Applicazioni pratiche di HMM

Genomica Computazionale, Corso di Laurea Specialistica in Genoma Funzionale, Università di Trieste

Fabrizio Mafessoni, Daniela Eugenia Nerelli, Dona Kireta

Applicazioni di Hidden Markov Models all'epigenomica

Analisi di ATACseq

Possiamo utilizzare hmmratac da solo installandolo con

```
conda install -c bioconda hmmratac
```

Tuttavia consiglio di usare MACS3.

```
conda create -n macs3 python=3.11 -y
conda activate macs3
pip install "cython==0.29.*"
pip install "macs3>=3.0.0"
```

oppure

```
pip install "macs3>=3.0a.4"
```

Controlliamo che sia installato correttamente:

```
macs3 --version
macs3 callpeak -h
macs3 hmmratac -h
```

Cercate su ENCODE il vostro organismo e tipo cellulare preferito. In alternativa, possiamo esaminare sequenze da dei motoneuroni di un donatore con sclerosi amiotrofica laterale:

```
mkdir chipseq
cd chipseq
wget https://www.encodeproject.org/files/ENCFF199OKA/@@download/ENCFF199OKA.bam
samtools index ENCFF199OKA.bam
samtools view -b ENCFF199OKA.bam chr21 > ATACseq_chr21.bam
samtools index ATACseq_chr21.bam
```

```
samtools sort -o ATACseq_chr21.sorted.bam ATACseq_chr21.bam

samtools index ATACseq_chr21.sorted.bam
```

Notate che macs3 può effettuare una gamma di metodologie di peak calling diverse, anche non utilizzando HMMs:

```
macs3 hmmratac -i ATACseq_chr21.sorted.bam -n ATACseq_chr21
```

Esploriamo le regioni accessibili/non coperte dai nucleosomi/espresse con:

```
less ATACseq_chr21_accessible_regions.narrowPeak

cd ..
```

Analisi di ChipSeq

A meno che abbiate scelto un tipo cellulare diverso su ENCODE, scarichiamo altre sequenze dallo stesso donatore, questa volta sequenziate con chip-seq per l'acetilazione della lisina in posizione 27 della coda N terminale dell'istone 3 (H3K27ac).

```
mkdir ATACseq

cd ATACseq

wget
http://hgdownload.cse.ucsc.edu/goldenPath/hg38/bigZips/hg38.ch
rom.sizes

wget
https://www.encodeproject.org/files/ENCFF869FUW/@@download/ENC
FF869FUW.bam

samtools index ENCFF869FUW.bam

samtools view -b ENCFF869FUW.bam chr21 > H3K27ac_chr21.bam

samtools index H3K27ac_chr21.bam
```

Per chiamare i picchi useremo per semplicità CHROMHMM, che è disegnato con lo scopo di combinare molte tracce epigenomiche ma può anche essere utilizzato per una singola traccia chipseq, ed utilizza un modello HMM semplificato per usare dati binari. Prepariamo i files binarizzando i nostri dati, utilizzando un modello che individua le finestre del genoma con valori significativamente piu' alti di presenza di reads.

Prima però installiamo CHROMHMM. CHROMHMM richiede delle librerie JAVA. Pertanto:

```
cd ..

mkdir CHROMHMM

cd CHROMHMM
```

```
sudo apt install default-jre   # Ubuntu / WSL

brew install openjdk           # macOS

wget
https://raw.githubusercontent.com/jernst98/ChromHMM/master/ChromHMM.jar
```

Ora iniziamo la preparazione dei dati e binarizziamoli. Notiamo che CHROMHMM necessita che i vari bam di input siano in una cartella e che siano specificati in un file tabulare come `cellmarkfile.txt`:

```
mkdir input_bam_H3K27ac_chr21

printf "Cell1\tH3K27ac\tH3K27ac_chr21.bam\n" >
input_bam_H3K27ac_chr21/cellmarkfile.txt

mv H3K27ac_chr21.bam* input_bam_H3K27ac_chr21

java -jar ChromHMM.jar BinarizeBam hg38.chrom.sizes
input_bam_H3K27ac_chr21
input_bam_H3K27ac_chr21/cellmarkfile.txt
binarized_H3K27ac_chr21
```

