Università degli Studi di Padova
Facoltà di Ingegneria

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE
CORSO DI LAUREA IN INGEGNERIA INFORMATICA
TESI DI LAUREA MAGISTRALE

RICERCA EFFICIENTE DI DIZIONARI DI K-MER DISCRIMINATIVI PER LA CORREZIONE DI LETTURE GENOMICHE

Efficient discovery of discriminative k-mer dictionaries for genotyping reads

---

Laureando:
Paolo Zinato

Relatore:
prof. Matteo Comin

Anno Accademico 2016/2017
Padova, 10 Aprile 2017

Indice

Introduzione

A partire dagli anni settanta del secolo scorso incominciano a svilupparsi i primi filoni di ricerca finalizzati al sequenziamento del DNA. Con tale processo si era in grado di determinare l'esatta sequenza di nucleotidi che compongono il genoma di una specie vivente. In breve tempo, a partire dal metodo Sanger in poi, furono messe appunto numerose metodologie, e il sequenziamento delle sequenze di DNA su grande scala divenne non solo possibile, ma indispensabile per la ricerca biologica di base e per numerosi campi applicati come la diagnostica, le biotecnologie, la biologia forense e la biologia sistematica[1].

Nell'ultimo ventennio l'introduzione di ulteriori progressi in questo campo, come ad esempio il sequenziamento di nuova generazione (NGS), hanno portato a una consistente diminuzione dei costi e ad una produzione di dati genomici senza precedenti[2]. Questa grande quantità di dati messa a disposizione ha accelerato significativamente le attività di ricerca scientifica, come ad esempio nel ramo della medicina, per identificare e diagnosticare malattie ereditarie e lo sviluppo di nuovi trattamenti e di nuove medicine, o nel campo della biologia per la catalogazione di diversi organismi animali e vegetali e di molti microrganismi. Inoltre, la rapidità del processo di sequenziamento è stato fondamentale per il sequenziamento su larga scala del genoma umano realizzato grazie al progetto di ricerca scientifica internazionale "Progetto Genoma Umano" (HGP).

Tale progetto aveva come obiettivo principale quello di determinare la sequenza delle coppie di basi azotate che formano il DNA umano e di identificare e mapparne i geni da un punto di vista sia fisico che funzionale. Il progetto è stato completato il 22 giugno 2003 quando, per la totalità del genoma umano, è stata identificata l'esatta sequenza dei tre miliardi di coppie di basi azotate dal quale è formato, e quindi la determinazione della posizione occupata da ciascun gene rispetto agli altri.

Questa grande mole di dati genomici di cui ora si è a disposizione ha fatto emergere alcune problematiche dovute alla sua gestione, immagazzinazione, elaborazione e trasmissione. Per superare queste difficoltà, in letteratura sono state proposte numerose tecniche di compressione che permettessero una memorizzazione efficiente dei dati e ne permettesse una veloce trasmissione, sia con perdita di informazione (lossy) che senza perdita (lossless).

Pertanto, in questo lavoro di tesi, si è ritenuto importante analizzare una particolare metodologia di compressione lossy, che fa uso di un dizionario di k-mer per ridurre l'aleatorietà dei file genomici, aumentando l'efficacia della compressione. In particolare è stato usato il software *Quartz*, ponendo grande attenzione sulla

creazione del dizionario di k-mer e proponendo un processo efficiente per la sua costruzione.

Nel primo capitolo vengono descritti in breve i passi del sequenziamento genomico che portano alla produzione delle letture genomiche. Vengono introdotte le principali problematiche legate alla loro grande mole e alcune soluzioni di compressione che sono state proposte in letteratura. In particolare, viene descritto il software *Quartz* ponendo l'attenzione su come opera un filtraggio dei dati mediante l'uso di un dizionario di k-mer. Viene descritta, infine, la pipeline di valutazione utilizzata per misurare la qualità delle letture genomiche dopo aver subito il filtraggio.

Nel secondo capitolo viene presentato il dizionario ufficiale utilizzato da *Quartz* per operare il filtraggio dei dati genomici e in seguito viene descritto il procedimento per la creazione efficiente di un dizionario di k-mer personalizzato. Tale procedimento è stato utilizzato per la creazione di quattro differenti dizionari descritti al termine del capitolo.

Nel terzo e ultimo capitolo viene descritto il set sperimentale utilizzato per i test di filtraggio delle letture genomiche, utilizzando *Quartz* con i dizionari realizzati. Viene descritta, inoltre, l'intera pipeline di valutazione usata per analizzare la qualità delle letture dopo aver subito il processo di filtraggio, ponendo attenzione sull'efficacia della compressione e sulla capacità di conservare il contenuto informativo. I dati raccolti sono stati, infine, analizzati e interpretati.

Capitolo 1

Compressione delle letture genomiche

1.1 Sequenziamento genomico

Con il processo di sequenziamento genomico si determina, a partire da un campione biologico, la sequenza esatta dei nucleotidi che compongono il genoma delle specie viventi. Per molti anni il sequenziamento genomico fu basato sul metodo Sanger di elettroforesi capillare. Tale metodo aveva lo scopo di sintetizzare un filamento complementare a un frammento di DNA e di registrare i segnali dell'incorporazione dei nucleotidi, consentendo così di ricostituire la sequenza di un singolo frammento di DNA.

Recentemente è stato introdotto un nuovo metodo, il metodo NGS (Next Generation Sequencing) che, pur basandosi su un principio analogo, ha il vantaggio di avere elevate prestazioni che permettono di ridurre significativamete i tempi ed i costi del sequenziamento. Nonostante la minore lunghezza dei frammenti analizzati, dà la possibilità di effettuare un gran numero di letture in parallelo, anzichè una soltanto, permettendo in questo modo di ottenere in poco tempo una grande mole di dati. Per questo motivo la NGS è nota anche come high-throughput sequencing. Diventa, perciò, necessario il ricorso ad avanzati sistemi bioinformatici per la gestione dell'informazione ottenuta dal sequenziamento, che spesso può arrivare fino a centinaia di terabyte.

Esistono diversi sistemi NGS, svilupati da varie aziende del settore, ma tuttavia è possibile identificare tre passi fondamentali comuni a ciascuno: la preparazione e l'immobilizzazione del campione DNA, ovvero la preparazione della libreria di sequenziamento (sequencing library), la reazione di amplificazione/clonazione e la reazione di sequenziamento[3].

1. *Preparazione della libreria di sequenziamento.* Nella prima fase, il campione di DNA viene sottoposto a un processo di frammentazione casuale, ad esempio, attraverso l'uso di ultrasuoni. I frammenti prodotti vengono fissati in un supporto adatto al processo di sequenziamento tramite l'aggiunta di alcune sequenze predefinite, note come adattatori. L'insieme dei frammenti di DNA e degli adattatori costituiscono la libreria di sequenziamento. Molto spesso

viene estratta solo una piccola porzione di genoma, quella che interessa allo scopo prefissato, mentre il restante materiale viene scartato.

2. *Processo di amplificazione.* Nella seconda fase, viene operato un processo che consente la moltiplicazione dei frammenti di DNA. La reazione di amplificazione attraverso la PCR (reazione a catena della polimerasi) consente di ottenere in vitro, molto rapidamente, una crescita esponenziale del materiale genetico, dovuto al fatto che ogni frammento viene clonato un grande numero di volte.

3. *Reazione di sequenziamento.* Nella terza e ultima fase avviene il sequenziamento vero e proprio, dove mediante complessi meccanismi fluidici che regolano il flusso dei reagenti che vanno a legarsi con i camioni di DNA immobilizzato. Ogni ciclo di sequenziamento consiste nel far reagire il DNA immobilizzato con una soluzione contenente un singolo nucleotide: se tale nucleotide è complementare alla sequenza, viene incorporato. Tale evento è caraterizzato dall'emissione di un lampo di luce che viene rilevato con un opportuno sistema di rilevazione per immagini. Tali informazioni vengono registrate e costituiscono la sequenza cercata di nucleotidi di ogni singolo frammento.

Quello che si ottiene da questo procedimento sono un insieme di "letture" di ogni frammento della sequenza, denominate solitamente *reads*. In un sequenziamento NGS di buon livello, ogni frammento che costituisce la sequencing library viene riletto più volte: a scopo di ricerca si producono almeno 30 reads per ogni frammento, mentre per la diagnostica almeno 50 ma altri casi si pò arrivare anche a 500 o 1000. Questi valori definiscono la *coverage* della lettura genomica. Un valore elevato di coverage, e quindi un numero elevato di reads, è necessario sostanzialmente perchè il sequenziamento NGS è talvolta impreciso e si ha bisogno di un elevato numero di letture per mitigare i segnali d'errore.

Per permettere la ricostruzione della sequenza originaria, le letture ottenute dalle piattaforme NGS sono progettate per avere le estremità sovrapposte tra loro. In questo modo, le reads possono essere riassemblate sovrapponendo le loro estremità e operando un allineamento con una sequenza di riferimento del genoma umano.

In figura 1.1[1] sono illustrate le tre fasi del processo di sequenziamento NGS appena descritto.

1.2 Letture genomiche

I dati genomici derivanti dal processo di sequenziamento sono di norma contenuti in file testuali di estensione `.fastq` o `.fq` e al loro interno sono costituiti dalle lettere rappresentanti i nucleotidi dei rispettivi frammenti genomici con associati i *quality*

[1] https://www.abmgood.com/Enzymes/images/NGS-Process.jpg

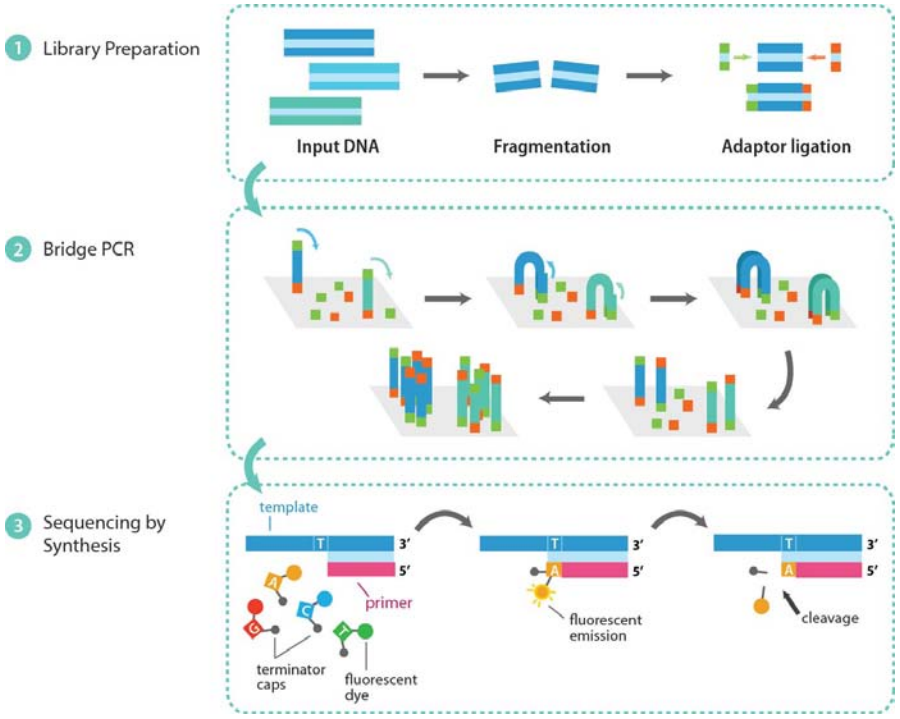


Figura 1.1: Fasi del sequenziamento NGS.

scores. Sia per i nucleotidi che per i quality scores viene adottata una codifica a singolo carattere. Tale formato è divenuto lo standard per la memorizzazione dei dati genomici derivanti dal sequenziamento di tipo high-throughput. In caso di file genomici di riferimento, i *quality scores* vengono omessi e i file sono costituiti solamente dai caratteri rappresentanti i nucleotidi in file di estensione `.fasta` o `.fa`. Sotto è riportato un frammento di file `.fastq`:

```
@SRR068164.2139 206F7ABXX100822:7:100:1354:10644/1
CCACTAGAACTTAAGTCGCTGAAATTTCTGAGACTGTATCTATCTGTTTCAATGCTTTGAGCCAGAAGAAGGTTGCATGCCT
+
GGEGGDG:DBGGBDGDGGGGGBEGBGDGGGGGDDGGGGDGD<EACGGGGB<CCEAEEBG<B@>B@##################
@SRR068164.2140 206F7ABXX100822:7:100:1354:11248/1
TGACTGCACCCCTCCGTTGACAGGTTTTGTCTTTTTGACTTCTCCTGATACTCACTTTTCTAATGTAAGATAACCTTAAAAA
+
HHHHBGG>EGGGGGGAHBHHG?GGBGGGDDGGGGGDA:GGGB@@8AAAA?*BA?##############################
```

In questi file sono contenute più letture e ciascuna di esse segue una struttura ben definita costituita dai seguenti elementi:

1. Una riga descrittiva che inizia con il carattere speciale "@" con a seguire l'identificatore della sequenza e un descrittore opzionale;

2. Una riga contenente la sequenza di nucleotidi rilevata dal processo di sequenziamento;

3. Il carattere di separazione "+" a volte seguito dallo stesso identificatore della sequenza;

4. Una riga contenente i *quality scores* della sequenza di nucleotidi contenuta nella seconda riga.

I *quality scores* seguono la scala Phred, appositamente creata per misurare la qualità delle letture automatizzate delle basi azotate durante il "Progetto Genoma Umano HGP" e ormai largamente utilizzata. La scala va da un minimo di 0 a un massimo di 93 e viene codificata usando un singolo carattere ASCII per ogni valore, dal simbolo 33 al simbolo 126. Il simbolo "!" rappresenta il più basso valore di qualità mentre il carattere "~" rappresenta quello più alto. Tali valori, associati a ciacuna lettera, vengono calcolati a partire dalla stima d'errore P assegnata a ogni singolo nucleotide secondo la funzione $Q = -10 \log_{10} P$. In questo modo si può assegnare alle basi una vasta gamma di probabilità di errore, da 1.0 (una lettura certamente sbagliata) fino a $10^{-9,3}$(una lettura estremamente accurata). Tale scala di valori, perciò, può essere impiegata sia per i dati grezzi derivati dal processo di sequenziamento sia per i dati derivanti da operazioni di post-processing, come le sequenze di riferimento, in cui sono presenti valori di accuratezza molto elevata. Molto spesso però, tali valori sono affetti da rumore e da incertezza e vengono utilizzati da molte applicazioni in maniera euristica.

