RNA Seq, ATAC-seq & ArchR

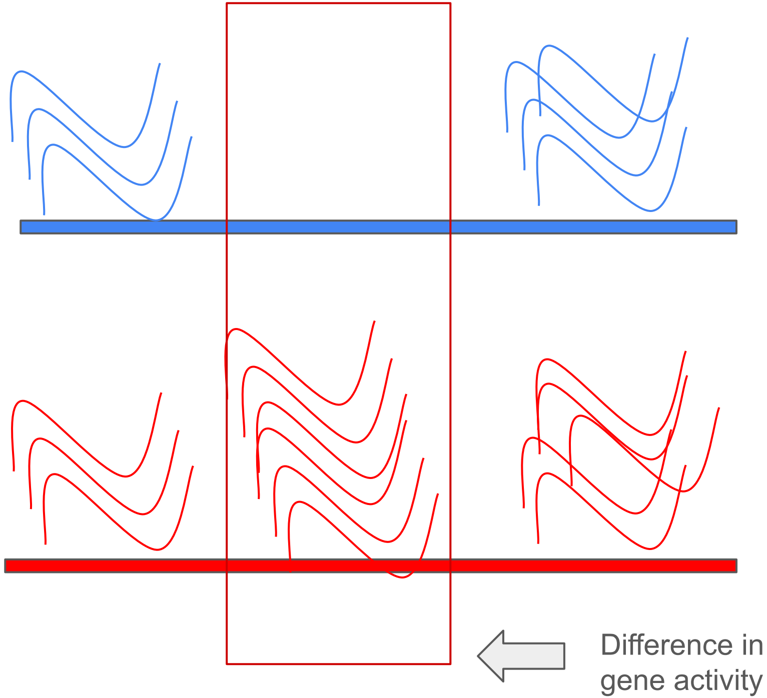
Varshini Vijay

High throughput sequencing

High-throughput or next-generation sequencing (NGS) allows for the rapid sequencing of millions of sequences of DNA and RNA. (Illumina)

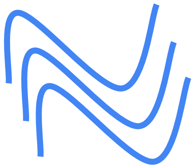
- RNA Sequencing
- ATAC Sequencing

Overview



RNA Sequencing can be applied to normal and mutant cells to identify differences in gene expression.

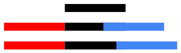
(1) Prepare RNA Seq library



1. Isolate RNA

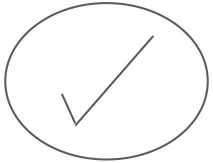
2. Break RNA into smaller fragments

3. Convert RNA into DNA which is more stable and easier to modify



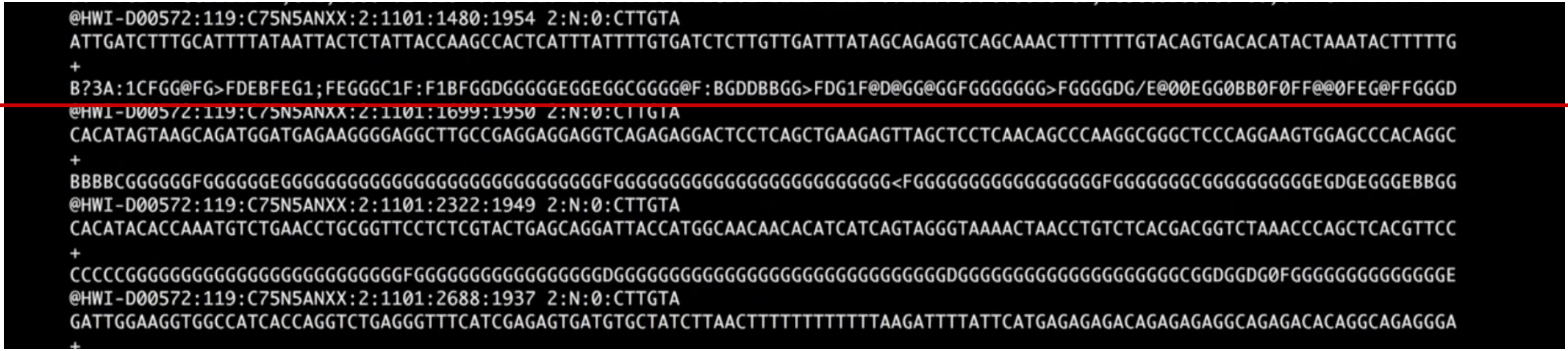
4. Add sequencing adaptors which allows the sequencing machine to recognize fragments

5. PCR Amplify (enhancement)

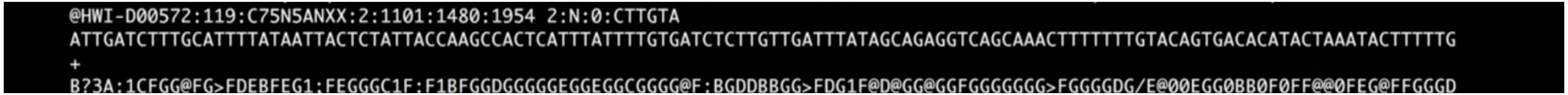


6. Quality control

Raw Data



```
@HWI-D00572:119:C75N5ANXX:2:1101:1480:1954 2:N:0:CTTGTA
ATTGATCTTTGCATTTTATAATTACTCTATTACCAAGCCACTCATTTATTTTGTGATCTCTTGTTGATTTATAGCAGAGGTCAGCAAACTTTTTTTGTACAGTGACACATACTAAATACTTTTTG
+
B?3A:1CFGG@FG>FDEBFEG1;FEGGGC1F:F1BFGGDGGGGGEGGEGGCGGGG@F:BGDDBBGG>FDG1F@D@GG@GGFGGGGGGG>FGGGGDG/E@00EGG0BB0F0FF@@0FEG@FFGGGD
@HWI-D00572:119:C75N5ANXX:2:1101:1699:1950 2:N:0:CTTGTA
CACATAGTAAGCAGATGGATGAGAAGGGGAGGCTTGCCGAGGAGGAGGTCAGAGAGGACTCCTCAGCTGAAGAGTTAGCTCCTCAACAGCCCAAGGCGGGCTCCCAGGAAGTGGAGCCCACAGGC
+
BBBBCGGGGGGFGGGGGGEGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGFGGGGGGGGGGGGGGGGGGGGGGGGGG<FGGGGGGGGGGGGGGGGGFGGGGGGGCGGGGGGGGGGEGDGEGGGEBBGG
@HWI-D00572:119:C75N5ANXX:2:1101:2322:1949 2:N:0:CTTGTA
CACATACACCAAATGTCTGAACCTGCGGTTCCTCTCGTACTGAGCAGGATTACCATGGCAACAACACATCATCAGTAGGGTAAAACTAACCTGTCTCACGACGGTCTAAACCCAGCTCACGTTCC
+
CCCCCGGGGGGGGGGGGGGGGGGGGGGGGGFGGGGGGGGGGGGGGGGGDGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGDGGGGGGGGGGGGGGGGGGGGCGGDGGDG0FGGGGGGGGGGGGGGE
@HWI-D00572:119:C75N5ANXX:2:1101:2688:1937 2:N:0:CTTGTA
GATTGGAAGGTGGCCATCACCAGGTCTGAGGGTTTCATCGAGAGTGATGTGCTATCTTAACTTTTTTTTTTTTAAGATTTTATTCATGAGAGAGACAGAGAGAGGCAGAGACACAGGCAGAGGGA
+
```



```
@HWI-D00572:119:C75N5ANXX:2:1101:1480:1954 2:N:0:CTTGTA
ATTGATCTTTGCATTTTATAATTACTCTATTACCAAGCCACTCATTTATTTTGTGATCTCTTGTTGATTTATAGCAGAGGTCAGCAAACTTTTTTTGTACAGTGACACATACTAAATACTTTTTG
+
B?3A:1CFGG@FG>FDEBFEG1;FEGGGC1F:F1BFGGDGGGGGEGGEGGCGGGG@F:BGDDBBGG>FDG1F@D@GG@GGFGGGGGGG>FGGGGDG/E@00EGG0BB0F0FF@@0FEG@FFGGGD
```

