Esercitazione: Allineamento di una Sequenza Sconosciuta con i due applicativi (blastx e diamond-blastx)

Supponiamo di aver isolato la seguente sequenza di DNA da un campione ambientale e di voler scoprire a quale organismo appartiene o se ha delle similitudini con altre sequenze note:

>seq1
AATTTTAGTAGTTTGCTGGTGTTCTCTCTTTTGTTTCTTATGTTTGGTTATCTTCTTTTTA
GAAATTTTTATTTGAGTAAGAAGATTGAATATCAGTGGTGAACTTTTATGTAGTATTTTCC
TTTAATTTTGGTTGCTGCCTTCTTTGAGTTTACTTTATTATTATGGTTTGATGAATCTTGAT
AGTAATTTGTCAAAGTTACTGGTCATCAGTGGTACTTTACGAGTTTAGTGATATTCCGGG
TAGAATTTGATTCTAAGTCTGTGGATCAGTTGGAGTTAGGTGAGCCTCGTTTGGAGGTT
TAATCGTTGTGTTGTTCCTTGTGATATTAACCGTTTTTGTATGGATGTTATTCATTCTTGT
TGGGCTTTGCCTAGCTATTAAGTTGGATGAGGGGTATTTTGTCTACTGTTTCTTATAGTT
TTCCTACTGTTGGTGATGGTCAATGTTCAGAGATTTGTGGGGCTATAGTTTTATGCCTAT
TGCTCTGGAAGTGTTATTG

Generico lancio in ambiente linux dell'applicativo blastx di NCBI da riga di comando

/bioinf/NCBI_BLAST_2_12_0/bin/blastx -query input.fasta -db database_name -out results.out

-evalue 1e-5 -max_target_seqs 500 -outfmt 6 -num_threads 4

Dove:

- ***input.fasta*** è il file che contiene le sequenze di nucleotidi in formato FASTA che si desidera confrontare con il database delle proteine.

- ***database_name*** è il nome del database di proteine contro cui eseguire il BLAST.

  Questo potrebbe essere uno dei database pre-costruiti disponibili attraverso il NCBI o un database personalizzato.

- ***results.out*** è il file di output dove verranno salvati i risultati.

Generico lancio in ambiente linux dell'applicativo blastx di NCBI da riga di comando

/bioinf/NCBI_BLAST_2_12_0/bin/blastx -query input.fasta -db database_name -out results.out

-evalue 1e-5 -max_target_seqs 500 -outfmt 6 -num_threads 4

Ci sono inoltre molti parametri opzionali per personalizzare l'esecuzione di BLASTx. Alcuni dei più comuni includono:

- ***-evalue*** per specificare il valore E massimo per gli allineamenti riportati. Un valore E più basso indica una corrispondenza più significativa.

- ***-max_target_seqs*** per limitare il numero di sequenze di destinazione riportate.

- ***-outfmt*** per specificare il formato dell'output. Ad esempio***, -outfmt*** **6** produce un output in formato tabulato, che è utile per l'analisi dei dati.

- ***-num_threads*** per specificare il numero di thread da utilizzare, che può accelerare l'esecuzione su macchine multi-core.

Lancio in ambiente linux dell'applicativo blastx di NCBI da riga di comando con seq1.fa come sequenza di input e SwissProt come database target

Per creare un database di sequenze che possa essere utilizzato con BLAST, si può utilizzare il comando **`makeblastdb`** che si trova tra gli eseguibili BLAST (sotto-cartella bin). Questo comando converte un insieme di sequenze in un formato ottimizzato per la ricerca, permettendo a BLAST di eseguire le query in modo molto più efficiente.