Q score	ASCII	Char	Q score	ASCII	Char	Q score	ASCII	Char
0	33	!	32	65	A	64	97	a
1	34	"	33	66	B	65	98	b
2	35	#	34	67	C	66	99	c
3	36	$	35	68	D	67	100	d
4	37	%	36	69	E	68	101	e
5	38	&	37	70	F	69	102	f
6	39	'	38	71	G	70	103	g
7	40	(	39	72	H	71	104	h
8	41	)	40	73	I	72	105	i
9	42	*	41	74	J	73	106	j
10	43	+	42	75	K	74	107	k
11	44	,	43	76	L	75	108	l
12	45	-	44	77	M	76	109	m
13	46	.	45	78	N	77	110	n
14	47	/	46	79	O	78	111	o
15	48	0	47	80	P	79	112	p
16	49	1	48	81	Q	80	113	q
17	50	2	49	82	R	81	114	r
18	51	3	50	83	S	82	115	s
19	52	4	51	84	T	83	116	t
20	53	5	52	85	U	84	117	u
21	54	6	53	86	V	85	118	v
22	55	7	54	87	W	86	119	w
23	56	8	55	88	X	87	120	x
24	57	9	56	89	Y	88	121	y
25	58	:	57	90	Z	89	122	z
26	59	;	58	91	[	90	123	{
27	60	<	59	92	\	91	124	|
28	61	=	60	93	]	92	125	}
29	62	>	61	94	^	93	126	~
30	63	?	62	95				
31	64	@	63	96	`			

Figura 1.2: Corrispondenza tra scala Phred e caratteri ASCII

1.3 Compressione delle sequenze

Comprimendo i file delle sequenze genomiche mediante compressori *lossless*, in assenza di perdita di informazione, i *quality scores* costituiscono circa il 70% della dimensione totale del file compresso[5]. Un valore così elevato è dovuto al fatto che all'interno dei file, i *quality score* costituiscono la parte più aleatoria dato che possono variare in un intervallo molto largo di valori (anche se spesso i file derivanti dal sequenziamento hanno valori compresi tra 0 e 60). Per questi motivi, in letteratura sono state proposte molteplici soluzioni *lossy,* ovvero compressioni con perdita di informazione sui valori di qualità. In generale, la strategia più addottata dalle varie soluzioni presentate, è quella di eseguire una quantizzazione dei *quality scores.* In questo modo è possibile ridurre la variabilità dei valori e perciò l'entropia intrinseca dei file, aumentando l'efficacia dei compressori che riescono a ridurre ulteriormente le dimensioni dei file e i requisiti di archiviazione. Il costo di tale procedura è, però, quello di introdurre una inevitabile distorsione nei dati dovuta al fatto che i *quality scores,* a seguito della compressione, possono differire da quelli originali.

Le sequenze genomiche sono spesso utilizzate per diversi studi biologici e diagnostici ed è, quindi, di fondamentale importanza analizzare gli effetti distorsivi che i compressori lossy producono su tali dati. Poiché esistono molteplici utilizzi ed applicazioni di tali dati per scopi diversi, risulta difficile considerare in maniera completa tutti i possibili effetti. Spesso si riduce il campo di studio sugli effetti che tali compressioni hanno sulle procedure più comunemente utilizzate nella pratica come la procedura di *SNP calling*, ovvero quella di scoperta delle varianti genetiche.

Alcune soluzioni di compressione, proproste recentemente in letteratura, rappresentano lo state dell'arte nel campo della compressione genomica e sono riportate nella prossima sezione.

1.3.1 Approci a perdita di informazione

Uno degli approcci più diffusi è il metodo Illumina binning[6]. Tale metodo non richiede materiale supplementare o operazioni di pre-processing sui dati e si basa sulla riduzione della risoluzione dei quality score, ovvero su una loro quantizzazione. Per fare ciò, viene creata una tabella con cui vengono raggruppati i quality score in *bins*, ovvero in intervalli rappresentati da un unico valore. Si crea, perciò, una mappatura in cui ad ogni valore di qualità viene assegnato un suo rappresentante in base all'intervallo di appartenenza.

Durante la scansione dei file genomici viene applicata tale mappatura con la quale, si opera la sostituzione dei valori. Nella tabella 1.3, ad esempio, i valori dell'intervallo 20-24 sono stati raggruppati in uno stesso *bin* rappresentato dal valore 22, perciò, durante la scansione del file, tutti i valori compresi tra 20-24 saranno sostituiti dal valore dei proprio *bin* di appartenenza: 22. Così facendo si diminuisce la variabilità dei valori di qualità, che ora sono ridotti al numero

Quality Score Bins	Example of Empirically Mapped Quality Scores
N (no call)	N (no call)
2–9	6
10–19	15
20–24	22
25–29	27
30–34	33
35–39	37
≥ 40	40

Figura 1.3: Esempio di raggruppamento in *Illumina bins* dei quality scores.

prescelto di *bins*, diminuendo così l'entropia dei file e aumentando l'efficienza di compressione.

La scelta dei *bins* è spesso di natura empirica, legata alla natura dei dati e agli strumenti di sequenziamento utilizzati. Tali valori devono tenere conto dell'aumento dell'efficacia di compressione a scapito della perdita di informazione e sono soggetti a continue ricalibrazioni.

Un altro approccio è il metodo P-block[7] che opera una quantizzazione dei punteggi di qualità separandoli in blocchi di dimensione variabile, dove tutti i valori contenuti in ciscuno di essi rispettano un determinato parametro p scelto in accordo a un criterio di misura scelto. Questa metodologia si basa sulla proprietà di località dei *quality scores* (Kozanitis et al., 2010), ovvero sull'osservazione che, di fatto, i punteggi di qualità in posizioni limitrofe hanno un valore simile.

Utilizzando P-block, la sequenza di *quality scores* viene devisa in blocchi e per ciasuno di essi viene memorizzata la sua lunghezza ed il proprio valore rappresentativo, che dipende dal criterio di misura scelto (vedi tabella 1.5). Ad esempio, se viene addottata come metrica di misura la distanza di Manhattan, i blocchi devono essere formati in modo tale che nessuno dei valori di ogni blocco differiscano al più di p dal proprio valore rappresentativo corrispondente. In definitiva, ogni blocco copre una gamma di *2p +1* differenti punteggi di qualità, dove il valore rappresentativo p è il punto medio del range di ciascun blocco i:

$$rep = \frac{(max_i qvals[i]) + (min_i qvals[i])}{2}$$
$$max_i \, |qvals[i] - rep| \leq p$$

Considerando l'utilizzo di varie metriche e la scelta di diversi valori di p, si possono creare molteplici sistemi di divisione in blocchi con differenti gradi di approssimazione dei valori originali. In figura 1.4 è riportato un esempio dell'uso di P-block, dove è stata addottata come metrica la distanza di Manhattan e *p=1*.

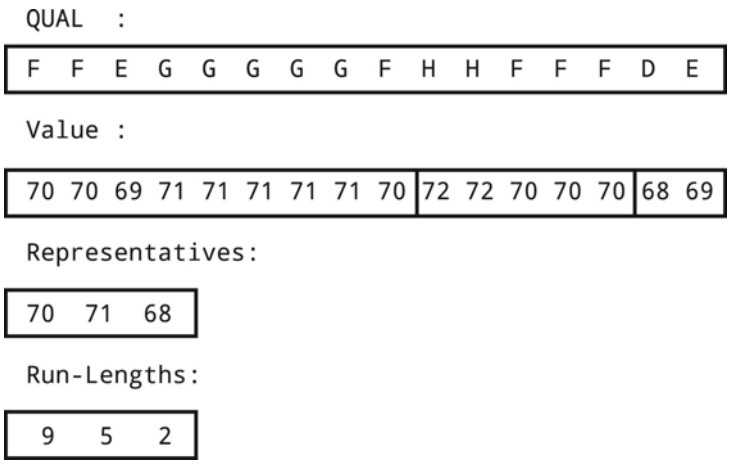


Figura 1.4: Esempio dell'uso di P-block adottando come metrica la distanza di Manhattan.

Measure	Function
Mean Manhattan Distance(X, Y)	$\frac{1}{\ell}\sum_{i=1}^{\ell} \lvert X_i - Y_i \rvert$
Max : Min Distance(X, Y)	$\max_{1 \le i \le \ell} \frac{\max(X_i, Y_i)}{\min(X_i, Y_i)}$
Mean Squared Error(X, Y)	$\frac{1}{\ell}\sum_{i=1}^{\ell} (X_i - Y_i)^2$
Chebyshev Distance(X, Y)	$\max_{1 \le i \le \ell} \lvert X_i - Y_i \rvert$
Soergel Distance(X, Y)	$\frac{\sum_{i=1}^{\ell} \lvert X_i - Y_i \rvert}{\sum_{i=1}^{\ell} \max(X_i, Y_i)}$
Lorentzian Distance(X, Y)	$\sum_{i=1}^{\ell} \log_2(1 + \lvert X_i - Y_i \rvert)$

Figura 1.5: Principali metriche per il calcolo della distanza tra due elementi

Anche in questo caso la scelta della metrica e il valore di p sono di natura empirica e dipendono dalla natura intrinseca dei dati.

Assieme a P-block, i medesimi autori, hanno presentato il metodo R-block. Tale metodo ha un funzionamento similare al precedente ma differisce nel conservare una buona precisione di compressione dei punteggi di qualità bassi. Molto spesso vengono ritenuti maggiormente importanti i valori di qualità bassi ovvero quei punteggi propri delle letture affette da incertezza e da errori. Per cercare di non penalizzare tali valori con le approssimazioni introdotte dal processo di quantizzazione, sono stati adottati alcuni accorgimenti. I valori di qualità subiscono dapprima una normalizzazione: viene sottratto dei valori di qualità il punteggio minimo ovvero 33, e riportati in un intervallo 0-93. In manier analoga al metodo P-block, i punteggi di qualità vengono suddivisi in blocchi di lunghezza variabile ma anzichè considerare un valore p prefissato nella loro costruzione, si considera

una metrica *max-min distance* e un valore r in accordo con le seguenti formule:

$$(max_i qvals[i])/rep < r$$
$$rep/(min_i qvals[i]) < r$$

Il rapporto fra il valore massimo di ciascun blocco i e il valore rappresentativo, e, il reciproco del rapporto fra il corrispettivo valore minimo e lo stesso valore rappresentativo, devono essere inferiori a una soglia massima r definita. In questo modo si garantisce una minore approssimazione dei valori più bassi della scala dei *quality scores* che vengono così meno penalizzati dal processo di quantizzazione. In figura 1.6 è riportato un esempio di come opera il metodo R-block con valore di r=1,055[8].

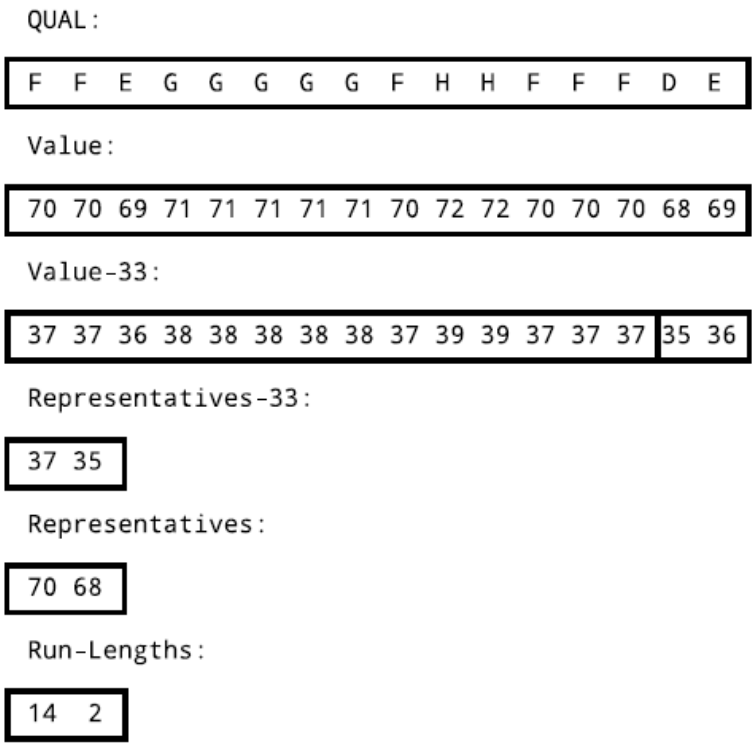


Figura 1.6: Esempio dell'uso di R-block con r=1,055.

I criteri descritti sono, al momento, tra i più diffusi e utilizzati e sono ben presenti in letteratura e spesso adottati come riferimento negli studi. Rapresentano, in definitiva, lo stato dell'arte attuale in termini di compressione lossy di file genomici. Altre soluzioni sono state presentate recentemente, come ad esempio QVZ[9], GeneCodeq[10] e Quartz[11].

QVZ si basa sulla codifica dei valori di qualità mediante l'algoritmo Lloyd-Max e sull'uso delle catene di Markov per il calcolo delle probabilità di transizione fra punteggi limitrofi. In particolare la correlazione tra i *quality scores*, già oggetto di studio negli approci precedenti, viene vista come il risultato di un processo i.i.d. modellato come una catena di Markov di primo ordine, in cui la probabilità che

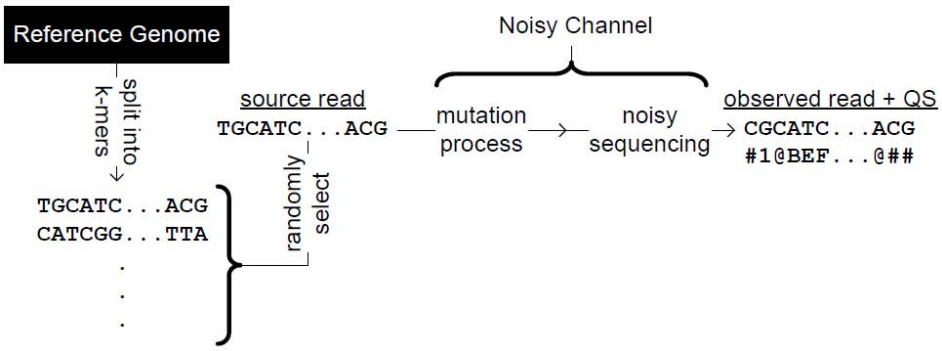


Figura 1.7: GenCodeq modella le differenze tra il rifermento genomico e le sequenze come il prodotto di una trasmissione attraverso un canale rumoroso.

ogni *quality score* abbia un determinato valore dipende dallo *score* immediatamente precedente.

Sia GeneCodeq che Quartz, basano il loro approccio sull'utilizzo di un dizionario di N k-mer, che viene utilizzato per descriminare i k-mer delle sequenze da comprimere e per modificarne i punteggi di qualità. GeneCodeq costruisce il dizionario partendo dai dati genomici stessi e da un genoma di riferimento, basandosi sulla teoria dei codici e sull'inferenza Bayesiana. In particolare, GeneCodeq, modella le differenze genomiche fra il riferimento e le sequenze da comprimere, come il prodotto di una trasmissione di tale genoma attraverso un canale rumoroso (figura 1.7). Con questa metodologia viene creato un insieme di k-mer che risultano avere una adequata distanza di Hamming rispetto al riferimento. Con tali k-mer viene calcolato un nuovo score per ogni singola base nucleica della sequenza che, se risulta migliore, sostituisce l'originale e viene quantizzato secondo la metodologia Illumina 8-bin.

Quartz è stato il software adottato in questo studio ed è stato illustrato accuratamente nella sezione succesiva.

1.3.2 Quartz

Quartz[11] è un software altamente efficiente e scalabile che opera la compressione dei file genomici, non operando una quantizzazione ma effettuando uno smooth di una larga parte dei valori di qualità. L'ipotesi alla base di questo approccio è che, rispetto a un riferimento genomico, ogni posizione divergente all'interno di un singolo k-mer possa essere un polimorfismo di un singolo nucleotide (SNP) o un errore di sequenziamento. Quartz conserva inalterati tali valori ma modifica i punteggi di qualità delle basi nelle posizioni in accordo con il riferimento portandoli a un certo valore di default. Questa operazione, come nella quantizzazione, comporta una forte diminuzione della variabilità dei punteggi di qualità aumentando la comprimibilità dei file genomici.

