Analisi della qualità di dati casuali, con esperimenti di generazione

Francesco Lorenzi, Maggio 2020

Sommario

In questa relazione mi propongo di analizzare i dati forniti cercando di evidenziare pattern e difetti nella casualità, usando sia metodi visivi che numerici di cui commenterò qualitativamente l'efficacia. Inoltre mostrerò due esperimenti per generare altri due set di dati, uno di natura fisica (estrazione di numeri casuali da rumore Johnson-Nyquist) ed uno di natura numerica (estrazione di numeri pseudo-casuali dall'automa cellulare "rule 30" di S. Wolfram). Anche questi dati verranno confrontati assieme agli altri.

1 Introduzione e teoria

Cerchiamo di formalizzare il problema in un quadro probabilistico: l'ipotesi che vogliamo verificare per ogni dataset è se sia stato ottenuto con un processo di Bernoulli con v.a. indipendenti di parametro 1/2.

$$(X_i)_{i=0}^{N} \quad t.c. \quad \forall i,\, X_i \sim \mathcal{B}\left(\frac{1}{2}\right)$$

Formalmente si tratta di un test d'ipotesi non parametrico. Piuttosto che puntare ad un metodo per testare questa ipotesi in modo diretto, problema matematico interessante ma delicato, in questa relazione useremo dei metodi per verificare, nei dati, la presenza di alcune delle caratteristiche che un processo del genere *dovrebbe* avere, andando contemporaneamente ad accertare che non ce ne siano altre che sappiamo invece non essere legate ad un processo di Bernoulli.

Le analisi sono svolte su due livelli, il primo basato sui bit, che rappresentano le realizzazioni del processo di Bernoulli, ed il secondo sui byte, ovvero sulla rappresentazione `uint8` associata ai bit raggruppati di 8 in 8. Nel secondo processo vorremmo verificare che le v.a. ottenute rappresentino un processo di v.a. uniformi. Infatti dato che in un processo di Bernoulli con variabili indipendenti tutte le realizzazioni di un gruppo di 8 v.a. sono equiprobabili, il problema si traduce nel verificare se questo nuovo processo soddisfa

$$(Y_i)_{j=0}^{\lfloor N/8 \rfloor} \quad t.c. \quad \forall j,\, Y_j \sim \mathcal{U}(\mathcal{A})$$
$$\text{dove } \mathcal{A} = \{0, 1, \cdots, 255\}$$

1.1 Scelta dei test

Avendo a disposizione due formati diversi di rappresentazione dei test puntiamo ad estrarre da ciascuno le metriche più informative.

1.2 Analisi a singoli bit

- Estrazione di media e varianza
- Autocorrelazione
- Analisi in frequenza

Osserviamo la definizione di autocorrelazione: una interessante metrica di confronto dovrà essere insen-

sibile alla media dei valori, che è già considerata in altra sede. Nell'ipotesi di un processo casuale WSS l'autocorrelazione è definita come:

$$r_j = E\left[X_i, X_{i+j}\right],$$

per qualsiasi i, mentre l'autocovarianza è:

$$K_j = E\left[(X_i - \mu_x), (X_{i+j} - \mu_x)\right],$$

dove $\mu_x = E[X_i]$ uguale per tutte le X_i.

Per analizzare i dati normalizzeremo questa autocovarianza ottenendo il cosiddetto *coefficiente di autocorrelazione* di Pearson in questo modo

$$\rho_j = \frac{E\left[(X_i - \mu_x), (X_{i+j} - \mu_x)\right]}{\sigma_x^2},$$

passando ai campioni otteniamo una definizione implementabile

$$\rho_m = \frac{\sum_{n=0}^{N-m} (x(n) - \widehat{\mu}_x)\,(x(n+m) - \widehat{\mu}_x)}{N\,\widehat{\sigma}^2},$$

dove $\widehat{\mu}_x$ è la media campionaria, e $\widehat{\sigma}^2$ è la varianza campionaria. Considereremo il test passato se il conteggio dei lag di autocorrelazione che escono dall'intervallo $\pm\phi^{-1}(1 - 0.1/2)/\sqrt{N}$ non supera il 10% dei campioni, ovvero stabiliremo un livello di significatività del 10%.

Se da un punto di vista intuitivo questa metrica rappresenta quanto il segnale, privato della media, è similare alla sua copia ritardata nel tempo, è allora interessante analizzare la trasformata di Fourier discreta (implementabile nel nostro caso con una FFT) per ricavare informazioni sulle periodicità del segnale. Sfrutteremo questa intuizione per rappresentare i risultati del test di autocorrelazione con dei grafici. Nel caso in cui i grafici mostrino delle periodicità marcate, potremmo accorgerci di un PRNG, che ha appunto la caratteristica di essere periodico.

