

1. Configuration file containing:
   1. Sample metadata
   2. Pre-processing options
   3. Analysis and Reporting options
2. Cloud Initialization Automation:
   1. Launch cloud instance with adequate computational resources and specified software based on the Configuration file
   2. Mount necessary storage drives
3. Download any reference files:
   1. wget call to Ensembl API or
   2. Pull genome/index files from other cloud storage location
4. Pre-processing:
   1. Download input files
      1. **gutil cp | aws s3 cp | fastq-dump**
   2. Index reference genomes -> bwa 0.7.17
      1. **bwa index ref.fa**
   3. Trim Adapters -> Cutadapt 4.4
      1. **cutadapt {input\_string} -m 15 -e 0.1 -O 5 --cores {threads}**
         1. {input\_string} = specifying 5’ and/or 3’ adapter sequences, output, and input
         2. -m = discard processed reads that are shorter than LENGTH
         3. -e = maximum ERROR RATE in relation to length of adapter
         4. -O = Minimum overlap between the read and the adapter sequence
         5. –cores = CPU cores dedicated to execution
   4. Quality Filtering -> Trimmomatic 0.40
      1. **trimmomatic {PE} -threads {threads} {input} {output} SLIDINGWINDOW:4:{qual\_cutoff}**
         1. {PE} = single or paired end input
         2. {qual\_cutoff} = specifies the average quality required
   5. Reference Genome Alignment, Sam to Bam and Sorting, Indexing-> bwa 0.7.17; samtools 1.17
      1. **bwa mem -t {threads} {index} {input} > {output}**
      2. **samtools view -@ {threads} -Sbh {input} | samtools sort -@ {threads} > {output}**
      3. **samtools index -@ {threads} {input.bam}**
   6. Read QC -> FastQC 0.12.0
      1. **fastqc**
   7. BAM Alignment Summary Statistics -> RSeQC 5.0.1
      1. **bam\_stat.py**
         1. Summarizing mapping statistics of a BAM or SAM file. This script determines “uniquely mapped reads” from mapping quality, which quality the probability that a read is misplaced
      2. **read\_GC.py**
         1. GC content distribution of reads
   8. CNV Analysis -> QDNASeq (R package)
      1. QDNASeq: Quantitative DNA sequencing for chromosomal aberrations. The genome is divided into non-overlapping fixed-sized bins, number of sequence reads in each counted, adjusted with a simultaneous two-dimensional loess correction for sequence mappability and GC content, and filtered to remove spurious regions in the genome.

**## obtain bin annotations**

**bins <- getBinAnnotations(binSize=15)**

**## Read in sample bam files**

**readCounts <- binReadCounts(bins, bamfiles='{sample}.bam')**

**## Plot raw copy number profile (read counts across the genome) and highlight bins to remove with default filtering**

**plot(readCounts, logTransform=FALSE, ylim=c(-50, 200))**

**highlightFilters(readCounts, logTransform=FALSE, residual=TRUE, blacklist=TRUE)**

**## Apply filters**

**readCountsFiltered <- applyFilters(readCounts, residual=TRUE, blacklist=TRUE)**

**## plot median read counts as a function of GC content and mappability**

**isobarPlot(readCountsFiltered)**

**## Estimate the correction for GC content and mappability**

**readCountsFiltered <- estimateCorrection(readCountsFiltered)**

**## plot for the relationship between the observed standard deviation in the data and its read depth**

**noisePlot(readCountsFiltered)**

**## apply the correction for GC content and mappability then normalize, smooth outliers**

**copyNumbers <- correctBins(readCountsFiltered)**

**copyNumbersNormalized <- normalizeBins(copyNumbers)**

**copyNumbersSmooth <- smoothOutlierBins(copyNumbersNormalized)**

**## Copy number profile after correcting for GC content and mappability**

**plot(copyNumbersSmooth)**

**## export as BED file**

**exportBins(copyNumbersSmooth, file="{sample}.bed", format="bed")**

* 1. Peak Calling -> macs2 2.2.9.1
     1. The peak calling tool MACS can call peaks in either narrow peak mode (for focused signals like transcription factor ChIPseq) or broad peak mode (for more defuse signals, like certain histone modifications).
     2. Narrow