Il riferimento genomico utilizzato da Quartz nel stabilire le divergenze è un dizionario di k-mer contenente i k-mer più frequenti estratti da un dataset di *reads* NGS reale. Il dizionario viene creato in modo tale che i k-mer al suo interno

siano presenti e "coprano" gran parte della lunghezza delle sequenze genomiche reali ammettendo una piccola differenza in termini di mismatch, fungendo quindi da *genoma consenso*. Tramite questo dizionario, Quartz scandisce le sequenze da comprimere e ogni qualvolta trova un valore di qualità corrispondente a una posizione in accordo con almeno un k-mer presente nel dizonario lo impostata a un valore predefinito altrimenti, lo mantiene invariato.

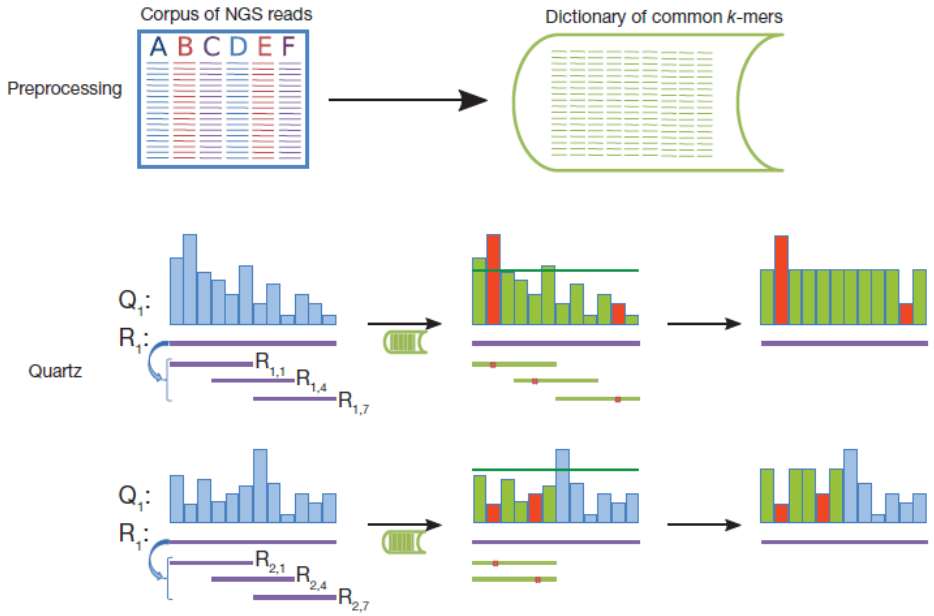


Figura 1.8: Schema riassuntivo del funzionamento di Quartz: creazione del dizionario e smooting dei *quality scores*.

Gli autori di Quartz, hanno individuato in 32 la lunghezza ottimale dei k-mer così da garantire che:

- Il numero totale di k-mer sia molto maggiore del numero dei k-mer unici per evitare eventuali collisioni fra k-mer non correlati e quindi conteggiati con una molteplicità non veritiera.
- La lunghezza dei k-mer sia sufficientemente corta da garantire che la probabilità che un k-mer contenga al suo interno più di un errore di sequenziamento sia bassa.
- La lunghezza sia multipla di quattro così che ogni frammento lungo 4 basi possa essere rappresentato mediante un byte: un 32-mer riesce così a essere contenuto in un intero a 64bit.

Durante la scansione delle sequenze, per ogni k-mer in esame, Quartz ricerca efficientemente nel dizionario tutti i *3k+1* 32-mer che differiscono di una distanza di Hamming pari a 1, ovvero tutti quelli che differiscono al massimo di una base, utilizzato un approccio efficiente basato sul metodo LSH "locality sensitive hashing" applicato al problema della ricerca del *nearest neighbor*. Rispetto ad altri tipi di

compressory lossy presentati, Quartz, riesce a ottenere un'ottima accuratezza di genotipo e un alto fattore di compressione eliminando più del 95% dell'informazione legata ai punteggi di qualità e in tempi molto rapidi. Durante il processo di smooting dei valori, viene eliminato gran parte del rumore di sequenziamento a vantaggio dell'accurezza. In figura 1.9 è riportato un esempio del funzionamento di Quartz.

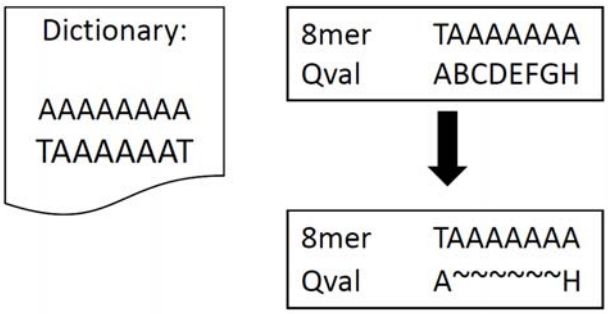


Figura 1.9: Esempio del funzionamento di quartz con k=8

La fase di costruzione del dizionario rappresenta la parte più onerosa dal punto di vista computazionale, ma una volta completata e ottenuto il dizionario, Quartz risulta un software molto scalabile, in grado di processare in tempi molto rapidi grandi moli di sequenze e interi genomi dell'ordine di grandezza di TB .

In questo progetto è stata utilizzata una versione modificata di quartz in cui si è reso possibile la lettura dei file `.fastq` rappresentanti le sequenze e i file dei dizionari, direttamente in formato compresso `.gz` o `.bzip2`. Questo non era previsto dalla versione originaria e ha permesso di risparmiare molto spazio di archiviazione durante la fase di test, a scapito però della velocità di esecuzione causato del tempo di decompressione dei file. Inoltre, in tale versione sono state eliminate alcune parti di codice ridondanti che avevano unicamente scopo di debug, rendendo lo stesso codice più pulito e il software più performante. Tale variante è stata denominata *myquartz*.

1.4 Processo di valutazione sulla chiamata degli SNP

Il modo più comunemente utilizzato per valutare l'accuratezza dei compressori a perdita di informazione e l'impatto che tali procedure hanno sui dati genomici, è basarsi sulla rilevazione delle variante genetiche. Tale processo prende il nome di "variant calling" è viene applicato sulle sequenze che hanno subito il processo di compressione con l'ausilio di un riferimento (spesso chiamato "gold standard" o "ground truth") nel quale sonospecificate le posizioni reali delle varianti, ovvero quelle ritenute attendibili. Le varianti genetiche sono le differenze tra le sequenze ottenute dal processo di sequenziazione e i riferimenti genomici e possono essere

suddivise in tre categorie: varianti a singolo nucleotide (SNV o SNP), inserzioni o delezioni (INDEL) e varianti strutturali.

Il processo che porta alla rilevazione delle varianti contenute nelle sequenze e la creazione di una lista in cui vengono specificate è composto da numerosi passaggi di elaborazione informatica a cui prendono parte molteplici software utilizzati in cascata all'interno di una "pipeline". I passaggi più importanti di questo processo sono sostanzialmente tre: l'allineamento delle read sul genoma di riferimento, la chiamata delle varianti e la valutazione finale.

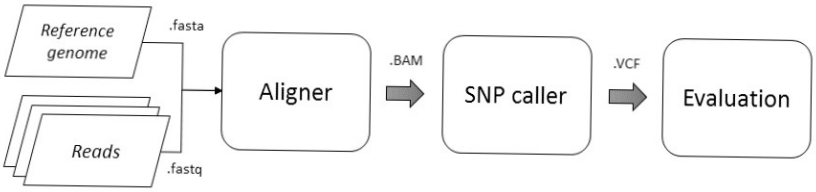


Figura 1.10: Pipeline per il processo di SNP calling

L'allineamento ha il compito di trovare la posizione corretta di ogni read lungo la sequenza di riferimento del genoma. Per fare ciò, particolari software chiamati *aligner* sfruttano le informazioni contenute nel genoma di riferimento e, tramite algoritmi di allineamento, identificano la posizione di ciscun frammento di DNA rappresentato dalle reads. Questa operazione risulta molto onerosa dal punto di vista computazionale dato l'elevato numero di frammenti prodotti dal sequenziameneto e la grande dimensione del genoma umano. Sotto è riportato un piccolo esempio[2] in cui vengono allineate sei sequenze a una sequenza di riferimento.

```
ref       aggttttataaaac----aattaagtctacagagcaacta
sample    aggttttataaaacAAATaattaagtctacagagcaacta
read1     aggttttataaaac****aaAtaa
read2      ggttttataaaac****aaAtaaTt
read3          ttataaaacAAATaattaagtctaca
read4              CaaaT****aattaagtctacagagcaac
read5                aaT****aattaagtctacagagcaact
read6                  T****aattaagtctacagagcaacta
```

Per operare in maniera efficiente tale operazione, sono state proposte in informatica diverse metodologie: le più comunui sono quelle utilizzate dai software BWA e BOWTIE2, come ad esempio l'utilizzo della trasformata *Burrows–Wheeler* e l'ausilio di strutture dati avanzate come i *Suffix Array*. Le principali difficoltà che rendono arduo il compito degli *aligner* sono, in primo luogo, due: la presenza di sequenze ripetute all'interno del genoma, che rende difficile trovare riscontri univoci per ogni sequenza, e il doppio verso di lettura di ogni read che costringe il software a duplicare la ricerca lungo tutto il genoma. In aggiunta, la presenza nelle sequenze di errori di sequenziamento, di mutazioni genetiche e di *indels* obbligano il software a valutare allineamenti "non perfetti" dove al loro interno vengono ammessi un certo numero di mismatch. Le scelte, le strategie e le soluzioni addottate

[2]https://bioinf.comav.upv.es/courses/sequence_analysis/snp_calling.html

per fare fronte a questi problemi rappresentano i punti centrali di differenza tra i vari software esistenti.

Al termine di questa fase viene creato un unico file di estensione *.sam,* o l'equivalente binario *.bam*, in cui è contenuto tutto l'allineamento calcolato. Il formato `SAM` (Sequence Alignment/Map format) è il formato testuale standard per gli output dei software di allineamento ed è sostanzialmente un file delimitato da tabulazioni contenente i dati di allineamento organizzati per linea di testo, in cui sono definiti 11 campi obbligatori, ed eventualmente da un'intestazione con alcuni metadati. Il formato `BAM` è un formato di codifica per i file `SAM`, compresso nel formato `BGZF`, che è un formato di compressione a blocchi implementato sullo standard *gzi*p, e ha lo scopo quello di fornire una buona compressione insieme alla possibilità di accedere al file `BAM` in modo non sequenziale per eseguire interrogazioni indicizzate.

Al termine della procedura di allineamento, un software per la chiamata delle varianti (SNP caller) sfrutta le informazioni ottenute dalla fase precendente per determinare i nucleotidi presenti nel genoma sequenziato e valutare, per ognuno, se si tratta di una variante o meno. Tali software utilizzano modelli probabilistici per stimare il genotipo più corretto per ciscuna posizione basandosi, sostanzialmente, su alcuni parametri come ad esmpio: la qualità dell'allineamento della read, il valore di coverage per ogni base e il rapporto dei vari nucleotidi rispetto al totale delle basi allineate per ogni posizione. Nel caso in cui si rileva la presenza di alcuni nucleotidi che differiscono da quelli presenti nel genoma di riferimento, il software ha il compito di valutare la presenza o meno di una variante SNP stimandone la qualità e il livello di confidenza. Questo compito non è semplice dato che, molto spesso, le informazioni non sono a sufficienza a garantire un buon livello di confidenza. Ad esempio, il numero di read allineate in una determinata posizione potrebbe essere troppo poco per poter affermare che un nucleotide che differisce dal riferimento sia effettivamente presente nella sequenza reale, oppure, avere una qualità troppo bassa. Anche in questo caso, le varie strategie utilizzate, i vari parametri e le relative soglie sono i punti dove si diffrenziano i vari tool esistenti. Fra i *variant caller* più utilizzati in letteratura si possono trovare GATK, SamTools, Picard.

Nel momento in cui una variante genetica viene identificata, il software la memorizza in un file di estensione *.vcf* (Variant Call Format), un formato file standard per la definizione delle varianti genetiche (figura 1.11[3]). Tale formato è stato proposto all'interno del progetto *1000 Genomi* e poi adottato come standard. Anch'esso è delimitato da tabulazioni e contiene un'intestazione con un numero arbitrario di meta-dati, organizzati su più linee, e una linea di definizione per la struttura della sezione di dati. Le meta-informazioni nell'intestazione sono usate per descrivere alcune proprietà del file come la data di creazione, la versione della sequenza di riferimento, i software usati e tutte le informazioni rilevanti sulla storia del file. Nel corpo del file sono specificate le varianti una per riga e definite da otto campi obbligatori, come ad esempio: il campo CHROM specifica il cromosoma a

[3]https://bioinf.comav.upv.es/courses/sequence_analysis/snp_calling.html

cui appartiene la variante, il campo POS la sua posizione all'interno del genoma, il campo REF indica il nucleotide corretto specificato nella sequenza di riferimento e il campo ALT mostra le alternative/mutazioni trovate per tale nucleotide.

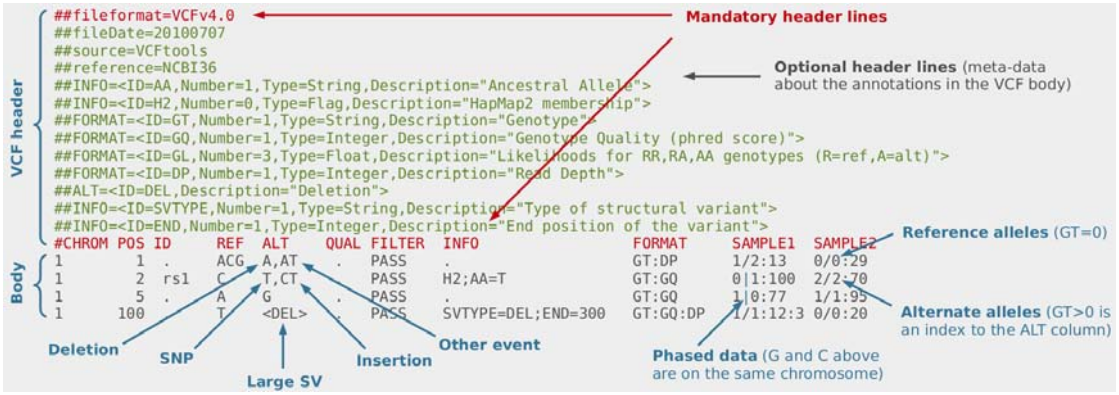


Figura 1.11: Esempio di file `VCF`, comprensivo di intestazione e parte centrale.

Infine, terminati i processi di allinemaneto e di chiamata delle varianti, il file `VCF` contenente le mutazioni viene valutato e confrontato con un file `VCF` di riferimento, denominato *gold standard,* nel quale sono contenute le varianti reali riconosciute attendibili dalla comunità scientifica. Dal confronto emergono alcuni valori tipici dei processi di evaluation come il numero dei *TP* (true positive), *FP* (false positive), *TN* (true negative) e *FN* (false negative), con i quali vengono calcolate le principali metriche quali la *Precision*, la *Recall* e la *F-measure.* Il confronto della bontà degli effetti che i vari compressori hanno sulle sequenze genomiche originarie si basa appunto su tali metriche, i cui valori sono il prodotto finale dell'intera pipeline di valutazione.