1.3 Analisi dei valori `uint8`

- Estrazione di media e varianza
- Ricerca visiva di pattern 2D / 3D
- Test χ^2 sulle frequenze attese

La ricerca visiva di pattern è un metodo molto veloce ed efficace, può venire usato come una condizione sufficiente per scartare un dataset che evidenzia problemi di ripetitività in pattern (in questo senso è similare al processo di qualifica che usa l'autorrelazione). Grazie alle diverse rappresentazioni grafiche "dirette" che possiamo adattare ai dati, è possibile riconoscere caratteristiche che diventano evidenti solamente in certi piuttosto che in altri approcci. In questo report, non disponendo di animazioni, mi limiterò alle analisi bidimensionali. Ci sono diversi modi per avere una rappresentazione visiva bidimensionale di una stringa di interi, dato che si possono vincolare diverse grandezze grafiche ai dati casuali. Un primo modo è quello di assegnare gli interi casuali alle coordinate x ed y di un punto nel piano, alternandole. Se si osservano agglomerati regolari di punti e differenze rispetto ad una distribuzione idealmente uniforme bivariata si può sospettare la non uniformità della stringa iniziale. Un secondo modo è quello di ordinare gli interi in celle disposte in ordine di lettura in una griglia, e assegnare, stabilito un codice, un colore ad ogni cella, corrispondente al valore dell'intero casuale contenuto nella cella.

Useremo il secondo modo, che nel caso di un analisi qualitativa come quella presentata, sembra essere più informativo.

Il test χ^2 permette di esprimere un giudizio sulle frequenze osservate rispetto alle frequenze attese, è significativo usare questo test su sequenze in cui i valori possibili sono in numero cospicuo, in modo che l'istogramma corrispondente abbia molte colonne da analizzare. Nel nostro caso la rappresentazione di interi senza segno a 8 bit fornisce 256 *bins* ovvero valori osservabili (da 0 a 255).

Per chiarire il significato del test d'ipotesi, notiamo che innanzitutto χ^2 è una v.a. funzione delle variabili aleatorie che rappresentano le frequenze osservate nei vari *bins*. Usando questa distribuzione siamo in grado, sotto opportune condizioni [1], di trovare la probabilità che, dato un processo veramente uniforme, vengano generati dei dati con frequenze uguali o peggiori (nel senso della somma esposta sotto) rispetto al dataset in analisi

$$\chi^2 = \sum_{i=0}^{255} \frac{(O_i - E_i)^2}{E_i} \tag{1}$$

dove O_i sono le frequenze osservate e E_i quelle attese (tutte identiche nel caso uniforme). χ^2 ha una distribuzione tabulata, che dipende dai gradi di libertà della variabile (ovvero il numero di *bins*-1), dalla quale si può selezionare il valore corrispondente ad una coda associata ad una certa probabilità. In base al valore assunto da χ^2 si può stabilire se si attesta prima o dopo questo valore critico, andando a stabilire la plausibilità o meno di un comportamento veramente uniforme. Nel nostro caso assumeremo che la probabilità della coda sia 5%.

Condurremo un'analisi aggiuntiva, sfruttando in particolare il fatto che il metodo usato, `scipy.stats.chisquare()`, restituisce anche il p-value, ovvero l'integrale della coda della v.a. limitata inferiormente proprio *dal valore calcolato*

> Se $A \sim \chi^2(255)$ e $\bar{A}$ è la sua realizzazione
> p-value $= P[A > \bar{A}]$

Usare il p-value ci pone nelle condizioni di non dipendere da una scelta particolare del livello di significatività. In ogni caso stabilendo un livello di significatività di 0.05, ovvero un p-value critico del 5%, l'esito del test sarà da interpretare negativo (i dati non sono coerenti con una distribuzione uniforme) per valori inferiori del parametro, positivo altrimenti.

1.4 Analisi entropica

Infine conduciamo una breve analisi che astrae dalla rappresentazione dei dati, e con un noto algoritmo di compressione, `zip`, visualizziamo la riduzione della dimensione dei file, indice della possibilità di rappresentare i dati in forma più compatta raggruppando caratteristiche prevedibili.

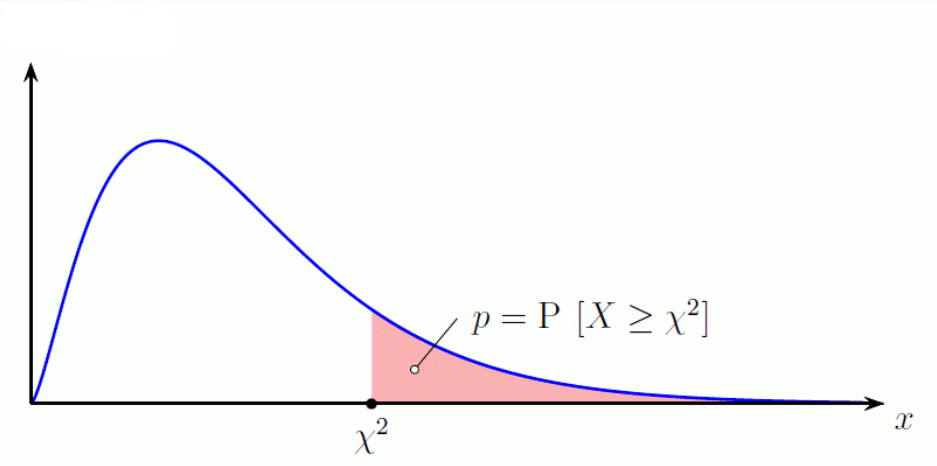


Figura 1: pdf della variabile χ^2 con un esempio di p-value relativo ad una data realizzazione.

