ChIP-Seq

Name: Dang Quynh Tram Nguyen

### Laden die FASTQ Files und Genom-File herunter

* Laden SRA (Sequence Read Archive) herunter und dann konvertieren in FASTQ Files mit SRA-Tools (fastq-dump)
* Laden Humangenom-Files (neuste Version)

### Sortiert alle Reads, die kürze als 30 bp sind, aus

* Filtern die FASTQ Files, damit sie nur die als 30bp längeren Reads enthalten.

Code: **cat <fastq-file> | awk 'BEGIN {FS = "\t"; OFS = "\n"} {header = $0 ; getline seq ; getline qheader ; getline qseq ; if (length(seq)>30) {print header, seq, qheader, qseq}}' > <fastq-file>**

### Mappen mit bowtie2 und zählen die Bedingung passende Reads ab

* Mappen die ChIP-Seq Datensätze mit hg38, welche als Suchindex von bowtie2 in der .bt2 Format aufgebaut wird

Code: **bowtie2-build hg38.fa hg38**

**bowtie2 -x hg38 -U <fastq-file> -S <sam-file>**

* Sortieren alle Reads aus, die mit Mappingqualität unter 20 gemappt werden.

Code: **samtools view -Scq 20 <sam-file>**

-S: arbeiten mit SAM Format

-q *INT*: vernachlässigen die MAPQ unten *INT*

-c: zählen die Reads

* Sortieren alle nicht gemappten Reads aus, dadurch bleiben nur die mappbaren Reads.

Code: **samtools view -ScF 4 <samfile>**

-F 4: außer den ungemappten Reads

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Insgesamt** | **Mappbaren Reads** | **Eindeutig gemappte Reads** | **Reads mit MAPQ > 20** |
| **MCF7 Control ER rep 1** | 33280770 | 32381745 | 24412556 | 28271412 |
| **MCF7 Control ER rep 2** | 13573224 | 13142690 | 9982761 | 11370321 |
| **MCF7 E2 rep 1** | 28962379 | 28194817 | 22445566 | 25146157 |
| **MCF7 E2 rep 2** | 15999069 | 15446909 | 12376329 | 13779699 |
| **MCF7 input DNA** | 24130002 | 23633863 | 16478868 | 19908763 |

### Filtern zu den Files mit nur eindeutig gemappten Reads

(benötigter Schritt, um Peaks ohne FDR-Fälle zu finden)

* Konvertieren SAM Files zu BAM Files

Code: **samtools view -h -S <sam file> -bo <bam file>**

-h: behalten den Header

-b: Bam Files

-o: Output

* Sortieren BAM Files

Code: **sambamba sort -t 4 <bam file> -o <sorted bam file>**

-t : Genutzte Threads

* Filtern eindeutig gemappte Reads (unique mapped reads)

Code: **sambamba view -h -t 8 -f bam -F “[XS] == null and not unmapped and not duplicate“ <sorted bam file> > <umr bam file>**

-h : mit Header

-f : Output Format

-F : (--filter) Filter setzen

-t : genutzte Threads

### Finden Peaks

* Finden Peaks

Code: **macs2 callpeak -t <umr e2 bam file> -c <umr control bam file> -n <name> --outdir <saved path> > <log file>**

-p: genutzte Threads (sollte nicht genutzt, weil der Prozess chaotisch wird und das Ergebnis nicht in der Reihe geordnet ist. Sowieso läuft der Befehl sehr schnell ab)

-t: Signal

-c: Control

-n: prefix

--outdir: directory to save the outputs

wc -l \*.narrowPeak

|  |  |
| --- | --- |
| **Bezeichnungen** | **Peaks** |
| Replikate 1 | 54031 |
| Replikate 2 | 28388 |
| Replikate 1 (mit Input) | 51402 |
| Replikate 2 (mit Input) | 49366 |

### Euler-Diagramm

(nutze VennDiagram-Paket in R)

* Importieren die csv Files
* Vergleichen control und signal mit Befehlen von VennDiagram

Code: **calculate.overlap(list(array1,array2)**

**draw.pairwise.venn(area1, area2, cross.area,…)**

**grid.newpage()**

(Die Werte des Arrays gehören zu den ‚abs\_summit‘ Spalten, wo die höchste Werte der Peaks anzeigen. Bei 2 Arrays wird alle Werte durch 100 verteilt vor dem Vergleichen. Weil der Abstand von 200-300 bp zwischen 2 Bindestellen beim Vergleichen ein Protein erscheinen kann.)

s