xmco

We deliver cybersecurity expertise

Dédé ! Des dés aux données

Analyse de tirages aléatoires avec l'outil Random Test Tool

3 novembre 2024

Sommaire

01 Nombres aléatoires

02 Analyse avec RTT

03 Prédiction de cartes

04 Bonnes pratiques de développement

05 QUESTIONS / RÉPONSES

Nombres aléatoires

A quoi servent-ils et comment les générer ?

1

Qu'est-ce qu'une suite de nombres aléatoires ?

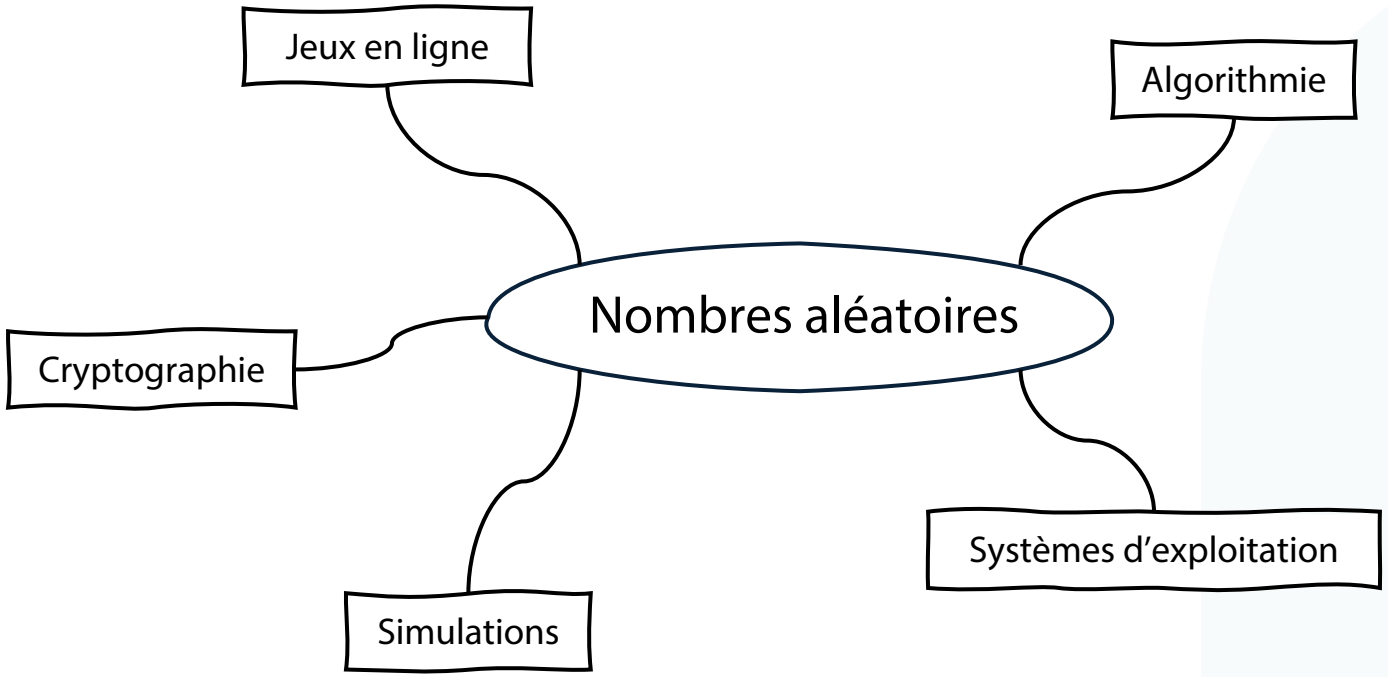
“

Une suite de **nombre aléatoires** est une suite de nombres où **aucune relation calculable** ne peut être établie entre un nombre et ses prédecesseurs.

Autrement dit, les nombres doivent être **imprévisibles**.

”

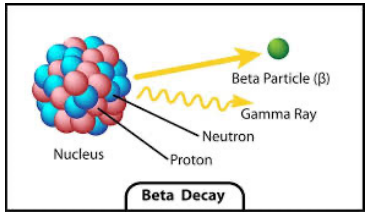
A quoi sert l'aléatoire ?



Comment générer des nombres aléatoires ?

