Évaluation de la qualité aléatoire des nombres générés par un LLM (Java)

Rapport final

Nom de l'étudiant : DOTONOU Mahukèdè Ulrich Idée
Code Permanent : DOTM23379109
Cours : Projet synthèse
Nom de l'enseignant responsable : Raphaël Khoury

AOUT 2025
UNIVERSITE DU QUEBEC EN OUTAOUAIS

Table des Matières

1 Introduction

La génération de nombres aléatoires est au cœur de nombreuses applications : simulation, cryptographie, tests statistiques, jeux, IA, etc. Dans ce projet, nous comparons trois sources d'« aléa » :

1. une séquence fournie par un **LLM** (modèle de langage),
2. le générateur pseudo-aléatoire standard de Java (**`java.util.Random`**),
3. le générateur à usage sécurité (**`java.security.SecureRandom`**, implémentation *SHA1PRNG*).

Même si les sorties d'un LLM "semblent" aléatoires, elles peuvent présenter des régularités (biais de chiffres, répétitions, corrélations locales) différentes de celles d'un PRNG classique. L'objectif est donc de **mesurer la compatibilité statistique avec l'aléa** et de **détecter d'éventuels écarts**.

Nous évaluons des séquences d'entiers compris entre **0 et 999**. Le corpus comprend :

- **5 fichiers** `sequence 1.txt` à `sequence 5.txt` (LLM), chacun de longueur 1000.
- **2 séquences générées** de longueur 1000 avec graines fixes pour la reproductibilité : `Random(seed=12345)` et `SecureRandom(SHA1PRNG, seed=98765)`.

Pour analyser la **stabilité temporelle**, chaque séquence est également découpée en **5 fragments de 200 valeurs**. Cela permet d'identifier des dérives locales (non-stationarités) qui peuvent rester invisibles à l'échelle globale.

Les tests appliqués (seuil **$\alpha = 5\ \%$**) sont : **fréquence χ^2**, **runs**, **poker** (sur 3 chiffres), **autocorrélation (lag = 1)**, **équidistribution** (50 classes), **séries k-uples** (k = 2, m choisi pour garantir un effectif attendu ≥ 5), et **gaps** (écarts) sur l'intervalle [0..99]. L'ensemble est **implémenté en Java**, avec impressions détaillées (statistique, degrés de liberté, valeur critique, verdict PASS/REJET).

2 Définition de la tâche

2.1 Objectif

Évaluer, comparer et interpréter la « qualité statistique » de suites d'entiers dans 0..9990..9990..999 provenant de trois sources :

- **LLM** : 5 fichiers `sequence 1.txt` à `sequence 5.txt` (N = 1000 chacun).
- **Random** : `java.util.Random(seed=12345)` (N = 1000).

- **SecureRandom** : `java.security.SecureRandom("SHA1PRNG", seed=98765)` (N = 1000).

Chaque suite est également **découpée en 5 fragments** de 200 valeurs pour analyser les variations locales.

2.2 Entrées

- Données : entiers 5 fichiers sequence 1.txt … sequence 5.txt, chacun **1000 entiers** (0–999).
- Paramètre global **α = 5 %**.

2.3 Sorties attendues

Pour chaque séquence (et pour chaque fragment) :

- Les **statistiques** de tests (valeur numérique), **ddl** si applicable, **valeur critique** au seuil 5 %, **verdict** (*PASS / REJET*).

2.4 Tests appliqués et critères d'acceptation

- **Fréquence χ^2 (10 classes par centaines)**
  Hypothèse H_0 : distribution uniforme sur 0..9990..9990..999.
  Décision : PASS si $\chi^2 \leq \chi^2\_\{0.95, ddl=9\} =$ **16.92**.
- **Runs (montées/baisses locales)**
  H_0 : alternances cohérentes avec l'aléa.
  Décision : PASS si $|z| <$ **1.96**.
- **Poker (sur 3 chiffres d'un nombre à 3 digits)**
  Catégories {tous différents, une paire, trois identiques} avec probabilités théoriques {0.72, 0.27, 0.01}.
  Décision : PASS si $\chi^2 \leq$ **5.99** (ddl = 2).
- **Autocorrélation (lag = 1)**
  H_0 : absence de corrélation linéaire au décalage 1.
  Décision : PASS si $|z| <$ **1.96**.
- **Équidistribution (B = 50 classes)**
  H_0 : répartition uniforme sur 50 bacs égaux.
  Décision : PASS si $\chi^2 \leq \chi^2\_\{0.95, ddl=49\} \approx$ **66.33**.
- **Séries (k-uples, k = 2)**
  Partition de l'intervalle en m bacs/axe, **m choisi** pour garantir un **effectif attendu ≥ 5** (sur N=1000, m=10 ; sur fragments N=200, m≈6).
  Décision : PASS si $\chi^2 \leq \chi^2\_\{0.95, ddl=m^2-1\}$ (ex. 99 ddl → **123.22**).
- **Gaps (écarts) sur 0..990..990..99**
  H_0 : distances entre occurrences suivent une loi géométrique de paramètre p (proportion empirique).