Una volta binarizzati i dati, andiamo a stimare emissioni e transizioni con l'HMM.

```
java -jar ChromHMM.jar LearnModel binarized_H3K27ac_chr21
output_H3K27ac_chr21 2 hg38
```

Queste possono essere esaminate con

```
cat output_H3K27ac_chr21/emissions_2.txt

cat output_H3K27ac_chr21/transitions_2.txt
```

Notate che qui il due rappresenta il numero degli stati. Cosa indicano emissioni e transizioni?

Infine esploriamo che frazione del genoma è caratterizzato con stato E1 e E2, contando la somma dei nucleotidi.

```
cat output_H3K27ac_chr21/Cell1_2_segments.bed | grep E1 | awk
'BEGIN{sum=0}{sum=sum+$3-$2}END{print sum}'

cat output_H3K27ac_chr21/Cell1_2_segments.bed | grep E2 | awk
'BEGIN{sum=0}{sum=sum+$3-$2}END{print sum}'
```

Che percentuale è coperta da E1?

```
cd ..
```

Segmentazione epigenomica

Ora che abbiamo almeno due tracce epigenomiche, proviamo ad analizzarle in modo combinato con CHROMHMM. Visto che sono solo due potremmo esplorarle visualmente, ma proviamo comunque a caratterizzare il genoma come caratterizzato da 3 stati.

```
mkdir CHROMHMM

cd CHROMHMM

mkdir input_bams_combined

printf
'Cell1\tH3K27ac\tH3K27ac_chr21.bam\nCell1\tATAC\tATACseq_chr21
.bam\n' > cellmark_combined.txt

cp ../chipseq/input_bam_H3K27ac_chr21/H3K27ac_chr21.bam*
input_bams_combined/

cp ../ATACseq/ATACseq_chr21.bam* input_bams_combined/

grep -w 'chr21' ../chipseq/hg38.chrom.sizes >
hg38_chr21.chrom.sizes
```

Ora binarizziamo:

```
java -jar ChromHMM.jar BinarizeBam hg38_chr21.chrom.sizes
input_bams_combined cellmark_combined.txt binarized_combined
```

Infine giriamo l'HMM:

```
java -jar ChromHMM.jar LearnModel binarized_combined
output_ATAC_H3K27ac_chr21_3states 3 hg38
```

Abbiamo già visto come interpretare i dati di CHROMHMM per la Chip-seq. Come interpretate questi dati della segmentazione? Cosa fa questo specifico tipo di acetilazione, inibisce o attiva l'espressione?

Esercizio 1: scaricate da ENCODE dati per altre modificazioni post-traduzionali degli istoni ed effettuate segmentazione con almeno 5 modifiche. Cosa notate?

Esercizio 2: scegliete un altro tipo cellulare di vostra scelta da ENCODE. Comparatelo con i motoneuroni. Quale stato è piu' abbondante e quale meno che nei motoneuroni?

Stima della contaminazione nel DNA antico

Usiamo la Hidden Markov Model AuthentiCT, che assume che il DNA contaminante non sia affatto deaminato, e prova per ogni reads a classificare le varie potenziali sostituzioni e citosine come situate in un una regione a doppio filamento (pertanto non deaminata) o a singolo filamento all'estremità di una read o al suo interno. Installiamolo usando github e pip:

```
git https://github.com/StephanePeyregne/AuthentiCT.git

pip3 install AuthentiCT/

cd AuthentiCT/

python3 setup.py install
```

Ora esploriamo il dataset per il test, appartenente ad un incisivo di circa 400.000 anni fa da un proto-Neandertal (probabilmente ascrivibile ad *Homo heidelbergensis*) dalla Spagna. Prima di tutto esploriamo il profilo di deaminazione con AuthentiCT deamination. Notate come le posizioni terminali delle reads abbiano una percentuale maggiore di sostituzioni C->T:

```
cd test_dataset

AuthentiCT deamination Sima.incisor.subset_test.sam
```

Ora usiamo l'HMM per stimare la contaminazione per selezionare le reads che hanno una probabilità (posterior) maggiore del 95% di essere endogene (antiche):

```
AuthentiCT deam2cont -o test_results -s 10000
Sima.incisor.subset_test.sam
```

cat test_results | awk '{print $(NF-3)}' | less -S

Quante sono in proporzione?