Capitolo 2

Dizionari di k-mer

In questo capitolo vengono descritti i dizionari utilizzati da *Quartz* per operare il filtraggio dei quality scores delle sequenze `fastq` nella fase di test. Viene descritto inizialmente il dizionario realizzato dagli autori di *Quartz,* illustrando il Dataset di file di partenza e l'algoritmo con qui è stato generato, ovvero, l'algoritmo Misra-Gries modificato. In seguito viene proposta una pipeline alternativa per la costruzione di dizionari, basata sul conteggio efficiente dei k-mer di un determinato Dataset e sul loro filtraggio basato su uno score. Tale score è basato sulla probabilità di ciascun k-mer e per il suo calcolo è stato adottato un modello statistico basato sui processi di Markov. Applicando tale pipeline a due Dataset reali di 20 e 40 file genomici, sono stati realizzati dieci diversi dizionari, i quali sono stati oggetto di valutazione.

2.1 Il dizionario di quartz

Il dizionario ufficiale di quartz, denominato *dec200* [1], è costituito da 2.497.777.248 k-mer distinti di lunghezza 32 e, codificato in binario e compressso, ha una dimensione di circa 12GB . Per generare il dizionario, gli autori di Quartz, hanno compilato una lista dei 32-mers più frequenti basandosi su un vasto dataset di file di reads genomiche. In particolare, tale dataset è costituito da 194 file FASTQ presi dal sito ufficiale di *1000 Genomes Project*, corrispondenti a coppie appaiate di reads di 97 individui umani di lunghezza variabile da 91-103 basi con una profondità totale di copertura di 700x. Tutti i file indicati sono liberamente disponibili e scaricabili dall'archivio ftp del *1000 Genomes Project*, in particolare dai sequenti URL corrispondenti alle coppie di file di ogni campione:

ftp://ftp.1000genomes.ebi.ac.uk/vol1/ftp/phase3/data/${SAMPLE_NAME}/sequence_read/${FASTQ_FILE}_1.filt.fastq.gz

ftp://ftp.1000genomes.ebi.ac.uk/vol1/ftp/phase3/data/${SAMPLE_NAME}/sequence_read/${FASTQ_FILE}_2.filt.fastq.gz

Dove *SAMPLE_NAME* e *FASTQ_FILE* sono rispettivamente il nome del campione e dei file FASTQ.

[1] Il dizionario è liberamente scaricabile all'indirizzo web: http://giant.csail.mit.edu/quartz/dec200.bin.sorted.gz

SAMPLE_NAME	FASTQ_FILE
NA19649	SRR189825
NA19657	SRR068163
NA19658	SRR189828
NA19663	SRR068160
NA19664	SRR068164
NA19669	SRR189827
NA19670	SRR189826
NA19720	SRR350122
NA19914	SRR350098
NA20362	ERR257982
NA20821	ERR257967
NA21125	ERR055396
NA21137	ERR055395
NA18966	ERR234335
NA18978	ERR234334
NA19351	ERR229817
NA19764	ERR229816
NA20357	ERR229818
NA20359	ERR229819
NA20503	ERR229821
NA20507	ERR229820
NA20513	ERR229822
NA20514	ERR229823
NA18528	ERR234331
NA18531	ERR234333
NA18533	SRR189816
NA18537	ERR034771
NA18572	ERR034774
NA18609	ERR034777
NA18611	ERR034778
NA18991	ERR052929
NA18488	SRR189829
NA18501	ERR234330

SAMPLE_NAME	FASTQ_FILE
NA18870	ERR234332
NA18881	ERR257965
NA19092	SRR189830
NA19093	ERR229810
NA19137	ERR229811
NA19138	ERR229812
NA19141	SRR211276
NA19143	SRR211272
NA19175	ERR229813
NA19210	ERR229815
NA19222	ERR229814
NA20900	ERR257972
NA21104	SRR768143
NA21120	ERR126299
NA21122	ERR125594
NA21123	ERR125595
NA21127	ERR257973
NA18504	SRR350142
NA18912	SRR350153
NA19200	SRR352199
NA19160	SRR352222
NA19171	SRR359061
NA18968	SRR359062
NA19204	SRR359064
NA18975	SRR359070
NA18981	SRR359083
NA18971	SRR359095
NA19131	SRR359096
NA19152	SRR359097
NA19119	SRR359106
NA18976	SRR359110
NA18974	SRR360136

SAMPLE_NAME	FASTQ_FILE
NA06994	SRR385751
NA11930	ERR233226
NA11932	ERR233225
NA11933	ERR233227
NA11992	SRR385759
NA11994	SRR385753
NA12003	SRR385756
NA12046	ERR239333
NA12154	SRR385769
NA12155	SRR385762
NA12234	ERR233302
NA12249	SRR211278
NA12282	ERR233301
NA12283	ERR239334
NA12286	ERR234321
NA12340	SRR075006
NA12716	ERR257986
NA12717	ERR257987
NA12748	ERR234323
NA12750	ERR257985
NA12751	ERR257988
NA12761	ERR257989
NA12827	ERR234322
NA12829	ERR234325
NA12830	ERR234324
NA12842	ERR234327
NA12843	ERR234326
NA12878	SRR622461
NA12889	ERR234328
NA12890	ERR234329
NA18969	SRR211274
NA18970	SRR211273

Figura 2.1: Elenco dei 97 file FASTQ usati nella generazione del dizionario ufficiale di Quartz.

A partire da questi dati, per una costruzione efficiente del dizionario di k-mers, gli autori di quartz, hanno utilizzato una variante dell'algoritmo di conteggio approssimativo Misra-Gries. Nella loro versione, gli autori, hanno preferito l'utilizzo di una hash table limitata a 65535 buckets, anzichè l'originale linked list prevista dalla versione originale, così da rendere più efficiente l'utilizzo della memoria. Una volta suddiviso lo spazio dei 32-kmer in 256 sottoinsiemi, generati facendo variare ciscuna base nelle sue quattro varianti, e inizializzata la hash table con tutti i rispettivi k-mer di lunghezza 32 per ciascuno di essi, viene lanciato l'algoritmo Misra-Gries su ciascun insieme. Il funzionamento è descritto dallo pseudocodice riportato di seguito[11].

Al termine del conteggio operato da Misra-Gries, gli autori, considerando il margine d'errore dovuto all'approssimazione effettuata dall'algoritmo, garantiscono che tutti i 32-mer che costituiscono il dizionario appaiano almeno 200 volte nel Dataset e che tutti i 32-mer che appaiono almeno 240 volte nel dataset sono compresi nel dizionario. Con tale soglia, il dizionario creato è costituito da 2.497.777.248 distinti 32-mer.

Lo svantaggio dell'implementazione di tale algoritmo sta nel fatto che è fortemente sequenziale. Anche se è di natura polinomiale, i tempi computazionali necessari ad elaborare un dizionario da un dataset, anche modesto, tendono a diventare estremamente lunghi.

Modified Misra-Gries:

1. Initialize a primary hash map A.
   Initialize 65535 hash sets $C_1, \ldots, C_{65535}$.
   Initialize the counter $D = 0$.
2. **update**(i): **if** $i \in A$, remove i from $C_{A[i]}$, increment $A[i]$ (unless $A[i] = 65535$), and insert i to the new $C_{A[i]}$.
   else if $|A| < m$, insert $(i, 1)$ into A and insert i into C_1.
   else Remove an arbitrary element j from C_D (while $|C_D| = 0$, increment D until nonempty). Remove j from A. Insert $(i, D+1)$ into A and i into C_{D+1}.
3. **query**(i): **if** $i \in A$ **then** output $A[i] - D$.
   else output 0

2.2 Pipeline per la creazione di un dizionario

Un approccio alternativo alla costruzione di un dizionario di k-mer rispetto al metodo visto, si può generalizzare in cinque fasi successive: nella prima vengono selezionati i file contenenti le reads dalla quali, nella seconda, invece, vengono estratti e conteggiati tutti i k-mer che le compongono. Nella terza fase si esegue il calcolo dello score di ciscun k-mer così da poter effettuare un filtraggio basato su tale valore nella fase successiva. Nell'ultima fase, con i k-mer filtrati, viene creato un file binario rappresentante il dizionario, idoneo al suo utilizzo con il software quartz. In figura 2.2 sono illustrate le fasi appena descritte mediante un diagramma a blocchi, mentre di seguito verranno descritte nello specifico.

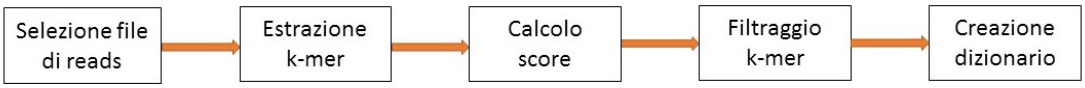


Figura 2.2: Fasi della costruzione del dizionario

La prima fase dunque è la creazione/selezione di un Dataset di file di reads. Il numero e la dimensione di questi file influenzano in maniera significativa la natura del dizionario finale: è necessario scegliere un numero sufficiente di file in modo da garantire un numero significativo di k-mer. A partire da questo Dataset, vengono estratti e conteggiati tutti i k-mer di lunghezza k. Questo processo di estrazione e conteggio dei k-mer è molto dispendioso in termine di risorse computazionali, dato che l'output tende ad esplodere per un numero elevato di reads in input. In questo progetto, si scelto di utilizzare il software JellyFish,[2] che opera il conteggio dei k-mer in maniera efficiente e ad elevato parallelismo, utilizzando una codifica efficiente per la funzione hash. Jellyfish, riceve in ingresso un elenco di file di reads in formato `.fastq` e, una volta processati, crea un database in formato `.jf` con tutti i k-mer estratti e conteggiati con la loro molteplicità. Anche se Jellyfish

[2]La versione di JellyFish utilizzata è 2.2.6, il software è liberamente scaricabile all'indirizzo web http://www.cbcb.umd.edu/software/jellyfish/

non supporta i file di input compressi (ad esempio in formato `.gz`), per maggior efficienza di spazio, i file del dataset possono essere mantenuti compressi e operare la decompressione direttamente in memoria al momento delll'utilizzo con Jellyfish, utilizzando una Unix pipe con il comando `gunzip -c` e specificando a JellyFish di leggere i file di input dallo standard input (/dev/fd/0).

```
gunzip -c $reads/*.fastq.gz | $jellyfish count -m 32 -L $soglia -C -o
    $dbfile -s 20G -t #thread /dev/fd/0
```

Per aumentare l'efficienza di calcolo è possibile indicare a Jellyfish di operare su più file simultaneamente. Questo viene fatto con l'utilizzo di *"generatori"*:

```
ls $reads/*.fastq.gz | xargs -n 1 echo gunzip -c > generators
$jellyfish count -g generators -G #generators -m 32 -L #soglia -C -s
    20G -o $dict -t #thread
```

Il database prodotto, al termine dell'esecuzione di Jellyfish, non è in formato leggibile, perciò, per estrarre le informazioni in formato testuale e poterlo analizzare ed elaborare, il software, mette a disposizione il comando `dump`. In questo modo è possibile ottenere in un file di testo l'elenco dei k-mer con i loro conteggio separati da una interruzione di riga.

L'output appena descritto è necessario per il calcolo delle probabilità statiche e di transizione delle basi nucleotidiche utili al calcolo dello *score* di ciascun k-mer. Lo *score* è calcolato nel seguente modo:

$$score_w = \frac{F_w}{E[w]} = \frac{F_w}{L * P_w} \simeq \frac{F_w}{F_{tot} * P_w}$$

Dove F_w rappresenta la frequenza in cui il k-mer w è presente nel database e $E[w]$ la sua aspettazione. Quest'ultimo valore si può approssimare moltiplicando la probabilità del k-mer per il numero totale di k-mer L presenti nel dataset di partenza. Tale valore può essere complicato da rilevare per grandi Dataset, dove molto spesso i file vengono mantenuti compressi. Si dovrebbe operare un preprocessing dei file nel quale questi vengono decompressi e scansionati memorizzando man mano il numero di k-mer che li compongono. Una soluzione più efficiente a questo problema, è quella di stimare il numero totale dei k-mer con il numero totale di k-mer presenti nel database (considerati con le loro molteplicità), che si ottiene sommando tutti i valori delle frequenze dei k-mer:

$$F_{tot} = \sum_w F_w$$

Il vantaggio di questa stima deriva dal fatto che il valore di F_{tot} può essere calcolato assieme al calcolo delle probabilità statiche e di transizione delle basi nucleotidiche, scansionando una sola volta l'elenco dei k-mer. Nel caso sia stato definito, durante la fase di estrazione dei k-mer, un limite minimo di apparizione dei k-mer, F_{tot} risulterà minore di L ma, per soglie basse (per esempio di 1 o 2), risulterà comunque vicino e ugualmente rappresentativo. P_w, invece, rappresenta la stima della probabilità associata al k-mer w. Tale probabilità è possibile stimarla utilizzando vari approcci e modelli matematici, in questo caso è stato scelto di

adottare un modello markoviano, dove ogni k-mer viene visto come il prodotto di un processo stocastico che gode della proprietà di Markov. Il modello adottato è stato approfondito nella sezione 2.3.

Allo scopo di effettuare il calcolo dello score, è stato creato un piccolo software, denominato *jftool*, che prende in ingresso il file di output del comando `dump` di Jellyfish ed esegue i conteggi. Per la precisione, *jftool* mette a disposizione più comandi utili alla manipolazione e all'analisi dell'output di *jellyfish*, ovvero seguenti comandi:

- `count:` prende in ingresso la lista dei k-mer conteggiati da jellyfish ed esegue il conteggio della frequenza di ciascun nucleotide, scandendo tutti i k-mer conteggiati e considerando le rispettive cardinalità. Al termine della scansione, calcola le probabilità stazionarie e di transizione per ciascuna base e le scrive su un file denominato `info.jft`.

- `filt:` prende anch'esso in ingresso la lista dei k-mer conteggiati da jellyfish e ne esegue il filtraggio basato sullo score di ciascuno, confrontandolo con una soglia minima e una massima. Per eseguire questo comando è necessario aver eseguito il comando `count` e aver, quindi, creato il file `info.jft` contenente le probabilità. L'output è la lista dei k-mer filtrati con associato lo score di ciascuno. E' possibile indicare con il flag 's' o 'ns' se inserire o non inserire nel file di output il valore dello score di ciascun k-mer. Con i flag 'z' o 'nz', invece, è possibile indicare al tool di creare o meno il file di output in formato compresso .gz.
  Al termine viene creato un file di testo contenente alcune statistiche riguardo i k-mer filtrati, fornendo alcune informazioni utili come ad esempio il valore minimo e massimo dello score, la media, e il conteggio dei k-mer suddiviso per ogni ordine di grandezza dello score.

Nel frammento di codice sotto riportato un esempio di utilizzo di jftool.

```
$jellyfish dump $dict | $jftool count
$jellyfish dump $dict | $jftool filt s nz *nomefile *soglia min
   max
```

Al termine di questa fase si ottiene una lista di k-mer aventi lo score compreso tra le soglie prescelte.

Allo scopo di operare un ulteriore filtraggio, la lista di k-mer deve essere ordinata: questa operazione può essere fatta con il comando unix-like `sort.` Una volta ottenuta la lista ordinata di k-mer, si può eliminare dal file il campo score, da qui non più necessario, ed eventualmente operare un ulteriore filtraggio dei k-mer con score più basso portandoli ad un numero prestabilito. Queste operazioni di manipolazione del file possono essere fatte con comandi unix, quali `sort`, `cat` e `tail`.

```
sort -k 2 -n --parallel=#nthread -S #RAM -T #outputdir -o #
   sortfile #inputfile
cat  #sortfile | cut -f1 -d " " > #sortkmerfile
tail -n  #numkmer #sortkmerfile > #dictfile
```

Una volta che si dispone del dizionario definitivo, prima di essere utilizzato da *quartz*, deve essere convertito in un formato idoneo. Quartz mette a disposizione due tool a questo scopo: `text2bin` e `sort_dict_file`. Il primo converte il file da formato testuale a binario mentre il secondo ordina i k-mer al suo interno basandosi sulla loro codifica binaria. Completate queste due operazioni il dizionario così creato è pronto all'utilizzo con *quartz*.

