Casualità

Algoritmi per l'intelligenza artificiale

Vincenzo Bonnici
Corso di Laurea Magistrale in Scienze Informatiche
Dipartimento di Scienze Matematiche, Fisiche e Informatiche
Universitá degli Studi di Parma

2025-2026

Numeri, casualità e simulazioni

Di seguito, continueremo ad accennare ad alcune tra le molteplici applicazioni nelle quali intervengono i cosiddetti **numeri casuali** ed a come la matematica consenta di affrontare problemi legati al concetto di caso.

Numeri casuali sono utilizzati per costruire simulazioni di natura probabilistica di fenomeni fisici (reattori nucleari, traffico stradale, aereodinamica), chimico-biologici (simulazione di reazioni metaboliche) di problemi decisionali e finanziari (econometrica, previsione Dow-Jones), informatica (progettazione VLSI, rendering) o come semplice fonte di divertimento (videogiochi).

Il forte legame che esiste tra il gioco e le simulazioni probabilistiche è sottolineato dal fatto che a tali simulazioni viene generalmente dato il nome di **metodi Monte Carlo** (in onore del famoso casinò a Monaco).

Generatori di numeri casuali

Innanzitutto cosa è un numero casuale?

Le proprietà statistiche che una sequenza di numeri casuali deve possedere sono uniformità ed indipendenza. Si supponga di dividere l'intervallo $[0, 1]$ in n sottointervalli di uguale ampiezza.

Conseguenza della **proprietà di uniformità**:
Se si eseguono N osservazioni di un numero casuale, il numero atteso in ogni sottointervallo è pari a N/n.

Conseguenza della **proprietà di indipendenza**:
La probabilità di ottenere un valore in un particolare intervallo è indipendente dai valori precedentemente ottenuti.

Generatori di numeri casuali

Un esempio che tutti conosciamo consiste nel lancio di un dado, in effetti l'imprevedibilità del numero ottenuto come punteggio, compreso tra 1 e 6, conferisce allo stesso una forma di casualità.

L'idea stessa di utilizzare un calcolatore (quindi un'oggetto puramente deterministico e di conseguenza prevedibile), per generare un numero casuale quindi imprevedibile sembra costituire una sfida impossibile.

In effetti nessun calcolatore è in grado di generare numeri puramente casuali, ma solo **numeri pseudo-casuali** o quasi-casuali ossia numeri generati da algoritmi numerici deterministici in grado di superare una serie di test statistici che conferiscono a tali numeri una apparente casualità.

Il metodo middle-square

Nei primi anni dell'era dei computer (1946) John von Neumann suggerì il famoso metodo middle-square per generare numeri pseudo casuali distribuiti in modo uniforme.

In tale distribuzione uniforme ogni possibile numero in un determinato intervallo è ugualmente probabile. Ad esempio se lanciamo un dado un certo numero di volte ognuna delle facce da 1 a 6 si presenterà circa 1/6 delle volte originando così una successione uniforme di numeri casuali compresi tra 1 e 6.

Oggi il metodo middle-square ha un'importanza solamente storica, ma nella sua semplicità evidenzia un aspetto importante nella generazione di numeri pseudo-casuali al computer, ossia la necessità di avere a disposizione molti numeri casuali ed in modo rapido.

Il metodo middle-square

Supponiamo di volere generare un numero casuale di 4 cifre, ossia un numero tra 0000 e 9999. Il metodo middle-square richiede come tutti i generatori di numeri casuali un valore iniziale, detto **seme** dal quale vengono generati i successivi valori.

Ad esempio a partire da 1234, elevando tale numero al quadrato abbiamo le otto cifre 01522756 delle quali teniamo solamente le quattro cifre di mezzo 5227. Da queste ripetendo il procedimento otteniamo 27321529 e quindi 3215.

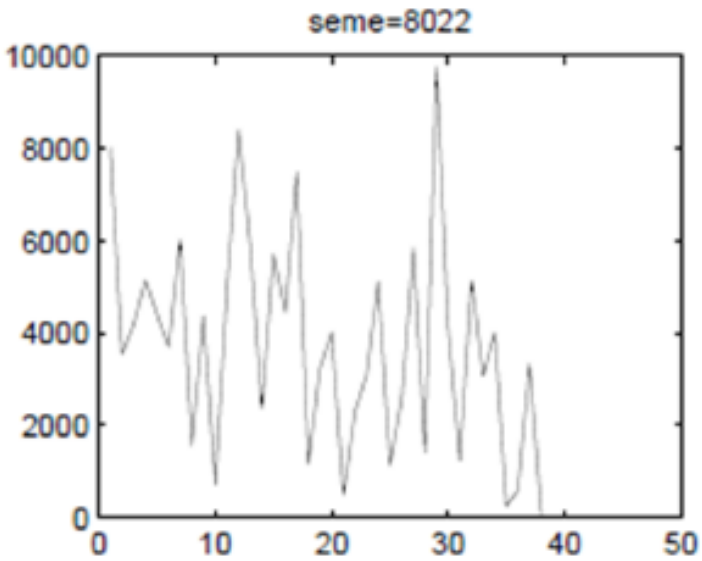
Ogni nuovo numero nella successione è **determinato univocamente** dal suo predecessore. La successione generata quindi non potrà essere casuale ma avrà solo il carattere di **apparente casualità**.

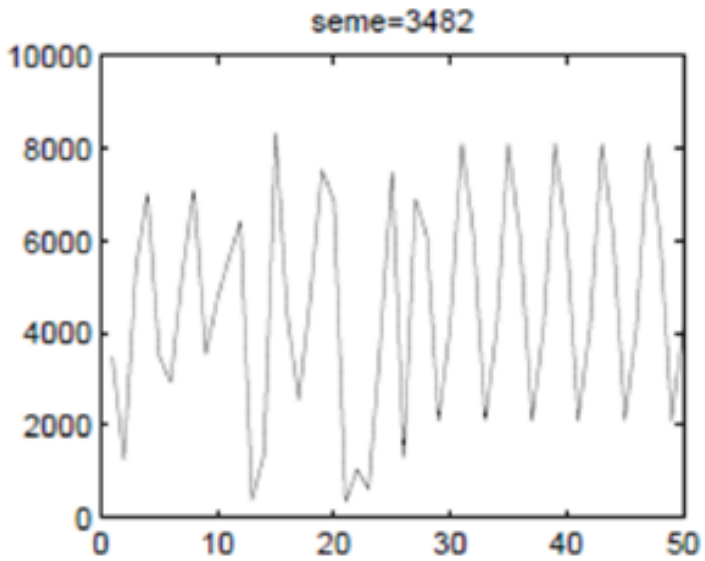
Il metodo middle-square: limiti

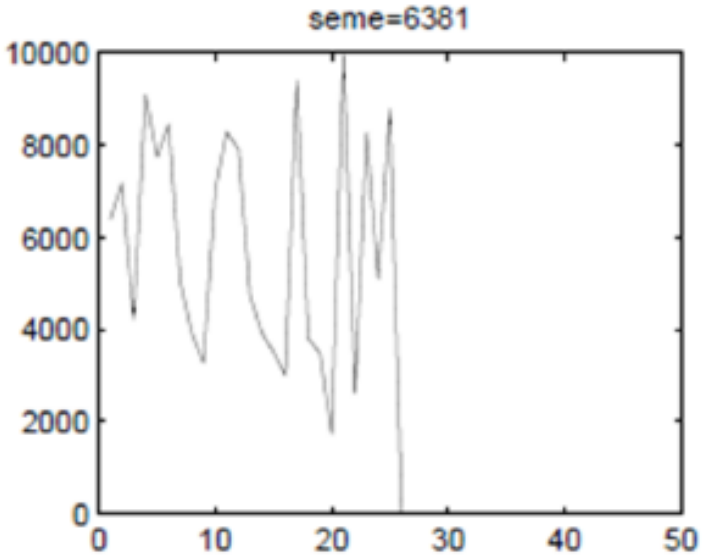
In particolare ogni successione di numeri generati da questo algoritmo inizierà a ripetersi prima o poi. Il numero di numeri della sequenza prima che intervenga una ripetizione è detto periodo della sequenza. La lunghezza di tale periodo può essere considerata una misura della bontà del generatore di numeri pseudo-casuali.

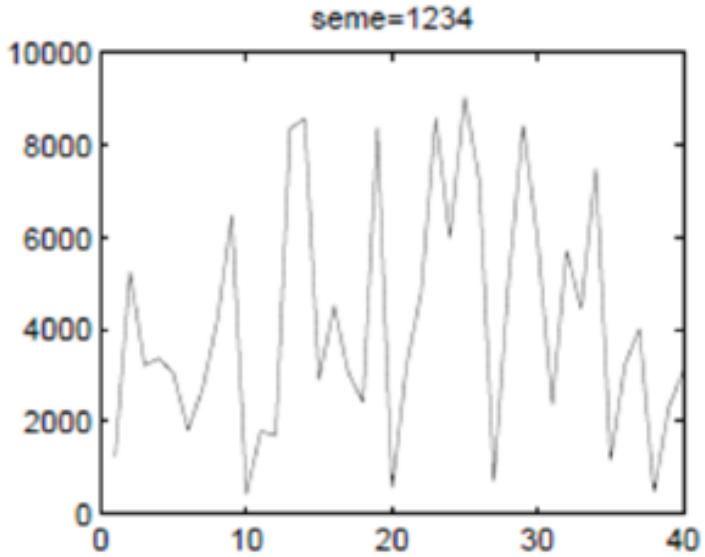
Sfortunatamente il metodo può degenerare in sequenze con periodi molto brevi, ad esempio a partire dal valore 0 la sequenza ha sempre periodo 1, oppure partendo con 43 e numeri a 2 cifre otteniamo la sequenza 43, 84, 05, 02, 00, 00, ...