Blueprint

(1) <unique sequence ID>

(2) <corresponding nucleotide sequence>

(3) +

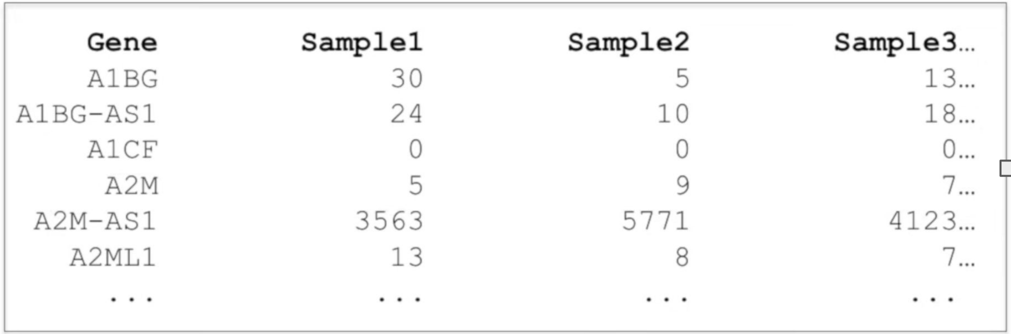
(4) <quality of base>

(2) Filter garbage reads

- Reads with low quality base cells
- Reads that are artifacts of two adapters binded to one another

(3) Align reads to genome

- Once chromosome and position for a read is determined, one can check if it falls within the coordinates of a gene



Gene	Sample1	Sample2	Sample3...
A1BG	30	5	13...
A1BG-AS1	24	10	18...
A1CF	0	0	0...
A2M	5	9	7...
A2M-AS1	3563	5771	4123...
A2ML1	13	8	7...
...	...	...	...

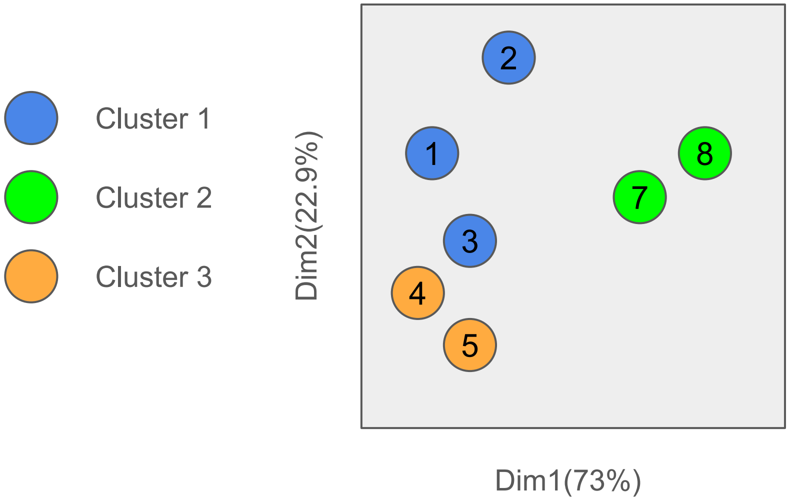
Columns contain counts (number of times an RNA sequence is detected) per sample sequenced.

(4) Normalize data

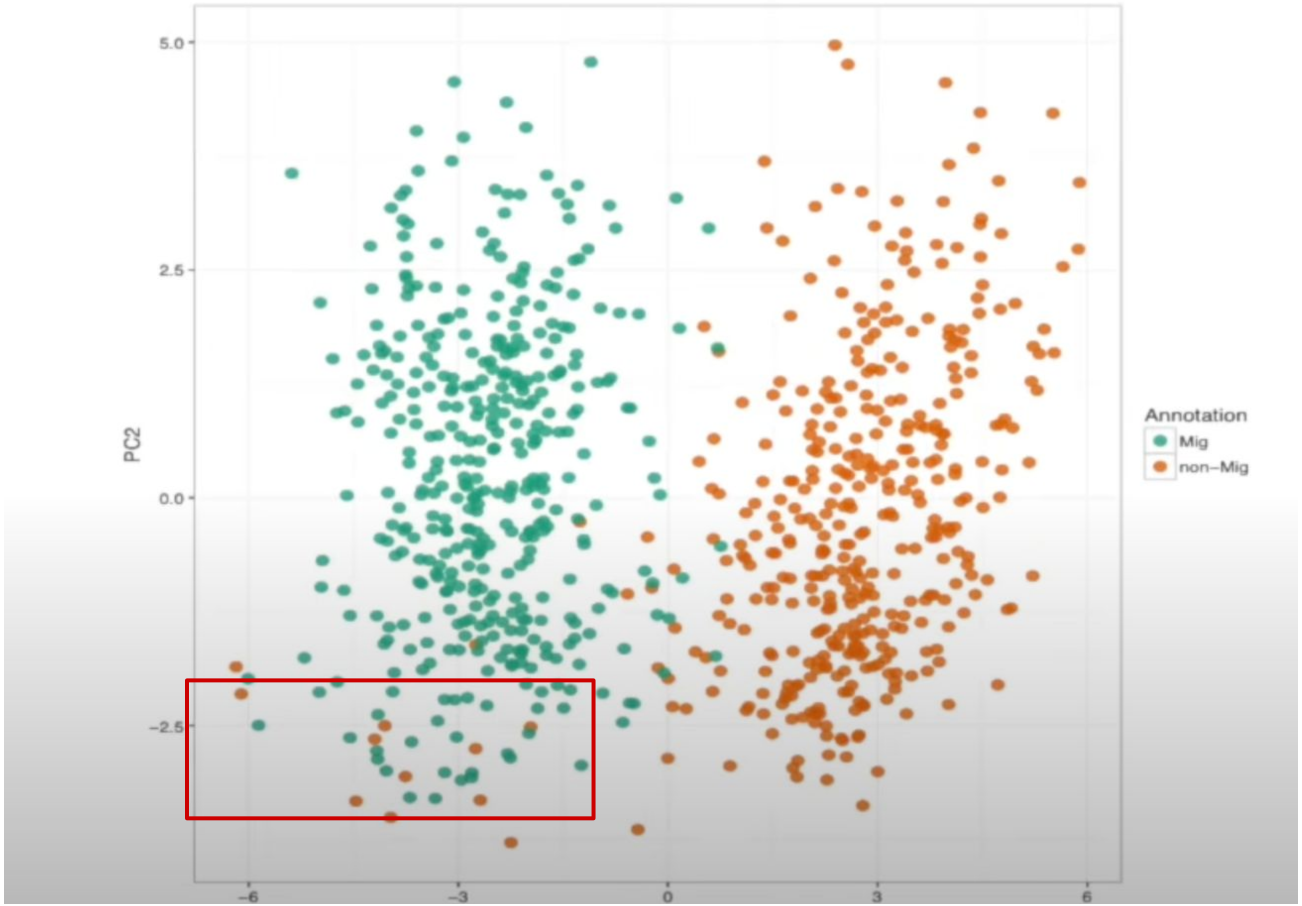
- Adjust reads per count to reflect differences in reads assigned per sample

