RNA-seq and CHIP-seq data analysis general pipeline

RNA-seq and CHIP-seq are two popular high-throughput sequencing technologies used to study gene expression and genomic regulatory elements, respectively. Integrating the results of these two technologies can provide a more comprehensive understanding of the molecular mechanisms underlying cellular processes. Here is a general outline of a possible RNA-seq and CHIP-seq integrative analysis pipeline:

1. Quality control

- Quality control

The very first step is to check the quality of raw sequencing data. It can be done with FASTQC software or using the FASTQC module on Rivanna in a Linux environment. The input should be a fastq file. FASTQC will output an HTML file that shows different aspects of the quality of the raw sequencing data, such as per base sequence quality, per base sequence content, and so on.

```
###quality control with FASTQC###
module load fastqc
fastqc -o [output_directory] [input_raw_data.fastq.gz]
```

- Data trimming

If the data fails in many modules, it's better to trim the reads to remove adapter sequences and low-quality regions. Trimmomatic is a good tool for doing this. The following example demonstrates a trimmomatic command for paired-end data.

```
###trimming###
module load trimmomatic/0.39
java -jar $EBROOTTRIMMOMATIC/trimmomatic-0.39.jar PE -threads 12
[input_raw_forward.fastq.gz] [input_raw_reverse.fastq.gz]
[input_raw_forward.paired.fastq.gz] [input_raw_forward.unpaired.fastq.gz]
[input_raw_reverse.paired.fastq.gz] [input_raw_reverse.unpaired.fastq.gz] LEADING:3
TRAILING:3 HEADCROP:[no._of_reads_to_be_trimed]
```

2. Preprocessing

- Alignment

The second step is to map the raw sequencing data to the genome using alignment tools, such as HISAT2 for RNA-seq data, and BOWTIE2 for CHIP-seq data. HISAT2 is a splice-aware aligner using a hierarchical indexing strategy to allow fast and efficient reads alignment. BOWTIE2 uses an index-based alignment strategy, reducing the search space and increasing the speed of the alignment process. The input is the fastq.gz file after quality control and the output will be a SAM file that contains sequence header information, alignment information, sequence data, and probable additional information. The alignment will also tell the mappability of the raw sequencing data which can be used to assess their quality. Before alignment, both HISAT2 and BOWTIE2 require an index directory. A pre-built index can be downloaded from their official websites. The following examples include using HISAT2 to align paired-end RNA-seq data and BOWTIE2 to map single-end CHIP-seq data.

```
###creating index (downloading from hisat2 official website)###
wget https://genome-idx.s3.amazonaws.com/hisat/hg38_genome.tar.gz
tar -zxvf hg38_genome.tar.gz

###alignment with hisat2 (RNA-seq)###
module load gcc/9.2.0
module load hisat2/2.1.0
cd hg38
hisat2 -x genome -1 [input_forward_paired_fastq.gz] -2 [input_reverse_paired_fastq.gz] -S
[output_sam_file.sam]

###creating index (downloaded from Bowtie2 official website)###
wget https://genome-idx.s3.amazonaws.com/bt/GRCh38_noalt_as.zip
unzip GRCh38_noalt_as.zip

###alignment with bowtie2 (CHIP-seq)###
cd GRCh38_noalt_as
bowtie2 -x GRCh38_noalt_as -U [input_fastq.gz] -S [output_sam_file.sam]
```

• Converting to BAM file

After alignment, SAM files are going to be converted into BAM files for further analysis. BAM is a compressed binary version of the SAM format, while SAM is a text-based format. SAM files are human-readable and can be viewed and edited using text editors, but they can be large in size, especially for large datasets. BAM files can be created by converting SAM files to a binary format using a tool such as SAMtools. The binary format allows for efficient storage and retrieval of the alignment data, leading to faster processing times and reduced storage requirements. The "view" function converts the SAM file to a BAM file without sorting it while the "sort" function can produce a sorted BAM file. The "index" function creates an index for sorted BAM files which is necessary for several analyses, such as visualization via IGV.

```
###converting sam file to bam file###
module load samtools/1.10
samtools view -bS [input_sam_file.sam] > [output_bam_file.bam]
samtools sort  [input_sam_file.sam] > [output_bam_file.sorted.bam] OR
[input_bam_file.bam] > [output_bam_file.sorted.bam]
samtools index [output_bam_file.sorted.bam]
```

3. Differential expression analysis using RNA-seq data

RNA-seq is a high-throughput sequencing technology used to study the transcriptome or the set of all RNA molecules expressed in a cell or tissue at a given time. RNA-Seq provides a comprehensive and quantitative picture of the gene expression landscape, allowing researchers to identify differentially expressed genes, alternative splicing events, and novel transcripts.