Profile HMM

Installate hmmer con

```
sudo apt-get install hmmer # Ubuntu o Windows

brew install hmmer # macOS with Homebrew

conda create -n hmmer python=3.9 hmmer -y #entrambi, con conda
```

Alternativamente scaricate il codice sorgente e compilatelo:

```
wget http://eddylab.org/software/hmmer/hmmer.tar.gz

tar -xzvf hmmer.tar.gz; cd hmmer-3.4

./configure

make

sudo make install
```

Ora scarichiamo il database di PFAM, che contiene i profili/allineamenti per tutte le possibili famiglie proteiche, e indicizziamolo con hmmpress:

```
wget
ftp://ftp.ebi.ac.uk/pub/databases/Pfam/current_release/Pfam-
A.hmm.gz

gunzip Pfam-A.hmm.gz

hmmpress Pfam-A.hmm
```

Ora creaiamo una sequenza di una proteina:

```
echo '>MyProtein

MKTIIALSYIFCLVFADYKDDDDKLVVSTQTALAIVLATTLVFLSKKQY

SPGAIELLLASANPGSFTKGDNKPFVPKQHIDSQKKAIEILKQSNSK' >
my_protein.fasta
```

Come primo esercizio, proviamo a caratterizzare un nuovo gene/proteina. Esplorate poi i risultati results.txt:

```
hmmscan --cpu 4 --tblout results.txt Pfam-A.hmm
my_protein.fasta
```

A quale famiglia assomiglia di piu'?

Ora invece assumiamo di essere interessati alle kinasi:

```
hmmstat Pfam-A.hmm | grep -i "kinase" | head -20

hmmfetch Pfam-A.hmm PF00696.34 > AA_kinase.hmm
```

Cerchiamole tutte nel genoma di E.coli:

```
wget
https://ftp.ncbi.nlm.nih.gov/genomes/refseq/bacteria/Escherich
ia_coli/reference/GCF_000005845.2_ASM584v2/GCF_000005845.2_ASM
584v2_protein.faa.gz

gunzip GCF_000005845.2_ASM584v2_protein.faa.gz

mv GCF_000005845.2_ASM584v2_protein.faa all_proteins.fasta
```

Quante proteine sono espresse in E.coli?

```
grep -c "^>" all_proteins.fasta  # ~4,300 proteins
```

Ora, quante AA_kinasi troviamo?

```
hmmsearch --tblout kinases.txt AA_kinase.hmm
all_proteins.fasta
```

HMM copy per copy number variation (in tumori ma anche in campioni normali)

Con conda installate R e create l'ambiente hmmcopy_env:

```
conda create -n hmmcopy_env -c bioconda -c conda-forge \
  r-base=4.3 \
  r-hmmcopy \
  r-genomicranges \
  r-iranges \
  r-genomeinfodb \
  r-rtracklayer \
  samtools
conda activate hmmcopy_env
```