Nel caso nostro il database da indicizzare è uniprot_sprot.fasta:

/bioinf/NCBI_BLAST_2_12_0/bin/makeblastdb -in /data/DATABASES/SP/uniprot_sprot.fasta -dbtype prot -out sp

Come risultato si creano i seguenti file di indice (evidenziati in viola):

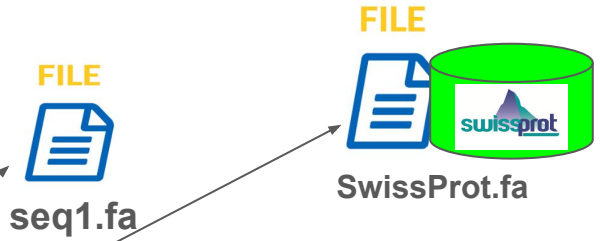
castri@raganella:/data/ESERCITAZIONI/9_4_2024$ ls

seq1.fa **sp.pdb** **sp.phr** **sp.pin** **sp.pot** **sp.psq** **sp.ptf** **sp.pto**

Lancio in ambiente linux dell'applicativo blastx di NCBI da riga di comando con seq1.fa come sequenza di input e SwissProt come database target

Per lanciare il comando blast ora abbiamo tutto quel che serve:

1) il file della sequenza query
2) il database indicizzato



/bioinf/NCBI_BLAST_2_12_0/bin/blastx -query seq1.fa -db sp -out results.out -outfmt 6 -num_threads 4

```
castri@raganella:/data/ESERCITAZIONI/9_4_2024$ cat results.out
seq1    sp|P24882|COX2_ASCSU    46.471  170     58      12      40      468     42      205     9.72e-18        80.5
seq1    sp|P24882|COX2_ASCSU    75.862  29      2       3       305     379     143     170     0.001   39.7
seq1    sp|P24882|COX2_ASCSU    69.231  13      4       0       465     503     206     218     0.001   22.3
seq1    sp|Q8SEM9|COX2_CAERE    44.118  170     62      12      40      468     41      204     2.29e-16        76.6
seq1    sp|Q8SEM9|COX2_CAERE    65.517  29      5       3       305     379     142     169     0.046   33.5
seq1    sp|Q8SEM9|COX2_CAERE    76.923  13      3       0       465     503     205     217     0.046   23.1
seq1    sp|Q8HEC3|COX2_CAEBR    44.118  170     62      12      40      468     42      205     2.33e-16        76.6
seq1    sp|Q8HEC3|COX2_CAEBR    65.517  29      5       3       305     379     143     170     0.047   33.5
seq1    sp|Q8HEC3|COX2_CAEBR    76.923  13      3       0       465     503     206     218     0.047   23.1
seq1    sp|P24894|COX2_CAEEL    44.118  170     62      12      40      468     42      205     4.54e-16        75.9
seq1    sp|P24894|COX2_CAEEL    65.517  29      5       3       305     379     143     170     0.047   33.5
seq1    sp|P24894|COX2_CAEEL    76.923  13      3       0       465     503     206     218     0.047   23.1
seq1    sp|Q37706|COX2_ARTSF    30.357  112     52      5       193     468     97      202     0.35    32.7
```

Lancio in ambiente linux dell'applicativo blastx di NCBI da riga di comando con seq1.fa come sequenza di input e SwissProt come database target

Per vedere di quale proteina si tratti e quale organismo usare l'interfaccia di uniprot:

https://www.uniprot.org/id-mapping

Lancio in ambiente linux dell'applicativo blastx di NCBI da riga di comando con seq1.fa come sequenza di input e SwissProt come database target

Per vedere di quale proteina si tratti e quale organismo usare l'interfaccia di uniprot:

