Relazione del progetto di BIOINFORMATICA

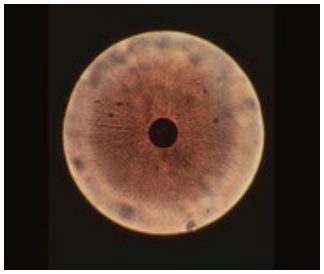
Data: 2 Aprile 2014

Studente: Daniele Ronzani

Docente: Prof. Giorgio Valle

Introduzione

In questa relazione verra fornita la documentazione relativa allo svolgimento del progetto di Bioinformatica. Il progetto pratico che viene proposto è il resequencing. Lo scopo di questa attività è acquisire particolari informazioni rilevanti studiando l'allineamento di una sequenza genetica di un campione di Acholeplasma laidlawii, con un riferimento genomico.



Lo studio riguarda principalmente le variazioni strutturali, come le insertion, le deletion e le inversion, rilevate grazie a caratteristiche particolari possedute dai mate-pair sequenziati.

E' stato messo a disposizione agli studenti un piccolo genoma di batterio di 1.496.992 basi, dato come riferimento, e un file *.sam* (*.bam*) contenente i risultati dell'allineamento di un campione di un batterio simile usando i read mate-pair.
I risultati che sono stati raggiunti dallo svolgimento di questo lavoro sono:

- calcolare la media e la deviazione standard della lunghezza degli inserti genomici.
- creare una traccia wig con la physical coverage.
- creare una traccia wig con la sequence coverage.
- creare una traccia wig con la copertura dei read multipli, ovvero che mappano in più di una posizione nel genoma.
- creare una traccia wig con la copertura dei singoli mate, ovvero quelli per cui il loro mate non mappa sul genoma.
- creare una traccia wig con la copertura dei mate-pair con direzioni uguali, quindi errate.

Il programma che implementa le funzioni che producono questi risultati è stato creato utilizzando il linguaggio C++, poichè la ormai nota efficienza ed efficacia, hanno permesso di ottenere una velocità di esecuzione più che accettabile.

Analisi del problema iniziale

Prima di poter procedere all'implementazione del programma, è stato necessario acquisire alcune conoscenze preliminari che riguardano la struttura dei file con estensione *".sam"*. Infatti occorreva conoscere le esatte posizioni dei vari campi contenuti in ogni riga del file, compreso l'header iniziale. Dopo aver interpretato correttamente tutte le informazioni rilevanti contenute nel file, è stato possibile cominciare a progettare la struttura principale del programma.
La cosa iniziale da fare è leggere tutte le righe del file, e memorizzarle in una struttura dati apposita, suddividendo tutti gli elementi necessari in variabili opportune. Successivamente, da questa struttura dati vengono prese le informazioni utili ad una certa funzione: ad esempio per

studiare le lunghezze degli inserti genomici verrà preso solo il campo relativo alla lunghezza di ogni singolo inserto.
Per quanto riguarda l'output del programma è stato scelto di usare lo Standard Output del terminale per presentare i risultati numerici (media, deviazione standard, dati quantitativi), e, come anche richiesto dalla traccia del problema, produrre dei file in formato *".wig"* contenenti le tracce che presentano dati quantitativi per ogni base. Questi file possono poi essere caricati in un browser genomico (per esempio IGV) per visualizzare graficamente i risultati ottenuti.

Implementazione del programma

Il programma è stato scritto in C++, utilizzando tre file principali: `main.cpp, utility.h, bio.h`.
Verranno prima illustrate le funzioni dei due file minori `utility.h` e `bio.h`, poi verrà discusso più ampiamente il file `main.cpp` che contiene il cuore delle funzionalità del programma.

`utility.h`

Questo file contiene tre funzioni utili all'esecuzione di alcune operazioni preliminari del programma. Particolarmente rilevanti sono le funzioni `void wig(vector<int> array)` e `void wig(vector<float> array)` che svolgono una operazione di scrittura dei risultati contenuti in un array in un file *.wig*. Si tratta semplicemente di copiare ogni elemento nell'array in una riga del file, inserendo inizialmente l'header opportuno alla prima riga.

`bio.h`

Questo file contiene la definizione della classe `SamLine`, che incorpora tutte le informazioni contenute in una riga del file *.sam.* Di seguito verranno presentate le variabili e le funzioni progettate.