(5) Plot data

- Principal component analysis can graph 4+ samples into 2D graph



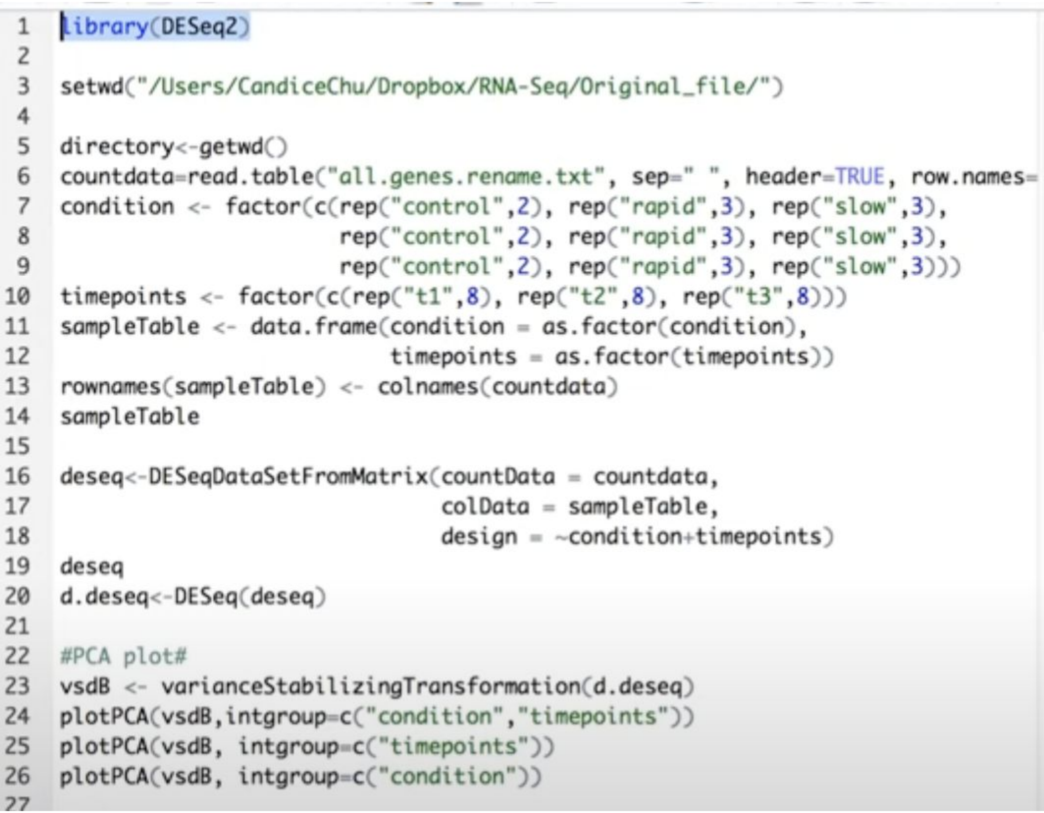
- **Dim1 (73%)**: This axis captures the most significant variance in the data (73%).
- **Dim2 (22.9%)**: This axis captures the next most significant variance (22.9%).
- Cluster 1 and 3 overlap, showing they are more similar to each other than to Cluster 2.



Takeaways

1. Green dots represent mutant cells and orange dots represent non-mutant cells
2. Separate clusters indicate differences in trends
3. May need to exclude highlighted data

DESeq2

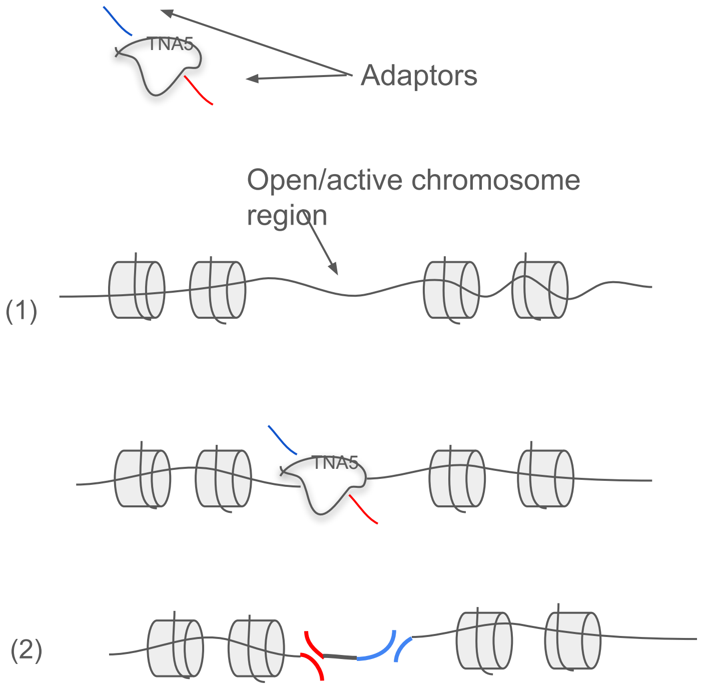


```
library(DESeq2)

setwd("/Users/CandiceChu/Dropbox/RNA-Seq/Original_file/")

directory<-getwd()
countdata=read.table("all.genes.rename.txt", sep=" ", header=TRUE, row.names=
condition <- factor(c(rep("control",2), rep("rapid",3), rep("slow",3),
                      rep("control",2), rep("rapid",3), rep("slow",3),
                      rep("control",2), rep("rapid",3), rep("slow",3)))
timepoints <- factor(c(rep("t1",8), rep("t2",8), rep("t3",8)))
sampleTable <- data.frame(condition = as.factor(condition),
                          timepoints = as.factor(timepoints))
rownames(sampleTable) <- colnames(countdata)
sampleTable

deseq<-DESeqDataSetFromMatrix(countData = countdata,
                              colData = sampleTable,
                              design = ~condition+timepoints)
deseq
d.deseq<-DESeq(deseq)

#PCA plot#
vsdB <- varianceStabilizingTransformation(d.deseq)
plotPCA(vsdB,intgroup=c("condition","timepoints"))
plotPCA(vsdB, intgroup=c("timepoints"))
plotPCA(vsdB, intgroup=c("condition"))

```

Source: Candice Chu, Sanbomics

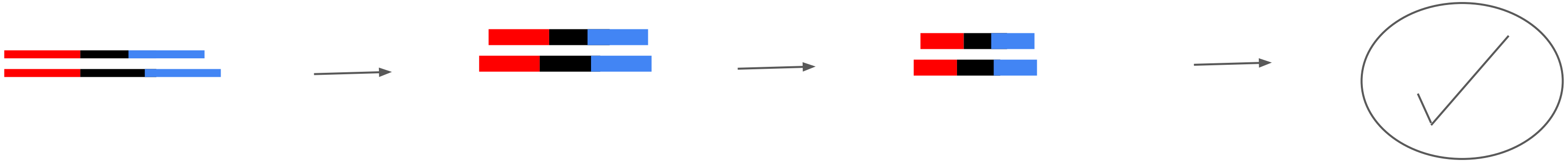
ATAC-Seq

- The assay for transposase-accessible chromatin with sequencing (ATAC-Seq) can determine chromatin accessibility across the genome.
- NGS adapters are loaded onto the TNA5 transposase, which allows simultaneous fragmentation of chromatin and integration of those adapters into open chromatin regions.



(1) TNA5 enzyme binds to open chromatin regions and cuts the DNA. (2) TNA5 inserts sequencing adapters into the DNA.