Semi e metodo middle-square









Un dado simulato

Proviamo ad utilizzare il metodo middle-square per effettuare un semplice esperimento di simulazione. Consideriamo la simulazione del lancio di un dado definendo il risultato ottenuto d come

$$d = 1 + [\frac{5ms}{10^4}]$$

dove *ms* e' il numero generato tramite il metodo middle-square.

Simulando 10 lanci consecutivi a partire dal seme 8022 otteniamo risultati che sembrano abbastanza realistici

5 3 3 4 3 3 4 2 3 1

Basta però aumentare il numero di lanci per giungere a risultati insoddisfacenti (la successione ha infatti periodo 38).

Generatori lineari congruenziali

Nel 1948 venne proposto un generatore di numeri casuali distribuiti uniformemente detto generatore lineare congruenziale o più brevemente LCG che a tutt'oggi è ancora utilizzato. Tale generatore venne presentato per la prima volta dal matematico D.H. Lemer esperto di teoria dei numeri. Il metodo LCG, analogamente al metodo middle-square, ha bisogno di un seme per inizializzare la sequenza di numeri secondo la seguente regola

$$x_{n+1} = (\alpha x_n + c) \bmod m; n \geq 0$$

dove a, c ed m sono opportuni numeri interi costanti. La notazione mod m, ossia modulo m, significa che $\alpha x_n + c$ viene diviso per m e x_{n+1} posto uguale al resto intero della divisione. Quindi x_{n+1} assume valori interi tra $0, 1, 2, \ldots, m-1$.

Generatori lineari congruenziali: esempi

Per esempio le scelte $a = 13$, $c = 0$ (ossia generatore puramente moltiplicativo) e $m = 31$ partendo dal valore iniziale $x_0 = 1$ otteniamo

$$1\ 13\ 14\ 27\ 10\ 6\ 16\ 22\ 7\ 29\ 5\ 3\ 8\ 11\ 19\ 30\ 18\ 17\ 4\ 21\ 25\ 15\ 9\ 24\ 2\ 26\ 28\ 23\ 20\ 12$$

Tale successione ha periodo 30 (ossia $m - 1$). Tutti i numeri da 1 a 30 compaiono per poi ripetersi. Questo non dipende dalla scelta del seme iniziale se non è nullo.

L'uso del seme 0 origina invece la successione costante uguale a 0 per $c = 0$ indipendentemente da a e m. Quindi il massimo periodo di un generatore puramente moltiplicativo è $m - 1$. Nel caso di $c \neq 0$ il massimo periodo sarà invece m, in quanto comparirà anche lo 0. Tale periodo massimo non è però raggiunto per tutte le scelte di a, c ed m, ad esempio per $a = 7$, $c = 7$ e $m = 10$ partendo dal seme 7 si ha per $n = 8$

$$7\ 6\ 9\ 0\ 7\ 6\ 9\ 0$$

che ha solo periodo 4.

Generatori lineari congruenziali: scelte sbagliate

Scelte sbagliate dei parametri per questo tipo di generatori hanno rovinato risultati sperimentali fin dai lontani anni 60 (generatore RANDU).
Ad esempio, un errore nella specifica dell'ANSI C ha portato alla propagazione di un terribile generatore per la rand() del C e del C++.

Quando usiamo metodi che richiedono la generazione di molti numeri casuali (metodi Montecarlo, meta-euristiche, ecc.) bisogna innanzitutto controllare quale tipo di generatore è utilizzato

I difetti più comuni di una routine di generazione di numeri casuali sono:

- numeri non uniformemente distribuiti
- discretizzazione dei numeri generati
- media o varianza non corrette
- presenza di variazioni cicliche

Bontá di un generatore

Il problema della scelta dei migliori valori per a, c ed m è quindi il punto cruciale del metodo. Cosa si intende però per migliori?

Un aspetto importante è la lunghezza del periodo da cui segue che m dovrà essere molto grande: per un dato m i valori di a e c dovranno essere tali che la successione abbia periodo massimo (se m è grande la differenza tra m e $m-1$ è irrilevante e ci si può restringere al caso di generatori moltiplicativi, $c = 0$, che sono più veloci). Una delle scelte più popolari è

$$m = 2^{32} - 1, a = 75, c = 0$$

Questo garantisce un periodo di $2^{31} - 2 = 2.147.483.646$ ossia oltre 2 miliardi di numeri pseudo-casuali (il fatto che $m = 2^{31} - 1$ sia un numero primo è fondamentale al fine di ottenere il massimo periodo).

Bontá di un generatore

Il problema della scelta dei migliori valori per a, c ed m è quindi il punto cruciale del metodo. Cosa si intende però per migliori?

Si dice che la sequenza ottenuta ha periodo pieno se il suo periodo è proprio m, e ciò si verifica quando sono verificate le seguenti condizioni:

- Se m e c sono primi tra loro
- Se m è divisibile per un numero primo b, per il quale deve essere divisibile anche $a - 1$
- Se m è divisibile per 4, allora anche $a - 1$ deve essere divisibile per 4

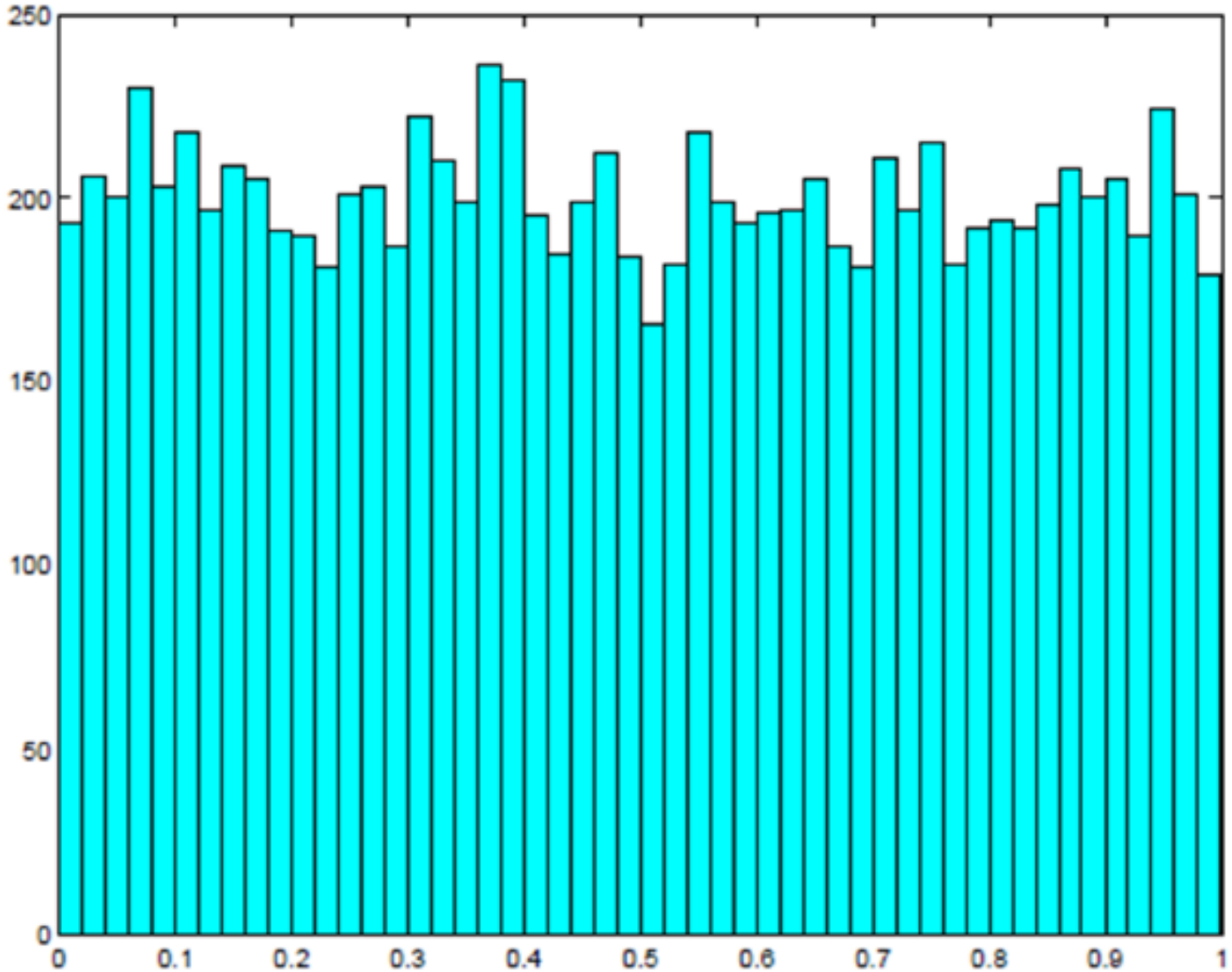
Istogrammi di frequenza

Da un punto di vista pratico molti generatori di numeri casuali restituiscono invece di $x_n + 1$ il suo valore diviso per m, al fine di evitare numeri troppo grandi. Inoltre evitano di reinizializzare la sequenza, se non esplicitamente richiesto.