**macs2 callpeak -B --SPMR -q {fdr} --keep-dup {keepdup} \**

**-g {genome\_size} -f {PE} \**

**-t {treatment} -c {control} --outdir {outdir} -n {name}**

* + 1. Broad

**macs2 callpeak -B --SPMR -q {fdr} --keep-dup {keepdup} \  
-g {genome\_size} -f {PE} \  
--broad --broad-cutoff {fdr} --nomodel --extsize {extsize} \  
-t {treatment} -c {control} --outdir {outdir} -n {name}**

* + - 1. -B = store the fragment pileup, control lambda, -log10pvalue and -log10qvalue scores in bedGraph files
      2. --SPMR = Save signal per million reads for fragment pileup profiles.
      3. -q = q-value (minimum FDR) cutoff
      4. --keep-dup = controls the MACS behavior towards duplicate tags at the exact same location
      5. -g = mappable genome size which is defined as the genome size which can be sequenced. Default value of hs (2.7e9) is recommended for human genome
      6. -f = Input format. Specify paired end if needed. Default will try to auto detect
      7. -t = treatment input file name
      8. -c = control genomic input file name
      9. -n = name string of the experiment. MACS will use this string NAME to create output files
      10. --broad = MACS will try to composite broad regions in BED12 ( a gene-model-like format ) by putting nearby highly enriched regions into a broad region with loose cutoff
      11. --broad-cutoff = cutoff for the broad region
      12. --nomodel = while on, MACS will bypass building the shifting model
      13. –extsize = while --nomodel is set, MACS uses this parameter to extend reads in 5'->3' direction to fix-sized fragments

1. Analysis:
   1. Peak Statistics / Scoring -> ChiPQC 1.36.0 (R package)
      1. ChiPQC: Bioconductor package that takes as input BAM files and peak calls to automatically compute a number of quality metrics and generates a ChIPseq experiment quality report

**## Create ChIPQC object**

**chipObj <- ChIPQC({sample\_metadata}, annotation="hg19")**

**## Create ChIPQC report**

**ChIPQCreport(chipObj, reportName={report\_title}, reportFolder={out\_dir})**

* + - 1. Key Output:
         1. Read depth, Read length, and Duplication rate
         2. RiP (Reads in Peaks; also known as FRIP). Percentage of reads that overlap ‘called peaks’.
         3. RiBL (Reads overlapping in Blacklisted Regions). Lower RiBL percentages are better than higher.
         4. SSD (Signal pile up standard deviation). Represents the uniformity of coverage of reads across the genome. A “good” or enriched sample typically has regions of significant read pile-up (larger differences in coverage) so a higher SSD is more indicative of better enrichment.
         5. FragLength, RelCC (Metrics related to the peak signal strength). RelCC values larger than 1 for all ChIP samples suggest good signal-to-noise & the FragL values should be roughly the same as the fragment length you picked in the size selection step during library preparation.
         6. Strand Cross-Correlation. A table of values mapping each base pair shift to a Pearson correlation value. These Pearson correlation values are computed for every peak for each chromosome and values are multiplied by a scaling factor and then summed across all chromosomes. Cross-correlation plot typically produces two peaks: a peak of enrichment corresponding to the predominant fragment length (highest correlation value) and a peak corresponding to the read length (“phantom” peak)
         7. Peak Profile. Peak profile plot shows the average peak profiles, centered on the summit (point of highest pileup) for each peak. Shape of these profiles can vary depending on what type of mark is being studied.
         8. Relative Enrichment of Genomic Intervals (REGI). Using the genomic regions identified as called peaks along with genome annotation information, we can see where reads map in terms of various genomic features.

POTENTIAL TODO:

1. Peak Statistics / Scoring -> deepTools
   * 1. deepTools: A suite of Python tools to process the mapped reads data for multiple quality checks, creating normalized coverage files in standard bedGraph and bigWig file formats, that allow comparison between different files

Create bigWig files:

**bamCoverage -b bam/{sample}.bam \**

**-o bigwig/{sample}.bw \**

**--binSize 20 \**

**--normalizeUsing BPM \**

**--smoothLength 60 \**

**--extendReads 150 \**

**--centerReads \**

**-p 6**

**bamCompare -b1 bam/{sample}.bam \**

**-b2 bowtie2/{control}.bam \**

**-o bigwig/{sample}\_norm.bw \**

**--binSize 20 \**

**--normalizeUsing BPM \**

**--smoothLength 60 \**

**--extendReads 150 \**

**--centerReads \**

**-p 6**

**Create computeMatrix for plotProfile and plotHeatmap**

1. Compare / Combine Sample Replicates (Optional) -> idr 2.0.3
   * 1. The IDR method compares a pair of ranked lists of identifications (such as ChIP-seq peaks). These ranked lists should not be pre-thresholded i.e. they should provide identifications across the entire spectrum of high confidence/enrichment (signal) and low confidence/enrichment (noise). The IDR method then fits the bivariate rank distributions over the replicates in order to separate signal from noise based on a defined confidence of rank consistency and reproducibility of identifications i.e the IDR threshold.
     2. **idr --samples {sample}\_rep1.narrowPeak {sample}\_rep2.narrowPeak \**

**--input-file-type narrowPeak \**

**--rank p.value \**

**--output-file {sample}\_idr \**

**--plot \**

**--log-output-file {sample}.idr.log**

* + - 1. –-rank = which column to use to rank peaks. Usually p.value
      2. -–plot = plot results .png
         1. Upper Left: Replicate 1 peak ranks versus Replicate 2 peak ranks - peaks that do not pass the specified idr threshold are colored red.
         2. Upper Right: Replicate 1 log10 peak scores versus Replicate 2 log10 peak scores - peaks that do not pass the specified idr threshold are colored red.
         3. Bottom Row: Peak rank versus IDR scores are plotted in black. The overlayed boxplots display the distribution of idr values in each 5% quantile. The IDR values are thresholded at the optimization precision - 1e-6 by default.

1. Exploratory Analysis -> Diffbind (R package); ChIPseeker (R package); Gvis (R package)
   * 1. Diffbind: Bioconductor package that is used for identifying sites that are differentially enriched between two or more sample groups. It works primarily with sets of peak calls (‘peaksets’), which are sets of genomic intervals representing candidate protein binding sites for each sample. It includes functions that support the processing of peaksets, including overlapping and merging peak sets across an entire dataset, counting sequencing reads in overlapping intervals in peak sets, and identifying statistically significantly differentially bound sites based on evidence of binding affinity (measured by differences in read densities)

**## Read in sample metadata and create object**

**samples <- read.csv('meta/samplesheet\_chr12.csv')**

**dbObj <- dba(sampleSheet=samples)**

**## Compute count information for each of the peaks/regions**

**dbObj <- dba.count(dbObj, bUseSummarizeOverlaps=TRUE)**

**## PCA plot to see how well the samples cluster with one another**

**dba.plotPCA(dbObj, attributes=DBA\_FACTOR, label=DBA\_ID)**

**## Correlation heatmap to evaluate the relationship between samples**

**plot(dbObj)**

* + 1. ChIPseeker: package implements functions to retrieve the nearest genes around the peak, annotate genomic region of the peak, statstical methods for estimate the significance of overlap among ChIP peak data sets, and incorporate GEO database for user to compare the own dataset with those deposited in database

**## Read in sample peak BED file**

**peak <- readPeakFile({sample}\_peaks.{narrow|broad}Peak)**

**## Calculate coverage of peak regions over chromosomes and generate coverage figure**

**covplot(peak, weightCol="V5")**

* + 1. Gviz: generation of tracks which can be, for example ChIP-seq signal along the genome, ChIP-seq peaks, gene models or any kind of other data / genome annotations

**## Read in peak BED files then**

**## plot tracks along with genomic features**

**plotTracks(c(input.track, rep1.track, rep2.track, bm, AT),**

**from=122530000, to=122900000,**

**transcriptAnnotation="symbol", window="auto",**

**type="histogram", cex.title=0.7, fontsize=10)**

1. Differential Enrichment Analysis -> DiffBind (R package)
   * 1. The core functionality of DiffBind is the differential binding affinity analysis, which enables binding sites to be identified that are statistically significantly differentially bound between sample groups. The core analysis routines are executed, by default using DESeq2 with an option to also use edgeR. Each tool will assign a p-value and FDR to each candidate binding site indicating confidence that they are differentially bound.