2 Analisi

2.1 Datasets

I file binari forniti, {`rng2, rng3, rng4, rng5, rng6, rng7`}, sono tutti di dimensione $976.6KiB$, la loro grandezza è un buon punto di partenza per le seguenti analisi, che possono intrinsecamente soffrire dalla scarsità dei dati (come è naturale aspettarsi). In aggiunta ai dati forniti, i due esperimenti di generazione, descritti alla fine dell'elaborato, portano anche dli altri file `gated_johnson` e `rule30`, che tuttavia sono più limitati nel volume di dati, essendo rispettivamente di $32.7KiB$ e $62.6KiB$. L'analisi non può portare ad un confronto paritetico tra i due gruppi, ma è utile osservare delle caratteristiche che si ripetono indipendentemente dall'effettiva dimensione del file.

2.2 Risultati in batch

In questa sezione riassumerò tutti i risultati dei test in modo da favorire i confronti tra i vari file. Per ogni test i risultati che denotano una buona risposta al test, ovvero che avvallano l'ipotesi di una vera stringa casuale, sono segnati in verde, viceversa vengono segnati in rosso gli esiti critici secondo i livelli limite posti. Il codice dei colori è limitato al caso del test specifico: per trarre conclusioni generali è necessario osservare il comportamento di un dataset nei vari scenari.

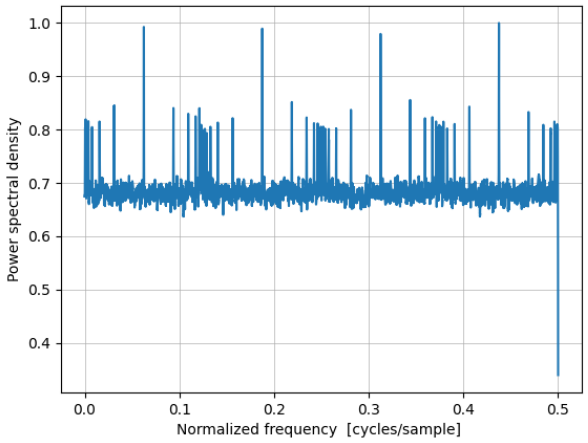
Procediamo ora ad analizzare le stringhe bit per bit

Risultati analisi a singoli bit:

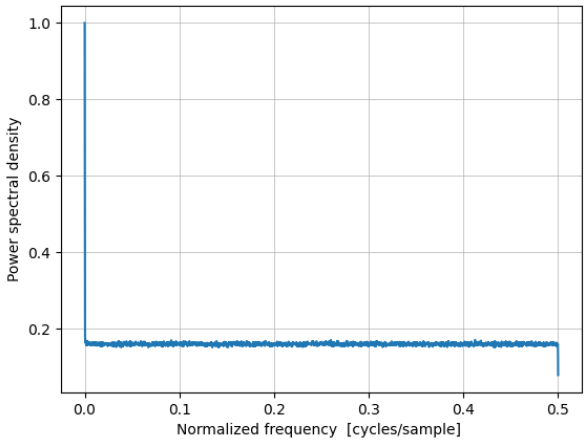
RNG	μ_x campion.	σ_x^2 campion.	fail Autocorrelazione (%)
`rng2`	0.5000	0.2500	5.92
`rng3`	0.5000	0.2500	6.10
`rng4`	0.4981	0.2500	5.71
`rng5`	0.5000	0.2500	12.50
`rng6`	0.5000	0.2500	5.82
`rng7`	0.4982	0.2500	6.35
`gated_johnson`	0.5041	0.2500	10.39
`rule30`	0.5008	0.2500	5.96

Osserviamo che il comportamento in media e varianza è poco significativo per trarre conclusioni, mentre, per quanto riguarda l'autocorrelazione, gli unici RNG che presentano criticità sono `rng5`, che sembra essere molto correlato, e `gated_johnson` che supera appena la soglia. Per avere una comprensione più profonda dei risultati sull'autocorrelazione calcoliamo come anticipato la sua DFT, con l'intento di evidenziare eventuali frequenze indesiderate. Se veramente siamo in presenza di TRNG dovremmo avere uno spettro completamente piatto, se invece i generatori sono pseudocasuali ci aspettiamo di vedere dei picchi a determinate frequenze.