(1) Library Preparation



Fragmented DNA with adaptors

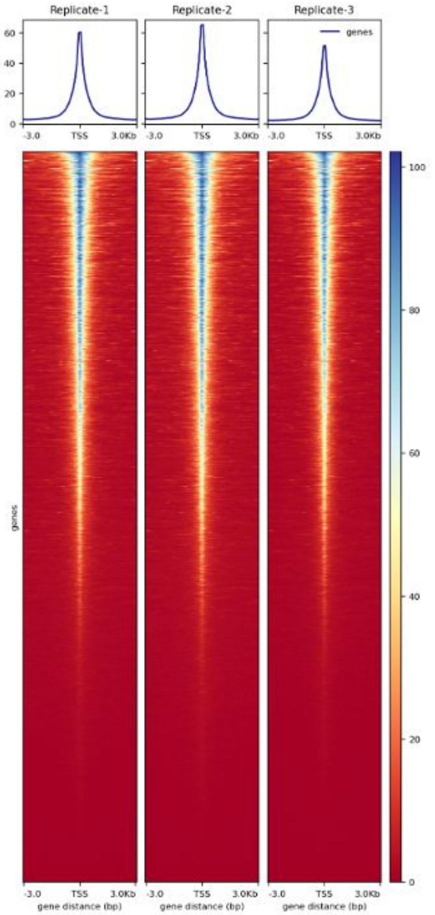
Library amplification

Size selection

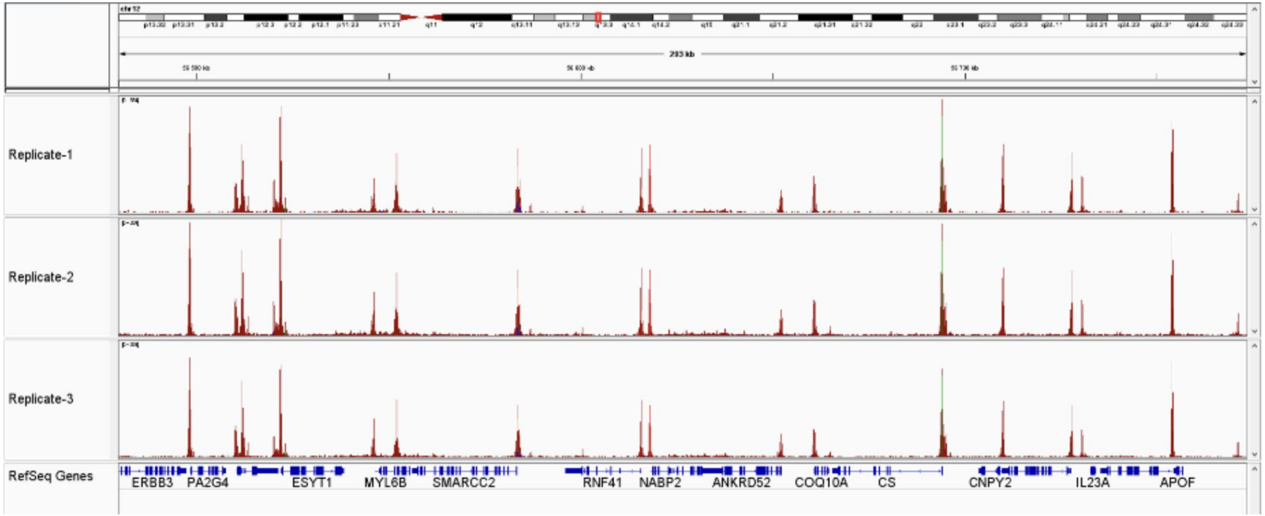
Quality control

(2) Differential Peak Analysis

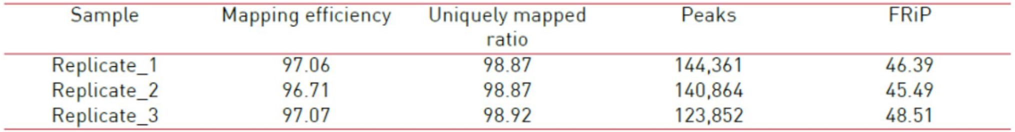
- **Consensus Peak-Based Tools**:
  - Assume negative binomial distribution and need biological replicates.
  - Generate peaks by pooling samples, intersection, or union operations.
- **Sliding Window Approaches**:
  - Evaluate all genome window
  - Need stringent filtering to reduce false positives.



Heatmaps that use color graduation to identify gene activity.

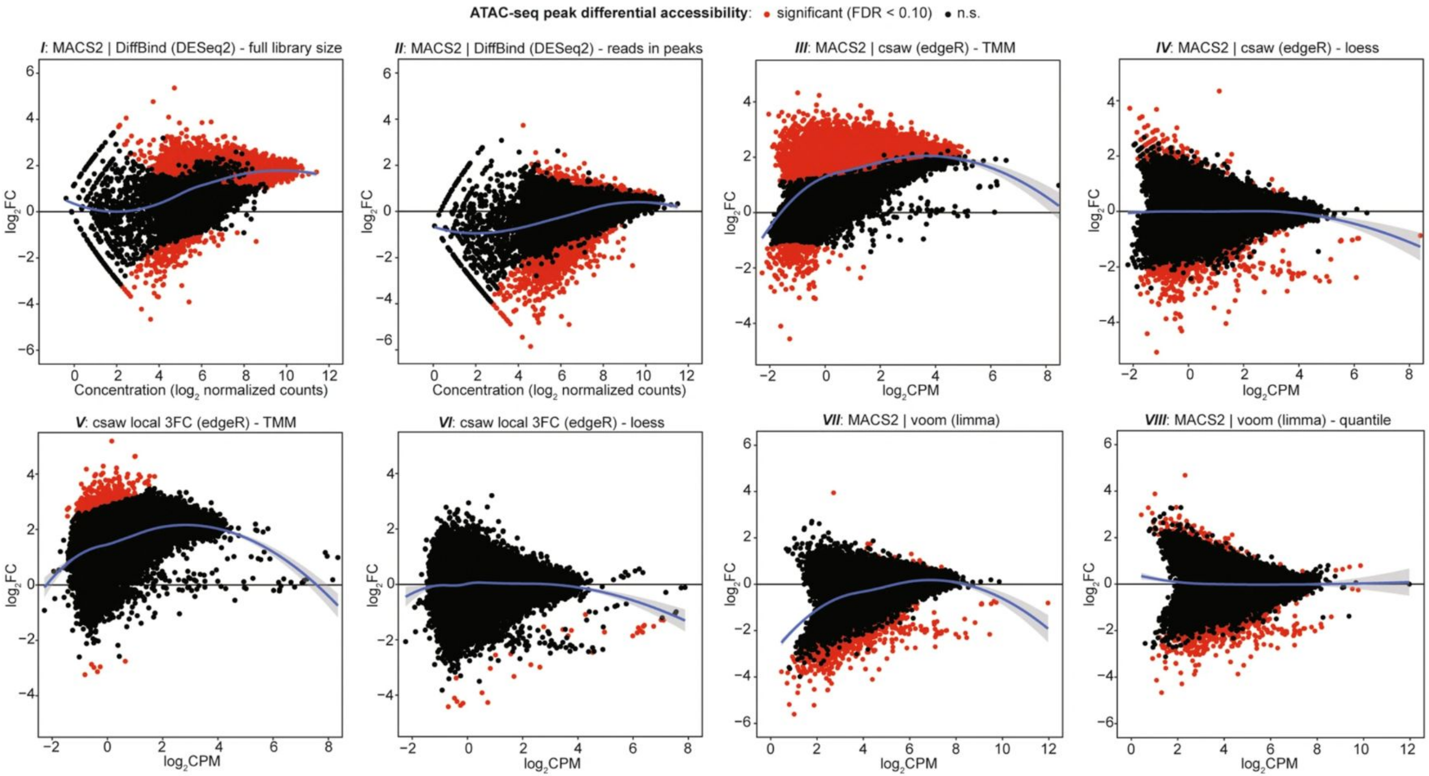


Bottom track shows the location of genes. Peaks indicate regions of accessible chromatin.