Generiamo una tabella di N valori pseudo-casuali compresi tra 0 ed 1 e verifichiamone l'uniforme distribuzione tramite l'istogramma di frequenza ottenuto suddividendo l'intervallo $[0, 1]$ in M sottointervalli di uguale ampiezza e calcolando quanti valori cadono in un dato intervallo. Nel nostro caso tale valore teorico dovrebbe essere pari a N/M(con fluttuazioni, appunto, casuali)

Istogrammi di frequenza

Istogramma di frequenza di N=10000 valori in M=50 sottointervalli, N/M=200

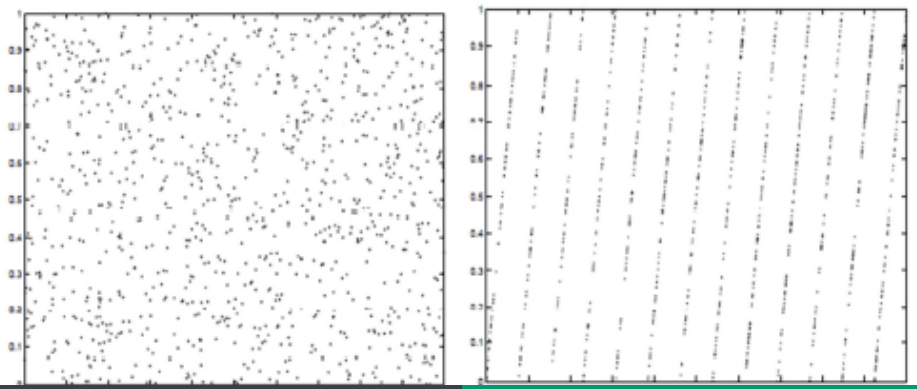


Test di casualitá

Una richiesta più importante nel valutare la bontà di un generatore uniforme di numeri pseudo-casuali è l'assenza di correlazione tra i numeri generati dall'algoritmo. In altre parole non deve emergere nessuna relazione tra x_n e x_{n+1} per $n > 0$.

Questa proprietà può essere verificata graficamente realizzando il grafico bidimensionale dei punti (x_n, x_{n+j}) per $j > 0$. In tale grafico non dovranno comparire linee, forme o altre strutture regolari.

In figura riportiamo il risultato per $j = 1$ con 1000 punti ottenuto con il generatore LCG con scelta ottimale e con lo stesso generatore modificato scegliendo come valori $m = 31$, $a = 13$ e $c = 0$.



Altri generatori

I generatori di numeri casuali più recenti non sono basati sul metodo LCG.

Essi sono una combinazione di operazioni di spostamento di registri e manipolazione sui bit che non richiedono nessuna operazione di moltiplicazione o divisione.

Mersenne twister

- Veloce e di alta qualità - permette di generare punti equidistribuiti in spazi fino a 623 dimensioni
- Periodo molto lungo ($2^{19937} - 1$) (i creatori di questo algoritmo hanno dimostrato questa proprietà)
- Utilizza i numeri primi di Mersenne (da cui il nome), e alcune delle costanti dell'algoritmo sono anch'esse numeri primi di Mersenne
- Ha passato numerosi test statistici di casualità, tra cui il test Diehard, ma non tutti i test della suite TestU01

Hapaxicitá

Un genoma G si definisce **k-hapax genoma** se
$\forall \alpha \in D_k(G) \Rightarrow mult_G(\alpha) = 1$.

Un genoma G si definisce **k-hapax completo** se G é un k-hapax genoma e se $D_k(G) = \Gamma^k$. Esso deve quindi avere una lunghezza di $4^k - k + 1$.

Genomi random

Fissato un k ed una data una lunghezza n, possono esistere piú di un genoma di lunghezza n con entropia k-esima massima. Gli altri genomi *ruotano* attorno ad essi.
Tuttavia...

Proposizione 14

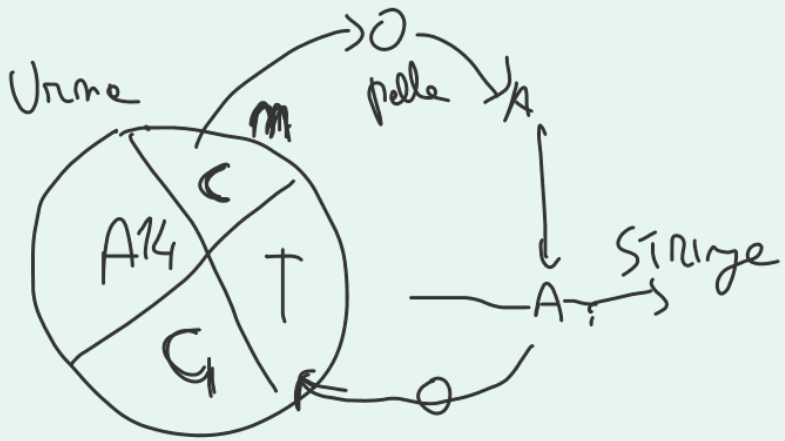
Un genoma k-hapax completo G di lunghezza n ha l'entropia k-esima massima rispetto tutti i genomi della stessa lunghezza.

Per esser vero ció, i suoi k-mer devono tutti avere frequenza uguale a $p = \frac{1}{n-k+1}$.

É quindi importante considerare il giusto valore k nell'analizzare un genoma.

Processo Bernoulliano

Definiamo un **genoma random** come una **stringa Bernoulliana**, cioé generata da un **processo Bernoulliano**.



Data una urna con n palline, un processo Bernoulliano é tale se ad ogni estrazione la pallina estratta viene rimessa nell'urna.

Genomi random

Randomicitá e principio di Bell

Nozione **negativa**: un genoma é random se esso passa tutti i test di randomicitá

Nozione **poisitiva**: un genoma é random se data una sua sottostringa non é possibile prevedere il simbolo successivo (Bell).

Classe random

Indichiamo con RND_n la classe dei genomi random di lunghezza n.

Indichiamo con $RND_{n,m}$ la classe dei genomi random di lunghezza n su un alfabeto di cardinalitá m.

Quindi, la totalitá dei genomi random é data da $RND = \bigcup_{m,n \in \mathbb{N}} RND_{n,m}$.

Principio di log-normalitá dei random (RLNP)

Sia α una stringa di lunghezza n su un alfabeto di m simboli, allora

- $\forall k \leq n$ ogni k-mer $\alpha \in \Gamma^k$ ha la stessa probabilitá a priori di occorrere in ogni posizione $1 \leq i \leq |\alpha| - k + 1$ di α e tale probabilitá equivale a $\frac{1}{|\Gamma^k|} = \frac{1}{m^k}$
- la probabilitá a posteriori di un k-mer di occorrere nella posizione i é uguale alla probabilitá condizionale di occorrere dato il suo prefisso proprio.
- $\Rightarrow \forall k < 2\lceil LG \rceil$ prob. a priori = prob. a posteriori di occorrere in posizione i.

Teorema 15: limite inferiore della hapax-totalitá in una stringa random

$\forall S \in RND_{n,m}, \forall k \geq \lceil 2LG(S) \rceil$: tutti i k-mers sono hapax.

Ricordiamo che $n = |S|$ lunghezza della stringa, $m = |\Gamma|$ caridnalitá dell'alfabeto e $\lceil 2LG \rceil = HB$ hapax bound

Sia k t.c. $|\Gamma^k| = m^k \geq n - k + 1$.
Per il principio di di log-normalitá dei random si deve avere che:

- $Prob(\alpha \in D_k(G)) = \frac{n-k+1}{m^k}$
- $Prob(\alpha \in D_k(G)) = \frac{1}{n-k+1}$

Da cui:

$$\frac{n-k+1}{m^k} = \frac{1}{n-k+1}$$

$$(n-k+1)^2 = m^k$$

$$k = 2log_m(n-k+1)$$

sostituendo k stesso nell'espressione, ed essendo $k << n$,

$$k = 2log_m(n-2log_m(n-k+1)+1)$$

$$2log_m(n-2log_m(n)) \leq k \leq 2log_m(n)$$

da cui

$$2log_m(n) - 2log_m(n-2log_m(n)) = 2log_m \frac{n}{n-2log_m(n)}$$

$$= 2log_m(1 + \frac{2log_m(n)}{n - 2log_m(m)})$$

per n che tende a infinito, $\frac{2log_m(n)}{n-2log_m(m)}$ tende a 0, quindi

$$= 2log_m(1) = 0$$

ovvero, la differenza dei due limiti che racchiudono k si approssima allo 0. Possiamo quindi dire che

$$k \approx 2log_m(n) \Rightarrow k \geq \lceil 2LG \rceil$$

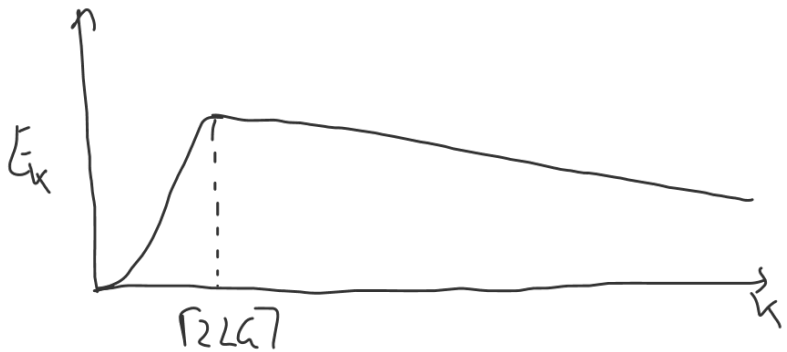
da cui la tesi. □

Teorema 16

Sia $S \in RND_{n,m}$ allora $mrl(S) + 1 = \lceil 2LG(S) \rceil$

La dimostrazione viene banalmente dal teorema precedente, in quanto per $k \geq \lceil 2LG(S) \rceil$ in una stringa random tutti i k-mer sono hapax. Quindi, necessariamente a $k - 1$ devo trovare i repeat di lunghezza piú lunga possibile.□

Il teorema implica che per $k = \lceil 2LG(S) \rceil$ si ottiene l'entropia massima di un random. Da tale k in poi, $D_k(S)$ é sempre e solo formato da soli hapax, tuttavia per $k = \lceil 2LG(S) \rceil$ si ha il dizionario di cardinalitá maggiore, che é quindi in grado di esprimere la massima entropia.