Processus Physiques

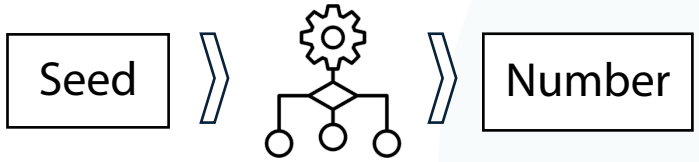
TRNG: True Random Number Generator



Lavarand (Cloudflare blog)

Algorithmes

PRNG: Pseudo Random Number Generator



Mersenne Twister
LCG
Xorshift Generator
Fortuna

Analyse avec RTT

Du générateur aléatoire d'un jeu de poker

2

Random Test Tool

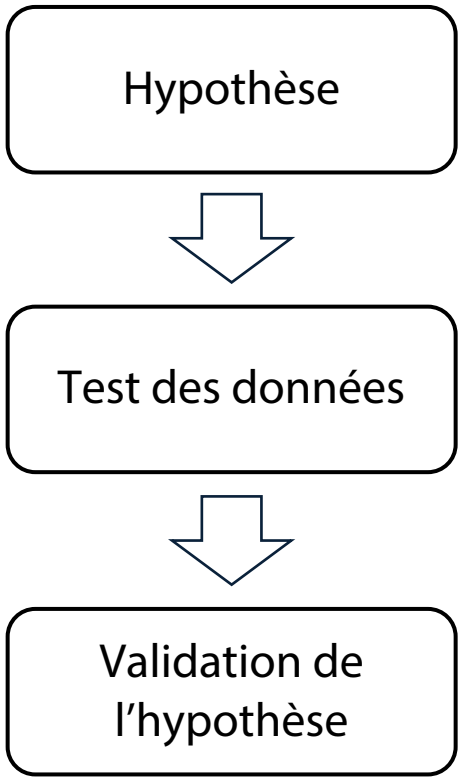
Teste statistiquement des suites de nombres aléatoires

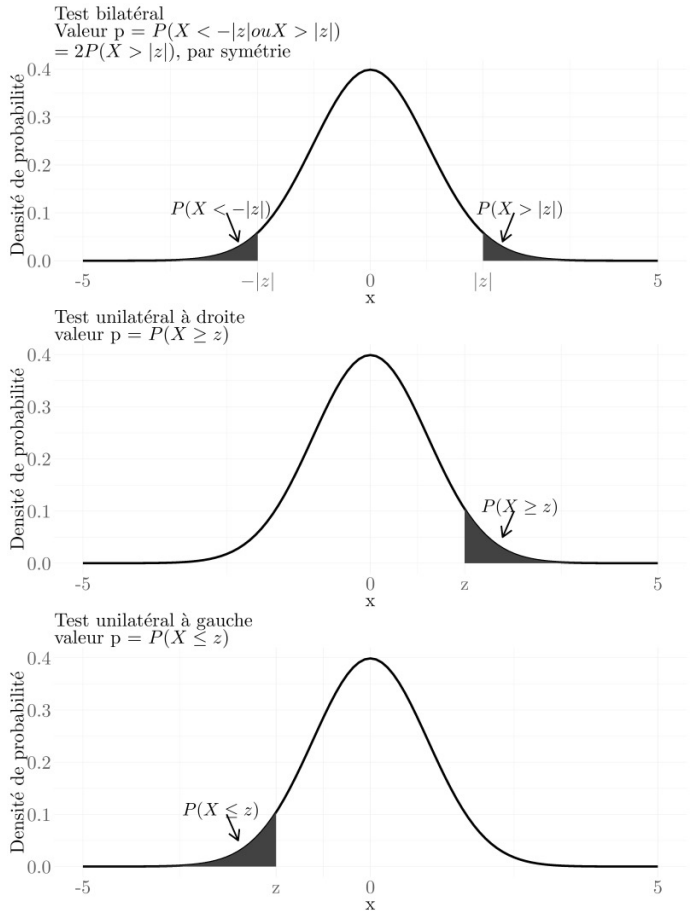
```
(venv) arigoureau@XMCO-ARI random_test_tool % random-test-tool -h
usage: random-test-tool [-h] [-i INPUT_FILES [INPUT_FILES ...]] [-d INPUT_DIR] [-o {terminal,csv,graph,html,all}] [-O OUTPUT_DIR] [-j {1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31}] [-t [STATISTICAL_TESTS ...]]
                        [-dt {int,bits,bytes}] [-s {\n, ,,,;}] [-ll {ALL,DEBUG,INFO,WARN,ERROR,FATAL,OFF,TRACE}] [-c CONFIG] [-C CHUNKS]

Script testing the randomness of a serie of integeror bits via statistical statistical_tests.

options:
  -h, --help            show this help message and exit
  -i INPUT_FILES [INPUT_FILES ...], --input_files INPUT_FILES [INPUT_FILES ...]
                        List of files to test.
  -d INPUT_DIR, --input_dir INPUT_DIR
                        Input directory, statistical_tests will be launched on each file.
  -o {terminal,csv,graph,html,all}, --output {terminal,csv,graph,html,all}
                        Output report options.
  -O OUTPUT_DIR, --output_dir OUTPUT_DIR
                        Output directory.
  -j {1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31}, --n_cores {1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31}
                        Number of processes used, 1 by default, maximum 31
  -t [STATISTICAL_TESTS ...], --test [STATISTICAL_TESTS ...]
                        Specifies which statistical_tests to launch. By default all statistical_tests are launched.
  -dt {int,bits,bytes}, --data_type {int,bits,bytes}
                        Used to select data type of sample, by default integer (int)
  -s {\n, ,,,;}, --separator {\n, ,,,;}
                        Separator used for integer files.
  -ll {ALL,DEBUG,INFO,WARN,ERROR,FATAL,OFF,TRACE}, --log_level {ALL,DEBUG,INFO,WARN,ERROR,FATAL,OFF,TRACE}
                        Log level (default: INFO).
  -c CONFIG, --config CONFIG
                        Configuration file of the run, if used will override other options.
  -C CHUNKS, --chunks CHUNKS
                        If this option is given, the inputs will be merged and splited into n chunks then processed independently.
```

