RNA-Seq Workshop

May 28 & May 30

elinor.velasquez@berkeley.edu

https://github.com/elinorv21/RNA-Seq_workshop/

Structure of the Workshop

Day 1: Lecture and discussion

1. Sandra’s slides: Introduction to bulk RNA-Seq

2. How to analyze bulk RNA-Seq data

3. Tools (software)/coding

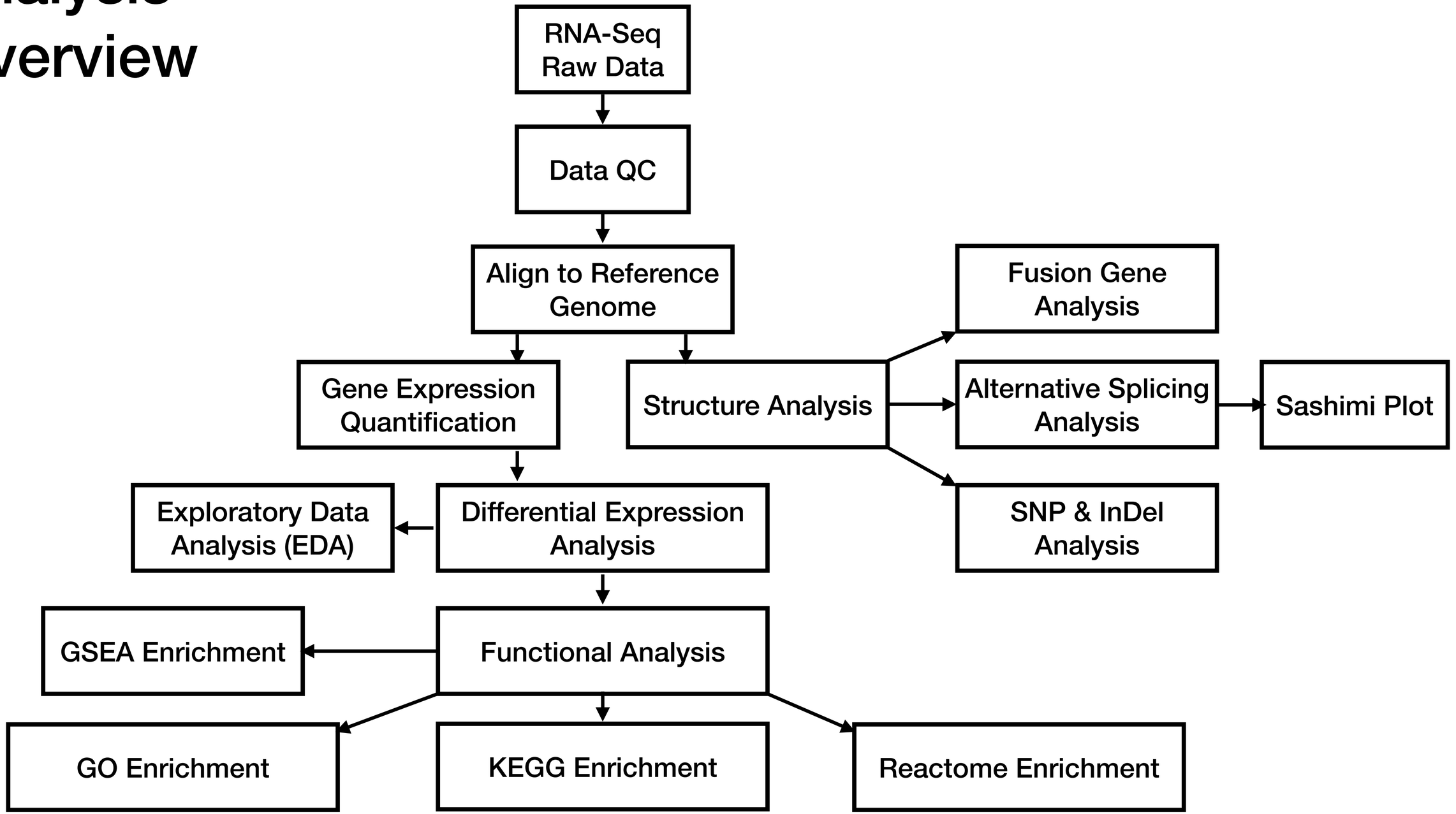
4. Galaxy website

5. Chat-GPT-o3

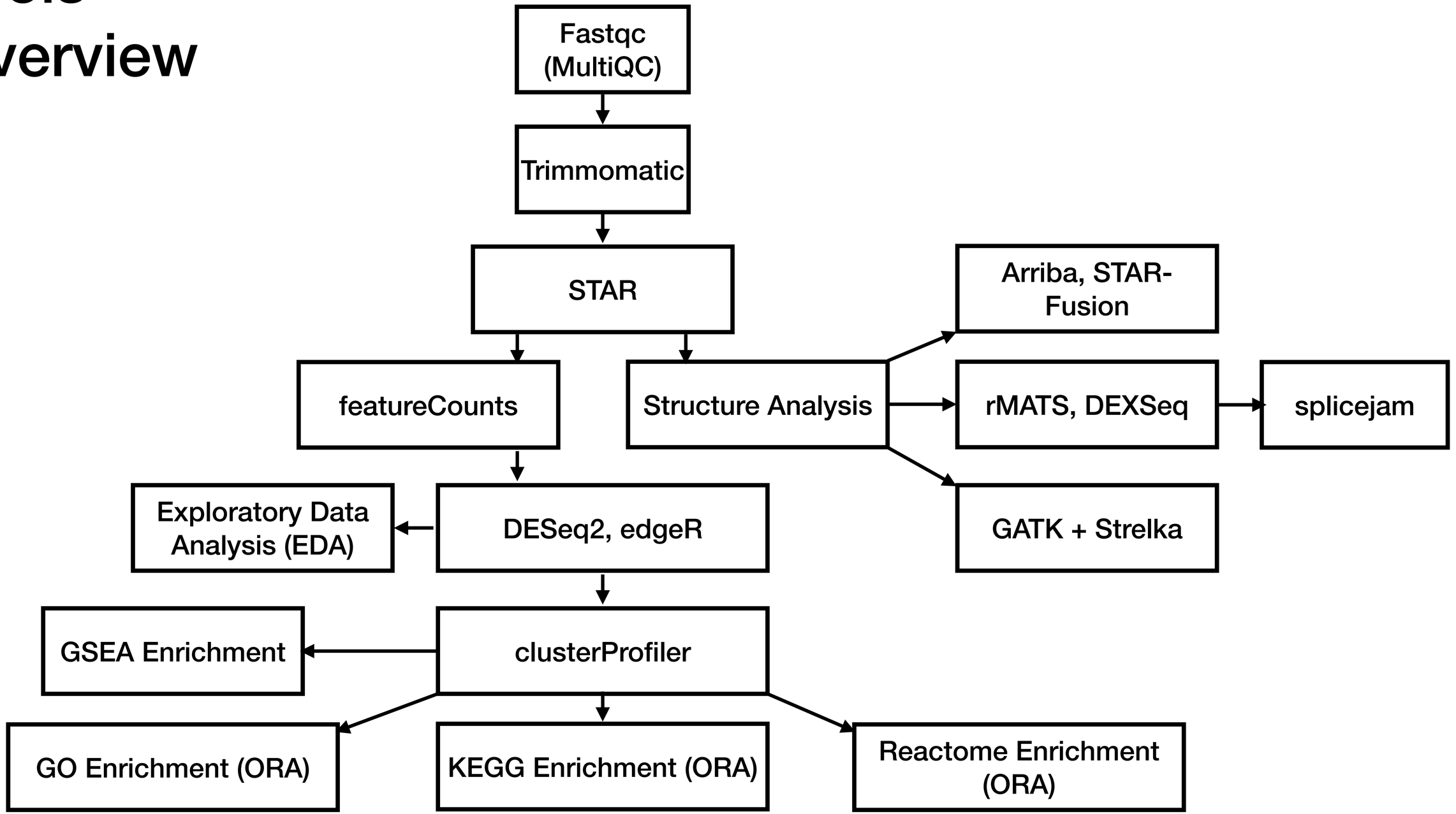
Day 2: Analyze data (Install software on Savio and analyze toy data or your own data)