La figura 2.3riassume in uno schema a blocchi la pipeline software cha ha portato alla costruzione dei dizionari.

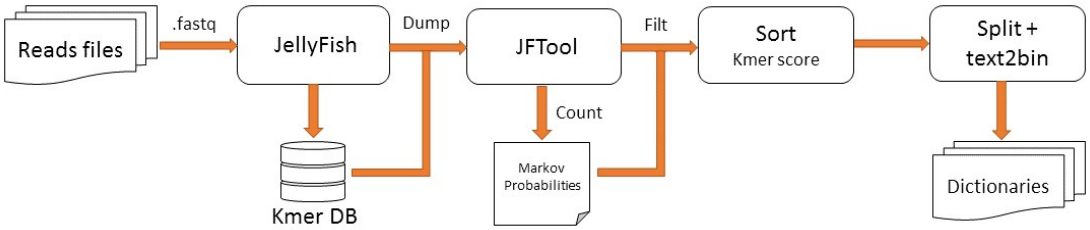


Figura 2.3: Pipeline software per la costruzione del dizionario

2.3 Stima della probabilità dei k-mer

Le probabilità dei k-mer sono state calcolate addottando un modello statistico Markoviano, assumendo che ogni k-mer sia il prodotto di un processo stocastico in cui vale la proprietà di Markov e in cui ogni nucleotide è una variabile aleatoria. Prima di spiegare in dettaglio come è stato operato il calcolo di queste probabilità, si è ritenuto importante riportare di seguito un breve richiamo alla teoria dei processi aleatori e alla teoria delle catene di Markov.

2.3.1 Processi aleatori discreti e Catene di Markov

Si definisce un processo stocastico a tempo discreto e stati discreti una successione di variabili aleatorie X:

$$X_0, X_1, \ldots, X_n, \ldots$$

dove ciascuna X_n è una v. a. discreta a valori nell'insieme S rappresentante lo *spazio degli stati*. Tale insieme può essere finito o numerabilmente infinito. Senza perdere di generalità si indica che S è un sottoinsieme dei numeri interi relativi $S \subseteq \mathbb{Z}$. L'indice n della variabile X_nè definito come *tempo* e i valori che tale variabile può assumere sono denominati *stati*. Al trascorrere del tempo n, tra l'istante n e l'istante successivo $n+1$ il processo più passare dalla stato i allo stato j: in questo caso avviene una *transizione*.

Dato un processo stocastico (X_n) è possibile calcolare alcune probabilità a esso associate:

- Probabilità stazionaria:

$$P(X_n = i) = p_i^{(n)}$$

che rappresentano la probabilità di osservare al tempo n il sistema nello stato i, dove $p^{(n)}$ indica il vettore delle componenti $p_i^{(n)}$e rappresenta la distribuzione della v.a. X_n. Per ogni istante n vale che $\sum_{i\in\mathbb{Z}} p_i^{(n)} = 1$.

- Probabilità di transizione:

$$P(X_{n+1} = j \mid X_n = i) = \frac{P(X_n = i, X_{n+1} = j)}{P(X_n = i)}$$

che rappresenta la probabilità del processo di trovarsi nello stato j al tempo *n+1* sapendo di essere nello stato i al tempo n e può essere scritta in tale modo se si applica la definizione di *probabilità condizionale.* Può essere calcolatà perciò servendosi della probabilità stazionaria $p_i^{(n)}$ e le probabilità congiunte $P(X_n = i, X_{n+1} = j)$.

Per il calcolo di probabilità di transizione più complesse, al variare di tutti gli n e tutti gli i, è necessario conoscere molteplici probabilità congiunte della forma $P(X_o = i_o, \ldots, X_n = i_n)$. Questo deriva dal fatto che un processo stocastico è completamente determinato statisticamente quando sono note tutte le densità congiunte. Tale conoscenza non è banale e risulta essere un problema molto arduo. Applicando ancora una volta il concetto di probabilità condizionale si può scrivere:

$$\begin{aligned} &P(X_o = i_o, \ldots, X_{n+1} = i_{n+1}) = \\ &\quad = P(X_{n+1} = i_{n+1} \mid X_0 = i_0, \ldots, X_n = i_n) * P(X_o = i_o, \ldots, X_n = i_n) \end{aligned}$$

Perciò, se fossero note le probabilità condizionali $P(X_{n+1} = i_{n+1} \mid X_0 = i_0, \ldots, X_n = i_n)$ si può procedere al calcolo delle probabilità congiunte $P(X_o = i_o, \ldots, X_{n+1} = i_{n+1})$ utilizzando in modo ricorsivo le probabilità $P(X_o = i_o, \ldots, X_n = i_n)$. A questo proposito vengono in aiuto le catene di Markov omogenee.

Un processo stocastico discreto $(X_n)_n$ si dice di Markov se:

$$P(X_{n+1} = j \mid X_n = i, X_{n-1} = i_{n-q}, \ldots, X_0 = i_0) = P(X_{n+1} = j \mid X_n = i)$$

Questa formula illustra la *proprietà di Markov* e afferma che la probabilità che la variabile X_{n+1}assuma un certo valore dipende solamente dalla conoscenza del valore della variabile immediatamente adiacente X che la precede e che la conoscenza dei valori di $(X_{n-1} \ldots X_0)$ diviene superflua perchè non dà ulteriori informazioni. Questa formula è valida al variare di tutti i parametri e qualora siano ben definite tutte le probabilità condizionate espresse. Tale processo si dice omogeneo se le probabilità di transizione p_{ij} non dipendono da n ma solo dai parametri i e j. Sotto questa condizione è possibile allora calcolare tutte le probabilità congiunte

conoscendo le sole probabilità di transizione $p_{ij} = P(X_{n+1} = j \mid X_n = i)$ e la distribuzione iniziale $p_i^{(0)} = P(X_0 = i)$:

$$P(X_o = i_o, \ldots, X_n = i_n) = p_i^{(0)} * p_{i_0 i_1} \ldots p_{i_{n-1} i_n}$$

Molto spesso è conveniente visualizzare una catena di Markov ricorrendo all'utilizzo di grafi e matrici. Si possono rappresentare gli stati i, j, ... della catena con i nodi (vertici) di un grafo e rappresentare le transizioni con gli archi orientati. A ogni arco $i \to j$ è associata la probabilità di transizione $p_{ij} > 0$ dalo stato i allo stato j. In particolare vale che:

$$p_{ij} \in [0, 1], \sum_j p_{ij} = 1 \, \forall i$$

ovvero che la somma di tutti gli archi uscenti di un nodo i deve essere pari a uno. Tale grafo prende il nome di *diagramma degli stati*. Una matrice quadata costituita da tutte le p_{ij}, invece, prende il nome di *matrice di transizione* e, in acordo con quanto detto fin'ora, è formata da tutti elementi non negativi e la somma di ogni sua riga è pari a uno.

2.3.2 Stima della probabilità mediante Markov

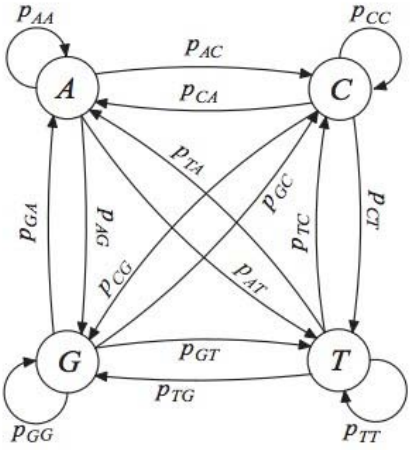
Il modello statistico adottato per la stima delle probabilità dei k-mer si basa sulla teoria illustrata. In particolare, le sequenze di DNA sono state considerate come il prodotto di un processo che può essere descritto come una sequenza di variabili casuali $X_1, X_2, \ldots, X_i$ dove X_i corrisponde al nucleotide di posizione i del k-mer. In questo modello, ogni variabile X_i casuale assume un valore dall'insieme degli stati {A, C, G, T} e la probabilità di ciscuna variabile dipende dal contesto locale, ovvero, dal valore dalla base immediatamente adiacente che la precede (tranne per la prima base di ogni sequenza di cui viene considerata la probabilità stazionaria associata). Ogni k-mer, perciò, viene visto come una catena di Markov di primo ordine dove la probabilità che Xi assume un valore particolare dipende solo dalla variabile precedente $Xi - 1$. In figura 2.4 è riportato il diagramma degli stati e la matrice di transizione che descrivono la catena adottata.

In questo modo è possibile calcolare la probabilità di ciascun k-mer conoscendo le probabilità stazionarie e di transizione delle basi. Scandendo tutti i k-mer estratti e conteggiando tutte le singole basi e tutte le coppie di basi contenute, è possibile calcolare queste probabilità nel seguente modo:

Prob. staz. $P_A = \frac{Tot_A}{Tot_{basi}}$, $P_C = \frac{Tot_C}{Tot_{basi}}$, $P_G = \frac{Tot_G}{Tot_{basi}}$, $P_T = \frac{Tot_T}{Tot_{basi}}$

Prob. trans. $P_{AA} = \frac{Tot_{AA}}{Tot_{Ax}}$, $P_{AC} = \frac{Tot_{AC}}{Tot_{Ax}}$ ... $P_{CA} = \frac{Tot_{CA}}{Tot_{Cx}}$ ... $P_{TT} = \frac{Tot_{TT}}{Tot_{Tx}}$

Dove le prime sono calcolate dividendo il numero totale di ogni singola base per il totale complessivo delle singole basi, mentre le seconde dividendo il totale di ogni coppia di basi per il totale delle coppie che hanno come primo elemento il primo elemento della coppia. Ottenute queste probabilità, è possibile procedere al calcolo della probabilità di ogni k-mer moltiplicando la probabilità stazionaria della prima



	A	C	G	T
A	P_{AA}	P_{AC}	P_{AG}	P_{AT}
C	P_{CA}	P_{CC}	P_{CG}	P_{CT}
G	P_{GA}	P_{GC}	P_{GG}	P_{GT}
T	P_{TA}	P_{TC}	P_{TG}	P_{TT}

Figura 2.4: Rappresentazione del diagramma degli stati e della matrice di transizione della catena di Markov adottata, dove gli stati corrispondono alle quattro basi nucleotidiche.

base di cui è costituito (k0) per le probabilità di transizione di tutte le coppie di basi seguenti $(k_0k_1, k_1k_2 \ ...k_{j-1}k_j)$, in accordo con la proprietà di Markov:

$$P_k = P_{k_0} * P_{k_0k_1} * ... * P_{k_{31}k_{32}} = P_{k_0} \prod_{j=0}^{31} P_{k_jk_{j+1}}$$

Si noti che, nella formula sopra riportata, è riportata la probabilità di un k-mer con pari a 32, ovvero di un 32-mer, ma in generale, una una volta ottenute le quattro probabilità stazionarie e le sedici probabilità di transizione, è possibile calcolare la probabilità di k-mer di qualsiasi lunghezza.

2.4 Dizionari creati

Per la creazione dei dizionari è stata utilizzata la pipeline software descritta nella sezione 2.2. Si è partiti da due differenti dataset di campioni genomici, entrambi realizzati campionado il dataset originale del dizionario di *quartz* formato da 97 coppie di campioni, uno costituito da 10 coppie di file genomici per una coverage totale di 75x e uno da 20 coppie ed una coverage totale di 136x . Con il primo dataset sono stati creati quattro dizionari di differente numero di k-mer, mentre dal secondo ne sono stati creati sei. Le coppie di file che che costituiscono i due dataset utilizzati sono riportati nella tabella seguente:

Dataset 1 - "DS20"		**Dataset 2 - "DS40"**			
Sample	*fastq file*	*Sample*	*fastq file*	*Sample*	*fastq file*
NA19649	SRR189825	NA19649	SRR189825	NA18870	ERR234332
NA19657	SRR068163	NA19657	SRR068163	NA19092	SRR189830
NA20362	ERR257982	NA19663	SRR068160	NA21104	SRR768143
NA19663	SRR068160	NA20821	ERR257967	NA19664	SRR068164
NA19664	SRR068164	NA18966	ERR234335	NA19658	SRR189828
NA12717	ERR257987	NA18978	ERR234334	NA18528	ERR234331
NA19670	SRR189826	NA19764	ERR229816	NA18531	ERR234333
NA19720	SRR350122	NA20357	ERR229818	NA19137	ERR229811
NA19914	SRR350098	NA20514	ERR229823	NA19175	ERR229813
NA19669	SRR189827	NA18533	SRR189816	NA19210	ERR229815

I file di questi dataset sono stati processati dal softaware Jellifish che ha operato il conteggio dei k-mer. Per entambi i dataset è stata specificata una dimensione di k-mer pari a 32 e soglia minima di due occorrenze per ciascun k-mer: al termine del conteggio, perciò, non sono stati estratti tutti i 32-mer trovati ma solo quelli che comparivano almeno due volte. Questa scelta di operare un pre-filtraggio su tutti i 32-mer presenti è stata fatta con l'intenzione di eliminare tutti i k-mer che comparivano solo una volta, ritenuti questi trascurabili e fonte di rumore nella determinazione dei k-mer descriminativi. Inoltre, la riduzione del numero dei k-mer totali conteggiati comporta una significativa riduzione dello spazio occupato su disco dal database di Jellyfish e una diminuzione dei tempi computazionali.

2.4.1 Dataset "DS20"

Dal dataset costituito da 10 coppie di file, denominato "DS20", si è ottenuta una lista contenente 3.506.581.098 distinti 32-mer di molteplicità variabile da 2 a 10.493.252. Il numero totale di k-mer, ovvero F_{tot}, è risultato di 132.072.477.998. Per maggior precisione si è voluto trovare anche il valore dei k-mer totali L presenti nel dataset, così da confrontarlo con il valore F_{tot}: tale valore è risultato essere di 158.284.666.960. Quindi considerando F_{tot} anzichè L per il calcolo dello score, si commette in questo caso un errore del 16% circa. Nella tabella sottostante sono riportate tutte le statistiche calcolate per questo dataset:

Statistiche k-mer - Dataset DS20	
Totali (L)	158.284.666.960
Singoli	26.212.188.962
Tot. > 2 (Ftot)	132.072.477.998
Distinti	3.506.581.098
Freq. max	10.493.252

La lista dei k-mer estratti è stata scansionata e analizzata con software *jftool*, il quale ha permesso di calcolare per ciascuno di essi la propria probabilità in accordo con la medodologia illustrata nella sezione 2.3.2 e, quindi, il proprio *score*. Nella tabella sottostante sono riportate le probabilità stazionarie e le probabilità di transizione calcolate considerando tutti i 32-mer conteggiati. In accordo con le proprietà delle catene di Markov, si noti come la matrice delle probabilità risulti stocastica essendo la somma dei valori dei ciascuna riga pari a 1.