Variabili

- **`name`**: contiene il *QNAME* del mate, ovvero il nome di identificazione (a seconda di che tipo di file sam viene analizzato, questo campo può essere uguale o diverso: questo particolare verrà discusso più avanti).
- **`num`**: contiene il numero (1 o 2) del mate nella coppia. Quando due read sono accoppiati, alcuni file sam distinguono i due read con due numeri: 1 e 2.
- **`flag`**: contiene il numero *FLAG* che rappresenta una stringa di 12 bit. Ogni bit descrive una particolare informazione, come l'orientamento del read o la sua mappatura. Per questo, viene definita una funzione che trasforma il numero decimale in un array di 12 interi, che possono assumere solo i valori 0 e 1.
- **`chromo`**: contiene il campo *RNAME* (Reference sequence NAME). Nel nostro problema sarà sempre *"Chromosome"* se il read è mappato, "*" altrimenti.
- **`pos`**: contiene il campo *POS* che rappresenta la posizione iniziale (la prima base) del read mappato sul riferimento.
- **`mapq`**: contiene il campo *MAPQ* che rappresenta la qualità.

- **`cigar`**: contiene il campo *CIGAR*, che rappresenta il numero e la posizione delle basi che fanno match su genoma di riferimento. Questa informazione è utile per conoscere la presenza di piccoli insertion e deletion.
- **`mrnm`**: contiene il campo *RNAME* del mate (la sua controparte). Se i due mate hanno nome uguale, viene scritto semplicemente il simbolo "=".
- **`mpos`**: contiene il campo *MPOS*, ovvero la posizione del mate.
- **`size`**: contiene il campo *TLEN*, ovvero la distanza tra i due mate o, visto in ottica diversa, la lunghezza dell'inserto genomico. Questo numero è stato usato per calcolare la media e la deviazione standard della lunghezza degli inserti.

Altre variabili che sono risultati vitali per il progetto sono:
- **`lenght`**: contiene la lunghezza reale del read mappato sul riferimento: viene calcolato sulla base dei match effettivi rilevati dal campo *CIGAR* di ogni read.
- **`vector<int> match`**: è un vettore contenente numeri che possono assumere 0 o 1. Questo vettore è utile soprattutto in fase di calcolo della copertura dei read sul riferimento, in quanto le basi che non sono matchate nel riferimento, di fatto non vengono conteggiate. Questo vettore viene calcolato sempre sulla base delle informazioni contenute nel campo *CIGAR*.
- **`int vFlag[12]`**: array di interi che contiene lo stato dei bit del campo *FLAG*.

Funzioni
Sono state implementate 4 funzioni per questa classe:
- **`bool valid()`**: verifica che una riga del file sia valida, cioè che un read sequenziato sia considerato valido per essere studiato; se non è valido, significa che non è mappato nel genoma.
- **`void readInfo(string l)`**: questa funzione legge una stringa (che contiene una riga del file *".sam"*) suddividendo i vari campi e recuperando tutte le informazioni necessarie. Ad ogni variabile descritta precedentemente viene assegnato il valore associato.
- **`void flagToBinaryVector()`**: inserisce nell'array `vFlag` tutti i valori dei bit rappresentati dal valore del parametro F*LAG*.
- **`int cigarValues()`**: svolge un processo di parsing della stringa *CIGAR* dividendo quindi le tracce della sequenza le cui basi sono matchate (match, M), quelle le cui basi sono aggiunte (inserts, I) e quelle le cui basi sono invece cancellate (deletions, D). Con questa funzione è quindi possibile anche riconoscere le *SNP* (Single Nucleotide Polimorfism), ovvero variazioni della sequenza che coinvolgono una base.

`main.cpp`
Il programma principale è contenuto nel file `main.cpp`. Sono presenti alcune importanti variabili che vengono di seguito descritte. Inoltre verranno illustrate le principali funzioni che calcolano i risultati richiesti dal problema.