Types of Analysis

Analysis Overview



Tools Overview

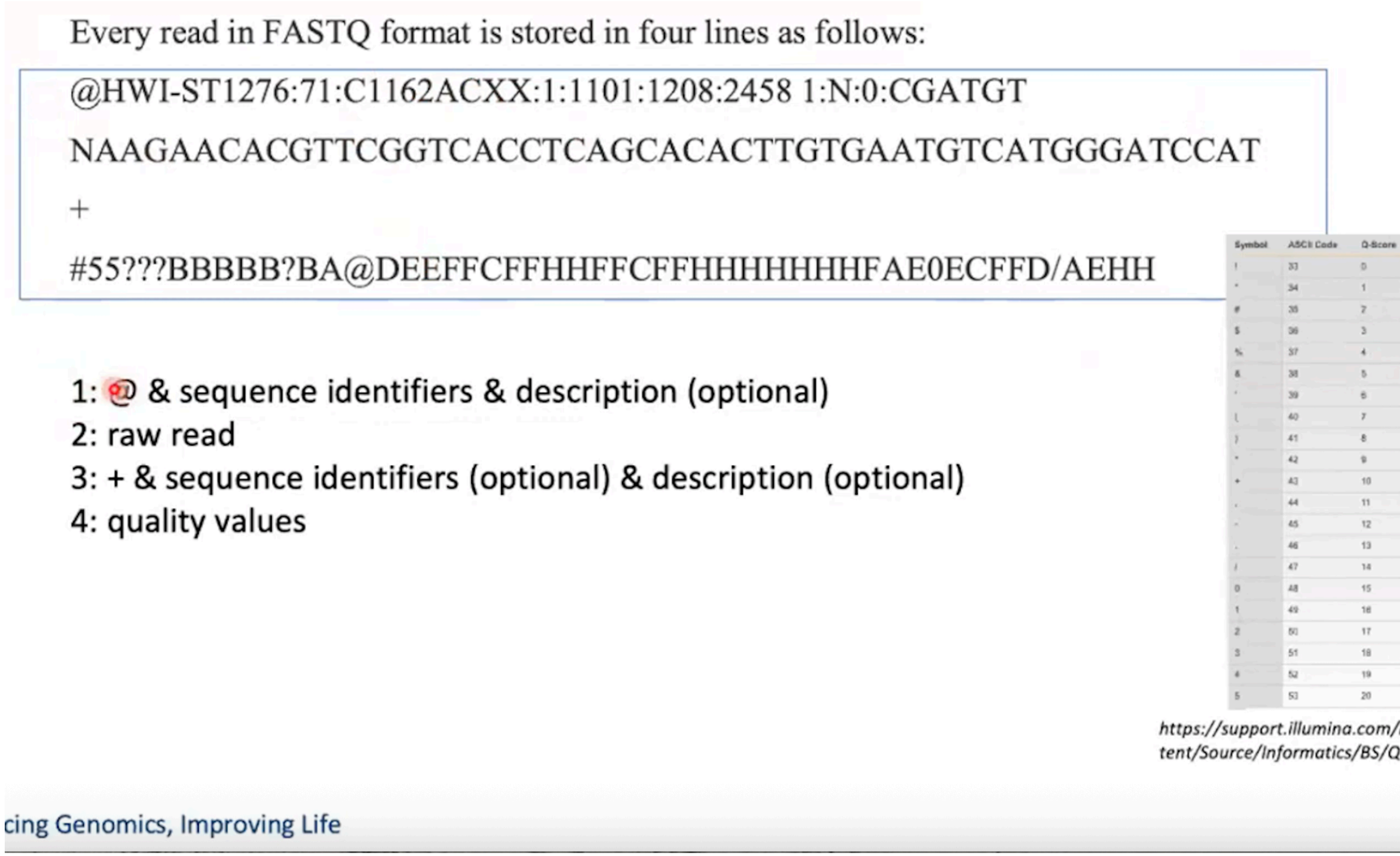


RNA-Seq Raw Data

- Novogene gives you raw data in a fastq format

- Example: data_1.fastq.gz, data_2.fastq.gz (forward and reverse) for a replicate of a sample type (baseline-control, treatment1, treatment2)

- Important to know:
  1) Length of reads [150]
  2) Paired reads? [paired]
  3) Library prep kit #: unstranded (non-directional) or strand-specific? [unstranded]



Reference: Novogene Sequencing Center

Data Quality Control

- Determine quality of data: Report GC content distribution, for example (Software: FastQC and MultiQC - Use FastQC before and after data filtering). Use MultiQC after data filtering and applying software for more investigation into data quality.

- Data filtering used to remove adapter sequences and sequences with low mean quality scores (Software: Trimmomatic - best for Illumina data)

Reference Genome and Annotation File

- Most organisms have a reference genome (genome.fa). If not, there are techniques for assembling a reference genome.

- Annotation file (annotation.gtf): Gives information about what biological features are positioned at each region of coordinates and how those features relate to each other. For example, features are: feature type (e.g., gene, exon, etc), metadata (e.g., protein-coding), genomic coordinates (e.g., start, stop, chromosome, contig name), gene ID (e.g., Ensembl ID, entrez ID), gene name, and so on.

Alignment of Reads

- Transcript = biological molecule (complete RNA molecule produced from a gene; possibly many transcript isoforms from one gene)
- Read = data (sequence of nucleotides) generated by a sequencing machine
- Goal: Align reads to a reference genome, such as the mouse genome
- Aligning these reads to the genome enables us to count how many reads map to each gene
- Recommended software: STAR (Inputs: Fastq files, genome index; Outputs: BAM files)
- Several assumptions of alignment tools:

  - Given reads can span splice junctions

  - Paired-end sequencing: Each read pair (sample1_forward and sample1_reverse) is sequenced from both ends of one fragment (cDNA).

Gene Expression Quantification

- Determine the abundance of RNA transcripts for each gene (or possibly at isoform level) in a collection of cells (e.g., mouse brain tissue).

- Take the raw sequencing reads and compute gene counts

- Recommended software: featureCounts

  - Inputs: BAM files, gene annotation file (gtf file)

  - Output: Table of rows (genes) and columns (sample type & replicate number per column) (text file)

Differential Gene Expression (DGE) Analysis

- The goal of differential gene expression (DGE) analysis is to identify those genes which show significant differences in expression levels between the different experimental groups (e.g., treatment vs. control, psilocybin vs. vehicle).

- Typically, the software also filters out genes with low counts and genes which show zero to low variance across experimental conditions in order to increase statistical power (power is the probability of detecting an effect when the effect truly exists - correctly reject a false null hypothesis)

- Recommended software (R packages): DESeq2, edgeR, limma (Bioconductor)

Exploratory Data Analysis (EDA) for Differential Gene Expression

- Heatmaps - Plot all or groups of genes, such as the top differentially expressed genes (based on p-value and fold change from the DESeq2 output or the edgeR output).