https://www.uniprot.org/id-mapping

5 IDs were mapped to **5** results
1 ID was not mapped:
seq1

From	Entry	Entry Name	Protein Names	Gene Names	Organism	Length
P24882	P24882	COX2_ASCSU	**Cytochrome c oxidase subunit 2**[...]	**COII**	**Ascaris suum (Pig roundworm) (Ascaris lumbricoides)**	232 AA
Q8SEM9	Q8SEM9	COX2_CAERE	**Cytochrome c oxidase subunit 2**[...]	**cox-2**, coII	**Caenorhabditis remanei (Caenorhabditis vulgaris)**	230 AA
Q8HEC3	Q8HEC3	COX2_CAEBR	**Cytochrome c oxidase subunit 2**[...]	**cox-2**, coII	**Caenorhabditis briggsae**	231 AA
P24894	P24894	COX2_CAEEL	**Cytochrome c oxidase subunit 2**[...]	**ctc-2**, coII, cox-2, MTCE.31	**Caenorhabditis elegans**	231 AA
Q37706	Q37706	COX2_ARTSF	**Cytochrome c oxidase subunit 2**[...]	**COII**, CO-II	**Artemia franciscana (Brine shrimp) (Artemia sanfranciscana)**	228 AA

Identificazione della sequenza mediante il software diamond da riga di comando in ambiente linux

- **downloading di diamond**

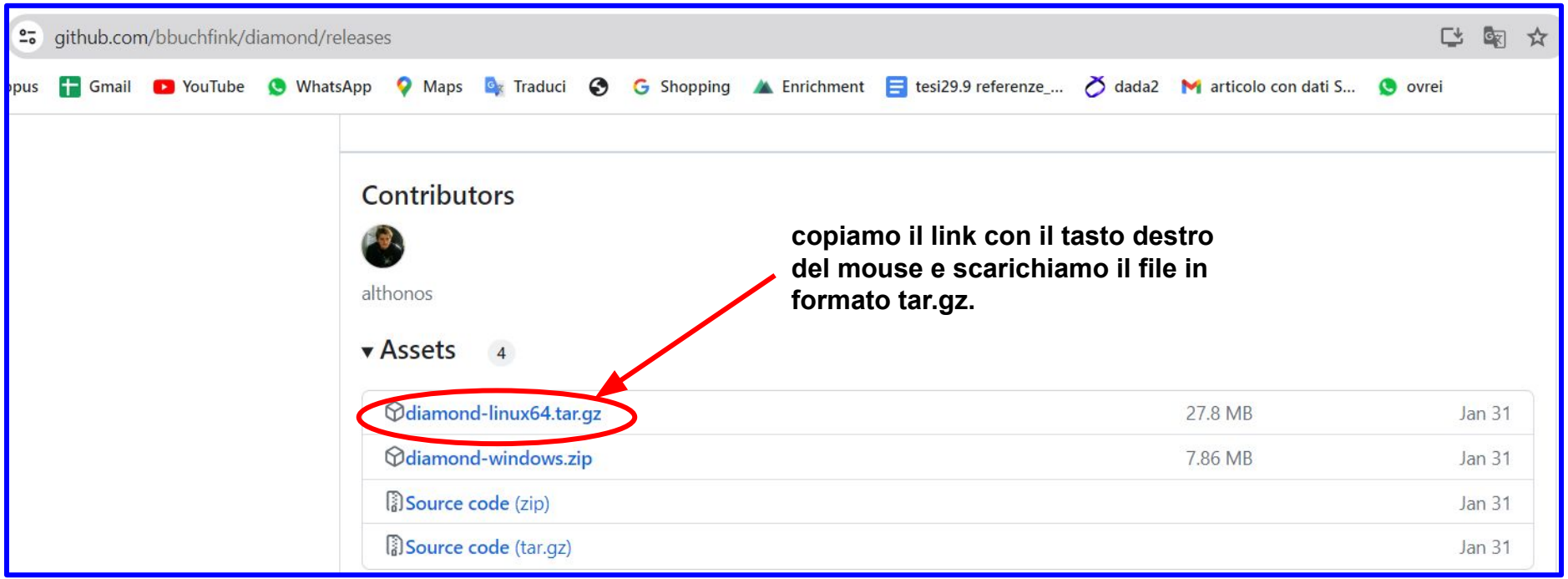
Colleghiamoci al link github delle releases di diamond:

https://github.com/bbuchfink/diamond/releases

GitHub è una piattaforma di hosting per il controllo di versione e la collaborazione tra sviluppatori, che permette loro di lavorare insieme a progetti software da qualsiasi parte del mondo. Fondato nel 2008, GitHub si basa sul sistema di controllo di versione distribuito Git, inventato da Linus Torvalds, il creatore di Linux. GitHub facilita la gestione di progetti software, consentendo agli utenti di caricare codice sorgente, collaborare con altri, tracciare e risolvere problemi, e implementare funzionalità di controllo di versione per tenere traccia e combinare modifiche al codice in un progetto condiviso.