Variabili

- **`int ln`**: intero che rappresenta la lunghezza del riferimento genomico; viene letto dall'header del file *.sam* e, in questo caso, conterrà 1.496.992, il numero di basi del batterio che si sta studiando.
- **`int mappedReads`**: variabile che contiene il numero di reads sequenziate che sono mappate nel genoma di riferimento.
- **`int notMappedReads`**: variabile che contiene il numero di reads sequenziate che non mappano nel genoma di riferimento.
- **`float mean`**: variabile che rappresenta la media della lunghezza di tutti gli inserti genomici presenti nel file *.sam*.
- **`float sd`**: variabile che rappresenta la deviazione standard della lunghezza degli inserti genomici.
- **`vector<SamLine> sam`**: vettore che contiene tutti gli elementi di tipo `SamLine` che rappresentano ogni read (e le sue caratteristiche) nel file *.sam*.

Funzioni

- **`void loadData(const char *path, int ValueOutlier)`**: questa funzione è la prima ad essere invocata e carica in un vettore tutti i campi presenti in ogni riga del file .sam. Inoltre, vengono calcolati contemporaneamente anche il numero dei read mappati, quelli non mappati e la media delle lunghezze. Per distinguere il numero di read mappati da quelli non mappati viene utilizzata la funzione `valid()`.
- **`void st_dev()`**: calcola la deviazione standard della lunghezza degli inserti genomici sulla base della media calcolata.

Le prossime funzioni calcolano una caratteristica particolare del sequenziamento, in riferimento al genoma dato, e tutte memorizzano i loro risultati su un file *.wig* differente. Questo file viene impostato con l'header **`fixedStep chrom=Chromosome start=1 step=1`**, il che significa che si ha un passo fisso, mappato sul cromosoma *"Chromosome"* (l'unico), partendo dalla base in posizione 1, con passo 1. Quindi i risultati vengono posti con una risoluzione di singola base.

- **`void physicalCov(const char * path)`**: calcola la copertura fisica degli inserti genomici, ovvero per ogni base viene contato il numero di volte che un read ci passa sopra. Questo conteggio tiene anche conto della reale copertura dei read sul riferimento, indicato dal campo *CIGAR*: per esempio, se una base del riferimento non è coperta dal read, questa non viene considerata.
- **`void sequenceCoverage(const char * path)`**: calcola la copertura tenendo conto della lunghezza totale degli inserti, ovvero conteggia anche le basi presenti in mezzo, tra i due mate che formano la coppia (quelle che non sono mappate direttamente dai reads). Oltre a questo, la funzione calcola anche la lunghezza media degli inserti che passano sopra una base.

Errata Corrige: per queste funzioni è descritto correttamente ciò che fanno, ma i nomi delle funzioni sono errate: in realtà la physical coverage è calcolata dalla funzione sequenceCoverage e viceversa.

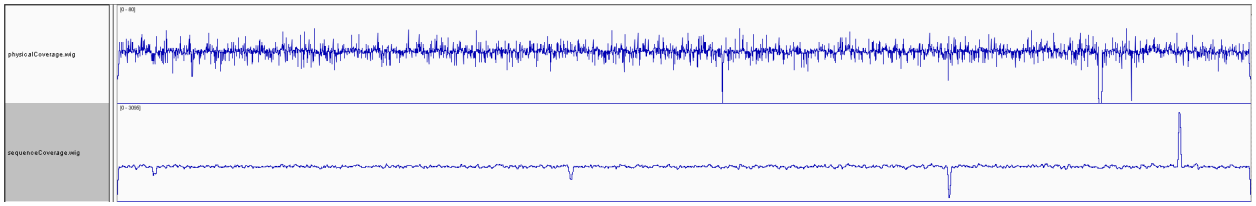
- **`void multiCoverage(const char * path)`**: questa funzione considera tutti quei read che mappano in più posizioni genomiche. Quindi viene memorizzata sul file la copertura di tali frammenti di sequenza. Per far ciò, viene utilizzato un altro file, differente da *"illumina.sam"*, che contiene tutti i read ordinati per nome; in questo modo, tramite un ciclo, vengono prese tutte le coppie di read adiacenti l'una all'altra, controllando se due di queste contengono la stessa stringa *NAME*: se è così, significa che lo stesso read mappa due volte nel genoma, nelle posizioni indicate nel campo *POS.*
- **`void wrongMate(const char * path)`**: analizza l'orientamento dei mate pair e verifica che due mate abbiano direzione opposta (giusta). Se due read possiedono direzioni uguali, significa che hanno direzioni errate (significa cioè che c'è stata una inversione nel campione), e viene quindi conteggiata la copertura di questi reads.
- **`void singleMate(const char * path)`**: l'ultima funzione calcola la copertura dei mate singoli, ovvero quelli il cui mate di appartenenza non è mappato. Questo viene controllato trammite il campo *FLAG*, che contiene un bit (bit 3) che indica se il mate del read è mappato o no.

