad si\ les\ nombres\ sont\ dépendents$
- Si les nombre sont non-corrélés
  - Pour les M larges, la distribution de $\hat{\rho}_{i,m}$ peut être approximé par une loi normale.

Test d'auto-corrélation

- Statistique

$$Z_0 = \hat{\rho}_{i,m} / \hat{\sigma}_{\hat{\rho}_{i,m}}$$

- Approximation normale

$$Z_0 \sim N(0,1)$$

avec

$$\hat{\rho}_{i,m} = \frac{1}{M+1}\left(\sum_{k=0}^{M} R_{i+km} R_{i+(k+1)m}\right) - 0{,}25$$

$$\hat{\sigma}_{\hat{\rho}_{i,m}} = \frac{\sqrt{13M+7}}{12(M+1)}$$

Test d'auto-corrélation

- Si $\rho_{i,m} > 0$, la sous-séquence est positivement corrélée.
- Si $\rho_{i,m} < 0$, la sous-séquence set négativement corrélée.

Problèmes

- Le test n'est pas très sensitif pour les M petits.

random: Générateur de nombres pseudo-aléatoire en Python

- Initialisation du générateur de base
  - Par défaut, l'horaire du système est utilisé.
  - random.seed
  - os.urandom
- Entiers
  - random.randrange([start], stop[, step]):
    Retourner un élément aléatoirement sélectionné de range(start, stop, step)
    Fonction équivalente : choice(range(start, stop, step)),
  - random.randint(a, b); randrange(a, b+1) :
    Retourner un entier aléatoire N entre a et b

random: Générateur de nombres pseudo-aléatoire en Python

- Séquences
  - random.choice(seq)
    Retourner un élément aléatoire de la séquence seq donnée
  - random.shuffle(x[, random])
    Retourner la séquence x mélangée en place. Par défaut, la fonction random() est utilisée.
  - random.sample(population, k)
    Retourner une liste de longueur k composé par uniquement les éléments choisis de la séquence ou l'ensemble population.
    (Re)-échantillonage aléatoire sans remplacement

random: Générateur de nombres pseudo-aléatoire en Python

- Nombre aléatoire standard (nombre réel)
  - random.random()
    Retourner le prochain nombre aléatoire de type float dans [0,0;1,0)
  - random.uniform(a, b)
    Retourner un float aléatoire N tel que
    a <= N <= b si a <= b;
    b <= N <= a si b < a.

```
>>> random() # Random float:  0.0 <= x < 1.0
0.37444887175646646
>>> uniform(2.5, 10.0) # Random float:  2.5 <= x < 10.0
3.1800146073117523
>>> randrange(10) # Integer from 0 to 9 inclusive
7
>>> choice(['win', 'lose', 'draw'])      # Single random element from a sequence
'draw'
```

```
>>> deck = 'ace two three four'.split()
>>> shuffle(deck)                    # Shuffle a list
>>> deck
['four', 'two', 'ace', 'three']
>>> sample([10, 20, 30, 40, 50], k=4)    # Four samples without replacement
[40, 10, 50, 30]
```

Tests : scipy.stats

- Test de Kolmogorov-Smirnov
  - kstest
- Test de Chi carré
  - chisquare
  - chi2_contingency