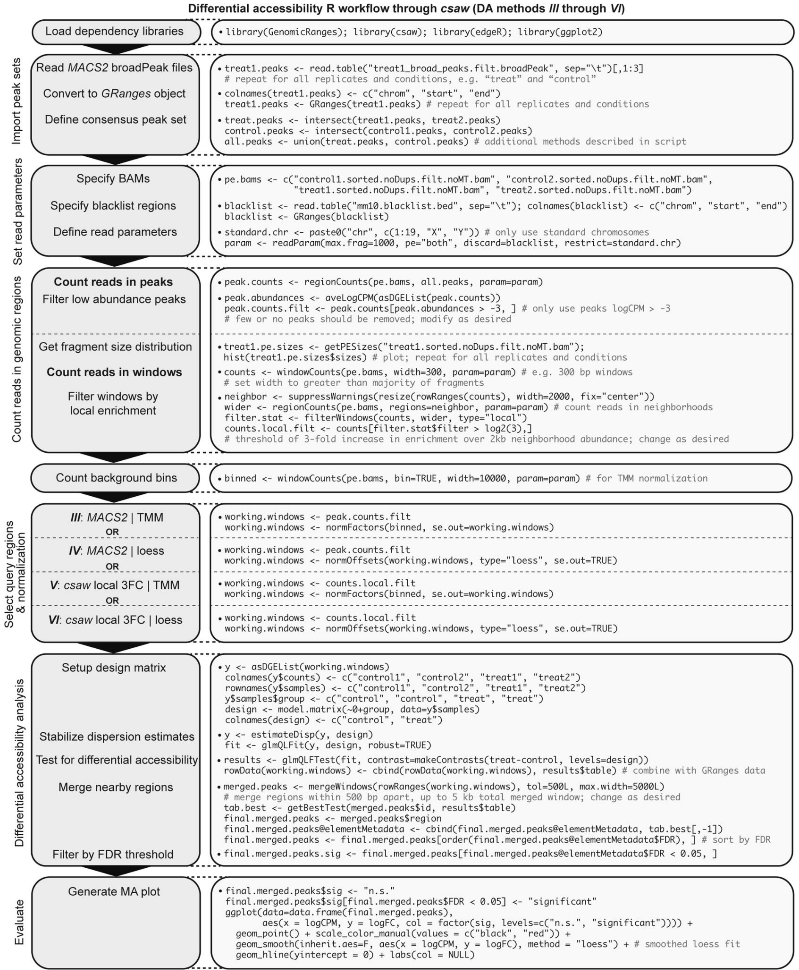
Sample	Mapping efficiency	Uniquely mapped ratio	Peaks	FRiP
Replicate_1	97.06	98.87	144,361	46.39
Replicate_2	96.71	98.87	140,864	45.49
Replicate_3	97.07	98.92	123,852	48.51

(1) The percentage of reads that mapped to the reference genome. (2) The proportion of mapped reads that map to a single location on the genome. (3) The number of peaks (regions of open chromatin) in each sample (4) (Fraction of Reads in Peaks): The percentage of total reads that fall within the identified peaks.



MA plot X-axis represents average ATAC signal abundance at that region, while Y-axis is the log2 difference in ATAC signal between the two conditions. Black dots represent non-significant regions, and red dots represent significant (FDR < 0.10) DA regions. Blue lines are loess fits to each distribution with 95% confidence intervals shaded in gray.

R program manual



Differential accessibility R workflow through *csaw* (DA methods *III* through *VI*)

Load dependency libraries
Import peak sets
Read MACS2 broadPeak files
Convert to GRanges object
Define consensus peak set
Set read parameters
Specify BAMs
Specify blacklist regions
Define read parameters
Count reads in genomic regions
Count reads in peaks
Filter low abundance peaks
Get fragment size distribution
Count reads in windows
Filter windows by local enrichment
Count background bins
Select query regions & normalization
III: MACS2 | TMM
OR
IV: MACS2 | loess
OR
V: csaw local 3FC | TMM
OR
VI: csaw local 3FC | loess
Differential accessibility analysis
Setup design matrix
Stabilize dispersion estimates
Test for differential accessibility
Merge nearby regions
Filter by FDR threshold
Evaluate
Generate MA plot

```
• library(GenomicRanges); library(csaw); library(edgeR); library(ggplot2)
```

```
• treat1.peaks <- read.table("treat1_broad_peaks.filt.broadPeak", sep="\t")[,1:3]
  # repeat for all replicates and conditions, e.g. “treat” and “control”
• colnames(treat1.peaks) <- c("chrom", "start", "end")
  treat1.peaks <- GRanges(treat1.peaks) # repeat for all replicates and conditions
• treat.peaks <- intersect(treat1.peaks, treat2.peaks)
  control.peaks <- intersect(control1.peaks, control2.peaks)
  all.peaks <- union(treat.peaks, control.peaks) # additional methods described in script
```

```
• pe.bams <- c("control1.sorted.noDups.filt.noMT.bam", "control2.sorted.noDups.filt.noMT.bam",
               "treat1.sorted.noDups.filt.noMT.bam", "treat2.sorted.noDups.filt.noMT.bam")
• blacklist <- read.table("mm10.blacklist.bed", sep="\t"); colnames(blacklist) <- c("chrom", "start", "end")
  blacklist <- GRanges(blacklist)
• standard.chr <- paste0("chr", c(1:19, "X", "Y")) # only use standard chromosomes
  param <- readParam(max.frag=1000, pe="both", discard=blacklist, restrict=standard.chr)
```

```
• peak.counts <- regionCounts(pe.bams, all.peaks, param=param)
• peak.abundances <- aveLogCPM(asDGEList(peak.counts))
  peak.counts.filt <- peak.counts[peak.abundances > -3, ] # only use peaks logCPM > -3
  # few or no peaks should be removed; modify as desired
```

```
• treat1.pe.sizes <- getPESizes("treat1.sorted.noDups.filt.noMT.bam");
  hist(treat1.pe.sizes$sizes) # plot; repeat for all replicates and conditions
• counts <- windowCounts(pe.bams, width=300, param=param) # e.g. 300 bp windows
  # set width to greater than majority of fragments
• neighbor <- suppressWarnings(resize(rowRanges(counts), width=2000, fix="center"))
  wider <- regionCounts(pe.bams, regions=neighbor, param=param) # count reads in neighborhoods
  filter.stat <- filterWindows(counts, wider, type="local")
  counts.local.filt <- counts[filter.stat$filter > log2(3),]
  # threshold of 3-fold increase in enrichment over 2kb neighborhood abundance; change as desired
```

```
• binned <- windowCounts(pe.bams, bin=TRUE, width=10000, param=param) # for TMM normalization
```

```
• working.windows <- peak.counts.filt
  working.windows <- normFactors(binned, se.out=working.windows)
```

```
• working.windows <- peak.counts.filt
  working.windows <- normOffsets(working.windows, type="loess", se.out=TRUE)
```

```
• working.windows <- counts.local.filt
  working.windows <- normFactors(binned, se.out=working.windows)
```

```
• working.windows <- counts.local.filt
  working.windows <- normOffsets(working.windows, type="loess", se.out=TRUE)
```

```
• y <- asDGEList(working.windows)
  colnames(y$counts) <- c("control1", "control2", "treat1", "treat2")
  rownames(y$samples) <- c("control1", "control2", "treat1", "treat2")
  y$samples$group <- c("control", "control", "treat", "treat")
  design <- model.matrix(~0+group, data=y$samples)
  colnames(design) <- c("control", "treat")
• y <- estimateDisp(y, design)
  fit <- glmQLFit(y, design, robust=TRUE)
• results <- glmQLFTest(fit, contrast=makeContrasts(treat-control, levels=design))
  rowData(working.windows) <- cbind(rowData(working.windows), results$table) # combine with GRanges data
• merged.peaks <- mergeWindows(rowRanges(working.windows), tol=500L, max.width=5000L)
  # merge regions within 500 bp apart, up to 5 kb total merged window; change as desired
  tab.best <- getBestTest(merged.peaks$id, results$table)
  final.merged.peaks <- merged.peaks$region
  final.merged.peaks@elementMetadata <- cbind(final.merged.peaks@elementMetadata, tab.best[,-1])
  final.merged.peaks <- final.merged.peaks[order(final.merged.peaks@elementMetadata$FDR), ] # sort by FDR
• final.merged.peaks.sig <- final.merged.peaks[final.merged.peaks@elementMetadata$FDR < 0.05, ]
```

```
• final.merged.peaks$sig <- "n.s."
  final.merged.peaks$sig[final.merged.peaks$FDR < 0.05] <- "significant"
  ggplot(data=data.frame(final.merged.peaks),
         aes(x = logCPM, y = logFC, col = factor(sig, levels=c("n.s.", "significant")))) +
    geom_point() + scale_color_manual(values = c("black", "red")) +
    geom_smooth(inherit.aes=F, aes(x = logCPM, y = logFC), method = "loess") + # smoothed loess fit
    geom_hline(yintercept = 0) + labs(col = NULL)
```