$ pip install random-test-tool

github.com/xmco/random_test_tool

Tests statistiques et p-value

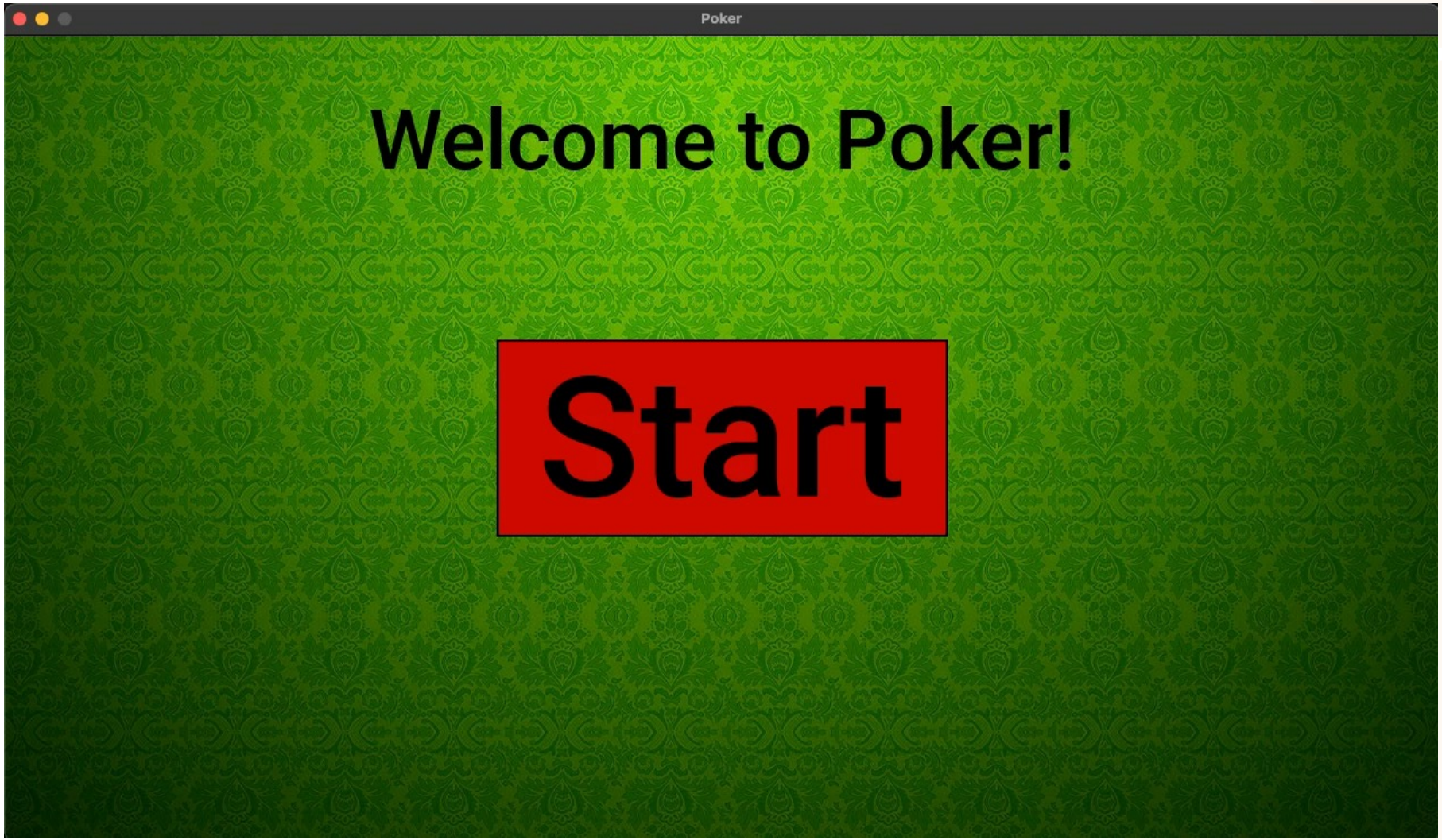




Valeur-p (Wikipedia)