Teorema 17

Sia $S \in RND_{n,m}$ allora $mhl(S) \leq \lceil LG(S) \rceil$

Per $k = \lceil LG(S) \rceil$,

Se esiste almeno un hapax abbiamo la tesi.

Se $D_k(S)$ é formato da soli repeat, se S é veramente random per il principio RLNP la prob. a posteriori deve eguagliare la prob. a priori. Ovvero, un daro prefisso non puó darmi alcuna informazione su un ipotetico repeat β che occorre in posizione i.
La prop. di occorrere al posizione i é $\frac{1}{n-k+1}$, e la prob. di occorrere in $i+1$ deve obbligatoriamente essere diversa perché ho giá osservato β in posizione i.
Quindi, la prob. a posteriori deve essere maggiore della prob. a priori:

$$\frac{1}{n-k+1} > \frac{1}{m^k}$$

Genomi random

$$m^k > n - k + 1$$

$$log_m(n - k + 1) > k$$

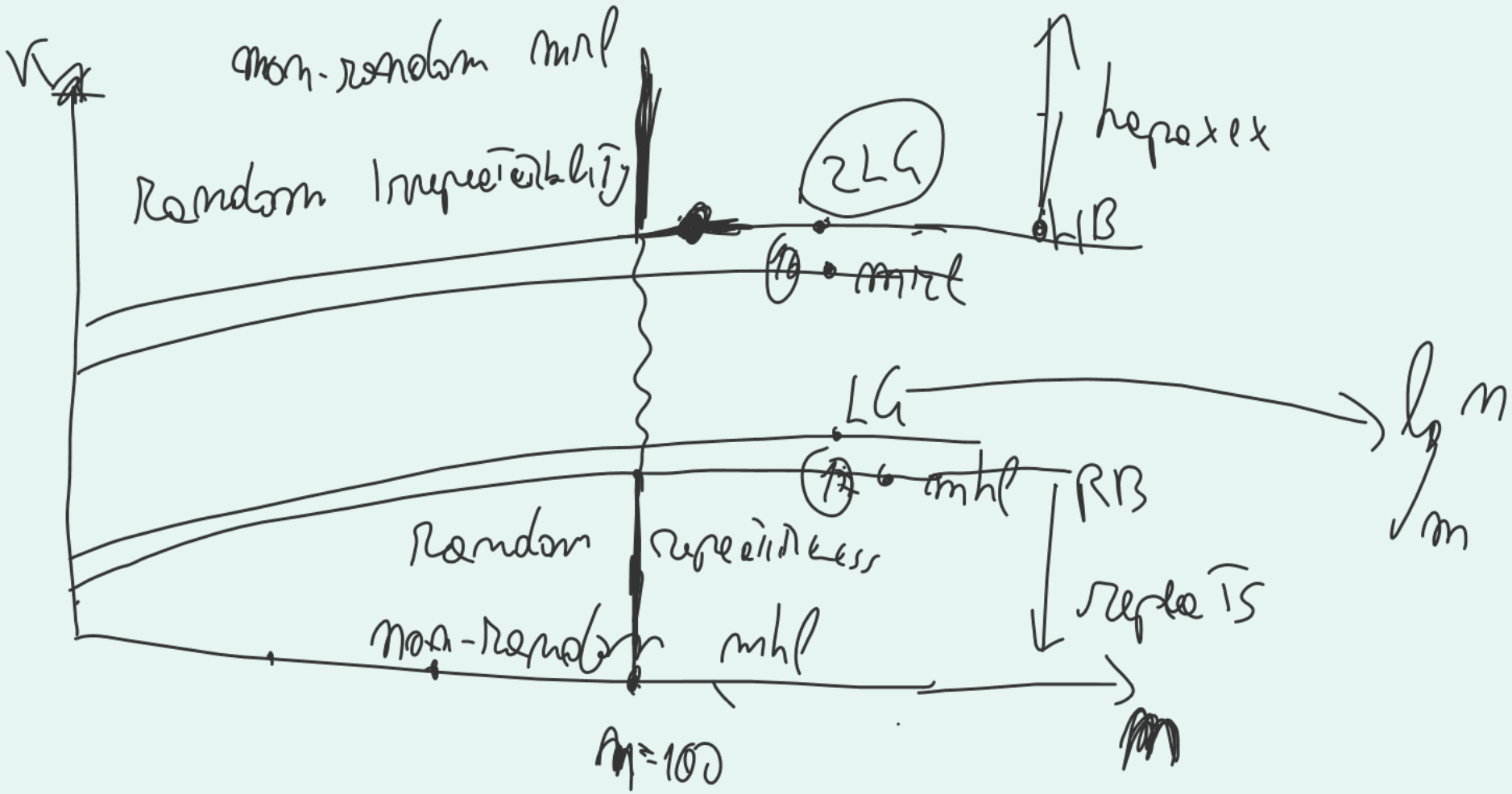
ricordandoci che $k << n$ e che $LG = log_m(n)$, si ha che

$$log_m(n) > k \Rightarrow mhl \leq \lceil LG \rceil$$

Il $\leq$ viene introdotto dal fatto che stiamo togliendo un fattore $-k + 1$ dalla equazione. $\square$

Genomi random

Grafico della relazione tra i vari indici informazionali nelle stringhe random e loro applicazione per interpretare la ripetitivitá



Leggi genomiche informazionali

Data una misura, é spesso utile investigare quali sono i limiti entro cui tale misura i fenomeni reali esistono.
Un esempio concreto é il ph (o altezza di aciditá). Sappiamo che valori troppo bassi o troppo alti di tale indicatore sono dannosi per la salute o addirittura incompatibile con la vita.

In questa lezione introdurremo delle misure, basate sul contenuto informazionale dei genomi, che sono state dimostrate essere indicative dei limiti entro i quali l'informazione *é contenuta* nei genomi. Essi sono stati associati al livello di complessitá di un genoma se esso é rappresentato sotto forma di stringa.
Introdurremo tali misure e vedremo quali sono le leggi che le regolano nei genomi reali.

La *risoluzione* ideale

Punto cruciale di questa nostra analisi sará la giusta *risoluzione* con cui estrarremo il contenuto informazionale dei genomi.

Dato un genoma reale G, tale risoluzione é la lunghezza di parola k per cui in un genoma random della stessa lunghezza di G si ha entropia massima. Abbiamo giá visto che tale k é $2LG$.

Cercheremo quindi di estrarre l'**informazione utile** di un genoma reale **confrontando** la sua entropia con quella di un **genoma random della stessa lunghezza**.

Leggi ellittiche

Un proprietá universale di $E_{2LG(G)}(G)$ é la **legge ellittica**, che costituisce la prima delle leggi che prenderemo in esame. Essa stabilisce che

$$LG(G) < E_{2LG(G)} < 2LG(G)$$

Per semplicitá in quanto segue ometteremo di scrivere (G).

Entropia ed anti-entropia

Si definisce **componente anti-entropica** di un genoma la misura

$$AC = 2LG - E_{2LG}$$

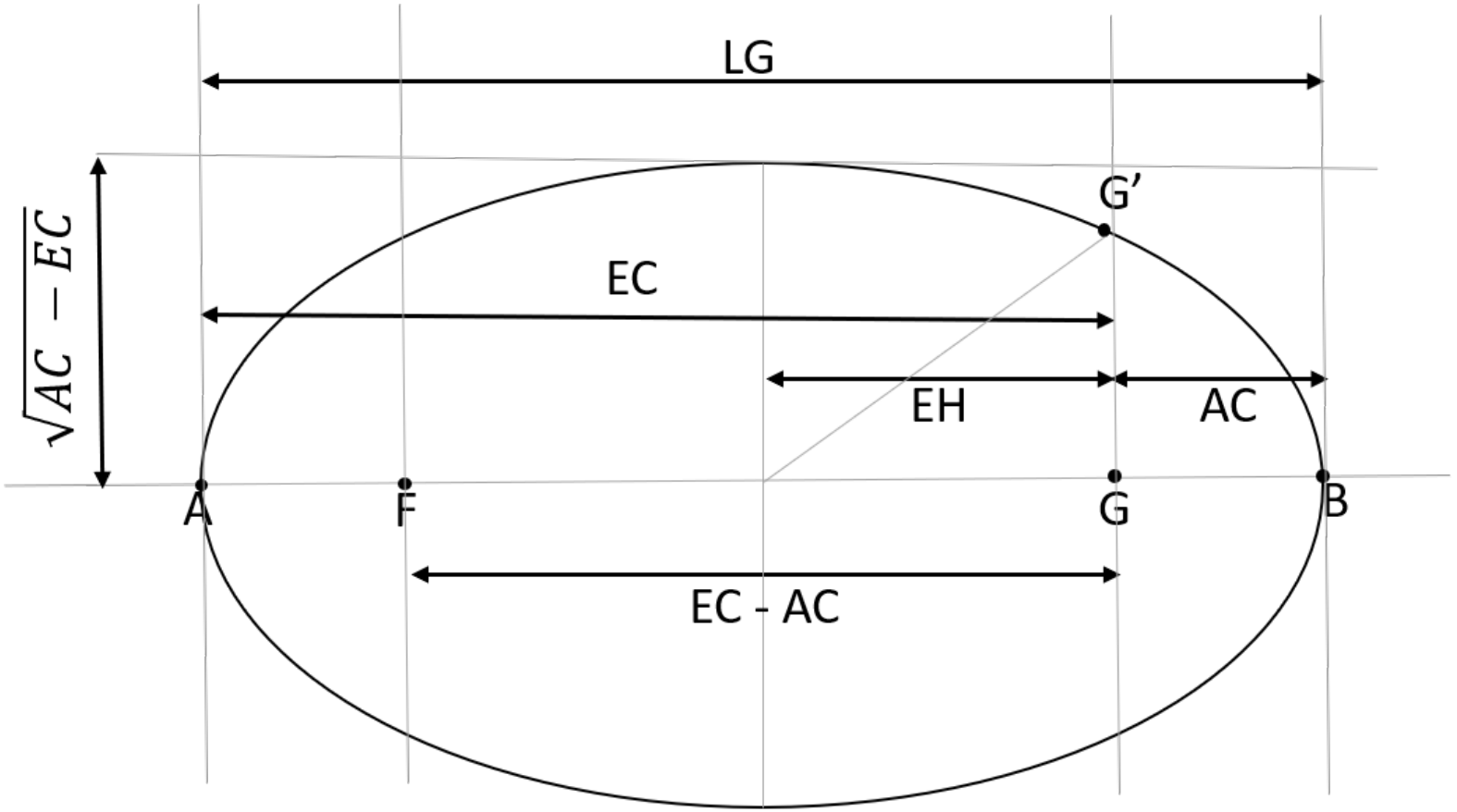
Si definisce **componente entropica** di un genoma la misura