Risultati ottenuti

Verranno ora presentati i risultati ottenuti dalle tracce create da ogni funzione descritta sopra. In particolare si farà riferimento all'analisi delle **Structural Variations** (SV): insertions, deletions, e inversions, che sono, a volte, identificate da più funzioni.

Physical Coverage e Sequence Coverage

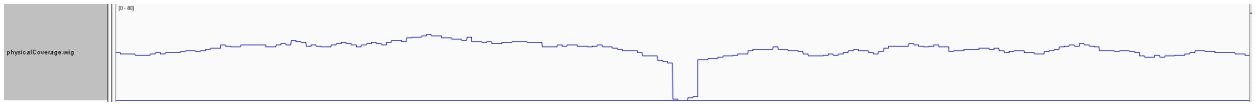
Come si può osservare nella figura sottostante esistono diverse sezioni del genoma che non sono coperte dai frammenti sequenziati. Questo significa che sono presenti delle **deletion** nel campione da cui è stato eseguito il sequenziamento, il che spiega il non mappamento di alcune reads nel genoma di riferimento. Nella figura, osservando la prima traccia, si possono notare tre posizioni in cui le basi non sono mappate, ma effettuando uno zoom in, esistono, in realtà, molte più sezioni in cui non c'è mappamento. Quest'ultima osservazione è stata ottenuta grazie all'analisi del campo *CIGAR* che indica la delezione a livello di singole basi.



Tracce contenenti la physical coverage e la sequence coverage

La seconda traccia offre una panoramica della copertura degli inserti genomici. Come si può notare sono presenti alcune parti del genoma che non vengono coperte, questo perchè non vengono considerati i read unici, che non possiedono la loro controparte, perciò in questi casi

non viene fatto nessun calcolo.



Frammento di traccia che presenta una deletion di 4 basi

Mappamento multiplo

La traccia seguente presenta il risultato ottenuto dall'analisi dei read che mappano in più posizioni genomiche.

Traccia con la copertura dei mate che mappano in più posizioni

Si possono osservare due loci in cui si ha una grande concentrazione di read multipli. Ciò significa che molto probabilmente quelle due zone contengono la stessa sequenza di basi.

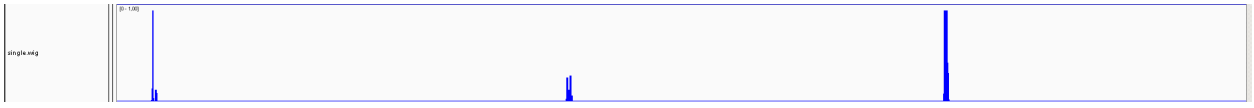
Orientamento

La traccia che rappresenta l'orientamento dei mate viene calcolata in termini di mate pair orientati in modo errato. Questa analisi è indispensabile se si vogliono trovare delle **inversions**. Nella traccia che segue si possono identificare 6 zone in cui si hanno mate con direzione uguale. **Errore**: la traccia non tiene conto dei mate singoli, che vengono presi in considerazione e vengono erroneamente considerati con orientamento sbagliato. Quindi togliendo i picchi associati a questi mate, si ottengono 3 posizioni nel genoma che sono correttamente attribuibili a delle inversioni.

Traccia con la copertura delle coppie di read che hanno direzioni uguali

Mate singoli

L'analisi dei mate singoli permette di identificare le **insertion**, che si presentano quando, appunto, un read viene mappato e l'altro no. Infatti quando sono presenti delle parti di genoma nel campione che non ci sono nel genoma di riferimento, i read che mappano in quelle zone non possono essere mappate nel riferimento. La traccia che segue mostra la copertura dei read le cui controparti non sono mappate.



Traccia con la copertura dei mate singoli

Si può notare che sono 3 i punti in cui si hanno mate singoli: perciò è possibile dedurre che, con molta probabilità, in quelle zone sono presenti dei frammenti che nel riferimento non c'erano.