Groupement des classes pour **attendus ≥ 5**.
Décision : PASS si $\chi^2 \leq \chi^2\_\{0.95, \text{ddl=nb\_bacs}-1\}$.

3 Évaluation et observations

3.1 Vue d'ensemble par source (plein format, N = 1000)

- **LLM (sequence 1→5)**
  Tous les tests passent au seuil 5 % : **Fréquence χ^2**, **Runs**, **Poker**, **Autocorr (lag = 1)**, **Équidistribution (B=50)**, **Séries (k=2)** et **Gaps**.

- **Random (java.util.Random, seed=12345, N=1000)**
  Tous les tests en plein format **PASS**.

- **SecureRandom (SHA1PRNG, seed=98765, N=1000)**
  Poker (χ^2 = 7,91 > 5,99) **REJET** et **Autocorr** (z = 2,29) **REJET**. Le reste **PASS**.

3.2 Observations par fragments (5×200 par suite)

Le découpage en fragments (N=200) **augmente la variance** des statistiques et **multiplie le nombre de tests**, on s'attend mécaniquement à quelques rejets au seuil 5 % (≈5 % de faux positifs en moyenne).

- **LLM (fragments) — rejets notables :**
  - `sequence 3.txt` **frag3** : **Équidistrib** χ^2 = 74,50 > 66,33 (**REJET**) et **Séries** χ^2 = 50,47 > 49,80 (**REJET**).
  - `sequence 4.txt` **frag2** : **Gaps** χ^2 = 4,38 > 3,84 (**REJET**) avec 2 bacs.
  - `sequence 4.txt` **frag5** : **Runs** z = 2,36 (**REJET**) (légère sur-alternance).
- **Random (fragments) — rejets ponctuels :**
  - **frag3** : **Équidistrib** χ^2 = 66,50 > 66,33 (**REJET**, de justesse).
  - **frag5** : **Poker** χ^2 = 6,63 > 5,99 (**REJET**).
- **SecureRandom (fragments) — rejets ponctuels :**
  - **frag2** : **Runs, Équidistribution, Gap** rejetées.
  - **frag4** : **Équidistribution, Séries, Gap** rejetées.
  - **frag5** : **Séries, Autocorrélation** rejetées.

3.3 Interprétation

- **Effet α=5 % + multiplicité des tests** : avec 7 tests × (5 suites + 2 générateurs) × (1 global + 5 fragments), on exécute **des centaines de tests**. Même si les sources sont parfaitement aléatoires, on **attend quelques rejets** (≈5 % en moyenne) par simple

fluctuation. Les rejets isolés, proches du seuil (p.ex. $\chi^2 = 66{,}50$ vs 66,33), vont dans ce sens.

- **Taille d'échantillon** : les fragments (N=200) sont plus **instables**. Les tests χ^2 (Équidistrib, Séries, Gaps) sont **sensibles** aux faibles effectifs ; un regroupement des classes et l'ajustement de *m* limitent mais n'éliminent pas cet effet.
- **Profil par source** :
  - **LLM** : très **stable** en global, quelques **écarts locaux** (seq3-frag3, seq4-frag2/5).
  - **Random** : comportement **attendu** d'un PRNG classique, rejets **rares** et **isolés**.
  - **SecureRandom** : **deux rejets** en global (poker, autocorr) et plusieurs rejets en fragments.

4 Méthodologie

4.1 Données, générateurs et taille des échantillons

- **Jeu "LLM"** : 5 fichiers `sequence 1.txt` … `sequence 5.txt`, chacun avec **N = 1000** entiers dans [0..999].
- **Générateurs Java** :
  - `Random(seed=12345)` → **N = 1000**.
  - `SecureRandom("SHA1PRNG", seed=98765)` → **N = 1000**.
- **Découpage en fragments** : chaque suite de N=1000 est découpée en **5 fragments** de **200** (N=200) pour observer la stabilité locale.

4.2 Tests statistiques retenus (α = 5 %)

1. **Fréquences χ^2 (10 classes "centaines")** — ddl=9, seuil **16,92**.
2. **Runs "pics/creux"** — statistique **z**, acceptation si $|z| < \mathbf{1{,}96}$.
3. **Poker (3 catégories 000–999 : tous diff., 1 paire, 3 identiques)** — ddl=2, seuil **5,99**.
4. **Autocorrélation (lag = 1)** — $z \approx r \cdot \sqrt{N}$, acceptation si $|z| < \mathbf{1{,}96}$.
5. **Équidistribution (B=50)** — ddl=49, seuil **66,33**.
6. **Séries (k=2)** — partition en **m** classes par chiffre (choisi pour avoir un **attendu ≥ 5**) → ddl = $m^2 - 1$, seuil selon ddl (ex. m=10 ⇒ ddl=99, seuil **123,22**).
7. **Gaps** — intervalle [0..99], classes 0..K−1 et "K+" (K choisi pour attendus ≥ 5), ddl = nb_bacs − 1.