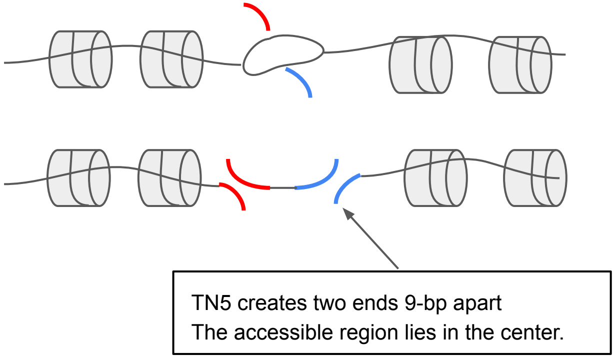
ArchR

The concepts in the following slides briefly cover topics surrounding data analysis in ArchR. For a more in-depth review, refer to the following document: https://docs.google.com/document/d/130NC8BjWexVGTI-oYMwNZQI4kUmAN3EXWekp-LqqXEM/edit?usp=sharing

Chapter 1: Getting Started with ArchR

Fragment: sequenced DNA molecule

- TN5 transposase binds to the DNA with 9-bp between the Tn5 molecules
- To find the center of the accessible region, an offset is applied where plus-stranded insertion events by +4 bp and minus-stranded insertion events by -5 bp

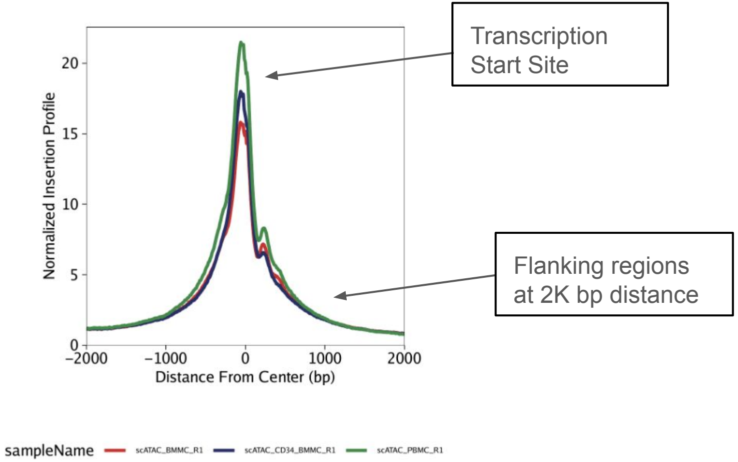


TN5 creates two ends 9-bp apart
The accessible region lies in the center.

ArchRProject: relates arrow files
Arrow file: stores data associated with an individual sample

TSS Enrichment Score

- By looking at per-basepair accessibility centered at these TSS regions, we see a local enrichment relative to flanking regions (1900-2000 bp distal in both directions).
- The ratio between the peak of this enrichment (centered at the TSS) relative to these flanking regions represents the TSS enrichment score.



Transcription Start Site

Flanking regions at 2K bp distance

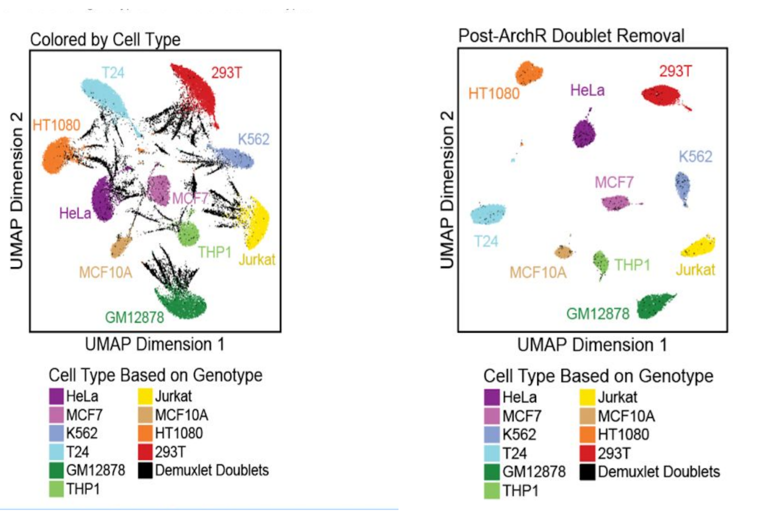
Chapter 2: Doublet Inference with ArchR

Doubtlet

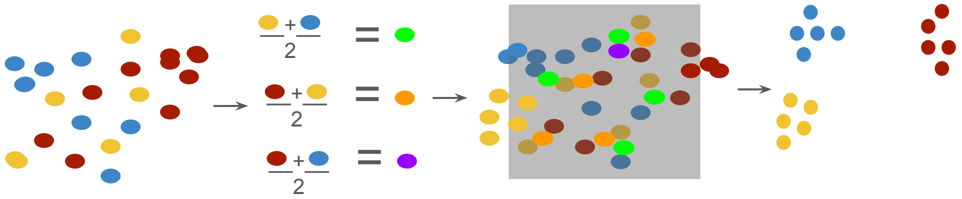
- refers to a single droplet that received a single barcoded bead and more than one nucleus
- causes the reads from more than one cell to appear as a single cell that is effectively the average of the two cells.

Detect doublets using KNN-based inference

- Metrics for quality control:
    - Doublet enrichments: enrichment of simulated doublets nearby each single cell
    - Doublet scores: significance of simulated doublets nearby each single cell `-log10(binomial adjusted p-value)`
    - Doublet density: density of the simulated doublet projections

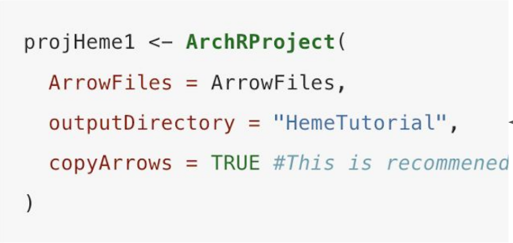


Before and after doublet removal data analysis



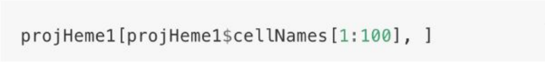
Doubtlet removal procedural explanation

Chapter 3: Creating an ArchR Project



```
projHeme1 <- ArchRProject(
  ArrowFiles = ArrowFiles,
  outputDirectory = "HemeTutorial",
  copyArrows = TRUE #This is recommened
)
```