Probabilità stazionarie:

A	C	G	T
0.327325	0.205650	0.199991	0.267034

Probabilità di transizione:

	A	C	G	T
A	0.354038	0.171087	0.226795	0.248080
C	0.383233	0.247756	0.052373	0.316638
G	0.325129	0.201273	0.245657	0.227942
T	0.234428	0.213226	0.249887	0.302458

Tra le probabilità stazionarie, il nucleotide A sembra essere quello più probabile con un valore di circa 0.33, mentre quelli meno probabili sembrano essere C e G con un valore di circa 0.20 per entrambi. Le probabilità di transizione, invece, variano quasi tutte all'interno di un intervallo compreso tra 0.20 e 0.35 circa, tranne per la transizione C->G che risulta la meno probabile in assoluto con un valore di circa 0.05. In generale, sembra che la transizione più probabile di uno stato A, C, G sia in A, mentre quella di uno stato T sia di rimanere in T. Ci si aspetta, quindi, nella composizione dei 32-mer di avere una maggioranza di nucleotidi A e una maggioranza di sotto-sequenze composte da T concatenate.

Calcolati gli score di tutti i 32-mer con l'utilizzo del software *jftool*, è stato possibile analizzarne la loro distribuzione: i valori dello score sono risultati essere distribuiti su un intervallo molto ampio, andando da un valore minimo di 16.419 e un valore massimo di circa $5 * 10^{21}$. In figura 2.5 è riportato l'istogramma e il grafico a torta riassuntivi. Dai grafici è facile notare come la quasi totalità dei k-mer, il 98%, abbia uno score di ordine di grandezza compreso tra 10^6e 10^{10}, in particolare di 10^8(circa il 44% del totale).

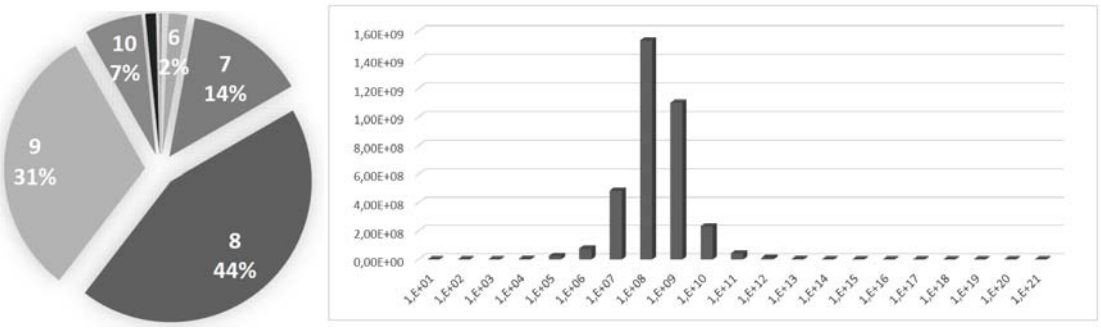


Figura 2.5: Distribuzione degli score dei k-mer per ordine di grandezza.

Basandosi sullo score, i 32-mer sono stati ordinati in modo crescente. Si è deciso di operare vari filtraggi, selezionando il 25%, 50%, il 75% e il 100% dei k-mer con score più alto e creare, quindi, quattro liste da cui creare e i rispettivi quattro dizionari:

Fraz.	**Nome**	**# k-mer distinti**
25%	DS20_top25	876.645.275
50%	DS20_top50	1.753.290.549
75%	DS20_top75	2.629.935.824
100%	DS20_all	3.506.581.098

2.4.2 Dataset "DS40"

Dal dataset costituito da 20 coppie di file, denominato "DS40", si è ottenuta una lista contenente 4.331.088.631 distinti 32-mer di molteplicità variabile da 2 a 21.844.459. Il numero totale di k-mer, ovvero F_{tot}, è risultato di 267.764.667.466. Anche in questo caso, si è voluto trovare anche il valore dei k-mer totali L presenti nel dataset, così da poter confrontarlo con il valore di F_{tot}: tale valore è risultato essere di 292.201.552.590. Quindi considerando F_{tot} anzichè L, si commette un errore di 8,4% circa. Da questi dati si noti come, in confronto al dataset "DS20", con il raddoppio dei file sia raddoppiato circa anche il valore L (ovvero il totale dei k-mer), di Fmax e la molteplicità massima. Il numero dei k-mer distinti, invece, è aumentato del 20% circa, ma non ha raddoppiato. Questo risultato è spiegabile dal fatto che, in un dataset reale, non sono presenti tutti i possibili 32-mer e che una buona parte dei 32-mer non compaiono mai fisicamente nelle sequenze genomiche. Nella tabella sottostante sono riportate tutte le statistiche calcolate per questo dataset:

Statistiche k-mer - Dataset DS40	
Totali (L)	292.201.552.590
Singoli	24.436.885.124
Tot. $>$ 2 (Fmax)	267.764.667.466
Distinti	4.331.088.631
Freq. max	21.844.459

Anche in questo caso, la lista dei k-mer estratti è stata scansionata e analizzata con software *jftoo*l che permesso di calcolare per ciascuno di essi la propria probabilità in accordo con la medodologia illustrata nella sezione 2.3.2. Nella tabella sottostante sono riportate le probabilità stazionarie e le probabilità di transizione calcolate considerando tutti i 32-mer conteggiati. A segno di correttezza del processo di calcolo, anche in questo caso, la matrice delle probabilità risulta stocastica, essendo la somma dei valori dei ciascuna riga pari a 1.

Probabilità stazionarie:

A	**C**	**G**	**T**
0.331213	0.201968	0.196523	0.270296

Probabilità di transizione:

	A	**C**	**G**	**T**
A	0.359079	0.167207	0.221469	0.252245
C	0.385783	0.247489	0.048180	0.318548
G	0.324770	0.200176	0.245589	0.229466
T	0.242237	0.206560	0.244210	0.306993

Le probabilità riportate nelle tabelle, calcolate con il dataset *DS40*, sono in accordo con le probabilità calcolate con il dataset *DS20:* in partcolare si conferma essere il nucleotide A quello più probabile e quelli meno probabili C e G. Ugualmente anche per le probabilità di transizione, invece, variano quasi tutte all'interno del medisimo intervallo, e la transizione C->G si conferma essere la meno probabile in assoluto. In generale, i valori delle probabilità calcolate con questo dataset sono in linea con quelle calcolate con il dataset più piccolo e differiscono in minima parte solo se si considera la terza cifra decimale.

Anche la distribuzione degli *score* rispecchia quella ottenuta in precedenza per l'altro dataset: i valori sono risultati essere distribuiti ancora su un intervallo estremamente ampio, da un valore minimo di 6.701 e un valore massimo di circa $2,3 * 10^{22}$. La maggior parte dei k-mer, circa il 96%, è risultato avere uno score di ordine di grandezza compreso tra 10^6e 10^{10}, in particolare nell'intervallo di 10^8(circa il 38% del totale). In figura 2.5 è riportato l'istogramma e il grafico a torta riassuntivi.

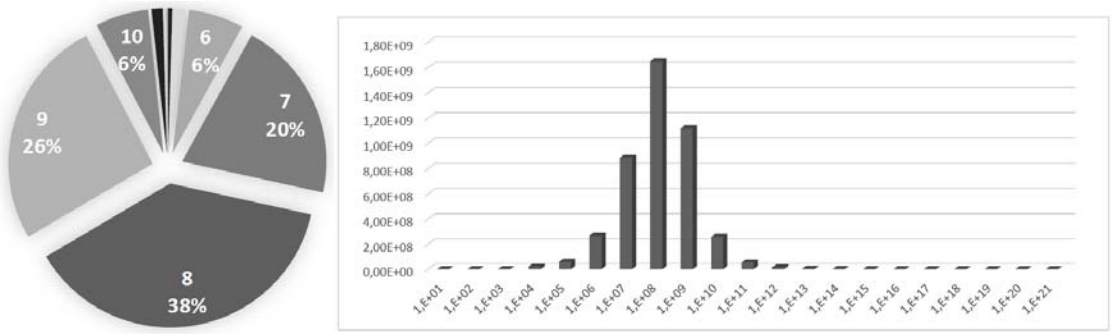


Figura 2.6: Distribuzione degli score dei k-mer per ordine di grandezza.

Si è deciso di operare vari filtraggi, selezionando il 20%, 40%, 58%, 61%, 81% e il 100% dei k-mer conteggiati e creare, quindi, sei liste da cui creare sei dizionari. Nella realizzazione delle liste, si è scelto di selezionare un numero di k-mer che coincidesse con quello dei precedenti dizionari realizzati e con quello ufficiale di quartz, così da poter operare dei confronti in maniera più agile. Sono stati creati, così, i seguenti dizionari:

Fraz.	**Nome**	**# k-mer distinti**
20%	*DS40_top20*	876.645.275
40%	*DS40_top40*	1.753.290.549
58%	*DS40_top58*	2.497.777.248
61%	*DS40_top61*	2.629.935.824
81%	*DS40_top81*	3.506.581.098
100%	*DS40_all*	4.331.088.631

Capitolo 3

Test e risultati

3.1 Pipeline di valutazione

Prima di procedere ai test nei quali sono stati messi sotto esame la capacità discriminativa dei dieci dizionari realizzati ci si è occupati di realizzare una pipeline per la valutazione delle prestazioni, in accordo con la procedura vista nella sezione 1.4. Nello specifico, sono stati utilizzati i software BWA come *aligner,* SAMtools e BCFtools come *SNP caller e* RTGeval come *evaluetor.* Inoltre per il calcolo della regione genomica è stato utilizzato il tool *BEDtools.* In figura3.1è riportato uno schema riassuntivo dell'intera pipeline software utilizzata, mentre nei prossimi paragrafi sono descritti in maniera accurata tutti gli step di cui è composta e le rispettive configurazioni software adottate.

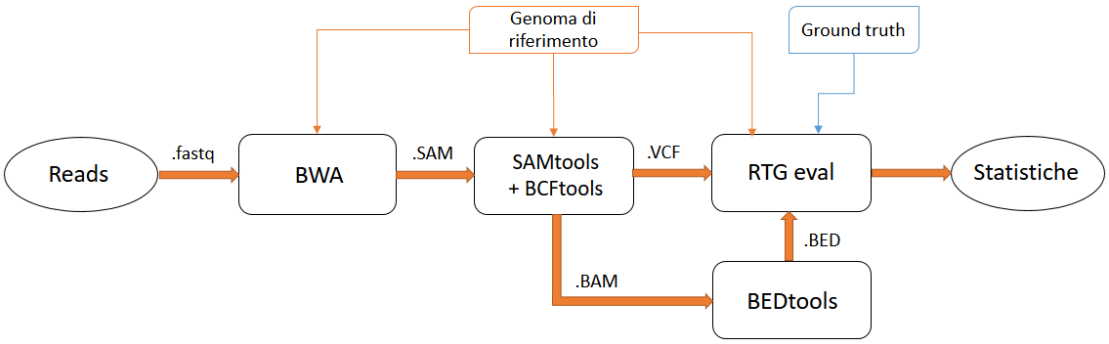


Figura 3.1: Pipeline software per la valutazione degli SNP

3.1.1 Allineamento con BWA

Per l'allineamento genomico è stato utilizzato il software bwa. BWA è un pacchetto software per mappare sequenze con bassa divergenza rispetto a vasti genomi di riferimento, come quello umano. Consiste di tre algoritmi: BWA-backtrack, BWA-SW e BWA-MEM. Il primo è progettato per sequenze fino a 100pb, mentre gli altri due per sequenze più lunghe da 70pb a 1Mpb. BWA-MEM e BWA-SH condividono funzionalità molto simili, come la gestione di lunghe sequenze,

ma BWA-MEM, che è il più recente, è generalmente quello raccomandato perché più veloce e più accurato permettendo di ottenere risultati di maggior qualità. BWA-MEM ha prestazioni migliori anche rispetto a BWA-backtrack per sequenze Illumina 70-100pb. I differenti algoritmi di allineamento vengono rispettivamente invocati tramite i sotto-comandi: `aln/samse/sampe` per BWA-backtrack, `bwasw` per BWA-SW e `mem` per BWA-MEM. Come altri software, bwa sfrutta il metodo dell'indicizzazione (indexing method), costruendo prima un indice del genoma di riferimento tramite il sotto-comando `index`. Ciò consente a BWA di ridurre man mano le zone di possibile localizzazione della read all'interno del genoma

In definitiva, BWA è in grado di seguire l'allineamento di *single-end reads* e di *paired-end reads* fornendo in uscita il miglior allineamento possibile trovato. I file di output contententi l'allinemento possono essere di due tipi: SAM (Sequence Alignment/Map) o la rispettiva versione binaria BAM (Binary Alignment/Map). Il contenuto informativo dei due file è il medesimo, l'unica differenza sta nella rappresentazione: alfanumerica nel primo caso e binaria nel secondo. In generale, è preferibile l'utilizzo di file BAM essendo questi di dimensione più ridotta dei rispettivi file SAM, così da poter risparmiare spazio di archiviazione e di memoria RAM nelle elaborazioni.

A scopo di avere risultati il più possibile confrontabili con lavori già esistenti in letteratura, è stato deciso l'utilizzo dell'algoritmo di allineamento BWA-backtrack utilizzando i comandi `aln` e `sampe`.

```
Usage:   bwa aln [options] <prefix> <in.fq>
Options:
-t INT    number of threads [1]

$bwa index $REF
$bwa aln -t 8 $REF $INPUT1 > $SAI1
$bwa aln -t 8 $REF $INPUT2 > $SAI2

Usage:   bwa sampe [options] <prefix> <in1.sai> <in2.sai> <in1.fq> <in2.fq>
Options:
-P        preload index into memory (for base-space reads only)
-f FILE   sam file to output results to [stdout]

$bwa sampe -P $REF $SAI1 $SAI2 $INPUT1 $INPUT2 > $SAM
```

3.1.2 Variants calling con SAMTOOLS

SAMtools è un insieme di utility per la post-elaborazione di allineamenti di DNA in formato SAM, BAM o in formato CRAM. Questi file, come si è visto, sono generati come output dalla fase di allineamento, come ad esempio dal software BWA. Le operazioni supportate da questo insieme di strumenti permettono attività complesse come il variants calling e la visualizzazione di allineamento, ma anche l'ordinamento, l'indicizzazione, l'estrazione dei dati e la conversione di formato. Come per altri software, oltre ai file SAM, che possono essere molto grandi (solitamente decine di GB), vengono utilizzati file compressi come BAM o CRAM per permettere una maggior efficienza di esecuzione.

Similmente a molti comandi Unix, i comandi di SAMtools seguono un modello di flusso, in cui i dati separati dall'operatore pipe "|", attraversano ogni comando

in cascata. Questo permette combinare più comandi in un pipeline di elaborazione date ottenere risultati molto complessa con un numero limitato di semplici comandi, spesso riassunti in una riga. SAMtools fornisce i seguenti comandi:.

view Con il comando `view` si possono visualizzare gli allineamenti contenuti in un file o, tramite opportune opzioni, eseguirne un filtraggio come ad esempio l'estrazione di un sottoinsieme di dati in un nuovo file o la conversione tra formati BAM/SAM e viceversa. Con questo comendo non viene modificato l'ordinamento dei dati.

sort Il comando `sort` ordina un file di input in base alla sua posizione nel riferimento o, eventualmente, in base al nome. In uscita il file ordinato viene scritto in un nuovo file o essere re-diretto a stdout. Per queste operazioni di ordinamento vi è un uso intensivo di memoria, perciò è previsto l'uso di un file temporaneo e una modalità di sezionamento per utilizzare al più una specifica quantità di memoria e generare file di output multipli che possono essere poi uniti in un unico file ordinato.

index Il comando `index` crea un nuovo file indice che permette velocemente il look-up dei dati in un file SAM o BAM . Vengono così creati dei file supplementari per ciascun file di input (`*.sai` per i file SAM o `*.bai` per i file BAM) che permettono ai programmi che li supportano di processarli con maggior efficienza.

faidx Il comando `faidx` indicizza il file *FASTA* di riferimento o, se specificata una determinata regione genomica ne estrae la relativa sottosequenza. Se tale regione non viene specificata viene creato un file di supporto `*.fai` relativo all'indice.

mpileup Il comando `mpileup` produce un file in formato pile-up VCF (o compresso BCF) contenenti gli SNP individuati dal confronto col genoma di riferimento. E' necessario che l'output di questo comando venga reindirizzato in input al tool BCFtools `call`.

