POLITECNICO
DI MILANO

SCUOLA DI INGEGNERIA INDUSTRIALE E DELL'INFORMAZIONE

Progetto di Ingegneria Informatica

a.a 2018/2019

Silvia Ferraris

matricola 854767 codice persona 10528353

docente responsabile: Marco Masseroli

Tutors: Anna Bernasconi, Arif Canakoglu

Github repository: https://github.com/silviaferraris/encode2020

INTRODUZIONE

Il progetto prevede **l'implementazione** di un codice in linguaggio Scala al fine di integrarsi, estendo la classe "Transformer" all'interno di GeCo.

Il codice permette la trasformazione di file di dati genomici in formato TSV, in particolare i "gene quantification" e i "transcript quantification", arricchendoli con informazioni prelevate dai file di annotazione (GENCODE) relativi.

Inoltre ho prodotto una **tabella** contenente i vari tipi di dati dalla fonte ENCODE facilmente riconducibili al formato BED (in formato excel, all'interno della cartella "delivery")

GENE QUANTIFICATION E TRANSCRIPT QUANTIFICATION FILES

Per l'implementazione del codice ho trattato due tipi di output file TSV dall'annotation matrix di Encode: i cosidetti "gene quantification" files, e i "transcript quantification" files.

Di seguto, per ognuno di questi, ho contato il numero di file "released", considerando entrambi gli "assembly" (GRCH38 e HG19) e ho riportato gli schemi più ricorrenti specificando il significato dei vari campi dell'header.

GENE QUANTIFICATION

totale : 6138

di cui 4588 col seguente schema:

gene_id transcript_id(s) length effective_length expected_count TPM FPKM posterior_mean_count posterior_standard_deviation_of_count pme_TPM pme_FPKM TPM_ci_lower_bound TPM_ci_upper_bound FPKM_ci_lower_bound FPKM_ci_upper_bound

Significato dei campi

Sono file contenenti stime di espressione a livello di gene. La prima riga contiene nomi di colonna separati da \t (un tab). Il formato di ogni riga nel resto di questo file è:

I campi sono separati dal carattere di tabulazione. I campi posterior_mean_count posterior_standard_deviation_of_count pme_TPM pme_FPKM TPM_ci_lower_bound TPM_ci_upper_bound FPKM_ci_lower_bound FPKM_ci_upper_bound sono facoltativi.

'transcript_id(i)' è un elenco separato da virgole di transcript_ids appartenenti al gene in questione. Se non vengono fornite informazioni,i campi "gene_id" e "transcript_id/(s)" sono identici ("transcript_id").

I campi “length” e "effective_length" sono definiti come la media ponderata delle lunghezze e delle lunghezze effettive delle trascrizioni di un gene (ponderate da "IsoPct").

TRANSCRIPT QUANTIFICATION

Totale:4601

Di cui 4568 col seguente schema:

transcript_id gene_id length effective_length expected_count TPM FPKM IsoPct posterior_mean_count posterior_standard_deviation_of_count pme_TPM pme_FPKM IsoPct_from_pme_TPM TPM_ci_lower_bound TPM_ci_upper_bound FPKM_ci_lower_bound FPKM_ci_upper_bound

Significato dei campi

'transcript_id' è l’identificatore di questa trascrizione. "gene_id" è lìidentificatore del gene a cui appartiene la suddetta trascrizione (denota questo gene come gene “genitore”). Se non vengono fornite informazioni sui geni, "gene_id" e "transcript_id" sono uguali.

'length' indica la lunghezza della sequenza della trascrizione. 'effective_length' conta solo le posizioni che possono generare un frammento valido. Se non viene aggiunta alcuna coda poli(A), 'effective_length' è uguale alla lunghezza della trascrizione.

La lunghezza media di un frammento è 1. Se la lunghezza effettiva di una trascrizione è inferiore a 1, le stime di lunghezza e abbondanza effettive di questa trascrizione sono impostate a 0.

'expected_count' è la somma delle probabilità inferiori di ogni lettura proveniente da questa trascrizione su tutte le letture. Perché ogni lettura allineata a questa trascrizione ha la probabilità di essere generata dal “background noise” oppure perchè RSEM può filtrare alcune letture allineate di bassa qualità. La somma dei conteggi previsti per tutte le trascrizioni è generalmente inferiore al numero totale di letture allineate.