$$EC = LG - AC$$

Si definisce **frazione entropica** di un genoma la misura

$$AF = \frac{AC}{LG}$$

Leggi genomiche informazionali



Eccentricitá

L'**eccentricitá** della ellisse é data da:

$$EH = \frac{\bar{FG}}{\bar{AB}} = \frac{|EC - AC|}{LG}$$

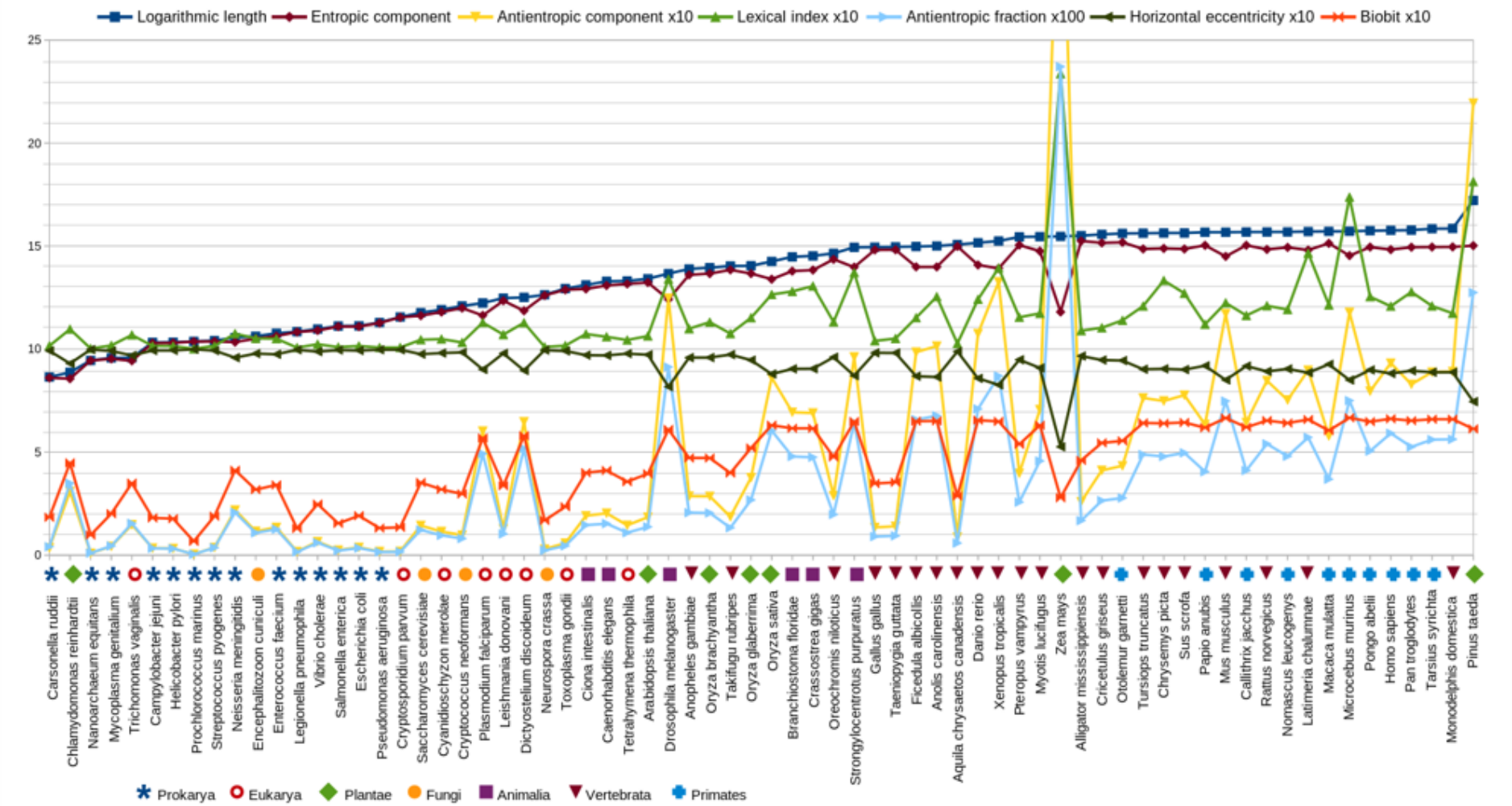
dato che $LG = EC + AC$ e $EC = LG - AC$, si ha che

$$EH = \frac{EC}{LG} - \frac{AC}{LG}$$

ed essendo $AF = \frac{AC}{LG}$

$$EH = \frac{LG - AC}{LG} - AF = \frac{LG}{LG} - \frac{AC}{LG} - AF = 1 - 2AF$$

Distribuzioni genomiche



Leggi genomiche informazionali

Come introdotto all'inzio del corso, in un genoma ci sono due forze distinte e contrapposte:

- la forza entropica la cui funzione é essenzialmente evolutiva e che implementa la visita delle possibilitá evolutive
- la forza anti-entropica il cui scopo é di preservare la struttura funzionale della stringa genoma

tali forze devono essere in un certo equilibrio per far si che la vita persista.

Principio di Pareto

Pareto, uno studioso italiano vissuto a cavallo tra il 1800 e il 1900, enunció in seguente principio:

circa il 20% delle cause provoca l'80% degli effetti

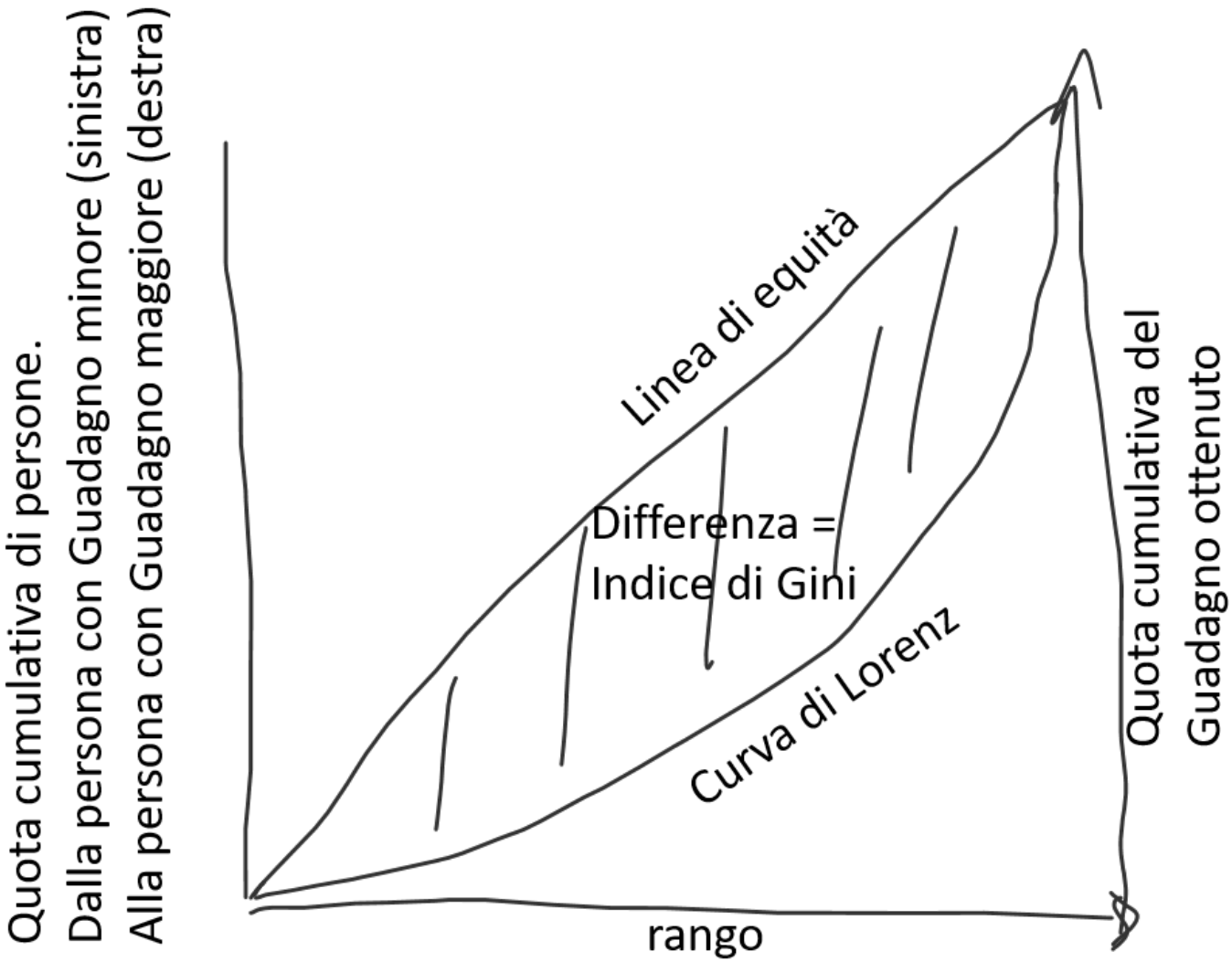
.