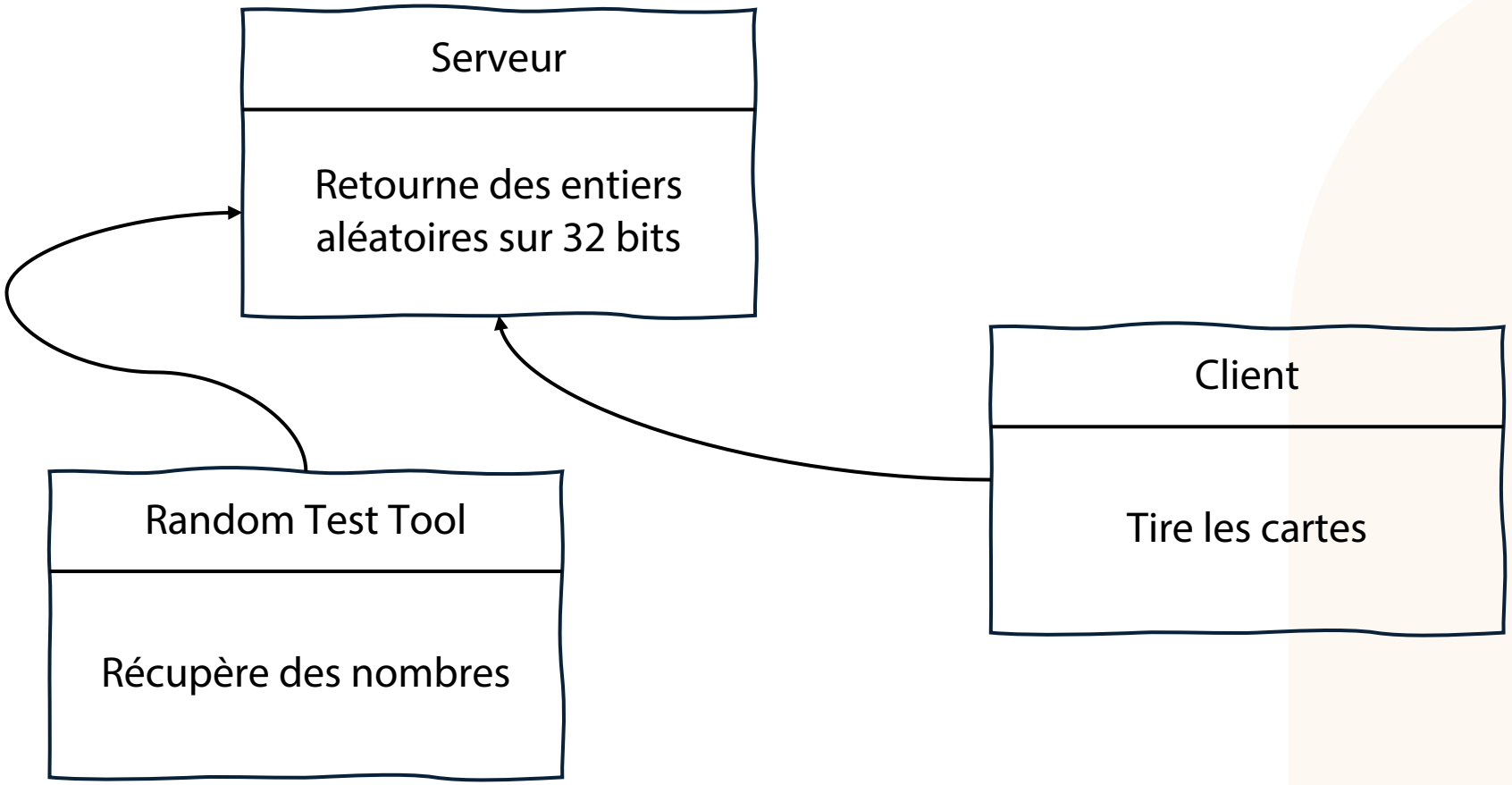
Une **p-value** représente la probabilité d'obtenir des données aussi extrêmes que celles observées au cas où l'hypothèse est correcte.