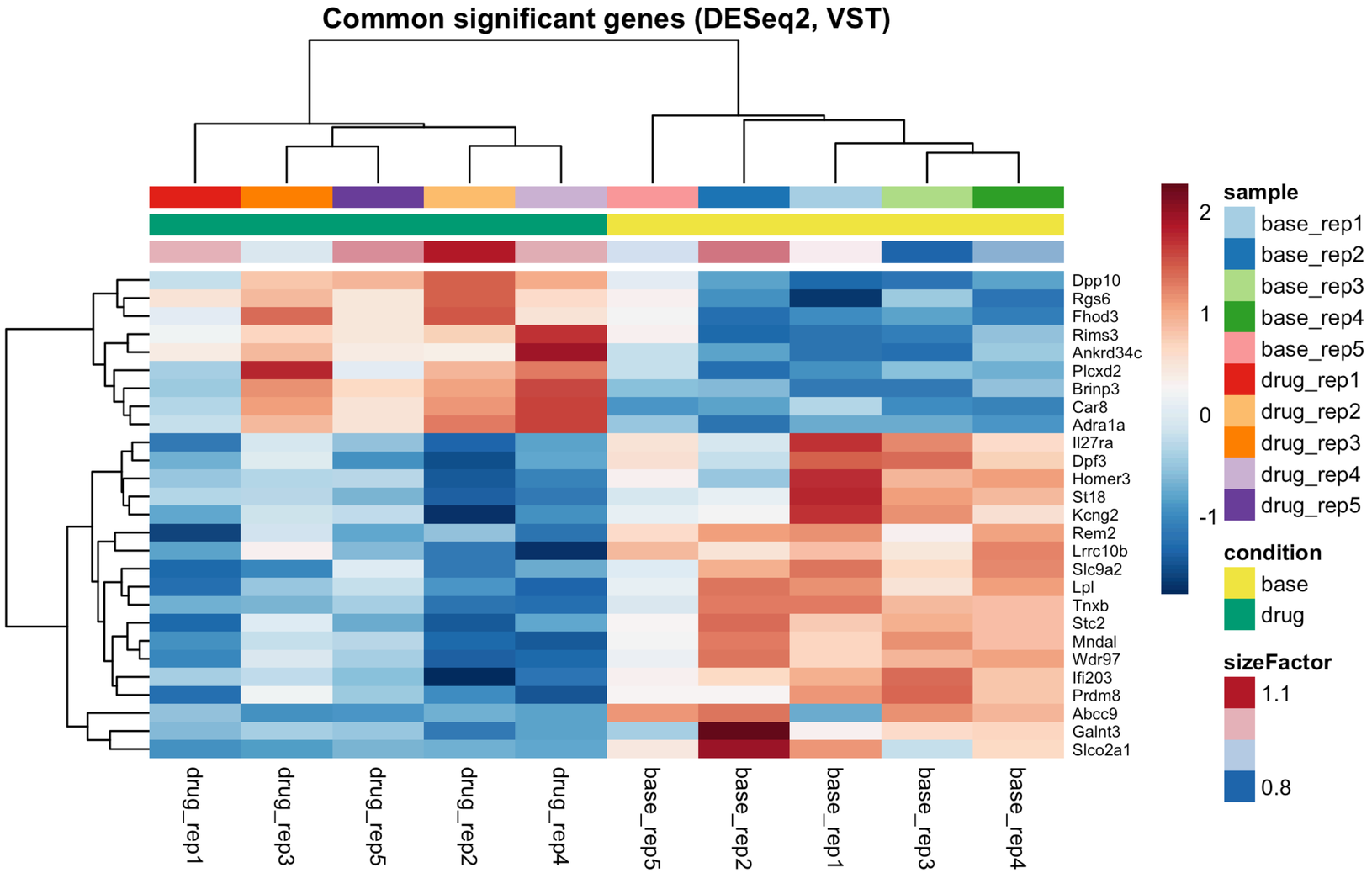
- PCA plots - Look for clustering patterns, outliers, and percentage of variance explained by each principal component. See if there is batch bias.

- Volcano plots - For overall differential gene expression analysis - also, genes that appear at the corners of the plot are statistically significant, thus useful for further investigation.

- MA plots - x-axis = abundance (average), y-axis = log2 fold change; points should cluster around the M = 0 line if data is unbiased