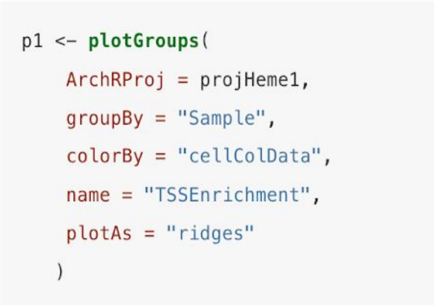
Create an ArchR Project “projHeme1”



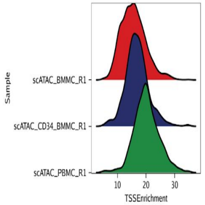
```
projHeme1[projHeme1$cellNames[1:100], ]
```

Subset the project based on certain cell names

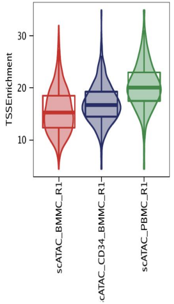
Set ridge or violet plot by altering the “plotAs” in plotGroups function



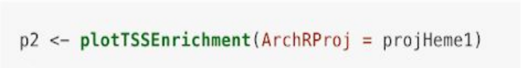
```
p1 <- plotGroups(
    ArchRProj = projHeme1,
    groupBy = "Sample",
    colorBy = "cellColData",
    name = "TSSEnrichment",
    plotAs = "ridges"
   )
```



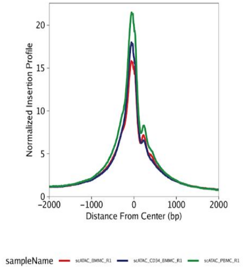
Ridge plot

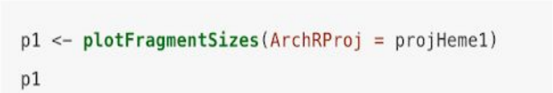


Violet plot

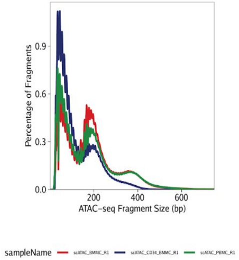


```
p2 <- plotTSSEnrichment(ArchRProj = projHeme1)
```





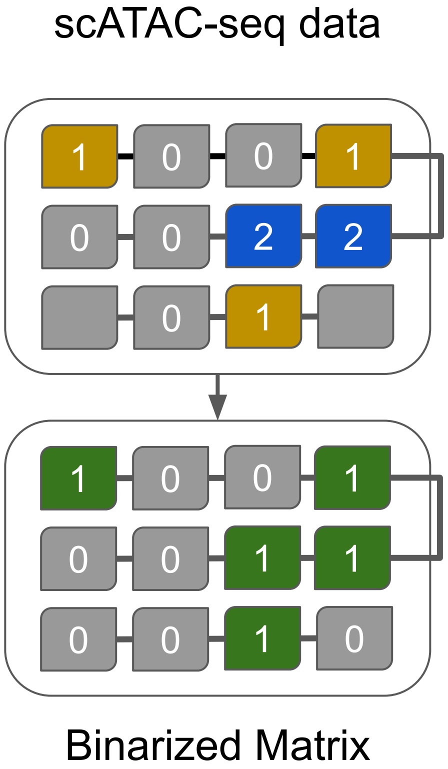
```
p1 <- plotFragmentSizes(ArchRProj = projHeme1)
p1
```

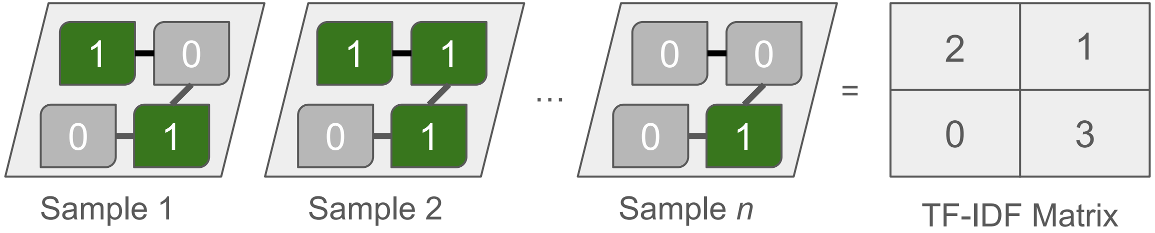


TSS enrichment profiles should show a clear peak in the center and a smaller shoulder peak right-of-center which is caused by the well-positioned +1 nucleosome.

Chapter 4: Dimensionality Reduction in ArchR

- 0 in scATAC-seq could mean “non-accessible” or “not sampled” and these two inferences are very different from a biological standpoint. This low information content is what makes our scATAC-seq data *sparse*.
- Latent Semantic Indexing (LSI): reduces the dimensionality of the sparse insertion counts matrix
    - Calculate term frequency by depth normalization per single cell.
    - Normalize values by the inverse document frequency
        - The resultant (TF-IDF) matrix reflects how important a word (aka region/peak) is to a document (aka sample).
    - Singular value decomposition (SVD): the most valuable information across samples is identified and represented in a lower dimensional space



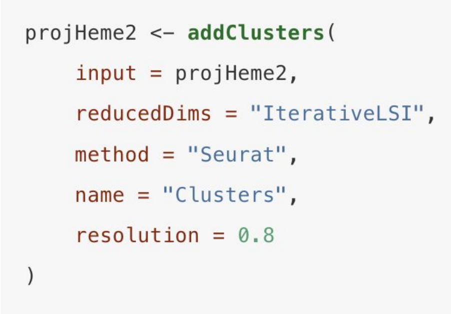


2	1
0	3

TF-IDF Matrix

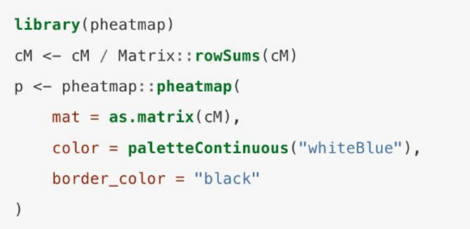
Chapter 5: Clustering with ArchR

Most single-cell clustering methods focus on computing nearest neighbor graphs in reduced dimensions and then identifying “communities” or clusters of cells.



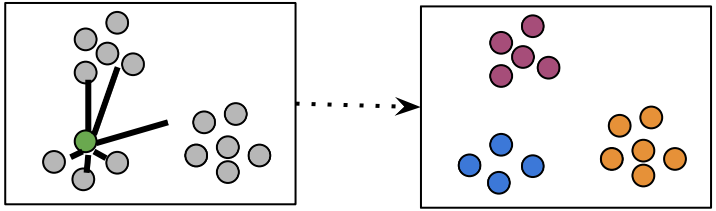
```
projHeme2 <- addClusters(
    input = projHeme2,
    reducedDims = "IterativeLSI",
    method = "Seurat",
    name = "Clusters",
    resolution = 0.8
)
```

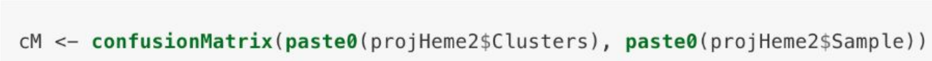
Performs clustering



```
library(pheatmap)
cM <- cM / Matrix::rowSums(cM)
p <- pheatmap::pheatmap(
    mat = as.matrix(cM),
    color = paletteContinuous("whiteBlue"),
    border_color = "black"
)
```

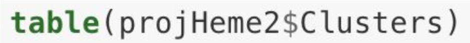
Plots confusion matrix as heatmap





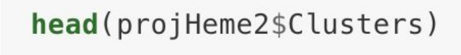
```
cM <- confusionMatrix(paste0(projHeme2$Clusters), paste0(projHeme2$Sample))
```