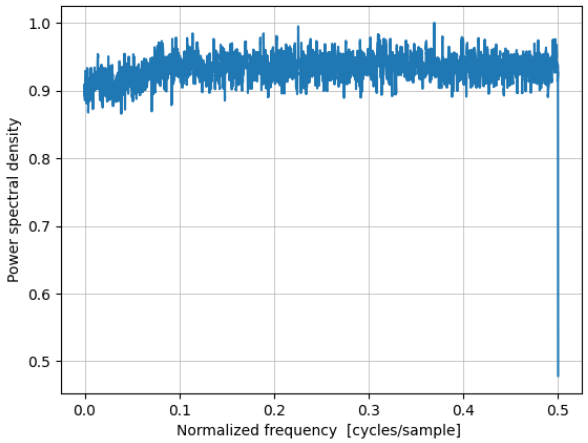
2.2.1 Visualizzazione PSD



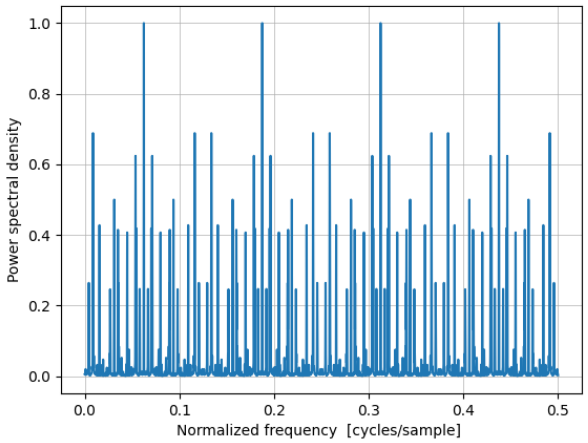
(a) rng2



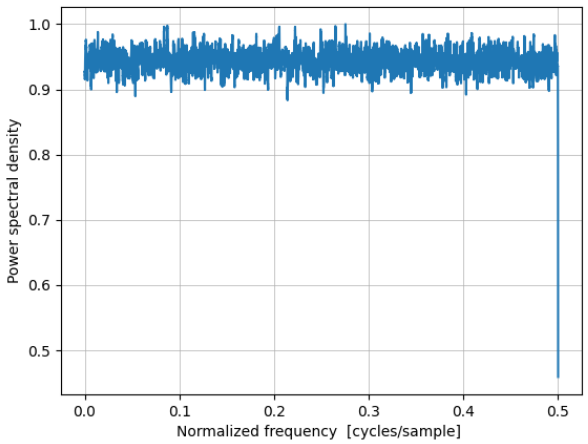
(b) rng3



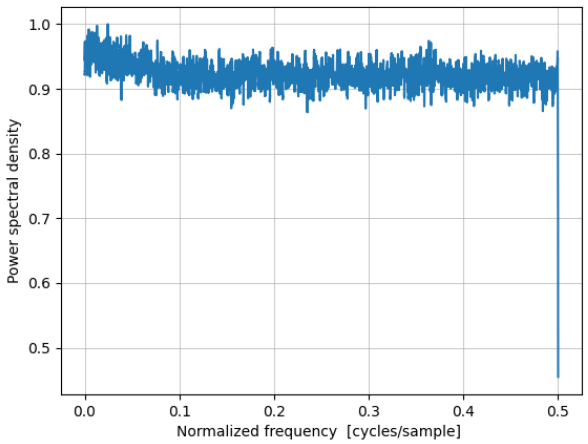
(c) rng4



(d) rng5

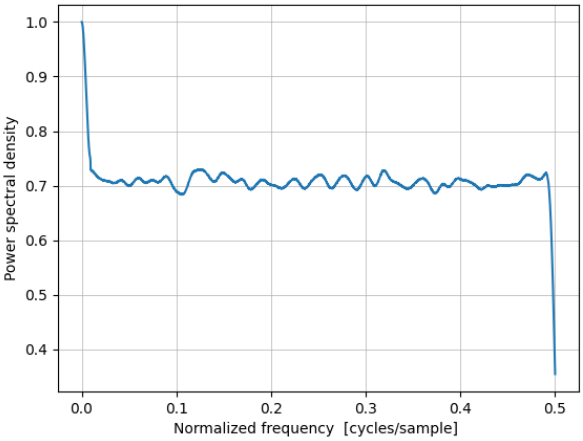


(e) rng6

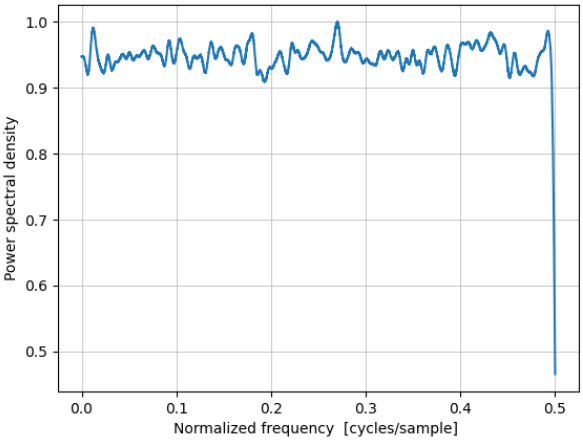


(f) rng7

Figura 2: PSD



(a) gated_johnson



(b) rule30

Notiamo appunto che i grafici rispecchiano quello che avevamo intuito dalle misurazioni dell'autocorrelazione. `rng5` ha un pessimo comportamento in frequenza, con dei picchi marcati, anche però `rng2` esibisce delle periodicità, seppur lievi. Gli altri RNG si differenziano solamente per il comportamento alle basse frequenze (si noti che `gated_johnson` e `rule30` hanno purtroppo meno campioni degli altri e quindi risultano meno dettagliati).