Tale principio era dovuto al fatto che in Italia l'80% delle terre era posseduta dal 20% della popolazione. Tuttavia si é visto che tale principio si riscontra in molti sistemi complessi reali.

$$H = G = |2A - 1| = |1 - 2B|$$

dove A=20% e B=80%. H é l'indice di **Hoover** che descrive la porzione di profitto da redistribuire per avere una distribuzione uniforme. G é l'indice di **Gini** che rappresenta una misura statistica per la dispersione del guadagno.

Leggi genomiche informazionali



Leggi genomiche informazionali

É stato osservato che sui genomi reali valgono le seguenti **leggi**:

$$EC > LX \cdot AC$$

$$LX \cdot EC - \frac{LG}{LX} > AC$$

$$1 < EH \cdot LX < 1 - AF$$

Complessitá genomica

Inoltre, data la seguente misura di **complessitá genomica**

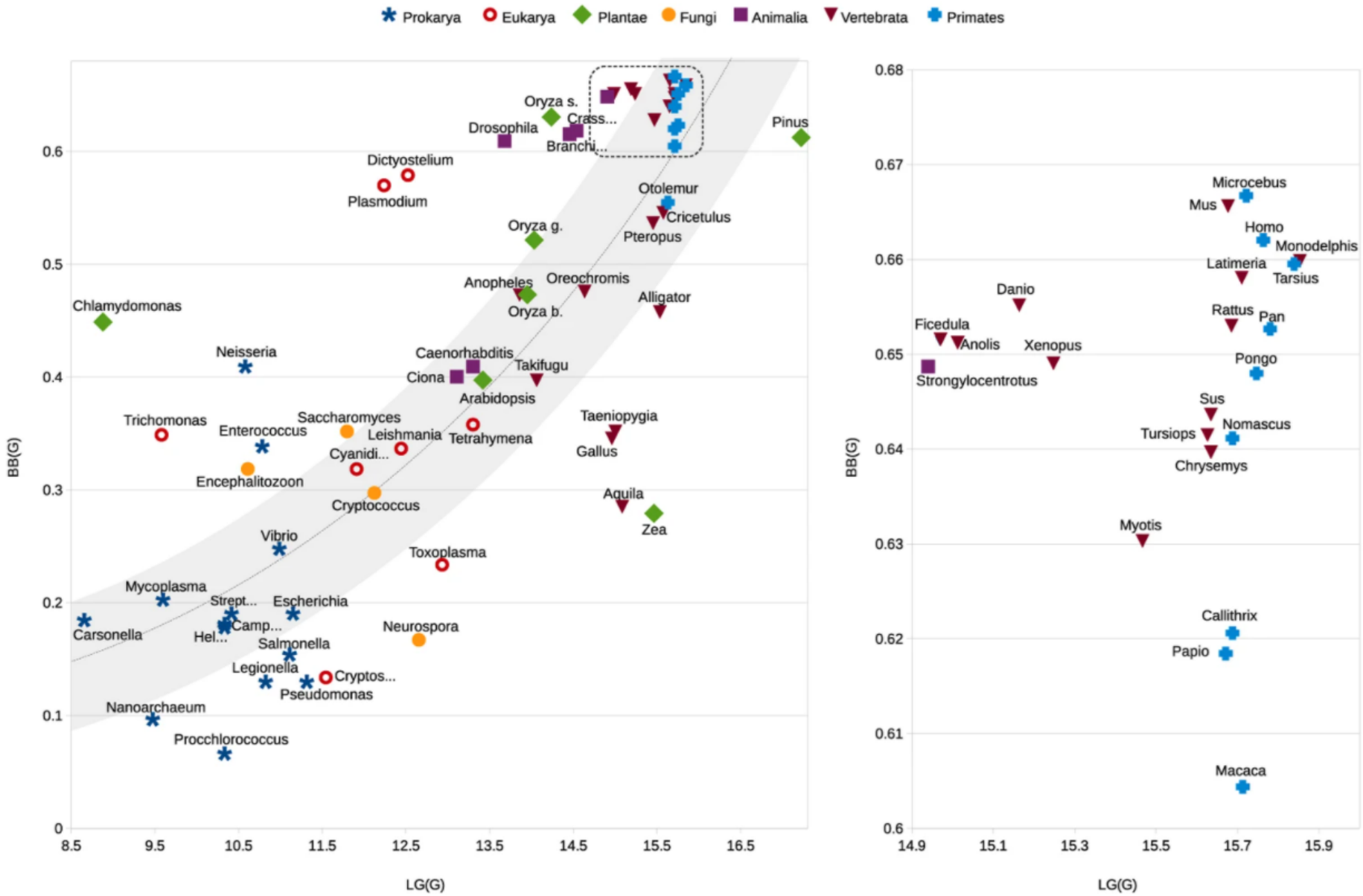
$$BB = AC^{\gamma}(\frac{LG - 2AC}{LG})^{3\gamma+2} = AC^{\gamma}(EH)^{3\gamma+2}$$

, dove γ é la costante di Eulero-Mascheri, si é visto che

$$BB(G_1) < BB(G_2)$$

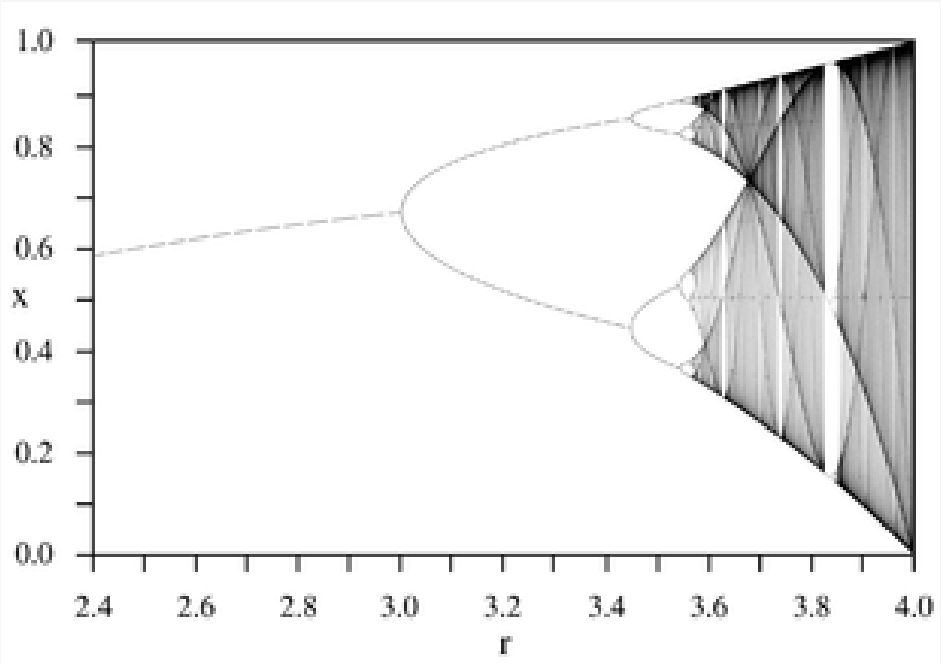
se G_1 é convenzionalmente considerato meno complesso di G_2

Leggi genomiche informazionali



Mappa logistica

La mappa logistica é una descrizione grossolana del caos e del fatto che esso dipende dalle condizioni iniziali. Viene spesso usata per *simulare* il caos.



Stringhe random

Definiamo un test di randomicitá tale che una stringa S é random se

- $mrl(S) = 2\lceil LG(S) \rceil - 1$
- $mhl(S) < \lceil LG(S) \rceil = log_m(|S|)$

Se prendiamo le cifre decimali di π e le convertiamo in una stringa nell'alfabeto $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ otteniamo per $|S| = n$ i seguenti indici

n	MCL	MHL	$\lceil LG \rceil$	MRL	$\lceil 2LG \rceil$	$\mu_{\lceil LG \rceil}$	$\mu_{\lceil 2LG \rceil}$
100,000	4	4	5	9	10	1.58 (0.81)	1.00 (0.00)
1,000,000	4	5	6	12	12	1.58 (0.81)	1.00 (0.00)
2,000,000	5	6	7	12	14	1.10 (0.33)	1.00 (0.00)
5,000,000	5	6	7	12	14	1.27 (0.54)	1.00 (0.00)
10,000,000	5	6	7	14	14	1.58 (0.81)	1.00 (0.00)
20,000,000	6	7	8	14	16	1.10 (0.33)	1.00 (0.00)
50,000,000	6	7	8	15	16	1.27 (0.54)	1.00 (0.00)

Table 1: Decimal digits of π.