Jeu de poker



Implémentation de base : github.com/drewtorg/poker

Implémentation du jeu



Analyse du générateur du jeu de cartes

Données:
1000 fichiers de 1 000 000 de bits (31250 entiers sur 32 bits)

Test	Objectif
Chi2	Distribution des nombres
Block Chi2	
Longest Run of Ones	
Run	Périodicité de la séquence
Maurer Test (compression)	
Spectral	
Sign	
Linear Complexity	Linéarité de la génération
Binary Rank	

Analyse du générateur du jeu de cartes

```
(venv) arigoureau@XMCO-ARI random_test_tool % random-test-tool -d poker_random_gen -dt bits -j 10 -o all
```

Résultats

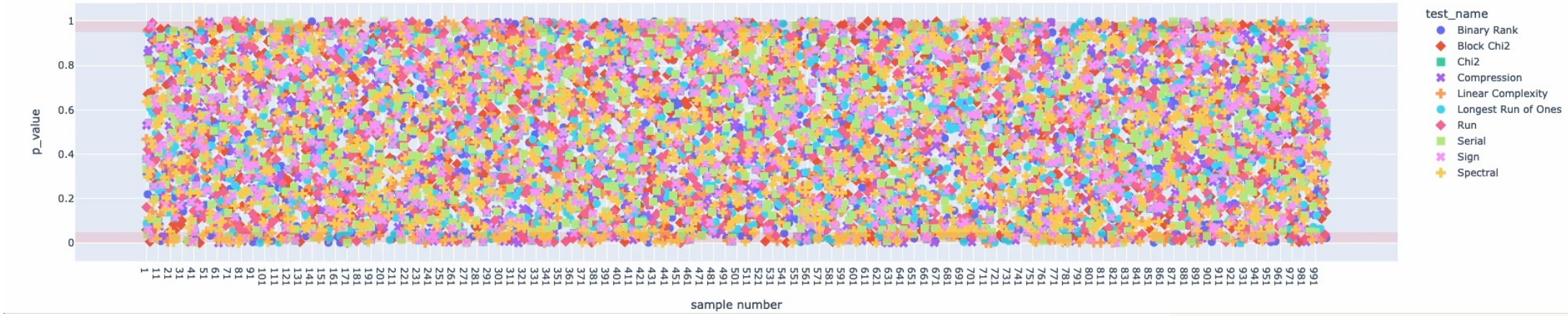
Run Results

The following table and graphs summarize the test results for the different chunks of data.

For additional information about results analysis please check: EXAMPLES.md

Test Name	OK Count	KO Count	KO 95% Interval	SUSPECT Count	SUSPECT 95% Interval
Binary Rank	895	24	[12, 30]	81	[82, 120]
Block Chi2	892	19	[12, 30]	89	[82, 120]
Chi2	883	18	[12, 30]	99	[82, 120]
Compression	893	30	[12, 30]	77	[82, 120]
Linear Complexity	881	33	[12, 30]	86	[82, 120]
Longest Run of Ones	904	14	[12, 30]	82	[82, 120]
Run	899	23	[12, 30]	78	[82, 120]
Serial	897	19	[12, 30]	84	[82, 120]
Sign	884	19	[12, 30]	97	[82, 120]
Spectral	865	26	[12, 30]	109	[82, 120]