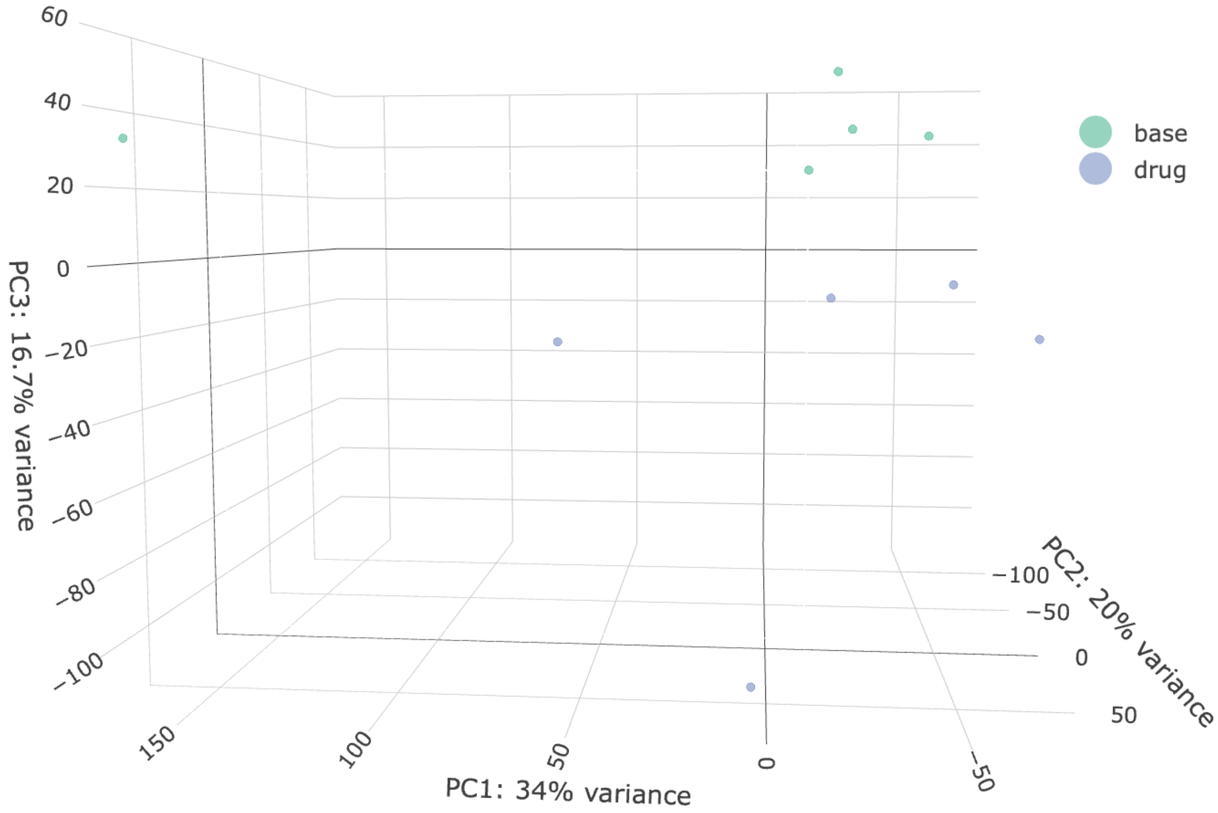
Heatmap: How to Interpret

- Each cell represents the expression level of a single gene in a single sample
- Red: Higher gene expression relative to mean of a gene's expression across all samples. Blue: Lower gene expression relative to mean of a gene's expression across all samples
- Dendrograms: (Left) Genes that are closer together are co-expressed (similar patterns of expression across samples or conditions); (Top) Samples that are closer together on the dendrogram are more similar
- (Example) Genes upregulated in "drug" samples and downregulated in "base" samples: Observe rows that are red in the "drug" columns and blue in the "base" columns (e.g., Dpp10, Rgs6)
- The replicates within each condition cluster together, implying good reproducibility