Identificazione della sequenza mediante il software diamond da riga di comando in ambiente linux

- downloading di diamond



Identificazione della sequenza mediante il software diamond da riga di comando in ambiente linux

- estrazione della suite di diamond

Per estrarre un file con estensione `.tar.gz` su un sistema Linux o macOS, puoi usare il comando `tar` dalla linea di comando. Il comando completo per estrarre un file `.tar.gz` è:

tar xzvf diamond-linux64.tar.gz

- `x`: indica a `tar` di estrarre i file.
- `z`: dice a `tar` di decomprimere l'archivio (poiché `.gz` indica che è compresso con gzip).
- `v`: sta per "verbose", facoltativo, fa sì che `tar` mostri i nomi dei file mentre vengono estratti.
- `f`: permette di specificare il nome del file archivio.

Identificazione della sequenza mediante il software diamond da riga di comando in ambiente linux

- estrazione della suite di diamond

DIAMOND SI SCARICA UFFICILAMENTE CON

wget https://github.com/bbuchfink/diamond/archive/refs/heads/master.zip

L'ESSEGUIBILE DI DIAMOND LO TROVATE SUL SITO:

http://deb.scienceontheweb.net/BioInformatica1/lezioni/diamond.o

ED E' SCARICABILE SU LINUX CON IL COMANDO

wget https://deb.scienceontheweb.net/BioInformatica1/lezioni/diamond.o --no-check-certificate

UNA VOLTA SCARICATO RINOMINARE IL FILE CON IL COMANDO:

mv diamond.o diamond

ULTERIORE ALTERNATIVA LINUX MAC/OS

```
brew install diamond
```

SWISSPROT INDEXING

1) **Per prima cosa decomprimiamo il file fasta di SwissProt:**

gunzip uniprot_sprot.fasta.gz

2) una volta fatto il gunzip sul file uniprot_sprot usiamo la funzione makedb di diamond per creare l'indice della banca dati:

diamond makedb --in nomefile.fasta -d nomeDatabase

- `--in nomefile.fasta`: specifica il file (banca dati in formato fasta) di input. Sostituisci `nomefile.fasta` con il nome del tuo file FASTA contenente le sequenze di proteine che vuoi indicizzare.

- `-d nomeDatabase`: specifica il nome dell'indice del database da creare. Sostituisci `nomeDatabase` con il nome desiderato per il tuo database indice. Questo nome verrà utilizzato quando esegui ricerche utilizzando questo indice.

SWISSPROT INDEXING

Applichiamo il comando 2) della slide precedente per creare l'indice della banca dati uniprot_sprot:

diamond makedb --in uniprot_sprot.fasta -d sp

- `--in` ***uniprot_sprot.fasta***: e' la nostra banca dati di sequenze

  di aminoacidi in formato fasta che utilizziamo come input per l'indicizzazione.

- `-d` ***sp***: specifica il nome dell'indice del database (***uniprot_sprot.fasta)*** da creare.

  Questo nome verrà utilizzato quando eseguirete le ricerche su ***uniprot_sprot.fasta,*** utilizzando questo indice che avrà estensione dmnd (***sp.dmnd***).