Autres exemples



Approche d'analyse de générateur aléatoire

Audit d'un générateur aléatoire

Revue de la **configuration** du générateur choisi

Audit de **l'algorithme** en lien avec le métier

Exécution de **tests statistiques** (RTT)

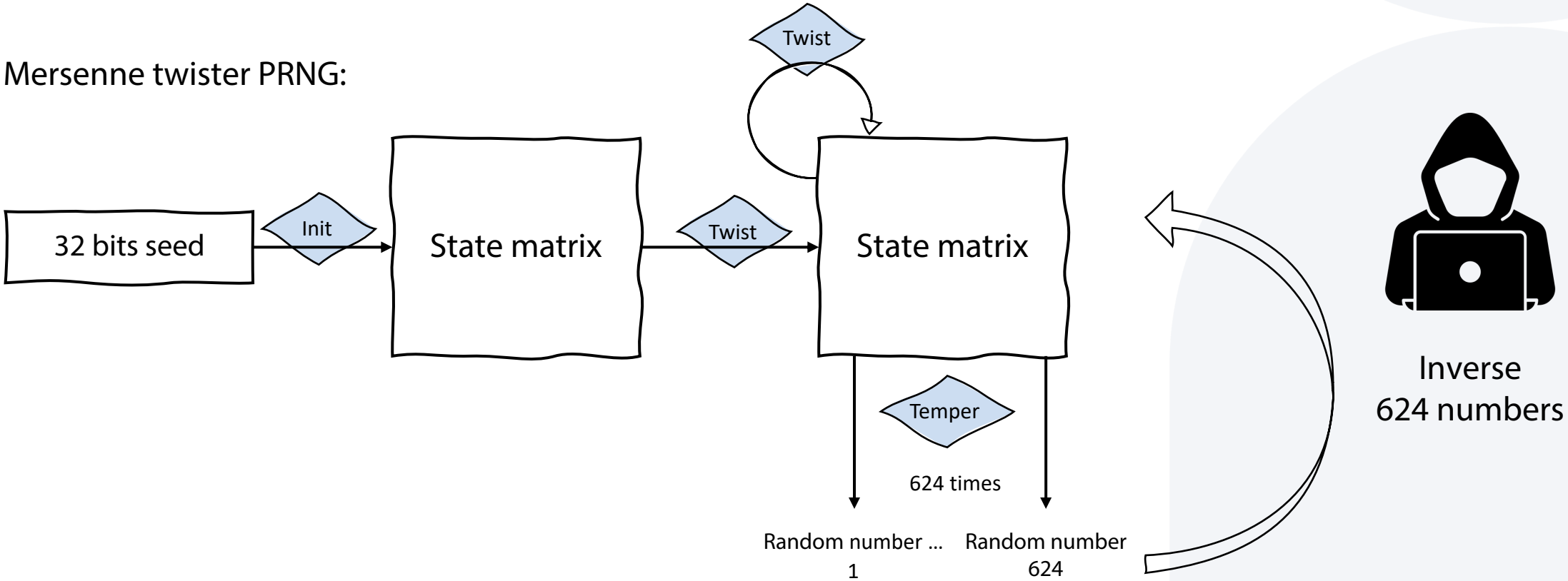
Prédictions de cartes

Ou comment tricher au poker ?