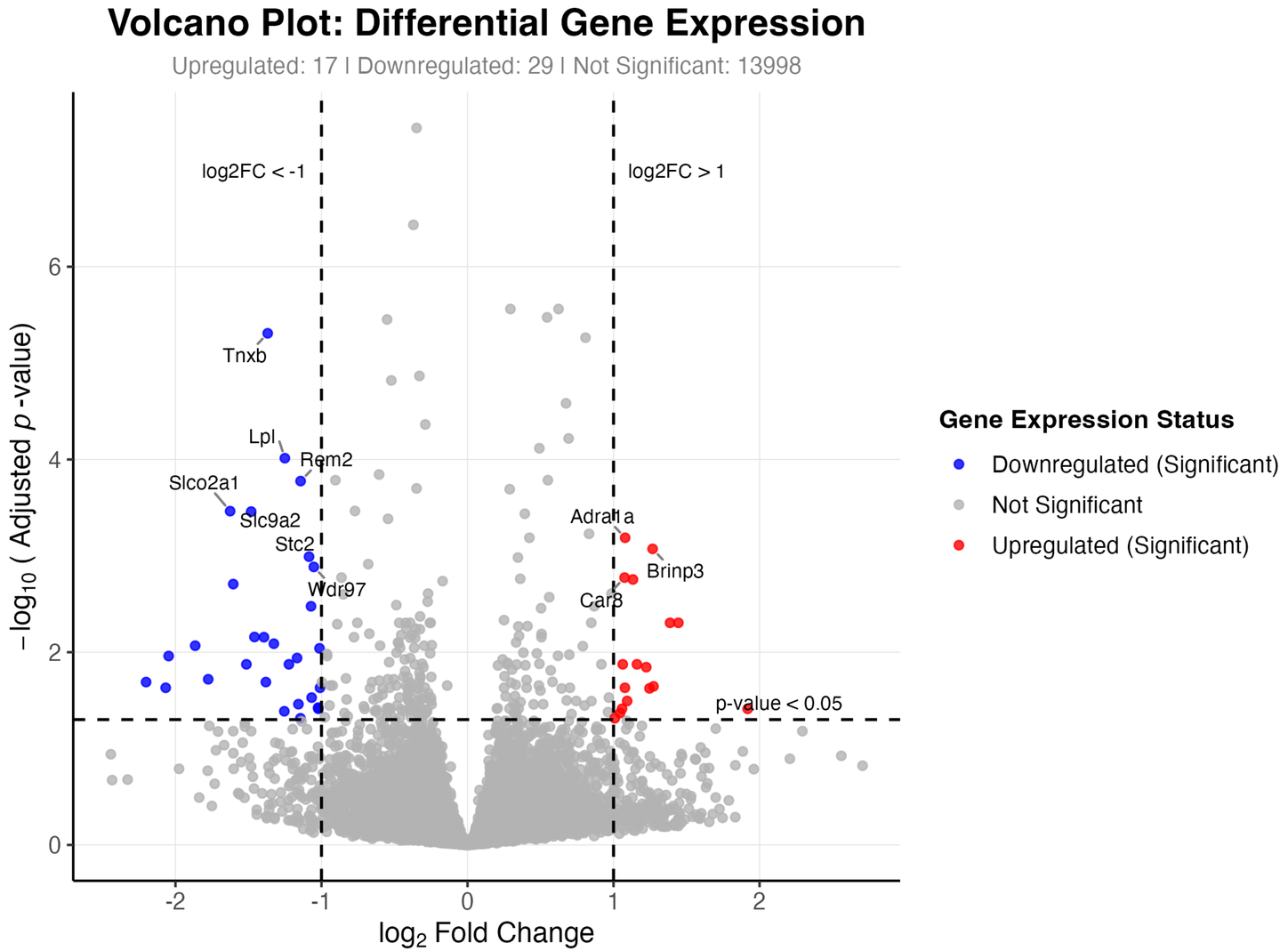
PCA Plots: How to Interpret

- Principal Component Analysis (PCA): Principal Components (PC1, PC2, PC3): These axes represent the principal components. Each component is a linear combination of the original variables (e.g., genes and their expressions).
- The goal of PCA is to reduce the dimensionality of the data while retaining as much variance as possible.
- Clustering samples: suggests they are similar in terms of the data.
- Good separation between groups indicates that the principal components effectively distinguish between them.
- Outliers suggest unique features in a sample or possible issues.
- Observation: The baseline (vehicle) samples cluster together and the drug (psilocybin) samples cluster together, but each cluster has one outlier. The two clusters are well-separated, and while no single principal component alone explains the majority of the total variance, the first three principal components together capture a substantial proportion (71%) of the total variability in the dataset.