Identificazione della sequenza mediante il software diamond da riga di comando in ambiente linux

IL COMANDO E'

/bioinf/diamond blastx -q seq1.fa -d sp.dmnd -o $path_file_output/file_output.tsv -p numeroprocessi –ultra-sentive 1> file_uscita.out 2>file_uscita.err

ESEMPIO (da scrivere su una unica riga senza andare a capo)

```
./diamond blastx -q transcripts.fasta.transdecoder.cds -d sp.dmnd
-o SP_blastx.tsv
-p 40
--ultra-sensitive
1>SP_blastx.out 2>SP_blastx.err
```

IL SIGNIFICATO DEI CAMPI ALLA SLIDE SUCCESSIVA

- `./diamond`: Questo indica che il comando `diamond` viene eseguito dalla directory corrente. Il `./` è utilizzato per eseguire un programma che si trova nella directory in cui ti trovi attualmente, anziché in una directory inclusa nel tuo `PATH` di sistema.
- `blastx`: Modalità di DIAMOND per l'allineamento di sequenze di nucleotidi contro un database di sequenze di proteine, effettuando la traduzione in tutte le sei frame di lettura.
- `-q fileQuery.fasta`: L'opzione `-q` specifica il file di query in formato FASTA. `fileQuery.fasta` è il file contenente le sequenze di nucleotidi da allineare.
- `-d sp.dmnd`: L'opzione `-d` specifica il database contro cui effettuare l'allineamento. `sp.dmnd` è il database pre-indicizzato di proteine, precedentemente creato con `diamond makedb`.
- `-o $path_file_output/file_output.tsv`: L'opzione `-o` specifica il nome del file di output. Qui viene utilizzata una variabile d'ambiente (`$path_file_output`) per indicare il percorso della directory di output, e `file_output.tsv` è il nome del file di output in cui verranno salvati i risultati dell'allineamento.

- ***-p numeroprocessi***: L'opzione ***-p*** specifica il numero di processori (o core) da utilizzare per l'allineamento. Sostituisci ***numeroprocessi*** con il numero effettivo di processori che vuoi che DIAMOND utilizzi.
- ***--ultra-sensitive***: Questa opzione aumenta la sensibilità dell'allineamento, rendendolo più accurato ma potenzialmente più lento. È utile per database molto grandi o per file query molto grandi.
- ***1> file_uscita.out***: Questa parte del comando reindirizza l'output standard (STDOUT) in un file chiamato ***file_uscita.out***. Di solito, questo include messaggi di stato e riassunti dell'esecuzione del comando.
- ***2>file_uscita.err***: Questa parte reindirizza lo standard error (STDERR) in un file chiamato ***file_uscita.err***. Questo file conterrà messaggi di errore, se ve ne sono stati durante l'esecuzione del comando.

TRANSDECODER

TRANSDECODER SI SCARICA UFFICILAMENTE CON

wget https://github.com/TransDecoder/TransDecoder/archive/refs/heads/master.zip

Poi si decomprime la cartella con:
unzip master.zip

Una volta decompresso si ha la cartella
TransDecoder-master

dove si trovano gli eseguibili

TransDecoder.LongOrfs
TransDecoder.Predict

TRANSDECODER

TransDecoder è uno strumento ampiamente utilizzato in bioinformatica per identificare le regioni codificanti proteine (ORF, Open Reading Frames) all'interno di trascritti di RNA più lunghi. È particolarmente utile nell'analisi di dati di trascrittoma, specialmente quando si lavora con organismi per cui non esiste un genoma di riferimento completo. Questo strumento può aiutare a identificare potenziali geni e le loro proteine prodotte all'interno di un insieme di dati di sequenziamento RNA-Seq, facilitando l'analisi funzionale e comparativa.

Di seguito, troverai i comandi di base per utilizzare i due eseguibili principali di TransDecoder: ***`TransDecoder.LongOrfs`*** per identificare i candidati ORF lunghi, e ***`TransDecoder.Predict`*** per predire gli ORF effettivi partendo dai candidati identificati.