'TPM' è l'acronimo di Transcripts Per Million. È una misura relativa della quantità di trascrizioni. La somma del TPM di tutte le trascrizioni è di 1 milione. 'FPKM' è l'acronimo di Fragments Per Kilobase di trascrizione per Milione di letture mappate. È un'altra misura relativa della quantità di trascrizioni.

La somma di FPKM non è costante , considerati vari campioni.

'IsoPct' sta per percentuale isoforme. È la percentuale di “abbondanza” di questa trascrizione sull’ “abbondanza” del gene padre. Se quest’ultimo ha un solo isoform o le informazioni sul gene non vengono fornite, questo campo è impostato al valore 100.

"posterior_mean_count", "pme_TPM", "pme_FPKM" sono stime calcolate dal campionatore Gibbs di RSEM. "posterior_standard_deviation_of_count" è la deviazione standard posteriore dei conteggi. 'IsoPct_from_pme_TPM' è la percentuale di isoform calcolata a partire dai valori 'pme_TPM'.

'TPM_ci_lower_bound', 'TPM_ci_upper_bound', 'FPKM_ci_lower_bound' e 'FPKM_ci_upper_bound' sono i limiti inferiori e superiori di intervalli di credibilità del 95% per i valori TPM e FPKM. I limiti sono inclusivi.

note: Nel codice,se si vogliono usare i file di annotazione “completi”, è necessario entrare nella cartella gz e decomprimerli dentro alla cartella gencode_files. Se invece si vogliono usare i file di annotazione “ridotti” bisogna cambiare, nel file di configurazione, la gtf_folder e sostituendo “gencode_files” con “gencode_files/trimmed”

IMPLEMENTAZIONE

1. Trasformazione

   La trasformazione e la scrittura dei file di output è eseguita dai metodi *transform, transformFiles, writeData, writeHeader.*

   1.1. transform

   Il metodo legge il file di configurazione, usando il metodo *readConfigFile,* carica i file di annotazione (.gtf).
   Ipotizzando che i file tsv da analizzare siano già stati scaricati e inseriti in una cartella che ho denominato “tsv”, contenente altre due cartelle per i differenti tipi di file “gene quantification” e “transcript quantification”, il metodo procede con la ricerca di tutti i file in formato .tsv all’interno delle suddette cartelle.
   Una volta ottenuta la lista dei file presenti, procede con la trasformazione prima dei “gene quantification” e dopo dei “transcript quantification” utilizzando il metodo *transformFiles*.

   1.2. transformFiles

   Verifica innanzitutto che le cartelle di output indicate nel file di configurazione esistano, altrimenti le crea.
   Successivamente, itera la lista dei file .tsv creata precedentemente e, per ogni file:
   - crea un oggetto di tipo tsv
   - crea il file di output, scrivendo l’intestazione
   - per ogni id, esegue il “match” (di cui si spigherà il funzionamento di seguito nel dettaglio) con i dati provenienti dagli annotation files, scrivendoli sul file di output.

   1.3. writeData

   Utilizza lo schema proveniente dal file di configurazione per comporre il nuovo file .tsv di output con le varie informazioni relative ricavate dalle HashMap del *TSV* e del *GTF*. (di cui si parlerà nel dettaglio in seguito)

1.4. writeHeader

Utilizza lo schema del file di configurazione per scrivere l'intestazione delle varie colonne del file di output.

2. Caricamento dei file di annotazione

- I file di annotazione (.gtf) vengono cercati nella relativa cartella (gtf_folder del file config.xml).
  Per ognuno di questi, il *GTFLoader* legge tutte le linee convertendole nel *GTFEntry*, il quale viene aggiunto ad un'istanza della classe *GTF*.
- Quando il *GTFLoader* crea un'istanza di *GTF*, vengono inizializzate tre HashMap (geneMap, transcriptMap, trnaMap).
- Il metodo loadOn del GTFLoader itera tutte le linee dei file di annotazione e per ogni linea produce un'HashMap contenente tutti i dati e differenzia i diversi "*futureType*" (tRNAscan, gene, transcript) .
- L'HashMap prodotta viene passata all'istanza di *GTF* attraverso il metodo addLine, il quale, basandosi sulla classificazione dei "*featureType*" inserisce nell'HashMap corrispondente l'Id, come chiave e, come valore, un oggetto di tipo *GTFEntry* istanziato usando l'HashMap precedentemente prodotta dal metodo loadOn della classe *GTFLoader*.