n	MCL	MHL	$\lceil LG \rceil$	MRL	$\lceil 2LG \rceil$	$\mu_{\lceil LG \rceil}$	$\mu_{\lceil 2LG \rceil}$
400,000	4	14	19	43	38	2.61 (1.80)	1.00 +3E-5(0.01)
4,000,000	4	18	22	49	44	3.77 (2.67)	1.00 +8E-5 (0.00)
8,000,000	4	19	23	49	46	4.08 (2.86)	1.00 +3E-5(0.00)
20,000,000	4	20	25	52	50	3.27 (2.26)	1.00 +1E-5(0.00)
40,000,000	4	22	26	57	52	3.54 (2.43)	1.00+ 1E-5(0.00)
80,000,000	4	23	27	57	54	3.86 (2.62)	1.00 (6E-4)
200,000,000	4	24	28	63	56	5.15 (3.37)	1.00 (5E-4)

Table 2: Binary conversion of decimal digits of π.

n	MCL	MHL	$\lceil LG \rceil$	MRL	$\lceil 2LG \rceil$	$\mu_{\lceil LG \rceil}$	$\mu_{\lceil 2LG \rceil}$
100,000	3	4	5	9	10	1.58 (0.82)	1.00 (0.00)
200,000	4	5	6	11	12	1.10 (0.33)	1.00 +5E-6(2E-3)
500,000	4	5	6	11	12	1.27 (0.54)	1.00 (0.00)
1,000,000	4	5	6	12	12	1.58 (0.81)	1.00 +1E-6(0.00 +1E-3)
1,200,000	5	5	7	12	14	1.06 (0.25)	1.00 (0.00)
1,500,000	5	6	7	12	14	1.08 (0.28)	1.00 (0.00)
2,000,000	5	6	7	12	14	1.10 (0.33)	1.00 (0.00)

Table 3: Decimal digits of Euler's constant e.

n	MCL	MHL	$\lceil LG \rceil$	MRL	$\lceil 2LG \rceil$	$\mu_{\lceil LG \rceil}$	$\mu_{\lceil 2LG \rceil}$
10,000	3	4	4	7	8	1.57 (0.82)	1.00 (0.00)
20,000	3	4	5	7	10	1.10 (0.32)	1.00 (0.00)
50,000	3	4	5	9	10	1.27 (0.54)	1.00 (0.00)
100,000	4	4	5	10	10	1.58 (0.82)	1.00 +1E-5(3E-3)
200,000	4	5	6	10	12	1.10 (0.33)	1.00 (0.00)
500,000	4	5	6	10	12	1.27 (0.54)	1.00 (0.00)
1,000,000	4	5	6	11	12	1.58 (0.81)	1.00 (0.00)

Table 4: Decimal digits of $\sqrt{2}$.

n	MCL	MHL	$\lceil LG \rceil$	MRL	$\lceil 2LG \rceil$	$\mu_{\lceil LG \rceil}$	$\mu_{\lceil 2LG \rceil}$
10	1	1	2	1	3	1.00 (0.00)	1.00 (0.00)
100	1	2	2	2	4	1.39 (0.59)	1.00 (0.00)
1,000	2	3	3	5	6	1.47 (0.73)	1.00 (0.00)
10,000	3	4	4	8	8	1.57 (0.85)	1.00 +4E-4(0.02)
100,000	4	5	5	11	10	1.75 (1.07)	1.00 +6E-5(0.01)
1,000,000	5	5	6	14	12	1.86 (1.29)	1.00 +6E-6(0.00)
10,000,000	6	6	7	17	14	2.10 (1.65)	1.00 +2E-6(0.00)

Table 5: Decimal digits of Champernowne's constant.

Stringhe random

n	MCL	MHL	$\lceil LG \rceil$	MRL	$\lceil 2LG \rceil$	$\mu_{\lceil LG \rceil}$	$\mu_{\lceil 2LG \rceil}$
100	1	2	2	3	4	1.65 (0.85)	1.00 (0.00)
1,000	2	3	3	5	6	1.54 (0.72)	1.00 (0.00)
10,000	3	4	4	7	8	1.59 (0.81)	1.00 (0.00)
100,000	3	4	5	9	10	1.58 (0.81)	1.00 (0.00)
1,000,000	4	5	6	11	12	1.58 (0.81)	1.00 (0.00)
10,000,000	5	6	7	13	14	1.58 (0.81)	1.00 (0.00)

Table 6: Pseudo-random decimal numbers generated by Java linear congruential generator.

n	$\lvert\Gamma\rvert$	MCL	MHL	$\lceil LG \rceil$	MRL	$\lceil 2LG \rceil$	$\mu_{\lceil LG \rceil}$	$\mu_{\lceil 2LG \rceil}$
3.00	179	1	1	4	44,690,374	8	116,331.09	79,754.59
3.57	127	1	1	4	38,934,707	8	155,688.61	85,245.89
3.60	674	1	2	3	14,783,855	6	22,452.50	8,111.06
3.70	1000	1	8	3	29,349,145	6	11,772.70	2,931.23
3.83	253	1	1	4	51,999,530	8	152,046.77	143,250.67
3.89	1000	1	2	3	62	6	9,475.22	1,620.34
3.90	1000	1	2	3	65	6	9,335.73	1,548.82
4.00	1000	1	2	3	51	6	7,490.64	833.61

Table 7: Strings generated by logistic maps with several values of the parameter r, and seed 0.1. Generated numbers are normalized in the interval $(0, 1)$ and discretized into 1000 digits.

Stringhe random

n	$\lvert\Gamma\rvert$	MCL	MHL	$\lceil LG \rceil$	MRL	$\lceil 2LG \rceil$	$\mu_{\lceil LG \rceil}$	$\mu_{\lceil 2LG \rceil}$
Alphabet size 10								
10	6	1	1	2	1	4	1.00	1.00
50	10	1	1	2	6	4	2.23	1.21
100	10	1	1	2	9	4	3.96	1.62
200	10	1	2	3	10	6	3.54	1.59
500	10	1	2	3	13	6	7.55	1.69
1,000	10	1	2	3	16	6	15.12	2.38
10,000	10	1	2	4	20	8	71.92	4.72
100,000	10	1	2	5	31	10	352.10	11.00
1,000,000	10	1	2	6	34	12	1,745.19	27.20
5,000,000	10	1	2	7	38	14	4,347.82	33.96
10,000,000	10	1	2	7	41	14	8,695.65	67.87
52,000,000	10	1	2	8	60	16	22,579.24	88.20
Alphabet size 1,000								
10	9	1	1	2	1	4	1.00	1.00
50	48	1	1	2	2	4	1.02	1.00
100	92	1	1	2	2	4	1.02	1.00
200	172	1	1	2	4	4	1.07	1.02
500	356	1	1	2	7	4	1.14	1.06
1,000	567	1	1	2	7	4	1.27	1.12
10,000	999	1	1	2	17	4	3.92	2.21
100,000	1000	1	2	2	24	4	33.89	7.37
1,000,000	1000	1	2	2	27	4	335.91	67.82
5,000,000	1000	1	2	3	33	6	720.36	163.26
10,000,000	1000	1	2	3	37	6	1,440.51	325.99
52,000,000	1000	1	2	3	51	6	7,490.64	833.61

Table 8: Strings generated by logistic maps with seed 0.1 and parameter $r = 4$. Generated numbers are normalized in the interval $(0, 1)$ and thus discretized into 10 and 1000 digits.

n	MCL	MHL	$\lceil LG \rceil$	MRL	$\lceil 2LG \rceil$	$\mu_{\lceil LG \rceil}$	$\mu_{\lceil 2LG \rceil}$
100,000	1	2	3	4	6	1.00 (0.05)	1.00 (0.00)
500,000	1	2	3	5	6	1.01 (0.12)	1.00 +2E-6(1E-3)
1,000,000	2	3	3	5	6	1.03 (0.18)	1.00 +1E-6(1E-3)
5,000,000	2	3	3	5	6	1.16 (0.40)	1.00 (0.00)
10,000,000	2	3	3	6	6	1.33 (0.60)	1.00 (3E-4)
50,000,000	2	3	4	6	8	1.01 (0.08)	1.00 (0.00)

Table 9: Raw quantum data (alphabet size 256).

n	MCL	MHL	$\lceil LG \rceil$	MRL	$\lceil 2LG \rceil$	$\mu_{\lceil LG \rceil}$	$\mu_{\lceil 2LG \rceil}$
100,000	2	3	4	6	8	1.03 (0.16)	1.0000 (0.00)
200,000	2	3	4	30	8	1.05 (0.24)	1.0003 (0.02)
500,000	2	3	4	30	8	1.14 (0.38)	1.0001 (0.01)
1,000,000	3	4	4	314	8	1.29 (0.56)	1.0004 (0.02)
1,500,000	3	4	4	314	8	1.45 (0.71)	1.0003 (0.02)
2,000,000	3	4	5	314	10	1.01 (0.12)	1.0002 (0.01)
2,500,000	3	4	5	314	10	1.02 (0.14)	1.0002 (0.01)
3,000,000	3	4	5	314	10	1.02 (0.15)	1.0001 (0.01)

Table 10: Roulette spins (alphabet size 37).

n	MCL	MHL	$\lceil LG \rceil$	MRL	$\lceil 2LG \rceil$	$\mu_{\lceil LG \rceil}$	$\mu_{\lceil 2LG \rceil}$
100	2	2	4	7	7	1.39 (0.70)	1.01 (0.10)
1,000	2	3	5	13	10	2.12 (1.67)	1.02 (0.12)
10,000	3	4	7	16	14	2.35 (2.55)	1.00 (0.04)
100,000	4	5	9	115	17	2.05 (2.22)	1.00 (0.12)
1,000,000	5	6	10	380	20	3.30 (5.47)	1.01 (0.14)
10,000,000	6	7	12	2,720	24	2.97 (5.47)	1.01 (0.19)
13,033,770	6	7	12	2,720	24	3.37 (6.70)	1.01 (0.20)

Table 11: Sorangium cellulosum's genome (alphabet size 4).

n	MCL	MHL	$\lceil LG \rceil$	MRL	$\lceil 2LG \rceil$	$\mu_{\lceil LG \rceil}$	$\mu_{\lceil 2LG \rceil}$
10,000	1	1	3	24	6	3.66 (6.25)	1.13 (0.64)
100,000	1	2	4	41	8	4.25 (9.65)	1.09 (0.53)
200,000	1	2	4	116	8	6.50 (17.27)	1.15 (0.74)
500,000	1	2	5	286	10	3.82 (9.96)	1.12 (0.67)
1,000,000	1	2	5	286	10	5.38 (16.86)	1.18 (1.02)
1,500,000	1	2	5	286	10	6.68 (23.02)	1.22 (1.24)
2,000,000	1	2	5	286	10	7.80 (28.56)	1.26 (1.44)
2,500,000	1	2	5	286	10	8.91 (34.09)	1.16 (0.98)
3,000,000	1	2	5	286	10	9.97 (39.62)	1.17 (1.07)
3,301,740	1	2	5	286	10	10.56 (42.73)	1.18 (1.11)

Table 12: Shakespeare's collection (alphabet size 26).