TRANSDECODER SI SCARICA UFFICILAMENTE CON

wget *https://github.com/TransDecoder/TransDecoder/archive/refs/heads/master.zip*

Poi si decomprime la cartella con:

unzip master.zip

Una volta decompresso si ha la cartella

TransDecoder-master

L'algoritmo alla base di ***TransDecoder*** segue alcuni passaggi chiave per identificare le CDS:

Identificazione delle Regioni di Traduzione: *TransDecoder* inizia analizzando le sequenze di trascritti (RNA) per identificare le regioni che potrebbero essere tradotte in proteine. Questo processo implica la ricerca di regioni che iniziano con un codone di inizio (tipicamente ATG) e terminano con un codone di stop (TAA, TAG, o TGA), attraverso tutte le sei *frames* di lettura possibili (tre per ogni *strand*).

Predizione delle Sequenze Proteiche: Una volta identificate le regioni di traduzione, *TransDecoder* predice le sequenze di aminoacidi corrispondenti. Questo passaggio permette di estrapolare quali trascritti potrebbero effettivamente codificare per proteine.

Identificazione delle Sequenze Conservate: *TransDecoder* utilizza vari criteri per valutare quali tra le sequenze predette sono più probabilmente codificanti, inclusa la lunghezza della sequenza di aminoacidi e la presenza di domini proteici conservati. Questo è spesso fatto attraverso il confronto con database di domini proteici noti, come ***Pfam***.

Scelta delle Migliori CDS per Trascritto: Se un trascritto contiene multiple regioni potenzialmente codificanti, *TransDecoder* seleziona la regione più probabile che codifica per una proteina basata su vari fattori, come la lunghezza della sequenza proteica e la presenza di domini conservati.

TRANSDECODER

`TransDecoder.LongOrfs` è una componente del software *TransDecoder*, progettata per identificare gli Open Reading Frames (ORFs) più lunghi all'interno di trascritti di RNA. L'obiettivo principale di ***`TransDecoder.LongOrfs`*** è di selezionare le ORFs che hanno maggiori probabilità di rappresentare regioni codificanti effettive all'interno di un trascritto.

TransDecoder.LongOrfs -t file_transcript.fasta

Dove ***`-t file_transcript.fasta`*** specifica il file FASTA dei tuoi trascritti da analizzare. ***`TransDecoder.LongOrfs`*** cercherà in questo file le sequenze che sono più probabili di essere tradotte in proteine, basandosi sulla lunghezza e altri fattori.

TRANSDECODER

`TransDecoder.Predict` è la fase successiva nell'uso di *TransDecoder*, che segue dopo l'identificazione delle lunghe Open Reading Frames (ORFs) con ***`TransDecoder.LongOrfs`***. Mentre ***`TransDecoder.LongOrfs`*** si concentra sulla ricerca e selezione delle ORFs più lunghe che potrebbero rappresentare regioni codificanti proteine nei trascritti, ***`TransDecoder.Predict`*** affina ulteriormente questa ricerca per identificare quali di queste lunghe ORFs sono effettivamente probabili sequenze codificanti proteine, basandosi su caratteristiche biologiche e bioinformatiche.

Ecco i passaggi chiave e le caratteristiche dell'analisi effettuata da ***`TransDecoder.Predict`***:

Analisi di Conservazione e Motivi: ***`TransDecoder.Predict`*** esamina le lunghe ORFs identificate per trovare evidenze di conservazione di domini proteici o motivi funzionali. Questo può includere il confronto delle sequenze con database di domini proteici noti, come Pfam, per identificare regioni che corrispondono a domini proteici conservati.

Valutazione Statistica: Il software utilizza modelli statistici per valutare la probabilità che una data ORF codifichi effettivamente per una proteina. Questo può includere l'analisi della composizione degli aminoacidi, la lunghezza della ORF e la presenza di segnali di inizio e fine traduzione che sono tipici delle proteine funzionali.

TRANSDECODER

.

Valutazione Statistica: Il software utilizza modelli statistici per valutare la probabilità che una data ORF codifichi effettivamente per una proteina. Questo può includere l'analisi della composizione degli aminoacidi, la lunghezza della ORF e la presenza di segnali di inizio e fine traduzione che sono tipici delle proteine funzionali.

