Università di Catania

Metodi Matematici e Statistici

Test per i numeri casuali in Python

Autore:
Marco Ardizzone

Matricola:
X81001077

Marzo 2021

Indice

1 Introduzione **2**

2 Python PRNG: Mersenne Twister **2**

3 Test di Uniformità **2**
3.1 Test del Chi-Quadro 3
3.2 Risultati Test di Uniformità 4

4 Up and Down Test **5**
4.1 Risultati Up and Down Test 6

5 Conclusioni **6**

Referenze **7**

1 Introduzione

I numeri *pseudo*casuali sono utilizzati in numerosi ambiti, dal gioco d'azzardo al settore bancario. Ma siamo davvero sicuri che siano generati in maniera randomica, seguendo una distribuzione uniforme? Questo progetto ha il compito di verificare, mediante un *Test di Uniformità* ed un *Up and Down Test*, se il RNG di Python segue davvero una distribuzione uniforme. Il codice utilizzato per questo progetto è consultabile qui

2 Python PRNG: Mersenne Twister

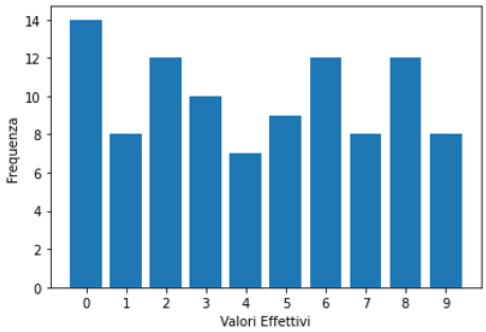
L'algoritmo alla base del PRNG di Python è il Mersenne Twister. Questo algoritmo utilizza un periodo molto lungo $T = 2^{19937} - 1$, il che è una condizione necessaria per avere un buon PRNG. Ad ogni iterazione non genera un solo numero random, ma 624 ed inoltre la sua funzione è reversibile, il che rende l'algoritmo non sicuro dal punto di vista crittografico, ma molto veloce ed efficiente dal punto di vista computazionale [1] [2].

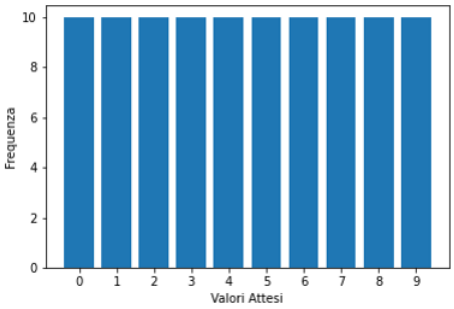
3 Test di Uniformità

Il test di uniformità utilizza il test del χ^2 per testare la bontà dell'adattamento. Serve a verificare che una popolazione sia distribuita secondo una specifica funzione di ripartizione [3]. Nel nostro caso, vogliamo verificare che i numeri random seguano una distribuzione uniforme.

3.1 Test del Chi-Quadro

Consiste nel suddividere l'intervallo [0,9] in k intervalli disgiunti (nel nostro caso, k=10). Si vuole verificare che in ogni intervallo cadano lo stesso numero di elementi.





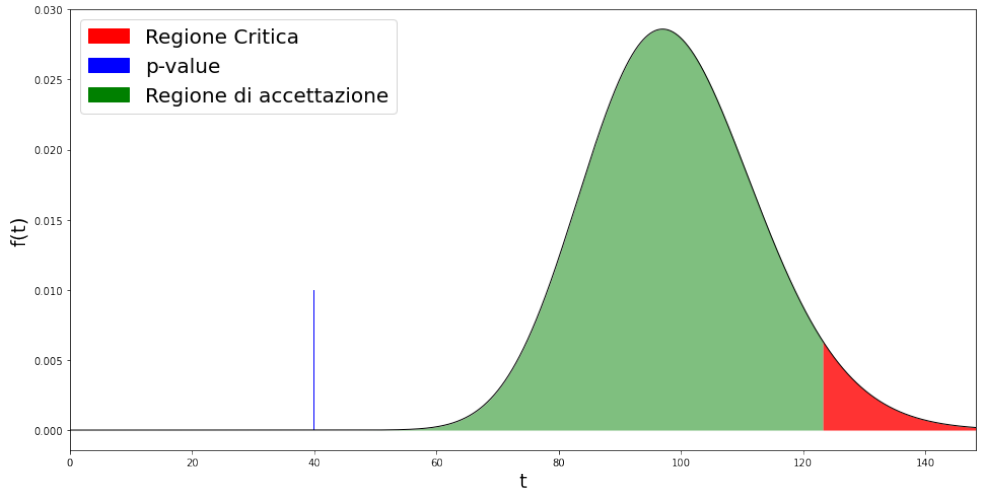
Dove X_i è l'i-esimo campione, O_i la frequenza effettivamente osservata ed A_i la frequenza che ci aspettiamo. Consideriamo la statistica:

$$W = \sum_{k=1}^{K} \frac{(O_k - A_k)^2}{A_k}$$

Se l'ipotesi nulla è vera, cioè se i numeri sono distribuiti uniformemente, W è distribuita come una χ^2 con $k-1$ gradi di libertà. Accettiamo l'ipotesi nulla se $W < W_{1-alpha}$, dove $W_{1-alpha}$ è il quantile di ordine 1-alpha, ovvero la regione di rifiuto. Per semplicità introduciamo il *p-value*, ovvero l'area della curva in $[W, +\infty[$. Accettiamo se $p - value > alpha$.