`rng3` ed `rng7` hanno delle componenti importanti in bassa frequenza, mentre il contrario avviene per `rng4`. `rng6` invece si mostra come praticamente piatto in tutta la banda.

2.2.2 Analisi a gruppi di 8 bit

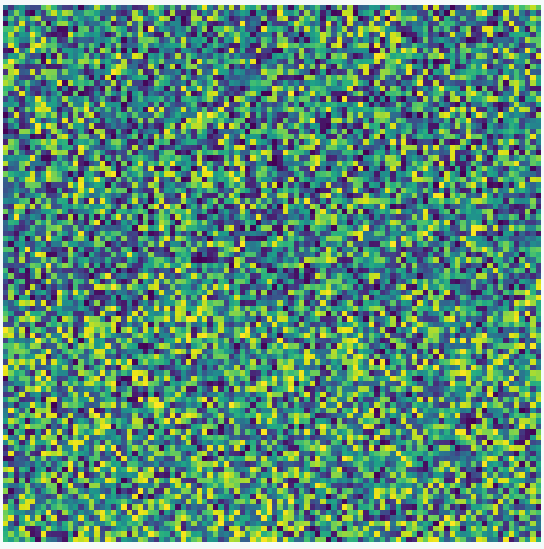
Risultati analisi `uint8`:

RNG	μ_x campion.	σ_x^2 campion.	χ^2 p-value ($P[data\|H_1]$¡0.05)
`rng2`	127.462	5457.575	0.3907
`rng3`	127.464	5479.409	0.0000
`rng4`	127.011	5419.679	0.0000
`rng5`	127.499	5461.237	1.0000
`rng6`	127.466	5457.757	0.2930
`rng7`	127.021	5495.074	0.0000
`gated_johnson`	128.724	5484.916	0.0941
`rule30`	127.749	5465.370	0.4344

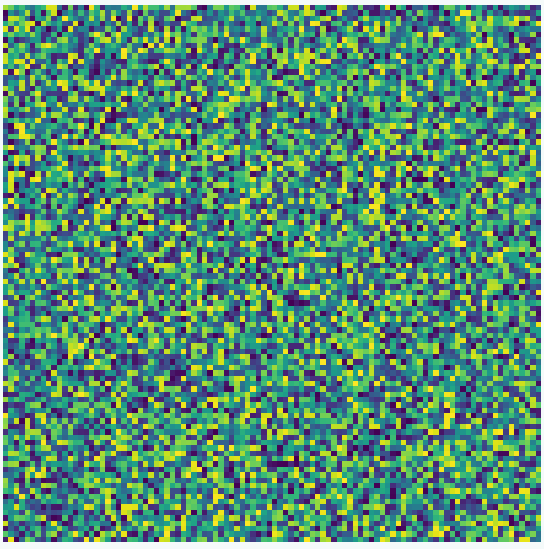
Il test χ^2 presenta un risultato non molto sorprendente: la periodicità di `rng5` è strutturata in modo che tutti i valori dell'intervallo vengano assunti con la stessa frequenza. Perciò il suo comportamento è praticamente perfetto. Lo stesso sembra valere anche per `rng2`, anche se in forma minore. Escludendo dall'analisi `rng5` ed `rng2` osserviamo che, tranne `rng6`, gli altri RNG hanno risultati piuttosto deludenti, mostrandosi lungi dall'essere uniformi nella loro distribuzione globale.

2.3 Analisi visive qualitative

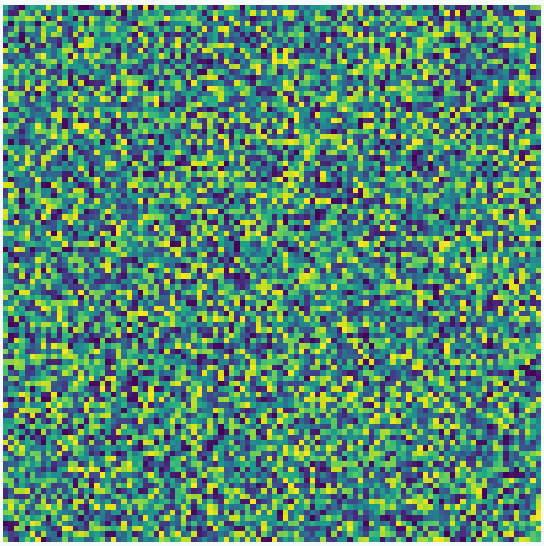
A causa della grande dimensione dei file, è possibile solamente osservare una porzione ridotta della quantità complessiva di dati (questo vale anche per le altre rappresentazioni visive, limitate sia dalla capacità di elaborazione del sistema grafico, sia dalla risoluzione finita dello schermo). L'assunzione implicita è che il processo che ha generato i datasets sia stazionario, altrimenti questa analisi concentrerebbe l'attenzione su un campione troppo piccolo per poter dedurre qualcosa di interessante. Allora sono sotto riportate le visualizzazioni dei primi 100x100 byte dei generatori, a cui è assegnato un codice colore associato al valore intero assunto ($0 \to 255$)