Per la determinazione delle varianti è necessario utilizzare un altro strumento che operi un filtraggio sui dati descriminando le qualità associate ai siti degli SNP. A questo scopo si utilizza il set di strumenti BCFtools, in particolare il comando `filter`, per eliminare gli SNP considerati fonte di errori corrispondenti a valori troppo elevati o troppo bassi di qualità che renderebbero i dati rumorosi. Il file VCF filtrato ottenuto in output è quella che viene usato per la valutazione.

```
Usage:  samtools mpileup [-EBugp] [-C capQcoef] [-r reg] [-f in.fa] [-l list] [-Q
    minBaseQ] [-q minMapQ] in.bam [in2.bam [...]]
Options:
-f FILE The faidx-indexed reference file in the FASTA format. The file can be
    optionally compressed by bgzip.
-u      Output uncompressed BAM. This option saves time spent on compression/
    decompression and is thus preferred when the output is piped to another
    samtools command.

Usage:  bcftools call [OPTIONS] FILE
```

```
Options:
-m      alternative modelfor multiallelic and rare-variant calling designed to
    overcome known limitations in -c calling model (conflicts with -c)
-v      output variant sites only

$samtools faidx $REF
$samtools view -bS -o $BAM $SAM
$samtools sort -T $BAM.tmp -o $SORTBAM $BAM
$samtools index $SORTBAM
$samtools mpileup -uf $REF $SORTBAM | $bcftools call -mv > $RAWVCF
$bcftools filter -s LowQual -e '%QUAL<20 || DP>100' $RAWVCF  > $VCF
```

3.1.3 Calcolo della regione genomica con BEDTOOLS

Prima di eseguire la valutazione del file VCF prodotto è necessario creare un file BED nel quale vengono specificate le regioni genomiche sulle quali sono presenti gli SNP da valutare. Per questa operazione viene utilizzato il set di strumenti BEDtools, che consentono di operare funzioni logico/aritmetiche sui tracciati genomici, come ad esempio, intersezioni, fusioni, conteggi, complementi, e shuffle degli intervalli genomici di uno o più file di molteplici formati, come BAM, BED, GFF / GTF, VCF. In particolare, è stato utilizzato il comando `bamToBed` con il quale, partendo dall'allineamento genomico contenuto nel file BAM, viene creato un file BED contente, una per riga, le specifiche di ciascuna regione genomica presente.

```
Usage:   bedtools bamtobed [OPTIONS] -i <bam>

$bedtools -i $SORTBAM > $BED
```

3.1.4 Vcf evaluation e curva ROC con i tool RTG

RTG Tools include diversi strumenti utili alla manipolazione di file VCF, sopratutto il tool `vcfeval,` che svolge in modo sofisticato il confronto fra file VCF. Questo strumento esegue il confronto a livello di aplotipo fra un file VCF da valutare e uno di riferimento usato come gold standard, in modo accurato e veloce. In uscita, vcfeval produce una serie di file VCF separati contenenti i risultati del confronto, le metriche di sintesi (TP, FP, TN e FN), e altri file contenenti dati che possono essere utilizzati dal tool RTG `rocplot` per visualizzare la curva ROC.

Infatti in accordo con il comando `vcfeval`, il comando `rocplot` fornisce in modo semplice, e anche interattivo, le curve ROC di uno o più confronti visualizzando il punteggio e le soglie di filtraggio.

```
Usage: rtg COMMAND [OPTION]...
Utility:
bgzip       compress a file using block gzip
            rtg bgzip [OPTION]... FILE+
index       create a tabix index
            rtg index [OPTION]... -f FORMAT FILE+
vcfeval     evaluate called variants for agreement with a baseline variant set
rocplot     plot ROC curves from vcfeval ROC data file

$rtg format -o $SDFREF $REF
$rtg bgzip $VCF
$rtg index -f vcf $VCF.gz
```

```
$rtg vcfeval -t $SDFREF -b $VCFREF --bed-regions=$VCFBED -c $VCF.gz -e $BAMBED -f
    QUAL -o $DIREVAL
$rtg rocplot $DIREVAL/weighted_roc.tsv.gz --svg ./$ROC.svg
```

3.2 Risultati

Per eseguire i test sui dizionari realizzati, e valutare della capacità di rilevazione degli SNP e di compresione, è stato utilizzato un campione genomico standard molto diffuso in letteratura: il campione NA12878. In particolare, sono state estratte dall'archivio "1000 Genome Project[1]" le coppie di file *SRR622461_1.fastq* e *SRR622461_2.fastq* relative a tale campione. Per questi campioni è stato usato come *gold standard* il dataset di varianti genetiche validate *vcfNA12878_GIAB_highconf_IllFB-IllGATKHC-CG-Ion-Solid_ALLCHROM_v3.2.2_highconf.vcf.gz*[2], mentre, il genoma di riferimento compatibile adottato, è stato il file *hs37d5.fa*. Le analisi sono state condotte su un cluster di calcolo dotato di 64 Processori quad core Intel Xeon E5450 (12MB Cache, 3.00 GHz) e un totale di 256 GB RAM.

I campioni genomici sono stati filtrati mediante *Quartz* con ciascun dizionario. Nella tabella 3.1 sono raccolti i tempi computazionali, in termini di *User Time*, e la memoria RAM massima occupata da Quartz durante il processo di filtraggio dei campioni genomici. La memoria massima utilizzata varia da circa 20 GB a 30 GB in base alla grandezza del dizionario, mentre, il tempo d'esecuzione è risultato molto variabile e difficilmente stimabile in maniera esatta.

Dizionari	**# Kmer**	**User Time [h]**	**RAM [GB]**
Quartz dict	2.497.777.248	03:59	26,0
DS20_top25	876.645.275	04:40	19,8
DS20_top50	1.753.290.549	03:42	23,1
DS20_top75	2.629.935.824	04:07	26,4
DS20_all	3.506.581.098	04:20	29,6
DS40_top20	876.645.275	04:26	19,8
DS40_top40	1.753.290.549	04:35	23,1
DS40_top58	2.497.777.248	04:02	26,0
DS40_top61	2.629.935.824	04:16	26,4
DS40_top81	3.506.581.098	04:09	29,6
DS40_all	4.331.088.631	04:21	32,8

Tabella 3.1: Tempi computazionali e spazio di memoria occupato dal software Quartz durante il filtraggio dei campioni genomici.

Successivamente, tutti i file filtrati prodotti sono stati processati dalla pipeline di valutazione descritta nella sezione 3.1, mediante la quale sono stati valutati in termini di accuratezza nella ricerca degli SNP. In particolare, al termine del processo di valutazione si ottengono i valori TP, FP, FP e FN definiti come:

[1] ftp://ftp.1000genomes.ebi.ac.uk/vol1/ftp/phase3/data/NA12878/sequence_read/
[2] ftp://ftp-trace.ncbi.nlm.nih.gov/giab/ftp/release/NA12878_HG001/NISTv3.2.2/

- TP: il numero dei *true positive*, ovvero il numero degli SNP identificati correttamente;
- FP: il numero dei *false positive,* ovvero il numero degli SNP identificati erroneamente;
- TN: il numero dei *true negative*, ovvero il numero dei non SNP identificati correttamente;
- FN: in numero dei *false negative*, ovvero il numero degli SNP non identificati.

Dati questi valori sono state calcolate alcune metriche classiche, quali la Precision, la Recall e la F-measure:

$$Precision = \frac{TP}{TP+FP},\ Recall = \frac{TP}{TP+FN}\ ,\ F-measure = 2\frac{Precision*Recall}{Precision+Recall}$$

In base a queste metriche si è operato un confronto sulle prestazioni dei dizionari considerati.

Inoltre, per valutare la differenza in termini di entropia e di comprimibilità dei file filtrati, sono stati compressi con l'utilizzo dei software di compressione *general purpose* GZIP e BZIP2. In questa sezione sono riportati tutti i risultati sperimentali ottenuti.

3.2.1 Confronto sulla rilevazione degli SNP

Sono stati valutati per primi i dizionari realizzati dal dataset *DS20* costituito da 10 coppie di file genomici. I risultati dei test condotti, riguardo la capacità di rilevazione degli SNP valutata attraverso la pipeline e le metriche precedentemente descritte, sono riportati nella tabella riportata 3.2. I valori sono stati confrontati con il campione genomico originale, a cui non è stato praticato lo smooting, e al campione filtrato con il dizionario ufficiale di Quartz.

Dizionari	**# 32-mer**	**Precision**		**Recall**		**F-measure**	
no dict	-	0,9327	-	0,6539	-	0,7688	-
Quartz dict	2.497.777.248	0,9244	-0,89%	0,6669	+1,99%	0,7748	+0,78%
DS20_top25	876.645.275	0,9267	-0,64%	0,6629	+1,38%	0,7729	+0,53%
DS20_top50	1.753.290.549	0,9254	-0,78%	0,6660	+1,85%	0,7746	+0,75%
DS20_top75	2.629.935.824	0,9246	-0,87%	0,6681	+2,17%	0,7757	+0,90%
DS20_all	3.506.581.098	0,9238	-0,95%	0,6683	+2,20%	0,7755	+0,87%

Tabella 3.2: Risultati dei test valutazione basata sulla rilevazione degli SNP mediante l'utilizzo dei dizionari realizzati con il dataset *DS40*.

Da tale tabella si può vedere come, all'aumentare del numero di 32-mer contenuti nei dizionari, il valore della Precision tenda a diminuire, passando da -0,64% del dizionario *DS20_top25* al -0,95% del dizionario *DS20_all* nei confronti della sequenza originale che non ha subito lo smooting dei valori. Tale risultato è in

Dizionari	# 32-mer	Precision		Recall		F-measure	
no dict	-	0,9327	-	0,6539	-	0,7688	-
Quartz dict	2.497.777.248	0,9244	-0,89%	0,6669	+1,99%	0,7748	+0,78%
DS40_top20	876.645.275	0,9272	-0,59%	0,6625	+1,32%	0,7728	+0,52%
DS40_top40	1.753.290.549	0,9261	-0,71%	0,6653	+1,74%	0,7743	+0,72%
DS40_top58	2.497.777.248	0,9256	-0,76%	0,6670	+2,00%	0,7753	+0,85%
DS40_top61	2.629.935.824	0,9255	-0,77%	0,6672	+2,03%	0,7754	+0,86%
DS40_top81	3.506.581.098	0,9253	-0,79%	0,6678	+2,13%	0,7757	+0,90%
DS40_all	4.331.088.631	0,9249	-0,84%	0,6676	+2,10%	0,7754	+0,86%

Tabella 3.3: Risultati dei test valutazione basata sulla rilevazione degli SNP mediante l'utilizzo dei dizionari realizzati con il dataset *DS40*.

linea, anche, con il risultato prodotto del dizionario ufficiale di *Quartz* che peggiora la Precision di -0,89%. Dal punto di vista della Recall, invece, si verifica un miglioramento: da +1,38% del dizionario *DS20_top75* al +2,20% del dizionario *DS20_all*. Considerando, il valore della F-measure, che combina entrambe le precedenti metriche, si verica un aumento all'aumentare del numero di k-mer contenuti nel dizionario. Il dizionario che ha migliori prestazioni, però, è il dizionario *DS20_top75* contenente 2.629.935.824 32-mer, superando anche quelle ottenute dal dizionario di *Quartz*. Il dizionario *DS20_all,* invece ha valore di F-measure minore, dato che riesce a migliorare di poco la Recall a scapito di un netto peggioramento in termini di Precision. Questo risultato è imputabile al numero troppo alevato di k-mer contenuti, pari alla totalità di quelli estratti dal dataset DS20, che ha portato a considerare anche k-mer aventi scarsa capacità discriminativa.

I succesivi test effettuati con i dizionari del dataset DS40 confermano quanto emerso dai test precedenti, evidenziando le stesse tendenze a riguardo di Precision, Recall e F-measure e sono riportati nella tabella 3.3.

Si riscontra anche in questo caso una diminuzione progressiva del valore di Precision associato ad un aumento del valore di Recall e di F-measure all'aumentare del numero di k-mer contenuti nei dizionari. Anche in questo caso, però, il dizionario contenente il numero più alto di k-mer, il dizionario *DS40_all,* non risulta essere il migliore e peggiora le prestazioni rispetto a quello che lo precede in ordine di k-mer, il dizionario *DS40_top81*. Tralasciando tale dizionario, si verifica una aumento del valore di F-measure dal +0,52% del dizionario *DS40_top20* al +0,90% del dizionario *DS40_top81*. Si noti che, a parità di k-mer, il dizionario *DS40_top58*, migliora le prestazioni rispetto al dizionario ufficiale di *Quartz*, sia in termini di Precision (da -0,89% a -0,76%), di Recall (da 1,99% a 2,00%) e di F-measure (da +0,78% a +0,85%).

In termini di valore assoluto, confrontando i due gruppi di dizionari ottenuti dai due differenti dataset, a parità di k-mer contenuti, i dizionari realizzati dal dataset *DS40* hanno riscontrato valori di Precison più alti rispetto agli altri. I risultati sono raccolti nella tabella 3.4.

Questo risultato è imputabile al fatto che l'insieme di k-mer dal quale sono stati estratti, era più piccolo e quindi meno rappresentativo rispetto all'universo

	Precision		Recall		F-measure	
# kmer	**DS20**	**DS40**	**DS20**	**DS40**	**DS20**	**DS40**
876.645.275	0,9267	0,9272	0,6629	0,6625	0,7729	0,7728
1.753.290.549	0,9254	0,9261	0,6660	0,6653	0,7746	0,7743
2.629.935.824	0,9246	0,9255	0,6681	0,6672	0,7757	0,7754
3.506.581.098	0,9238	0,9253	0,6683	0,6678	0,7755	0,7757

Tabella 3.4: Confronto tra dizionari contenenti lo stesso numero di kmer.

reale di k-mer. Dal punto di vista della Recall, invece, i dizionari del gruppo DS20 hanno avuto valori di poco più elevati in tutti e quattro i casi. Infine, analizzando i valori di F-measure si nota una forte vicinanza fra i risulatati dei due gruppi, che quasi coincidono. Entrambi i due gruppi raggiungono il medesimo valore di punta, 0,7757, anche se il primo lo raggiunge con un dizionario contenente un numero minore di k-mer rispetto all'altro.

Nel complesso, i dizionari contenenti un numero di k-mer superiore ai 2,5 miliardi riescono a ottenere ottimi risultati in termini di accuratezza di genotipizzazione, migliorando anche rispetto al dizionario di Quartz.

3.2.2 Confronto sulla compressione

Una volta sottoposti al filtraggio mediante Quartz, i campioni genomici presi in considerazione durante la fase di test sono stati compressi. Una volta effettuata tale operazione su tutti i file, ne sono state raccolte le dimensioni. Nella tabella 3.5 sono presenti tutti i valori raccolti.