3.2 Risultati Test di Uniformità

Mediante la libreria *stats* di *scipy* abbiamo condotto il test del chi quadro tra la distribuzione effettiva e quella attesa, ottenendo $W = 0.12599999$ ed un p-value pari a 0.9999999282, il che ci permette di affermare con probabilità di errore 5% che i dati sono distribuiti uniformemente.



E' possibile osservare graficamente che il p-value ricade nella regione di accettazione.

4 Up and Down Test

Questo test consiste nel verificare se una data serie è generata in maniera casuale o segue un qualche pattern. Data una serie di numeri L generati randomicamente, si assegna un 1 se $x_i < median(L)$, altrimenti si assegna 0. Un buon generatore di numeri random non dovrebbe produrre lunghe run di 1 oppure 0 [4].
Consideriamo la statistica:

$$Z = \frac{R - \bar{R}}{S_R}$$

Dove R è il numero di run effettive ed $\bar{R}$ è il numero di run attese, ovvero

$$\bar{R} = \frac{2n_1n_2}{n_1 + n_2} + 1$$

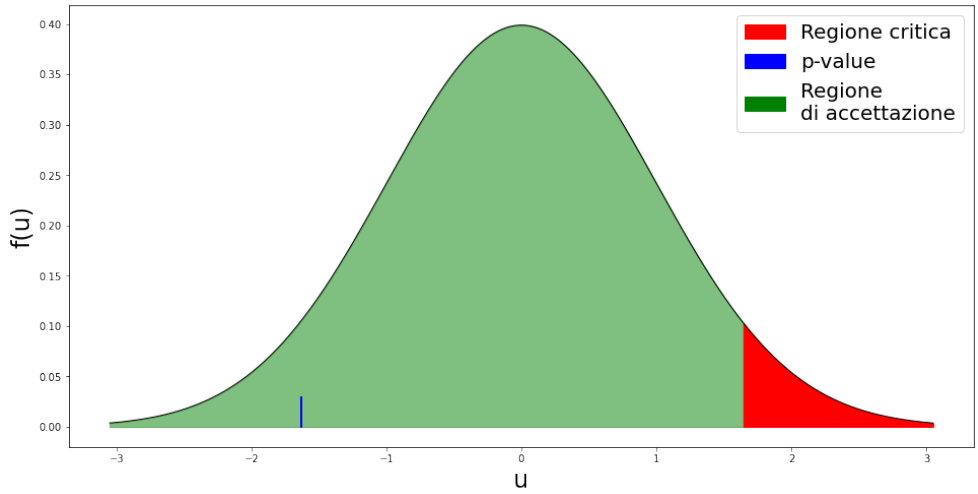
Mentre S_R è la devisione standard, ovvero

$$S_R^2 = \frac{2n_1n_2(2n_1n_2 - n_1 - n_2)}{(n_1 + n_2)^2(n_1 + n_2 - 1)}$$

Dove n_1 ed n_2 sono rispettivamente il numero di 1 e 0 prodotti dal test, ovvero i valori maggiori e minori della mediana della serie di numeri [5].
L'ipotesi che i numeri siano generati senza seguire alcun pattern è accettata con livello di confidenza alpha se $Z < Z_{1-alpha}$.
Dove $Z_{1-alpha}$ il quantile di ordine 1-alpha, ovvero la regione di rifiuto. [6].

4.1 Risultati Up and Down Test

Mediante le librerie *math* e *statistics* abbiamo ottenuto $Z = 0.94915847$ e $Z_{1-alpha} = 0.89915847$, essendo $Z > Z_{1-alpha}$, possiamo affermare che i numeri sono generati senza alcun pattern.



E' possibile osservare graficamente che il p-value ricade nella regione di accettazione.

5 Conclusioni

Abbiamo dimostrato che lo PRNG di Python è un generatore di numeri pseudocasuali che seguono una distribuzione uniforme, non seguono alcun pattern, il che rende l'algoritmo utilizzato da Python (Mersenne Twister) un ottimo e veloce PRNG.

Referenze

[1] Wikipedia : Mersenne Twister
https://en.wikipedia.org/wiki/Mersenne_Twister

[2] Cryptologie.net : How does the Mersenne Twister work
https://www.cryptologie.net/article/331/how-does-the-mersennes-twister-work/

[3] Orazio Muscato : Metodi Matematici e Statistici
https://www.dmi.unict.it/muscato/MMStat.pdf

[4] Kinds on the Genius : What is Run Test
https://kindsonthegenius.com/blog/what-is-run-test-in-statistics
-a-simple-explanation-with-step-by-step-examples/

[5] Wikipedia : Wald Wolfowitz Run Test
https://en.wikipedia.org/wiki/Wald-Wolfowitz_runs_test

[6] Geeks for Geeks : Runs test of Randomness in Python
https://www.geeksforgeeks.org/runs-test-of-randomness-in-python/