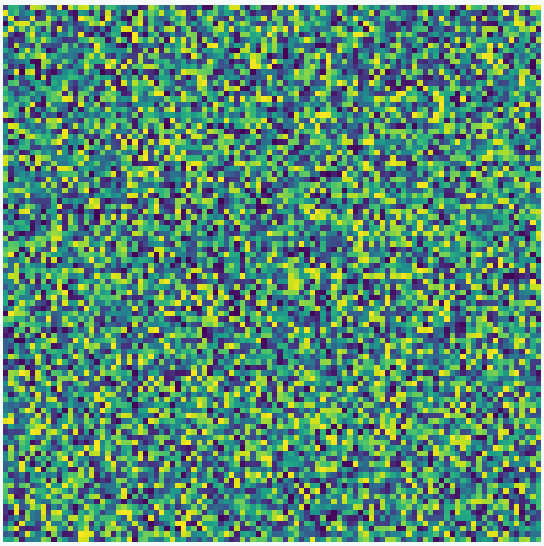
(a) gated_johnson



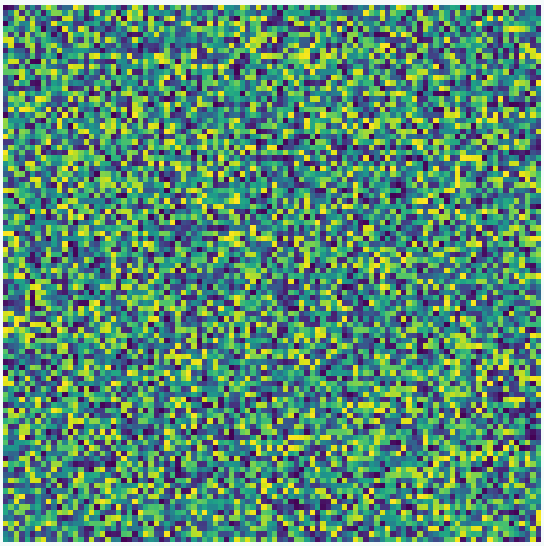
(b) rule30



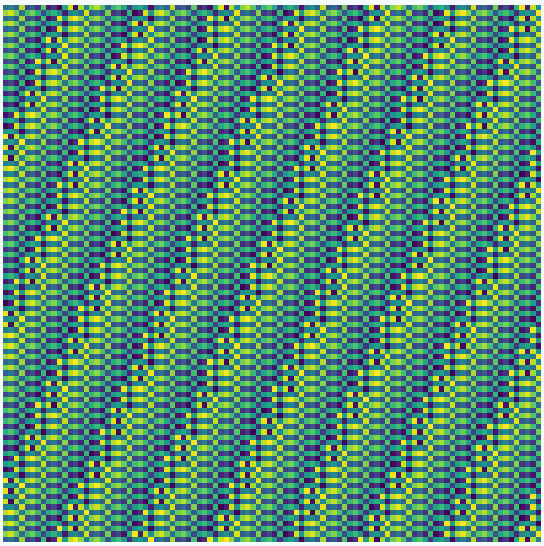
(a) rng2



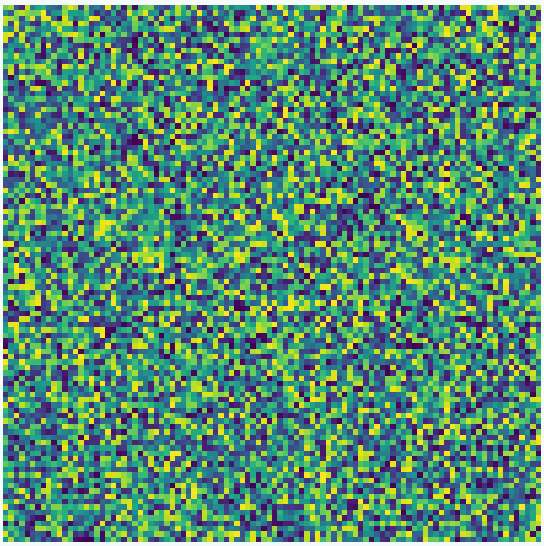
(b) rng3



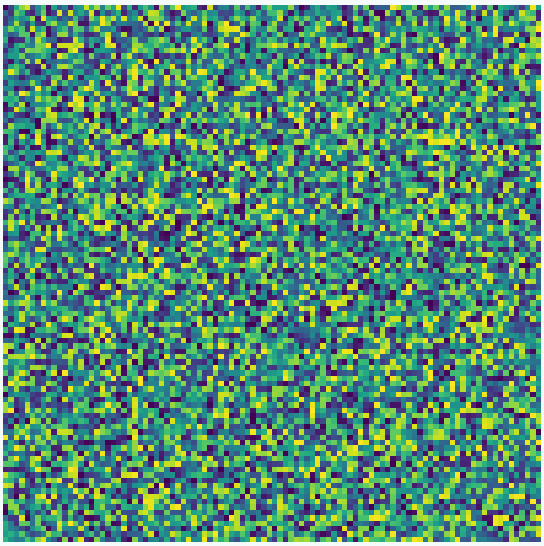
(c) rng4



(d) rng5



(e) rng6



(f) rng7

Si notano evidenti pattern ripetitivi in `rng5`, mentre in `gated_johnson` sembra che la metà superiore sia di colore leggermente più scuro rispetto alla inferiore. Una dinamica del genere suggerisce una giustificazione del motivo delle componenti in bassa frequenza osservate per l'analisi a singoli bit. Per modificare il comportamento di questa variazione di `gated_johnson` ho provato ad implementare il semplice metodo di Von Neumann per l'eliminazione del bias, supponendo che le fluttuazioni fossero sufficientemente prolungate nel tempo da permettere di valutare molte coppie di bit $0 \to 1$ e $1 \to 0$ nelle stesse condizioni. Purtroppo il metodo non ha dato i risultati sperati.

Il test visivo non aiuta nella decisione per i precedenti sospetti su `rng2` ed `rng4`. Guardando il risultato per `rng5` si può stimare la sua periodicità in $255 celle \cdot 8 bit/cella = 2040 bit$.

2.4 Analisi entropica con `zip`

Analisi entropica `uint8`:

RNG	diminuzione relativa dimensioni (%)
`rng2`	0.92
`rng3`	0.00
`rng4`	1.14
`rng5`	99.54
`rng6`	0.00
`rng7`	0.00
`gated_johnson`	-0.61
`rule30`	-0.16

Ormai gli indizi contro `rng5` ed `rng2` si accumulano in maniera vistosa, anche l'algoritmo di compressione riesce a ridurli di una quantità notevole. L'algoritmo fa dubitare anche di `rng4`, che evidentemente non ha entropia massima. Si noti che la diminuzione percentuale negativa dei due set sperimentali indica la rilevanza maggiore dell'header introdotto dal programma di conversione nell'archivio.

3 Interpretazione delle analisi e conclusioni