Les valeurs critiques du χ^2 proviennent d'une **table intégrée** (approx. Wilson–Hilferty au-delà de 30 ddl).

4.3 Implémentation (Java)

- Lecture des fichiers, **contrôle** des bornes [0..999], et **agrégats** par classes.
- Fonctions dédiées : `calculerChiCarre`, `testDesRuns`, `testPoker`, `testAutocorrelation`, `testEquidistribution`, `testSeries`, `testGapInterval`.
- **Fragments** : utilitaires `afficherTousLesTests` et `testerParFragments` (5×200).
- Graines : `Random(12345)`, `SecureRandom("SHA1PRNG",98765)`.

5 Résultats (synthèse)

5.1 Séquences complètes (n = 1000)

- **LLM (5 séquences)** : tous les tests **acceptés** (fréquence, runs, poker, autocorr, équidistribution, séries, gap).
- **Random (seed=12345)** : tous les tests **acceptés**.
- **SecureRandom (SHA1PRNG, seed=98765)** : **poker rejeté**, **autocorrélation rejetée** ; autres tests acceptés.

5.2 Fragments (5×200 par source)

Des rejets ponctuels apparaissent, ce qui est **attendu** avec α=5 % et de multiples tests. Exemples observés dans tes sorties :

- **LLM** :
  - `sequence 3 — frag3` : **Équidistribution & Séries** rejetées.
  - `sequence 4 — frag2` : **Gap** rejeté ; `frag5` : **Runs** rejeté.
- **Random** :
  - `frag3` : **Équidistribution** rejetée (au seuil) ;
  - `frag5` : **Poker** rejeté.
- **SecureRandom** :
  - `frag2` : **Runs, Équidistribution, Gap** rejetées.
  - `Frag4` : **Équidistribution, Séries, Gap** rejetées.
  - `Frag5` : **Séries, Autocorrélation** rejetées.

5.3 Analyse

À l'échelle **n = 1000**, les séquences **LLM** et **Random (seed=12345)** passent l'ensemble des tests (fréquence, runs, poker, autocorr, équidistribution, séries, gap) — rien d'anormal. **SecureRandom (SHA1PRNG, seed=98765)** montre deux rejets (poker et autocorrélation) mais tous les autres tests sont OK ; deux rejets sur sept à **α = 5 %** restent plausibles sans indiquer un défaut systémique.

Sur les **fragments (5×200)**, des rejets ponctuels apparaissent pour toutes les sources, ce qui est **attendu** quand on multiplie les tests à α=5 %. Une partie des rejets (surtout équidistribution et poker) est probablement accentuée par des **attendus trop faibles** à n=200 (p. ex. B=50 → attendu ≈ 4 par classe, <5), rendant le χ^2 moins fiable.

6 Conclusion

Au vu des résultats, les cinq séquences **LLM** et la séquence **Random** sont **compatibles** avec l'hypothèse d'aléa aux tests retenus. La séquence **SecureRandom** présente deux rejets au global (poker et autocorrélation) et davantage de rejets locaux en fragments ; toutefois, compte tenu du nombre total de tests et de la taille modeste des sous-échantillons, ces observations restent **plausibles** sous l'hypothèse d'aléa et ne constituent pas, en l'état, une preuve de non-aléa. Des campagnes supplémentaires, avec davantage de données, des lags multiples et des corrections de multiplicité, permettraient de **confirmer** et **resserrer** ces conclusions

7 Perspectives

- Augmenter l'échantillon (**n ≥ 10 000**), multiplier les tirages et **moyenner** les statistiques.
- Explorer d'autres **lags** d'autocorrélation ; variantes de **runs** (au-dessus/au-dessous de la médiane).
- Appliquer des **corrections de multiplicité** (Bonferroni/FDR) dans les campagnes de tests.
- Ajouter des **batteries** standard (NIST SP 800-22, Dieharder, TestU01) et **comparer** plusieurs LLM.

8 Référence

• Plan de projet : « Évaluation de la qualité aléatoire des nombres générés par les LLMs (Java) »

• Rapport de progrès (27/06/2025) — mise en place du projet Java et premiers tests.

• Supports de cours : La sécurité logicielle, une approche defensive, chapitre 4.

• Code : Testdesnombres.java (implémentations des tests).