3

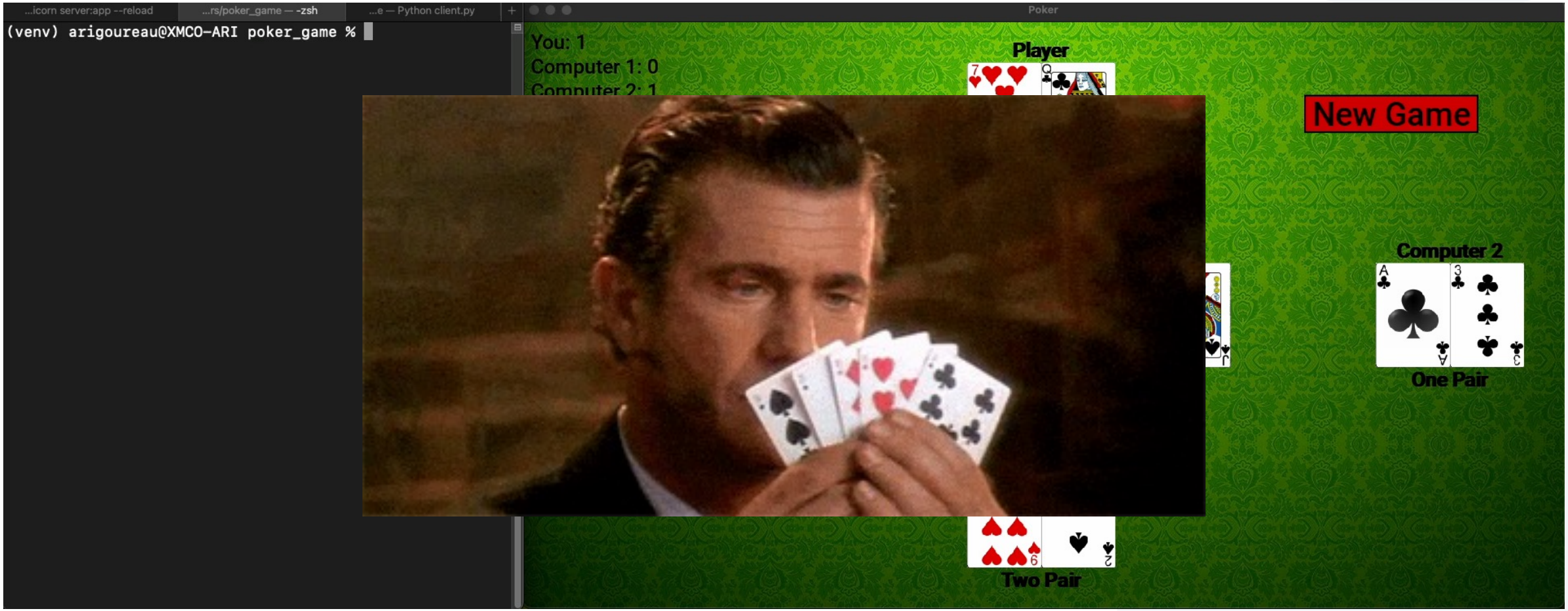
Un peu de théorie

Mersenne twister PRNG:



Module python *github.com/tna0y/Python-random-module-cracker*

Prédiction de cartes



Générateurs par défaut

Language	Instance	Générateur	Prédictible ?	Nombre d'entiers
Python	random	MT-19937	Oui	624
Java 21	Java.util.Random	LCG	Oui	3
glibc	random	LFSR	Oui	32
Node.js	Math.Random	MT-19937	Oui	624
Ruby	rand	MT-19937	Oui	624
C#	Random	Substractive Gen	Oui	55
Bash	$RANDOM	LCG	Oui	3

Bonnes pratiques

Bonnes pratiques de développement

4

Utiliser le bon générateur pour la bonne tâche

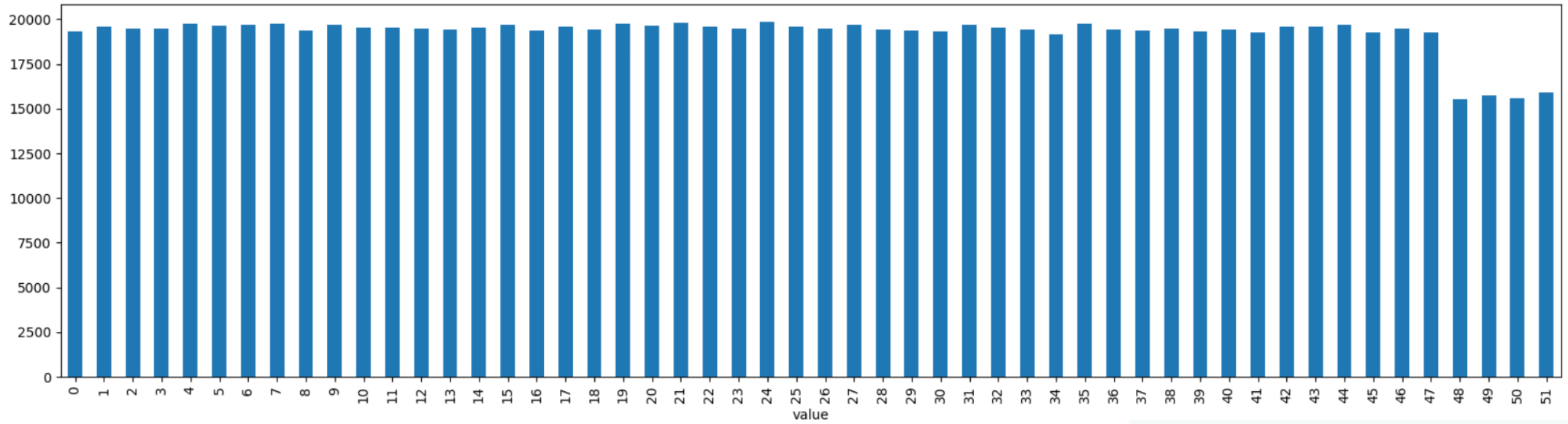
Tâche	Générateur	Référence
Simulation/statistiques	Numpy generator	numpy pcg64
Cryptography	Module secrets	secrets doc
Jeux en ligne (poker)	Module secrets	secrets doc
Certains algorithmes (load-balancer)	Module random	random doc