S**elezione delle ORFs Codificanti**: Basandosi sui criteri di analisi sopra menzionati, `TransDecoder.Predict` seleziona le ORFs che hanno le maggiori probabilità di essere sequenze codificanti proteine. Questo passaggio riduce il numero di falsi positivi, concentrandosi sulle ORFs che mostrano chiari segni di codificazione proteica.

Il risultato finale di ***`TransDecoder.Predict`*** è un set di file che descrivono le ORFs predette come codificanti proteine. Questi file includono sequenze di nucleotidi, sequenze di aminoacidi derivati e annotazioni relative ai domini proteici identificati. Queste informazioni possono essere utilizzate per ulteriori analisi funzionali, studi di espressione proteica o confronti con altre sequenze proteiche note.

TRANSDECODER

Il comando di TransDecoder.Predict è:

TransDecoder.Predict -t file_transcript.fasta

Confrontare il numero di sequenze di un trascrittoma con il numero di sequenze di ORFs predette. Dare una spiegazione del risultato.

Perché nella costruzione ed annotazione dei trascrittomi de-novo la *best practice* richiede di annotare gli ORFs predetti e non i trascritti direttamente?

L'analisi dei trascrittomi, soprattutto attraverso l'assembly de-novo, genera un vasto insieme di sequenze di RNA che possono variare significativamente in termini di funzione e struttura.

La predizione di Open Reading Frames (ORFs) e la successiva annotazione per omologia degli ORFs, anziché dei trascritti direttamente assemblati, è un approccio chiave per diverse ragioni scientifiche e tecniche. Vediamone alcune:

- **Specificità Funzionale:** I trascritti assemblati contengono sia regioni codificanti (che traducono in proteine) sia non codificanti. La predizione degli ORF permette di isolare specificamente le parti dei trascritti che hanno il potenziale di essere tradotte in proteine, facilitando l'identificazione della loro funzione biologica.

- **Efficienza:** Concentrandosi sugli ORFs, si riduce il rumore di fondo rappresentato dalle regioni non codificanti, migliorando l'efficienza computazionale e la qualità dell'annotazione funzionale.

- **Maggiore Precisione:** Annotare le sequenze basandosi sugli ORFs piuttosto che sull'intero trascritto aumenta la precisione dell'annotazione. Questo perché le sequenze proteiche conservano meglio le informazioni evolutive e funzionali attraverso le specie rispetto alle sequenze di RNA, rendendo più affidabile l'annotazione per omologia.

- **Splicing Alternativo:** Un singolo gene può produrre più trascritti attraverso lo splicing alternativo. Predire gli ORFs dai trascritti permette di identificare le diverse isoforme proteiche potenzialmente codificate dallo stesso trascritto, offrendo una visione più dettagliata della diversità funzionale all'interno del trascrittoma.

- **Qualità dell'Assemblaggio:** Gli assemblaggi de novo possono includere errori come fusioni errate di trascritti o inclusioni di sequenze non codificanti. Predire gli ORFs prima dell'annotazione aiuta a mitigare l'impatto di questi artefatti focalizzandosi sulle sequenze con un chiaro potenziale codificante.

- **Confronti Evolutivi:** Le proteine tendono a essere più conservate rispetto alle sequenze di RNA. Annotare gli ORFs per omologia permette confronti più significativi tra specie, aiutando a identificare ortologhi funzionali e a studiare la conservazione delle vie metaboliche e dei meccanismi cellulari.

In conclusione, la predizione e annotazione degli ORFs piuttosto che dei trascritti interi fornisce un approccio più mirato e accurato per l'analisi funzionale dei dati di trascrittomica, specialmente in contesti dove il genoma di riferimento non è disponibile o è incompleto. Questo metodo enfatizza le regioni effettivamente codificanti e funzionali, rendendo l'analisi più rilevante per scopi biologici e biotecnologici.