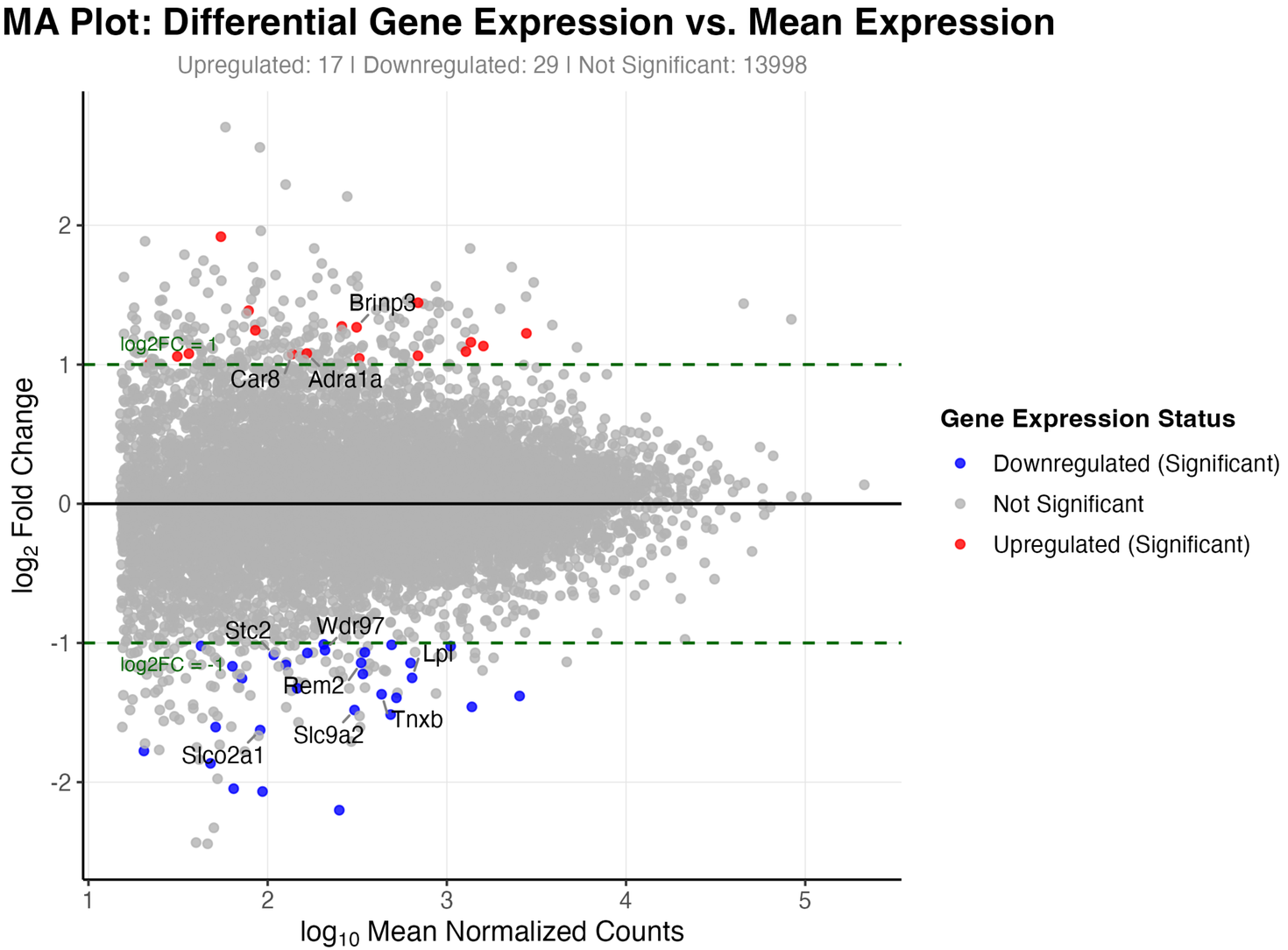
Volcano Plots: How to Interpret

- Volcano plots are helpful for displaying genes exceeding both statistical thresholds (red & blue dots); to visualize differential expression of genes between two conditions.
- Y-axis: The statistical threshold is given by -log10(0.05) = 1.301. Values above 1.301 are statistically significant.
- X-axis: The threshold is given by absolute_value[log2(fold change)] = 1, so the fold change threshold = +/-1: The greater the absolute value of fold change is beyond the threshold, the more up-regulated (or down-regulated ~ corresponds to sign of the fold change) is a gene between the two groups (e.g., base vs. drug).
- Outliers will show as triangles.
- The top 10 significant genes are labeled in the plot.



MA Plots: How to Interpret

- Y-axis (M): $y_{gene} = \log_2(E_2) - \log_2(E_1)$, with E_2 = expression value for a given gene in Group 2 (e.g., Treatment (Drug)), E_1 = expression value for given gene in Group 1 (e.g., Control (Baseline))
- X-axis (A): $x_{gene} = [\log_2(E_2) + \log_2(E_1)]/2$
- Points should be symmetrically clustered around M = 0 line. If plot is sloped, this shows a bias. Most genes should be in this cluster (since most genes are not significant).
- Top 10 significant genes are labeled



Quality Control: Determine Accuracy of Samples

- Create a list of positive control genes (cell type-specific marker genes) for biological tissues of interest. One source for finding such genes in mouse may be: https://mouse.brain-map.org/

- Determine the CPM thresholds for the marker genes from existing RNA-Seq datasets.

- Using edgeR to find the CPM values in your samples, determine if the marker genes in your samples meet the thresholds.

Functional Analysis:
The Gene Ontology

The Gene Ontology (GO) is a classification that describes the functions of genes and proteins across organisms. There are three main types:

- **Biological Process (BP):** Describes biological processes, such as cell growth, metabolism, etc.
- **Molecular Function (MF):** Describes activities performed by gene products at the molecular level, such as binding or catalysis.
- **Cellular Component (CC):** Describes the parts of a cell or its extracellular environment in which a gene product acts.

Functional Analysis:
KEGG Pathways

The **Kyoto Encyclopedia of Genes and Genomes (KEGG)** is a database that integrates genomic, chemical, and systemic functional information. KEGG is a collection of **pathway maps,** representing molecular interaction and reaction networks in biological processes, including:

- **Metabolism:** Covering global and specific metabolic pathways.
- **Genetic Information Processing:** Including transcription, translation, replication, and repair.
- **Environmental Information Processing:** Such as signal transduction and membrane transport.
- **Cellular Processes:** Like cell growth, death, and motility.
- **Organismal Systems:** Including immune, endocrine, and nervous systems.
- **Human Diseases:** Illustrating pathways involved in various diseases. (Consider mouse gene orthologs to human genes associated with a disease)
- **Drug Development:** Showing drug targets and mechanisms.

Functional Analysis: The Reactome

- Reactome is a manually curated database of biological pathways (human-centric, with ortholog projections to mouse and other species), including
- **Signal Transduction:** How cells communicate and respond to their environment.
- **Metabolism:** The chemical reactions that sustain life.
- **DNA Replication, Repair, and Recombination:** Processes maintaining the genome.
- **Transcription and Translation:** How genetic information is used to make proteins.
- **Immune System:** Pathways involved in defense against pathogens.
- **Cell Cycle and Cell Death:** Processes governing cell proliferation and elimination.
- **Transport:** Movement of molecules across cellular membranes.
- **Disease Pathways:** Aberrant pathways associated with various human diseases, often showing the impact of mutations or drug interventions.

Functional (Enrichment) Analysis: Over-Representation Analysis (ORA)

Over-Representation Analysis (ORA): These tools take a list of significant genes (e.g., differentially expressed genes based on a certain threshold such as FDR and log2(Fold Change)) and check if any functional categories (GO terms, KEGG pathways, Reactome pathways) are over-represented within that list compared to what would be expected by chance from the background gene population.