Attention aux opérations sur les nombres

[3]:
```
import pandas
import os
```

[4]:
```
s = pandas.DataFrame({'value':[x%52 for x in os.urandom(1000000)]})
s['value'].value_counts().sort_index().plot(figsize=(20,5),kind='bar')
```

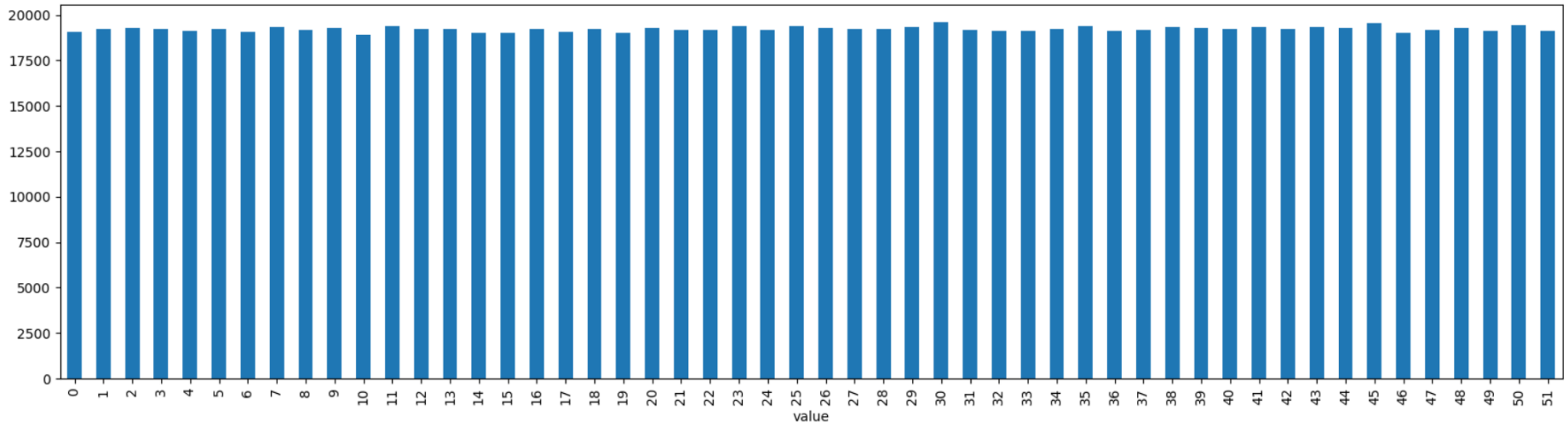
[4]:
```
<Axes: xlabel='value'>
```



Attention aux opérations sur les nombres

[11]:
```
values = []
rejects = 0
while(len(values)<=1000000):
    x = int.from_bytes(os.urandom(1), byteorder='little')
    if x < 208: # 4*52
        values.append(x % 52)
    else:
        rejects +=1
s = pandas.DataFrame({'value':values})
s['value'].value_counts().sort_index().plot(figsize=(20,5),kind='bar')
```

[11]:
```
<Axes: xlabel='value'>
```



[12]:
```
rejects
```

[12]:
```
232340
```

Utiliser directement les méthodes pré-implémentées (randrange, randint, …)

Contact et Références

Random test Tool
github.com/xmco/random_test_tool

Jeu de poker
github.com/drewtorg/poker

Randcrack
github.com/tna0y/Python-random-module-cracker

Tests statistiques
nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-22r1a.pdf

Antoine Rigoureau
antoine.rigoureau@xmco.fr
LinkedIn - Antoine Rigoureau

We deliver cybersecurity expertise