Ora esploriamo i dati di esempio di HMMcopy, sequenze di un tumore al seno mappate sul cromosoma 6. Create in vim o nel vostro editor preferito un file chiamato `run_hmmcopy_example.R` con questo contenuto:

```
#!/usr/bin/env Rscript

## ------------------------------------------------------------------
## Basic HMMcopy example using the built-in 'tumour' dataset
## ------------------------------------------------------------------

## (Optional) one-time installation, you do this before class
## install.packages("HMMcopy",
##   repos = c("https://bioc.r-universe.dev", "https://cloud.r-project.org")
## )

library(HMMcopy)

## 1. Load the example data (chromosome 6, tumour + normal)
:contentReference[oaicite:1]{index=1}
data("tumour")   # loads: tumour_reads, tumour_copy, normal_copy,
                 #        tumour_param, tumour_segments

## Quick peek at the tumour read counts (before correction)
tumour_reads       # RangedData: bins along chr6 with raw counts, GC, mappability
normal_copy        # copy number profile for the normal sample
tumour_copy        # copy number profile for the tumour (already corrected)

## 2. (Optional) Recompute copy number from raw read counts
##    This shows students the "GC + mappability correction" step.
:contentReference[oaicite:2]{index=2}
tumour_copy_new <- correctReadcount(tumour_reads, verbose = TRUE)

## 3. Run the HMM segmentation on the corrected copy number
:contentReference[oaicite:3]{index=3}
##    (You can use either tumour_copy or tumour_copy_new.)
tumour_segments_new <- HMMsegment(tumour_copy_new)

## 4. Plot bias correction, copy number and HMM segments
:contentReference[oaicite:4]{index=4}

## Ask before moving to each new plot in base R
par(ask = TRUE)

## (a) GC / mappability bias and correction – using the NORMAL sample
plotBias(normal_copy)

## (b) Copy number profile after correction – TUMOUR
plotCorrection(tumour_copy_new)
```

```
## (c) HMM segments and copy-number states along the chromosome
##     (segments in colours; each colour = a copy number state)
plotSegments(tumour_copy_new, tumour_segments_new)

## (d) HMM state parameters (means for each state, etc.)
##     Here we use the example parameter matrix shipped with the package.
plotParam(tumour_segments, tumour_param)

## Reset asking behaviour
par(ask = FALSE)
```

Eseguitelo e poi esplorate i plot generati. Non abbiamo trattato R in questo corso, ma modificando questo script potreste analizzare un qualsiasi bam file.

```
chmod +x run_hmmcopy_example.R
```

```
./run_hmmcopy_example.R
```

Uso di Bedtools

Scarichiamo le annotazioni geniche per il genoma di referenza umano

```
wget
https://ftp.ebi.ac.uk/pub/databases/gencode/Gencode_human/rele
ase_48/gencode.v48.basic.annotation.gtf.gz
```

```
gunzip gencode.v48.basic.annotation.gtf.gz
```

Convertiamo il gtf, formato tipico delle annotazioni, in .bed

```
GTF=gencode.v48.basic.annotation.gtf
OUT=genes.v48.basic.bed

awk 'BEGIN{FS=OFS="\t"}
     $3=="gene" {
       # parse attributes to extract gene_name (fallback to
gene_id)
       gene_id="NA"; gene_name="NA";
       split($9, a, ";");
       for (i in a) {
         gsub(/^ +| +$/, "", a[i]);   # trim
         if (a[i] ~ /^gene_name/) {
           split(a[i], b, " ");
           gene_name = b[2];
           gsub(/"/, "", gene_name);
         }
         if (a[i] ~ /^gene_id/) {
           split(a[i], c, " ");
           gene_id = c[2];
```

```
        gsub(/"/, "", gene_id);
      }
    }
    if (gene_name=="NA") gene_name=gene_id;
    # BED is 0-based start
    print $1, $4-1, $5, gene_name
  }' "$GTF" | sort -k1,1 -k2,2n > "$OUT"
```

Esploriamo il nostro file .bed genes.v48.basic.bed

```
less genes.v48.basic.bed
```

Installiamo bedtools

```
sudo apt install bedtools #linux o windows
```

```
brew install bedtools #mac
```

```
conda install -c bioconda bedtools
```

Usando la sintassi

```
bedtools intersect -a file1.bed -b file2.bed
```

trovate i geni che sono all'interno dei picchi acetilati o espressi con ATAC seq. E quali nelle varie regioni trovate da CHROMHMM? E quali non (consiglio: usate per questo bedtools subtract).