Dizionari	**# kmer**	**SRR622461_1**	**SRR622461_2**
No zip	-	21.304.030.772	21.304.030.772
zip	-	4.515.787.433	4.711.853.279
Quartz dict	2.497.777.248	3.034.015.640	3.118.836.272
DS20_top25	876.645.275	4.082.318.977	4.229.895.080
DS20_top50	1.753.290.549	3.667.682.022	3.784.898.637
DS20_top75	2.629.935.824	3.281.569.954	3.371.419.418
DS20_all	3.506.581.098	3.152.453.001	3.228.911.422
DS40_top20	876.645.275	4.102.657.033	4.252.861.099
DS40_top40	1.753.290.549	3.701.503.857	3.821.487.676
DS40_top58	2.497.777.248	3.393.492.350	3.489.937.039
DS40_top61	2.629.935.824	3.345.131.932	3.437.870.664
DS40_top81	3.506.581.098	3.187.784.419	3.264.408.034
DS40_all	4.331.088.631	3.267.466.744	3.340.023.106

Tabella 3.5: Confronto della dimensione dei file compressi con GZIP dopo il processo di smooting di *quartz* e i diversi dizionari.

Da tali dati si osserva che all'aumentare del numero di k-mer contenuti nei dizionari, aumenta anche l'efficienza di compressione producendo file di minore dimensione. Questa tendenza è presente sia con i dizionari del dataset *DS20* sia con quelli del dataset *DS40*, escluso il valori relativi al dizionario *DS40_all*, che seppur contenendo più k-mer del dizionario *DS40_top81*, ha una efficacia di compressione minore. Tale fenomeno, come ipotizzato in precedenza, è imputabile alla presenza di un numero troppo elevato di k-mer, fra i quali anche non discriminativi. I dizionari *DS20_all* e *DS40_top81*, contenenti entrambi lo stesso numero di k-mer, riescono a ottenere valori di compressione vicini a quello ufficiale di Quartz, anche se non riescono ad eguagliarli o a migliorarli.

Si è voluto, inoltre, mettere a confronto l'efficacia di compressione dei dizionari utilizzando un altro software di compressione oltre a GZIP, ovvero BZIP2. I campioni genomici filtrati con i dizionari realizzati dal dataset *DS40* sono stati, quindi, compressi anche con quest'ultimo software. I risultati ottenuti sono riportati nella tabella 3.6 e illustrati nel grafico 3.2.

Dizionari	**# kmer**	**SRR622461_1**		**SRR622461_2**	
		bzip2	**gzip**	**bzip2**	**gzip**
No zip	-	21.304.030.772	21.304.030.772	21.304.030.772	21.304.030.772
zip	-	4.053.514.929	4.515.787.433	4.216.436.538	4.711.853.279
Quartz dict	2.497.777.248	2.869.282.478	3.034.015.640	2.928.463.658	3.118.836.272
DS40_top20	876.645.275	3.706.380.123	4.102.657.033	3.823.487.225	4.252.861.099
DS40_top40	1.753.290.549	3.379.079.227	3.701.503.857	3.470.454.325	3.821.487.676
DS40_top58	2.497.777.248	3.139.735.427	3.393.492.350	3.211.045.748	3.489.937.039
DS40_top61	2.629.935.824	3.103.185.474	3.345.131.932	3.171.333.843	3.437.870.664
DS40_top81	3.506.581.098	2.984.673.970	3.187.784.419	3.039.500.462	3.264.408.034
DS40_all	4.331.088.631	3.038.979.251	3.267.466.744	3.091.846.020	3.340.023.106

Tabella 3.6: Tabella di confronto delle dimensioni dei file genomici campione compressi mediante i compressori GZIP e BZIP2.

Dai valori in tabella si può notare come attraverso l'utilizzo di BZIP2, si riescono a ottenere file compressi di minore dimensione, mediamente del 10%. E' perciò da preferire l'utilizzo di questo software di compressione per la compressione dei file genomici dopo aver operato il filtraggio dei *quality score*.

Con la variante apportata a Quartz in questo progetto, i file dei dizionari possono essere anch'essi compressi per risparmiare spazio di archiviazione ed essere utilizzati direttamente dal software. Per completezza, sono state effettuate delle analisi anche sulla comprimibilità dei file dei dizionari stessi. I file dei dizionari sono stati compressi utilizzando i software già utilizzati nei precedenti test, ovvero, GZIP e BZIP2. Le dimensioni dei file compressi sono riportate nella tabella 3.7. Si ricordi che tali file, durante l'ultima fase della pipeline di creazione dei dizionari presentata, vengono codificati in binario.

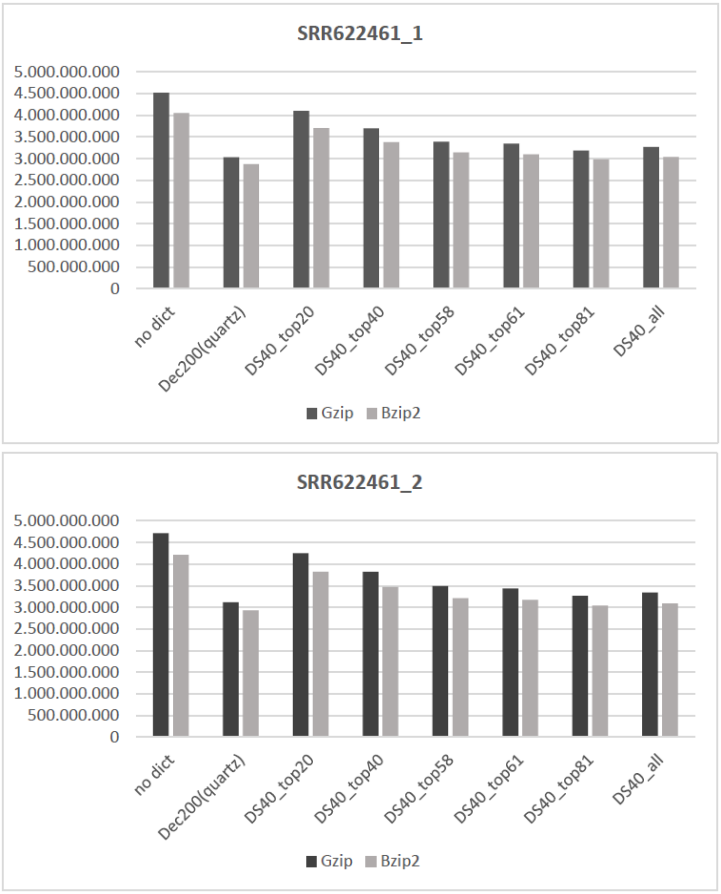


Figura 3.2: Grafici sul confronto tra la compressione dei due file genomici campione *SRR622461* mediante i compressori Gzip e Bzip2.

Dizionari	# kmer	File size	Gzip	Ratio	Bzip2	Ratio
Quartz dict	2.497.777.248	19.982.219.432	12.306.290.378	61,6%	15.990.659.875	80,0%
DS20_top25	876.645.275	7.013.162.208	4.337.990.875	61,9%	5.534.754.686	78,9%
DS20_top50	1.753.290.549	14.026.324.400	8.559.546.462	61,0%	11.018.921.102	78,6%
DS20_top75	2.629.935.824	21.039.486.600	12.430.915.551	59,1%	16.284.729.369	77,4%
DS20_all	3.506.581.098	28.052.648.792	15.524.163.968	55,3%	20.843.924.500	74,3%
DS40_top20	876.645.275	7.013.162.208	4.264.352.966	60,8%	5.474.476.189	78,1%
DS40_top40	1.753.290.549	14.026.324.400	8.466.949.121	60,4%	10.934.781.919	78,0%
DS40_top58	2.497.777.248	19.982.217.992	11.752.260.700	58,8%	15.387.777.144	77,0%
DS40_top61	2.629.935.824	21.039.486.600	12.300.738.315	58,5%	16.152.555.716	76,8%
DS40_top81	3.506.581.098	28.052.648.792	15.521.690.908	55,3%	20.841.848.007	74,3%
DS40_all	4.331.088.631	34.648.709.056	18.067.598.574	52,1%	24.762.645.382	71,5%

Tabella 3.7: Confronto della dimensione dei file dei dizionari compressi con Gzip e Bzip2.

Come si può vedere dalla tabella, i file dei dizionari risultano molto comprimibili e, dopo la compressione, risultano avere dimensioni molto ridotte. Dai dati

riportati, si evince che la compressione risulta più efficace quanto maggiore è la dimensione originale del file (e di conseguenza anche il numero di k-mer contenuti), e risulta essere più efficiente se viene eseguita con GZIP. Infatti, i file compressi nel formato *.gz* risultano più ridotti che nel formato *.bzip2* all'incirca del 20%. Si può notare, infatti, come i file dei dizionari compressi con GZIP abbiano dimensioni che varino circa dal 52% al 61% dei file originali mentre quelli compressi con BZIP2 varino dal 71% al 80% circa. Il grafico 3.3 rende ancora più evidente la differenza in termini di dimensione dei file fra le due tipologie di compressione prese in esame.

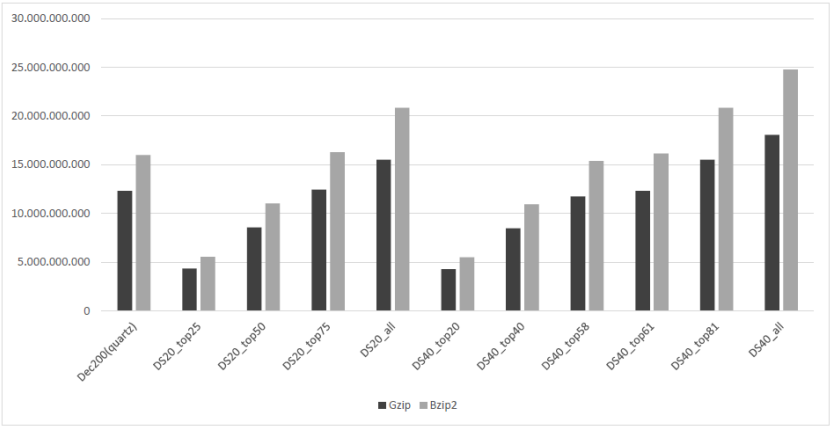


Figura 3.3: Grafico rappresentante la dimensione dei file dei dizionari compressi mediante GZIP e BZIP2

Questo comportamento sembra in contrasto con quanto visto in precedenza, dove sembrava essere il compressore BZIP2 avere prestazioni migliori. In verità, con un'analisi più attenta, si può osservare che i file di tipo alfanumerico in formato ASCII, come quelli dei campioni genomici, siano più comprimibili con BZIP2. Quelli in formato binario, come i file dei dizionari, sono invece più comprimibili con GZIP.

Conclusioni

I compressori a perdita di informazione si sono rivelati una soluzione alle recenti problematiche legate alla gestione, immagazzinazione e trasmissione della grande mole di dati prodotti dai nuovi sistemi di sequenziamento NGS. In particolare, in questo lavoro di tesi, è stata analizzata una particolare metodologia di compressione a perdita di informazione che fa uso di un dizionario di k-mer per ridurre l'aleatorietà dei file genomici. E' stato, quindi, utilizzato il software *Quartz* ponendo grande attenzione alla creazione del dizionario di k-mer proponendo un processo efficiente per la sua costruzione.

Dopo aver illustrato la natura dei dati genomici e le principali fasi del sequenziamento, nel primo capitolo sono state presentate le principali problematiche dovute alla natura di questi dati e alcune soluzioni proposte recentemente in letteratura, focalizzando l'attenzione sul software *Quartz*.

Nel secondo capitolo è stata illustrata una pipeline per la creazione efficiente di dizionari di k-mer a partire da una dataset di campioni genomici. Il punto centrale dell'intera pipeline è il calcolo dello score di ogni k-mer in cui si è adottato un modello statistico basato sulle catene di Markov. Ogni k-mer è visto come il prodotto di un processo aleatorio in cui ogni base nucleotidica che lo compone viene modellata come una variabile aleatoria, la cui probabilità dipende solo dal valore della variabile che la precede, in accordo con la teoria delle catene di Markov. Con il processo illustrato sono stati prodotti dieci dizionari, quattro da un dataset di 10 coppie di campioni genomici e sei da un dataset di 20 coppie di campioni genomici.

Questi dizionari sono stati oggetto di test ed accurate analisi, i cui risultati sono illustrati nel terzo capitolo di questa tesi. In quest'ultimo capitolo è stata definita interamente una pipeline software per la valutazione degli effetti distorsivi causati dal processo di compressione a perdita di informazione operato da Quartz. Tali effetti sono stati analizzati valutando l'accuratezza di genotipizzazione mediante le metriche *Precision*, *Recall* e *F-measure* in confronto a una ground truth validata. E' emerso che i dizionari realizzati, soprattutto quelli contenenti un numero di k-mer pari o superiore a circa 2,5 miliardi, riescono a ottenere ottimi risultati migliorando anche rispetto al dizionario di *Quartz*. Per quanto riguarda la comprimibilità dei file dopo aver subito il processo di filtraggio, i test sulla compressione hanno mostrato come tali dizionari garantiscono anche una buona compressione dei file genomici, di poco meno efficiente rispetto a quella operata mediante il dizionario di Quartz.

In conclusione, la metodologia di creazione dei dizionari proposta risulta ef-

ficiente e vantaggiosa, in quanto, permette di realizzare dizionari di k-mer con buone capacità discriminative partendo da un dataset modesto di campioni genomici. Abbinati al software Quartz, consentono di raggiungere un buon livello di compressione e di accuratezza di genotipizzazione paragonabili a quelle ottenute con l'utilizzo del dizionario ufficiale di Quartz, realizzato a partire da un dataset di dimensioni molto elevate e usufruendo di risolse di calcolo molto maggiori a quelle utilizzate in questo progetto.

Bibliografia

[1] S. Barocci, "Dal sequenziamento del DNA alla scoperta della reazione polimerasica a catena", http://www.chimicare.org/blog/metodi-e-approcci/dal-sequenziamento-del-dna-alla-scoperta-della-reazione-polimerasica-a-catena/

[2] National Human Genome Reserch Institute, *"The Cost of Sequencing a Human Genome*", https://www.genome.gov/sequencingcosts/, 2016.

[3] Heidi Chial, *DNA Sequencing Technologies Key to the Human Genome Project,* http://www.nature.com/scitable/nated/topicpage/dna-sequencing-technologies-key-to-the-human-828*,* Nature*,* 2008.

[4] Breda Genetics, "Next Generation Sequencing", https://bredagenetics.com/ngs-explained/?lang=it.

[5] Idoia Ochoa, *et al*, *Effect of lossy compression of quality scores on variant calling*, Stanford University, 2015.

[6] Illumina, "Reducing Whole-Genome Data Storage Footprint", https://www.illumina.com/documents/products/whitepapers/whitepaper_datacompression.pdf.

[7] R. Cánovas *et al.*, Lossy compression of quality scores in genomic data. Bioinformatics, 2014.

[8] R. Cánovas, Practical compression for multi-alignment genomic files. The University of Melbourne, 2015.

[9] G. Malysa *et al.*, QVZ: lossy compression of quality values. Bioinformatics, 2014.

[10] D. Greenfield *et al.*, GeneCodeq: quality score compression and improved genotyping using a Bayesian framework. Bioinformatics, 2016.

[11] Y. W. Yu *et al.*,Quality score compression improves genotyping accuracy. Nature Biotechnology. 2015.