Probabilitá discreta

Consideriamo il concetto di evento discreto. Un evento Ek rappresenta un elemento di un insieme, al quale è associato un numero non negativo, minore o uguale ad uno, p_k, che rappresenta la probabilità che l'evento k si verifichi:

$$P(E_k) = p_k$$

Probabilitá discreta

Alle probabilità degli eventi si associano le operazioni insiemistiche di unione e intersezione dati due eventi E_i ed E_j vale la relazione

$$P(E_i \cap E_j) = P(E_i) + P(E_j) - P(E_i \cup E_j)$$

Nel caso in cui due eventi E_i ed E_j non possano accadere entrambi simultaneamente si dicono **mutuamente esclusivi** (o eventi disgiunti). Allora si ha:

$$P(E_i \cup E_j) = P(E_i) + P(E_j)$$

$$P(E_i \cap E_j) = 0$$

Se $P(\bar{E}_i)$ indica il **complementare** di un evento $P(E_i)$ allora vale:

$$P(\bar{E}_i) = 1 - P(E_j)$$

Infine E_i ed E_j si dicono eventi **indipendenti** se:

$$P(E_i \cap E_j) = P(E_i)P(E_j)$$

Probabilitá discreta

In taluni casi ad un evento è possibile associare un numero. Ad esempio, nel caso del lancio dei dadi è naturale associare all'evento E_k il numero k che rappresenta il risultato.

Un evento casuale al quale possiamo associare un valore numerico, oltre che probabilità è detto variabile aleatoria. Una variabile aleatoria discreta X è caratterizzata quindi dall'insieme di valori $x_i, i = 1, \ldots, N$ che essa può assumere, e dalle probabilità con la quale essa assume tali valori.

$$p_i = P(\{X = x_i\})$$

La funzione $f(x_i) = p_i$ è detta funzione di probabilità di X.

Dadi ed eventi naturali multifattoriali

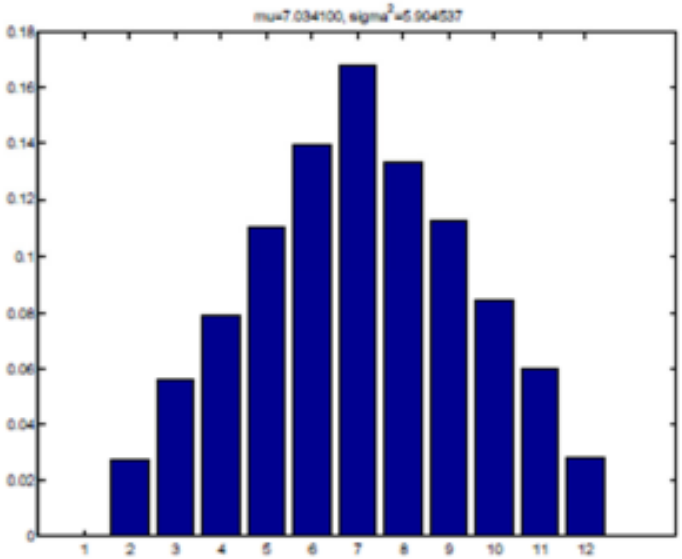
Consideriamo ora il problema del lancio simultaneo di n dadi ed indichiamo con X la variabile aleatoria che indica il punteggio ottenuto. Se $n = 2$ non è difficile calcolare tale probabilità. Infatti X potrà assumere i valori $2, \ldots, 12$ e è sufficiente considerare il rapporto tra i casi favorevoli ed il numero di possibili combinazioni del punteggio dei due dadi

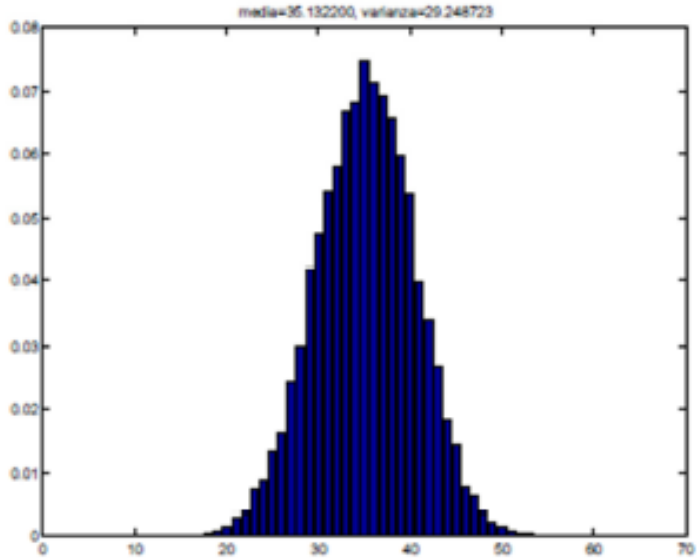
$$\begin{array}{cccc} (1,1) & (1,2) & (1,3) \ldots (1,6) \\ (2,1) & (2,2) & (2,3) \ldots (2,6) \\ \vdots & \vdots & \vdots & \vdots \\ (6,1) & (6,2) & (6,3) \ldots (6,6) \end{array}$$

Se n è molto grande il calcolo diretto risulta più complesso. Da un punto di vista probabilistico il lancio di n dadi è equivalente al lancio di un singolo dado effettuato n volte in modo indipendente.

Dadi ed eventi naturali multifattoriali

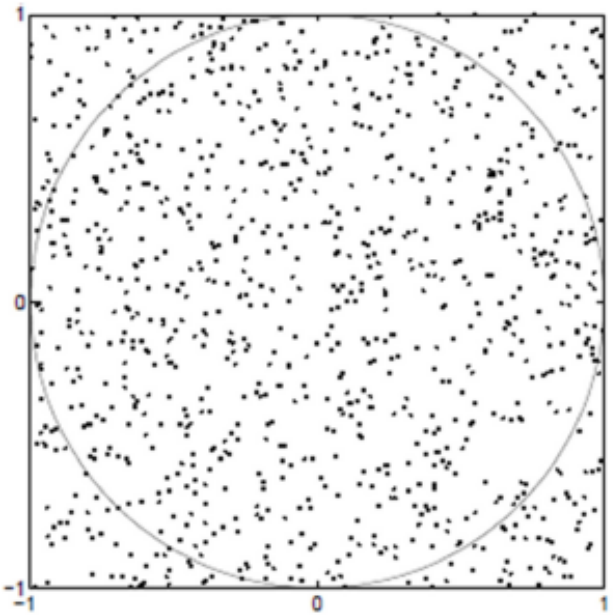
Nel caso di due dadi (figura a sinistra) il punteggio più probabile è 7 (probabilità 6/36=1/6) e quelli meno probabili 2 e 12 (probabilità 1/36). All'aumentare del numero dei dadi (destra) le frequenze tendono ad essere distribuite secondo una campana gaussiana (la distribuzione cosiddetta **normale**).





Monte Carlo per calcoli geometrici: stima di π

Supponiamo di lanciare N freccette ad un bersaglio formato da un quadrato di lato L contenente una circonferenza. Assumiamo che le freccette vengano lanciate casualmente all'interno del quadrato e che quindi colpiscano il quadrato in ogni posizione con uguale probabilità. In Figura riportiamo il bersaglio per $L = 2$ e la posizione dei primi 1000 lanci.

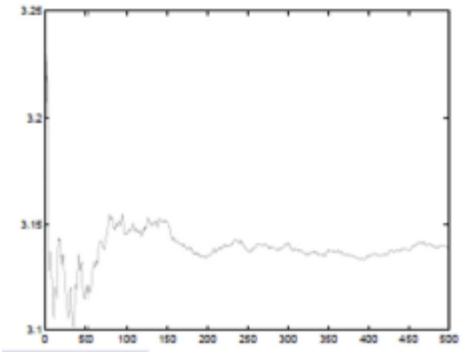


Monte Carlo per calcoli geometrici: stima di π

Dopo molti lanci la frazione di freccette che ha colpito la circonferenza sarà circa uguale al rapporto tra l'area della circonferenza e quella del quadrato. Quindi

$$\frac{\pi L^2}{4L^2} = \frac{1}{4}\pi \approx \frac{N_c}{N}$$

dove N è il numero totale di freccette lanciate e N_c indica il numero di freccette cadute all'interno della circonferenza. Potremo quindi usare il valore $4N_c/N$ come approssimazione di p.



La stima di π migliora all'aumentare di N ma che la convergenza è tutt'altro che veloce e uniforme. La presenza di fluttuazioni dovute all'approccio probabilistico è infatti una caratteristica dei metodi Monte Carlo.