• Summarizing read counts

The first step of identifying a differentially expressed gene is to summarize the read counts of each gene based on the mapping result. This step could be done using FeatureCounts (R) or HTSeq (Python). The following is an example of using FeatureCounts in R and Linux to find the read counts of each gene. FeatureCounts can process multiple BAM files in one command. The inputs are BAM files from alignment and an annotation GTF file. The GTF file format is a tab-delimited text file that contains information about gene structure, exon locations, and transcript information. It can be found on the GENCODE website. The following is an example of using FeatureCounts in R and Linux environments to summarize the read counts of paired-end RNA-seq data.

```
###obtainning annotation GTF file###
wget  https://ftp.ebi.ac.uk/pub/databases/gencode/Gencode_human/release_43/
gencode.v43.basic.annotation.gtf.gz
gunzip gencode.v43.basic.annotation.gtf.gz

###using FeatureCounts in Linux###
featureCounts -t exon -g gene_name -a [input_annotation_gtf_file.gtf] -o
[output_counts.txt][input_sorted_bam_file_1.sorted.bam]
[input_sorted_bam_file_2.sorted.bam]

###using FeatureCounts in R###
module load goolf/7.1.0_3.1.4  R
R
library(Rsamtools) #optional, allows RStudio to import BAM files
library(Rsubread)
bam_files <- c("input_bam_file_1.sorted.bam", "input_sorted_bam_file_2.sorted.bam")
gtf_file <- "input_annotation_gtf_file"
count_data <- featureCounts(files = bam_files, annot.ext = gtf_file, isGTFAnnotationFile
= TRUE, GTF.featureType = "exon", GTF.attrType = "gene_name", isPairedEnd = TRUE)
write.table(count_table, file = "output_file.txt", sep = "\t", row.names = FALSE)
```

• Differential expression analysis

After obtaining the read counts matrix, differential expression analysis can be performed using R packages such as DESeq2, edgeR, or Limma to identify differentially expressed genes between different conditions or samples. DESeq2 uses a negative binomial statistical model to test for differential expression, taking into account both the mean and dispersion of gene expression levels across samples. It requires an unnormalized read counts matrix as input. DESeq2 also provides a wide range of visualization and exploration tools, such as principal component analysis (PCA).

```
###using DESeq2 to perform differential expression analysis###
library(DESeq2)
library(dplyr)
counts <- read.table("input_table_is_FeatureCounts_output_counts.txt", header = TRUE,
row.names = "Geneid", sep = '\t', comment.char = "#", check.names = FALSE)
counts <- counts[,-c(1:5)] #delete the first five-row which is useless
counts <- counts[rowSums(counts)>10, ] #select rows that are significant (sum of counts
in all samples that is greater than 10)
#construct dds
samples <- data.frame(sampleID = c("treated_sample_1", "treated_sample_2",
"control_sample_1", "control_sample_2"), sample = c("treated", "treated", "control",
"control"))
```

```
row.names(samples) <- samples$sampleID
samples$sample <- factor(samples$sample, level = c(“treated”, “control”))
count <- as.matrix(counts[row.names(samples)])
dds <- DESeqDataSetFromMatrix(countData = count, colData = samples, design = ~ sample)
#differntial expression analysis
dds <- DESeq(dds)
#see the result and extract significant genes
analysis_result <- results(dds, contrast = c("sample", “treated”, “control”), parallel =
TRUE)
analysis_0.01 <- analysis_result[which(analysis_result$padj<0.01), ] #select significant
results with adjusted p-value smaller than 0.01
write.csv(analysis_0.01, file = “analysis_0.01.csv") #export result
#draw PCA graph and conduct PCA
vsd <- vst(dds, blind = F)
plotPCA(vsd, intgroup = c(“sample”))
plotPCA(vsd, intgroup = c("sample"), returnData = TRUE)
```

- Calculating FPKM

Furthermore, a differentially expressed gene can be up-regulated or down-regulated, referring to genes that have increased or decreased expression levels, respectively, compared to a reference condition. In other words, an up-regulated gene can activate the activity of a protein of interest, such as a transcription factor, while a down-regulated gene can repress it. The analysis can be completed by analyzing RPKM (Reads Per Kilobase per Million reads) or FPKM (Fragments Per Kilobase per Million fragments). Those are two normalization factors to correct for differences in sequencing depth and transcript length. RPKM is typically used with single-end RNA-seq data, while FPKM is used with paired-end RNA-seq data. FPKM can be calculated using Stringtie or self-written codes.

```
###calculating FPKM with Stringtie###
module load gcc stringtie
stringtie [input_sorted_bam_file.sorted.bam] -o [output_gtf_file.gtf] -G
[reference_annotation_gtf_file.gtf]
###calculating FPKM with R###
counts_fpkm <- read.table(“input_table_is_FeatureCounts_output_counts.txt”, header =
TRUE, row.names = “Geneid”, sep = '\t', comment.char = "#", check.names = FALSE)
for (clm in colnames(counts_fpkm)[column_of_1st_sample:column_of_last_sample]) {
  col_fpkm=paste0(clm, "_FPKM")
  total=sum(counts_fpkm[clm])
  counts_fpkm[col_fpkm] = (counts_fpkm[clm]*10^6) /
(counts_fpkm$Length*as.numeric(total)/1000)
}
```