**## Calculate contrast on certain measure**

**dbObj <- dba.contrast(dbObj, categories=DBA\_FACTOR, minMembers = 2)**

**or**

**dbObj <- dba.contrast(dbObj, contrast=c("Condition","Responsive","Resistant"))**

**## Perform differential analysis using both methods (DESeq2 and edgeR)**

**dbObj <- dba.analyze(dbObj, method=DBA\_ALL\_METHODS)**

**## Plotting a PCA but this time only use the regions that were identified as significant by DESeq2**

**dba.plotPCA(dbObj, contrast=1, method=DBA\_DESEQ2, attributes=DBA\_FACTOR, label=DBA\_ID)**

**## MA plots to visualize which of the data points are being identified as differentially bound. Plot FC**

**dba.plotMA(dbObj, method=DBA\_DESEQ2)**

**## MA plot of concentrations of each sample groups plotted against each other**

**dba.plotMA(dbObj, bXY=TRUE)**

**## Box plot see how the reads are distributed amongst the different classes of differentially bound sites and sample groups**

**dba.plotBox(dbObj)**

**## Table of full results from DESeq2**

**dba.report(dbObj, method=DBA\_DESEQ2, contrast = 1, th=1)**

1. Annotation and Functional Analysis -> ChIPseeker (R package); clusterProfiler (R package)
   * 1. ChIPseeker: supports annotating ChIP-seq data of a wide variety of species if they have transcript annotation. ChIPseeker package, uses the nearest gene method described above but also provides parameters to specify a max distance from the TSS. For annotating genomic regions, annotatePeak will not only give the gene information but also reports detail information when genomic region is Exon or Intron.

**## Perform annotation on sample peak BED files**

**library(EnsDb.Hsapiens.v75)**

**edb <- EnsDb.Hsapiens.v75**

**seqlevelsStyle(edb) <- "UCSC"**

**peakAnno <- annotatePeak(samplefiles[[1]], tssRegion=c(-3000, 3000),**

**TxDb=edb, annoDb="org.Hs.eg.db")**

**## Pie and Bar chart of genomic feature representation. Created multi panel with list obj**

**plotAnnoPie(peakAnno)**

**plotAnnoBar(peakAnno)**

**## View full annotation with overlap using combined vennpie with upsetplot**

**upsetplot(peakAnno, vennpie=TRUE)**

**## Distribution of TF-binding loci relative to TSS**

**plotDistToTSS(peakAnno, title="Distribution of TF-binding loci \n relative to TSS")**

**## Compare the number overlapping genes associated with peaks of replicate experiments or from different experiments**

**vennplot(genes)**

* + 1. clusterProfiler: Functional Enrichment analysis is a widely used approach to identify biological themes. Gene Ontology and KEGG enrichment analysis. Can use associated packages DOSE (Yu et al. 2015) for Disease Ontology, ReactomePA for reactome pathway.

**## Annotate sample peak**

**peakAnno <- annotatePeak(samplefiles[[1]], tssRegion=c(-3000, 3000),**

**TxDb=edb, annoDb="org.Hs.eg.db")**

**## GO enrichment clustering**

**ego <- enrichGO(gene = as.data.frame(peakAnno)$geneId,**

**keyType = "ENTREZID",**

**OrgDb = org.Hs.eg.db,**

**ont = "BP",**

**pAdjustMethod = "BH",**

**qvalueCutoff = 0.05,**

**readable = TRUE)**

**## KEGG pathway enrichment clustering**

**ekegg <- enrichKEGG(gene = as.data.frame(peakAnno)$geneId,**

**organism = 'hsa',**

**pvalueCutoff = 0.05)**

**## output tables and dot plots**

**View(ego)**

**dotplot(ego)**

**## Compare functional enrichment across samples. Gather gene ids**

**genes = lapply(peakAnnoList, function(i) as.data.frame(i)$geneId)**

**## Perform comparison of biological themes among gene clusters**

**compKEGG <- compareCluster(geneCluster = genes,**

**fun = "enrichKEGG",**

**pvalueCutoff = 0.05,**

**pAdjustMethod = "BH")**

**## Dot plot showing comparisons**

**dotplot(compKEGG, showCategory = 15, title = "KEGG Pathway Enrichment Analysis")**

1. Motif Analysis -> MEME
   * 1. MEME-ChIP: tool that is part of the MEME Suite that is specifically designed for ChIP-seq analyses. MEME-ChIP performs DREME and Tomtom analysis in addition to using tools to assess which motifs are most centrally enriched (motifs should be centered in the peaks) and to combine related motifs into similarity clusters