In base alle analisi fin qui fatte siamo sicuramente in grado di escludere `rng5` ed `rng2` dalla lista dei TRNG per le loro periodicità che sono molto evidenti. Anche `rng4` evidenzia delle carenze a livello dalla compressione di quasi l'1%, e necessita di analisi ulteriori. `rng3`, `rng7` hanno passato i test evidenziando solo criticità nel test di χ^2, mentre `rng6` non ha fallito nemmeno un test, ed è plausibilmente un dataset da un TRNG.

Infine i due esperimenti di generazione hanno risultati nel complesso soddisfacenti, ma non possono essere confrontati direttamente con gli altri RNG per via del troppo ridotto volume di valori generati.

Due esperimenti di generazione

3.1 Generatore con rumore di Johnson-Nyquist

Il setup sperimentale è diviso in una parte di generazione di rumore (analogica) e quantizzazione (a 1 bit), ed una parte di acquisizione. Il tutto è realizzato con l'intento di costruire un prototipo facile da assemblare e poco sensibile alle variazioni dei componenti. La parte analogica è costituita da un amplificatore a 1 stadio CS JFET, 2 stadi CE BJT, e 1 inseguitore di emettitore BJT. La configurazione permette di evitare autooscillazioni e ricezioni non intenzionali di interferenze RF, grazie all'introduzione di un polo dominante con una capacità di Miller. L'uscita del rumore (circa sui $2V_{RMS}$) viene addizionata ad una tensione continua di $2.5V$, filtrata da un semplice filtro RC passa basso, e mandata in ingresso ad un inverter CMOS, dalla cui uscita viene pilotato un pin di lettura digitale del microcontrollore di un Arduino UNO. Le frequenze in gioco fanno in modo che la frequenza di lettura sia molto più bassa dei quella di commutazione (media) dell'invertitore.

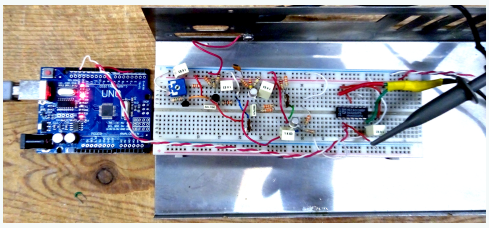


Figura 6: Setup sperimentale

Uno svantaggio di questo esperimento è la lentezza nell'acquisire i dati, che ha permesso di ottenere solamente $32.7KiB$ di dati. Tuttavia per avere una generazione più veloce sarebbe sufficiente un'innalzamento della frequenza di lettura dell'Arduino, unitamente ad un aumento della frequenza di taglio del filtro del rumore.