- Finding up-regulated gene or down-regulated gene

Up-regulated genes are genes that are activated and expressed at a higher level than their baseline expression (controlled sample). Down-regulation genes, in contrast, are genes that are suppressed or expressed at a lower level than their baseline expression. Both genes play important roles in biological processes and development of various diseases, such as cancer, metabolic disorders and so on. Up-regulated and down-regulated genes can be distinguished based on the log2FoldChange values from the summary of DESeq2 differential expression analysis. A positive log2FoldChange value indicates an increase of expression while a negative value means a decrease of expression. The classification process can be simply completed by the "Filter" funciton in excel or using the example R script below.

```
###distinguishing up-regulated from down-regulated genes###
up_regulated <- analysis_0.01[which(analysis_0.01$log2FoldChange>0), ]
down_regulated <- analysis_0.01[which(analysis_0.01$log2FoldChange<0), ]
```

4. Transcription factor binding site finding using CHIP-seq data

- Remove PCR duplication

PCR duplicates in ChIP-seq data can arise when multiple copies of the same DNA fragment are amplified during library preparation and sequencing. These duplicates can significantly impact the analysis of ChIP-seq data, leading to the over-representation of certain regions and artificially inflated signal levels. The following is an example of removing PCR duplication with Picard.

```
###removing PCR duplication###
module load picard
java -jar $EBROOTPICARD/picard.jar MarkDuplicates I=[input_sorted_bam_file.sorted.bam]
O=[output_bam_file.bam] M=[output_remove_duplicates_metrics.txt] REMOVE_DUPLICATES=TRUE
ASSUME_SORTED=TRUE
```

- Looking for binding sites

A binding site is a specific location on a DNA molecule where a transcription factor or other DNA-binding protein binds to regulate gene expression. The binding site can be a few base pairs in length and can be located within a gene promoter or enhancer region. The binding site determines the specificity of the protein-DNA interaction, and different proteins may bind to different binding sites to control gene expression. The binding site of a transcription factor can be found using peak-calling tools such as MACS2. The output of MACS2 contains six files: two bdg files, a summit bed file, a narrowpeak file, an excel file, and an R file. The narrowpeak file, which is normally used for downstream analysis, can be converted into a bed file by simply changing the suffix to .bed. A bed file is a tab-delimited text file that contains the genomic coordinates of each identified peak. It has six columns including the name of the chromosome where the peak is located, the start and end position of the peak, the name of the peak, a score that reflects the relative strength, and a strand that represents the orientation of the peak.

```
####peak_calling with q-value 0.01###
module load macs2
macs2 callpeak -t [output_rmdup_treated_sample.bam] -c
[output_rmdup_controlled_sample.bam] -f BAM -g hs -B -q 0.01 -o [/path/to/
output_directory]
```

- Distinguishing promoters and enhancers

Binding sites can be distinguished into two categories: promoters and enhancers. Promoters are regions of DNA located near the transcription start site(TSS). They contain binding sites for transcription factors that are involved in the initiation of transcription. Enhancers, on the other hand, are distal cis-regulatory elements located away from the gene's promoter and can be located upstream, downstream, or even within introns of a gene. They contain binding sites for transcription factors that can modulate gene expression. The BETA function can give the transcription start site (TSS) information. The TSS is the specific location on the DNA where the transcription of a gene begins. Promoter regions are usually 2kb upstream or downstream of TSS while enhancers are 2~10kb upstream or downstream of TSS. The input of BETA is the peak-calling bed file from MACS2. BETA-minus can output two txt files: “Targets_predicted_by_BETA” demonstrates the TSS, and “Targets_genes_associated_peaks” contains the information of binding sites that associate with each TSS, including the distance from TSS. Promoters regions can be found by filtering out the distances that is smaller than 2000 but greater than -2000. Similarly, enhancers region are the peaks with a distance greater than 2000 or smaller than 2000.

```
###looking for TSS and distinguishing promoters from enhancers###
BindingSite <- read.table("BETA_Targets_genes_associated_peaks.txt", header = TRUE, sep =
'\t', comment.char = "#", check.names = FALSE)
promoters <- BindingSite[which(BindingSite$distance<2000&BindingSite$distance>-2000), ]
enhancers <- BindingSite[which(BindingSite$distance>2000|BindingSite$distance<(-2000)), ]
```

- Demonstrating the genomic distribution of the binding sites

The Liu Lab from Harvard University develops a very powerful web-based server, cistromeAP, to help analyze and visualize the CHIP-seq data. The CEAS function can produce a graphical representation of the detailed genomic distribution of binding sites, such as the percentage of binding sites located in the promoter, enhancer, and other functional genomic regions.

5. Integrative analysis