3. Matching

- Innanzitutto, il file tsv scaricato viene passato alla classe TSV. Qui, viene estrapolato l'header del file in modo da classificarlo in base alla prima colonna (gene quantification se la prima colonna è *gene_id,* transcript quantification se la prima colonna è *transcript_id)*

- Tutte le linee del file tsv vengono trasformate in un'HashMap ed inserite in una lista di coppie "id-hashMap".

- Il metodo *trasformFiles* utilizza il metodo *foreach* della classe *TSV* per iterare tutte le coppie "id-hashMap" del file tsv.

- Per ognuna delle coppie, viene effettuato il "match" semplicemente passando al metodo *get* della classe *GTF* l'id ed un valore booleano che indica di quale tipo di file si tratta. Tale metodo cerca nelle HashMap la *GTFEntry* corrispondente all'id passato e se non la trova, restituisce una *GTFEntry* vuota.

- L'hashMap del *TSV* e il *GTFEntry* vengono passati al metodo writeData che scrive i dati sul file di output.

TABELLA DEI FILES

ANNOTAZIONI IMPORTANTI :

GDM sta per Genomic Data Model. È basato su dataset e campioni, e su due astrazioni: una per le regioni genomiche, che rappresentano cioe le porzioni del DNA e delle loro caratteristiche, una per i metadati.

DATASETS= collezioni di campioni

CAMPIONI= due parti- una regione con i dati che descrivono le caratteristiche a la locazione delle caratteristiche genomiceh e i metadati, che descrivono generalmente le proprietà dei campioni

Obiettivo:

Esplorare la matrice di dati su ENCODE matrix/experiments e analizzare nello specifico i "file type", escludendo i tipi di dati non processati (es fastq, bam, bigWig, bigBed...), concentrandosi sui bed e tutti quelli che sono riducibili a questo formato (bed è molto simile a GDM)

Alcuni di questi tipi di file(corrispondenti all'assay "transcription") sono stati aggiunti, attraverso gli appositi schema, all'interno del "Metadata-Manager".

COMMENTI AGGIUNTIVI RELATIVI AI VARI FORMATI :

BedExonScore=BED6+3

TagAlign=BED3+3

Bedpe=BED3+

peptideMapping=BED6+4

Bed narrowPeak=BED6+4

BedBroadPeak=BED6+3

BedRnaElements=BED6+3 Scores format

tssPeak

IdrPeak=BED6+8

bedLogR=BED9+1

Silvia Ferraris -progetto di Ingegneria Informatica 2018/19

PRINCIPALI FONTI:

https://www.encodeproject.org/help/file-formats/>

http://resources.qiagenbioinformatics.com/manuals/clccancerresearchworkbench/200/index.php?manual=Import_tracks.html

https://www.completegenomics.com/documents/DataFileFormats_Standard_Pipeline_2.5.pdf

http://software.broadinstitute.org/software/igv/?q=book/export/html/16

IDRpeak: https://github.com/ENCODE-DCC/encValData/edit/master/as/idr_peak.as

https://www.biostars.org/p/240291/

https://bedtools.readthedocs.io/en/latest/content/general-usage.html

https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSM2198555

https://github.com/arq5x/bedtools-python/blob/master/examples/ex1-intersect.py

VCF: http://samtools.github.io/hts-specs/VCFv4.1.pdf

GTF/GFF format: http://genomewiki.ucsc.edu/index.php/Genes_in_gtf_or_gff_format

http://gmod.org/wiki/GFF2

http://uswest.ensembl.org/info/website/upload/gff.html

IDAT: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3968891/

modPep: https://github.com/ENCODE-DCC/encValData/blob/master/as/modPepMap-std.as

GENOME ASSEMBLY:

- Hg19=GRCh37

  Description: Genome Reference Consortium Human Build 37 (GRCh37)

  Organism name: homo sapiens(human)

  BioProject: PRJNA31257

  Submitter: Genome Reference Consortium

  Synonyms: hg19

  Assembly type: haploid-with-alt-loci

  Assembly level: Chromosome

  Genome representation: full

- GRCh38

  Description: Genome Reference Consortium Human Build 38

  Organism name: homo sapiens (human)

  BioProject: PRJNA31257

  Submitter: Genome Reference Consortium

  Synonyms: hg38

  Assembly type: haploid-with-alt-loci

  Assembly level: Chromosome

  Genome representation: full