3.2 Generatore con cellular automata (regola 30 di S. Wolfram)

Il generatore qui descritto segue un noto schema di automa cellulare ideato da Stephen Wolfram e descritto nel suo celebre "A New Kind of Science" [3]. Si tratta di un automa ad una dimensione: partendo da una condizione iniziale in cui tutte le celle che rappresentano la prima riga sono a 0, tranne quella centrale, una regola di trasformazione permette di generare iterativamente tutte le celle della riga successiva

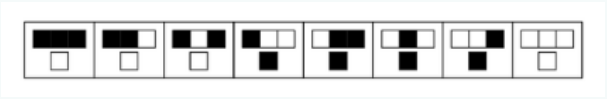


Figura 7: La regola di avanzamento fa dipendere una cella dalle 3 che le stanno sopra secondo questa geometria

la regola è chiamata "rule 30" perché, leggendo in binario la successione di bit dei risultati della regola si ha $00011110_2 = 30_{10}$. In effetti lo schema è molto semplice, ma abbastanza sorprendentemente, nella metà destra della figura, sembra esibire un comportamento caotico e imprevedibile, anzi simile alle rappresentazioni visuali che abbiamo ricavato per gli altri RNG, tant'è che nel linguaggio Mathematica è stato usato per la generazione di interi casuali [2].

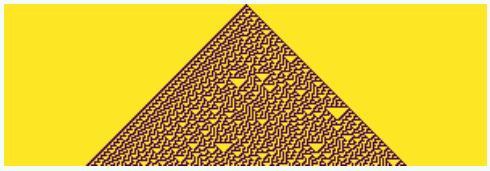


Figura 8: Cellular automata rule 30, apice

In particolare lo stream di bit casuali è estratto dalla colonna centrale dell'automa. Con un programma in C++ ho implementato l'automa con l'algoritmo naïf: anche in questo caso era troppo lento per ottenere molti dati in breve tempo, il volume totale di dati generati ammonta a $62.6KiB$.

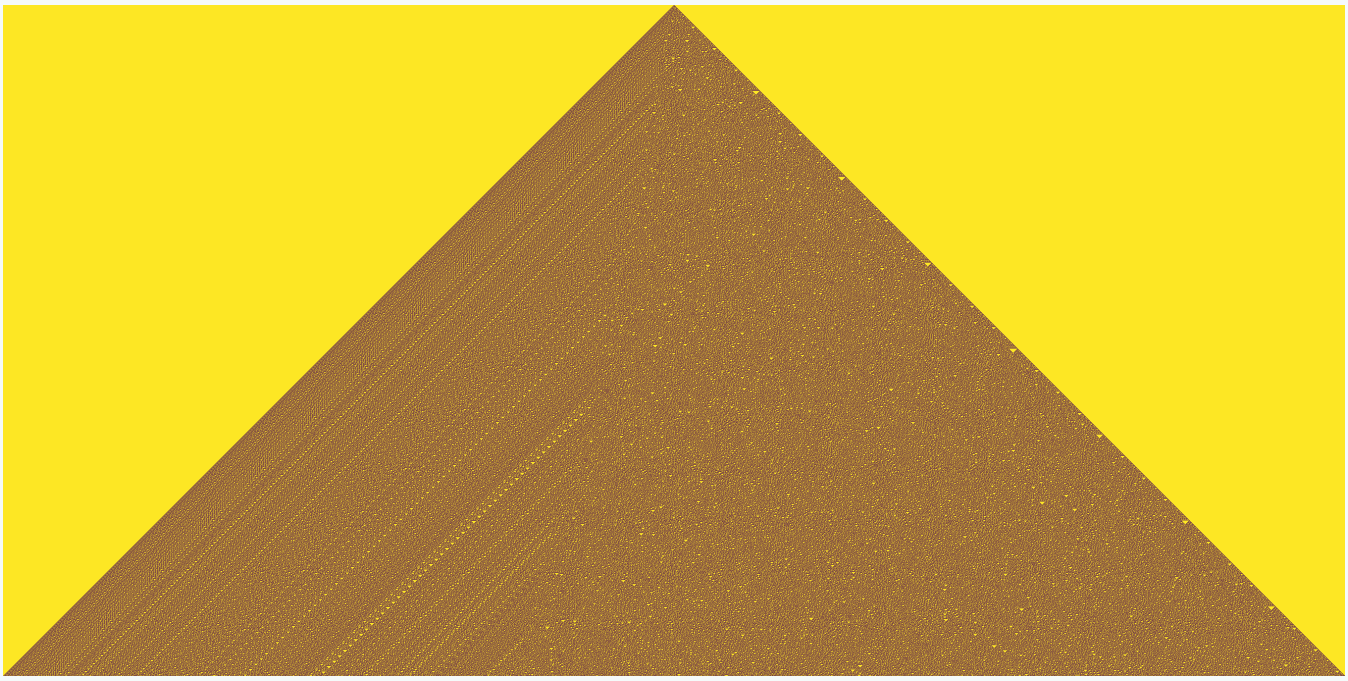


Figura 9: rule 30, estesa

Riferimenti bibliografici

[1] R.V.Hogg, J.W.McKean, A.T.Craig, *Introduction to Mathematical statistics*, 8th edition, Pearson.

[2] `https://mathworld.wolfram.com/Rule30.html`.

[3] `https://www.wolframscience.com/nks`.