Creates a cluster confusion matrix



```
table(projHeme2$Clusters)
```

Tabulates number of cells in each cluster



```
head(projHeme2$Clusters)
```

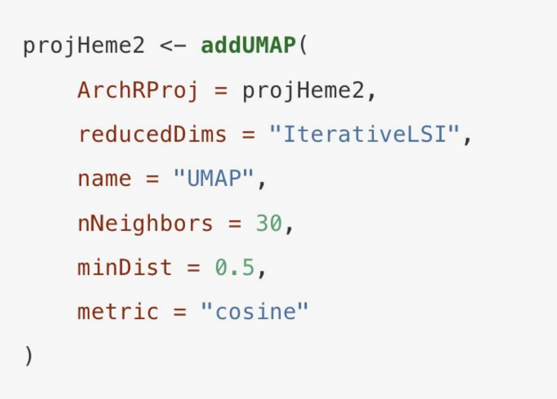
Shows the cluster ID for each cell

Chapter 6: Single-Cell Embeddings

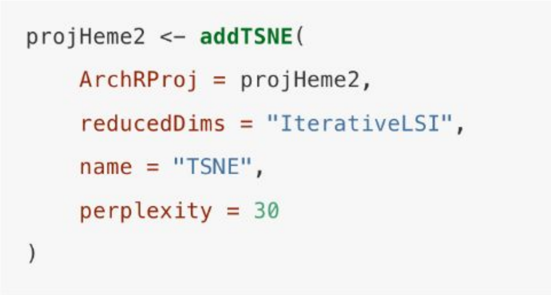
Uniform Manifold Approximation and Projection (UMAP) or t-distributed stochastic neighbor embedding (t-SNE), are used to visualize single cells in reduced dimension space without identifying them.

Differences include:

- UMAP and t-SNE is the interpretatino of the distance between cells or clusters
  - t-SNE is designed to preserve the local structure in the data
  - UMAP is designed to preserve both the local and most of the global structure in the data
- t-SNE shows much more randomness across multiple replicates of the same input than does UMAP
- UMAP works very well for a diverse set of applications
  - Standard choice
  - Can project new samples into the embedding
- Harmony
  - Implemented to correct batch variations
  - assess the effects of Harmony by visualizing the embedding using UMAP or t-SNE



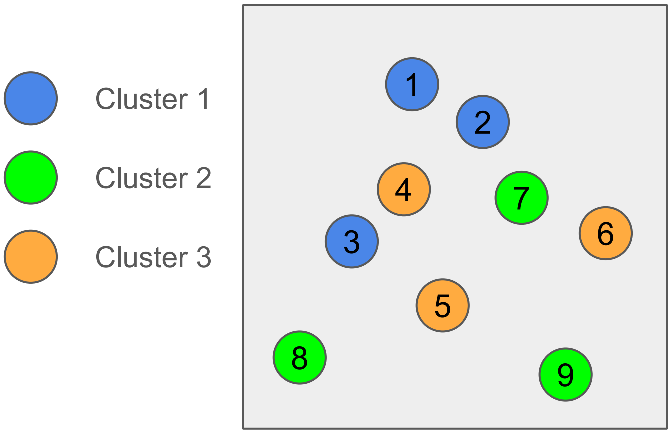
```r
projHeme2 <- addUMAP(
    ArchRProj = projHeme2,
    reducedDims = "IterativeLSI",
    name = "UMAP",
    nNeighbors = 30,
    minDist = 0.5,
    metric = "cosine"
)
```

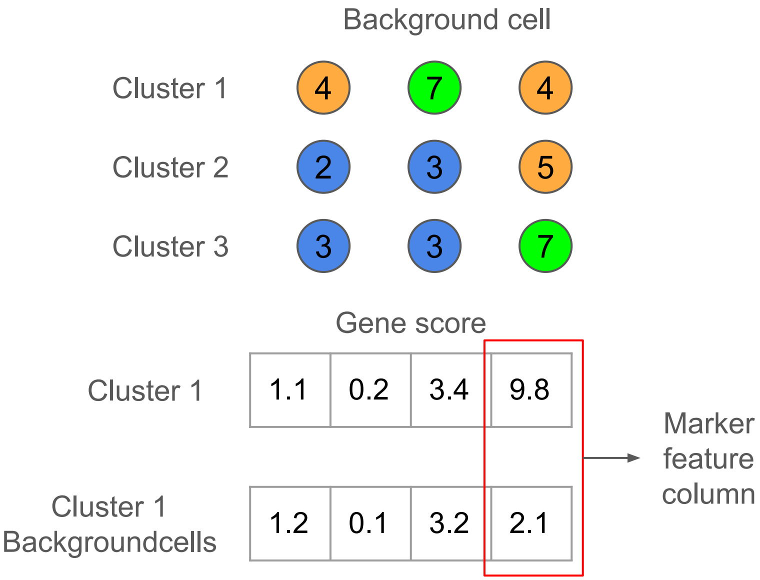


```r
projHeme2 <- addTSNE(
    ArchRProj = projHeme2,
    reducedDims = "IterativeLSI",
    name = "TSNE",
    perplexity = 30
)
```

Chapter 7: Gene Scores and Marker Genes

Gene Score Matrix correlates the cell type with cluster.





Gene score				
Cluster 1	1.1	0.2	3.4	9.8
Cluster 1 Backgroundcells	1.2	0.1	3.2	2.1

Chapter 8: Defining Cluster Identity with scRNA-seq

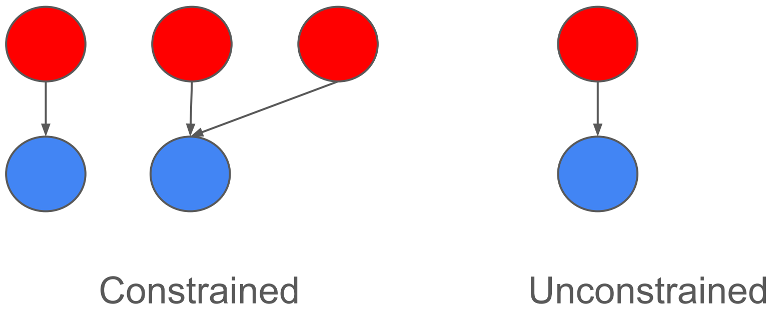
Integration works is by directly aligning cells from scATAC-seq with cells from scRNA-seq.

Unconstrained integration

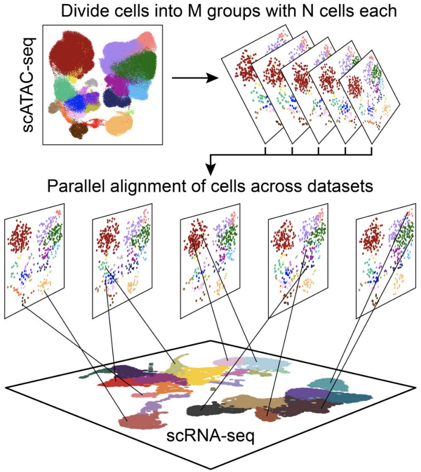
Takes all of the cells in your scATAC-seq experiment and attempt to align them to any of the cells in the scRNA-seq experiment

Constrained integration

Use prior knowledge of the cell types to limit the search space of the alignment and perform a more refined constrained integration.



Assume we knew that Clusters A, B, and C in the scATAC-seq data corresponded to 3 different T cell clusters, and we knew that Clusters X and Y in the scRNA-seq data corresponded to 2 different T cell clusters. Then we a=can akugn A, B and C to X and Y


Divide cells into M groups with N cells each
scATAC-seq
Parallel alignment of cells across datasets
scRNA-seq