ORA asks whether a set of differentially expressed set genes has a statistically significant overlap with specific functional categories (e.g., GO terms) compared to what would be expected by chance: Are these functions/pathways over-represented in the differentially expressed gene list?

Software: clusterProfiler

clusterProfiler Functions:
GO ORA terms: enrichGO( )
KEGG ORA pathways: enrichKEGG( )
Reactome ORA pathways: enrichPathway( )

How to Interpret GO enrichment (ORA): Bubble plot

- Perform Gene Ontology (GO) enrichment analysis to identify biological processes (BP), molecular functions (MF), and cellular components (CC) that are over-represented in your list of differentially expressed genes (DEGs).

- How to interpret the DGE data output via GO enrichment analysis (functional analysis):

  1. What biological processes are up/down-regulated?

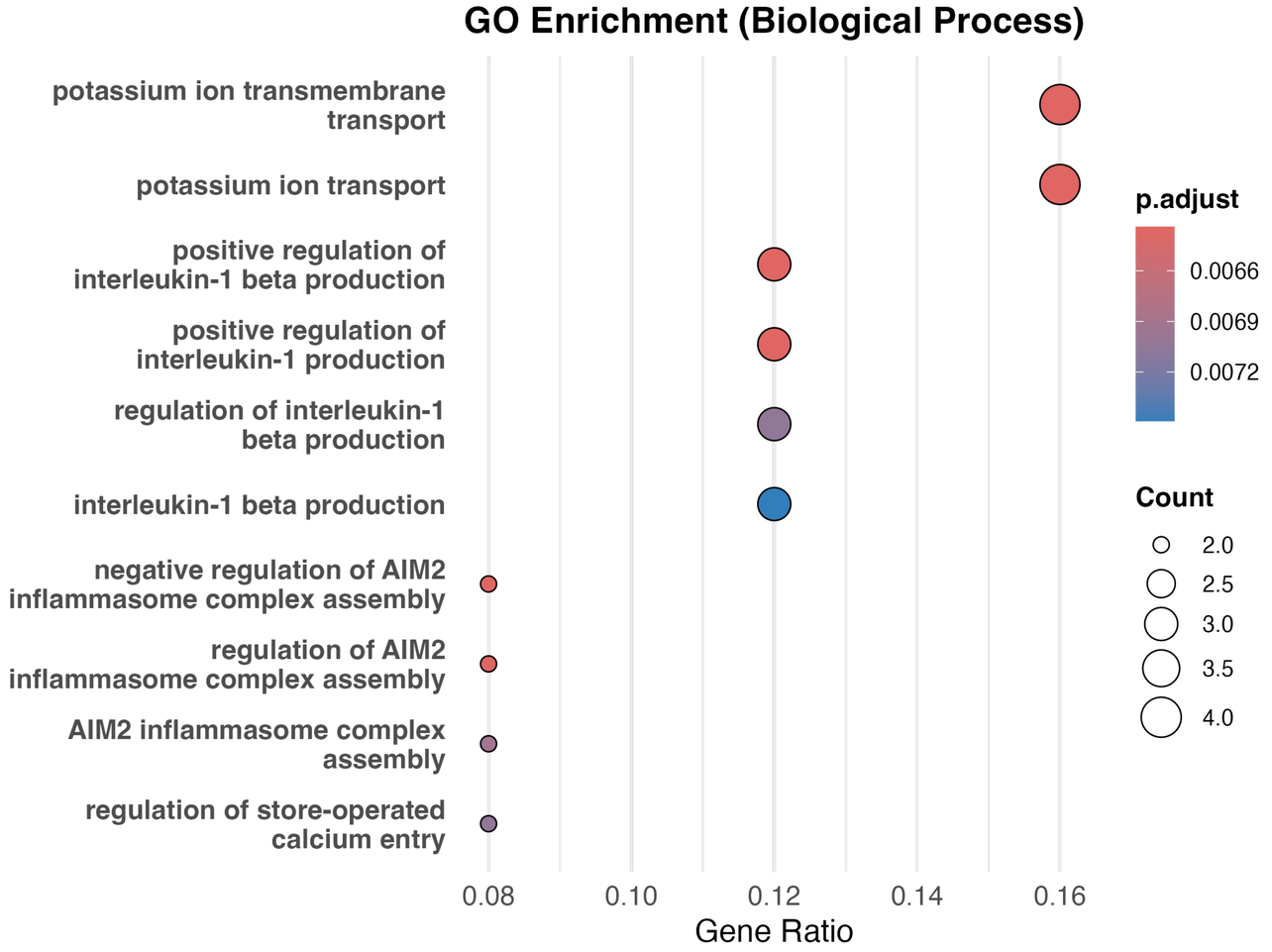
  2. Are these GO terms broad (e.g., “metabolic process”) or specific (e.g., “DNA repair”)?

  3. Do enriched terms match your experimental hypothesis?

  4. How do the different conditions (e.g. treatment (drug) vs control (baseline)) affect the functional themes?

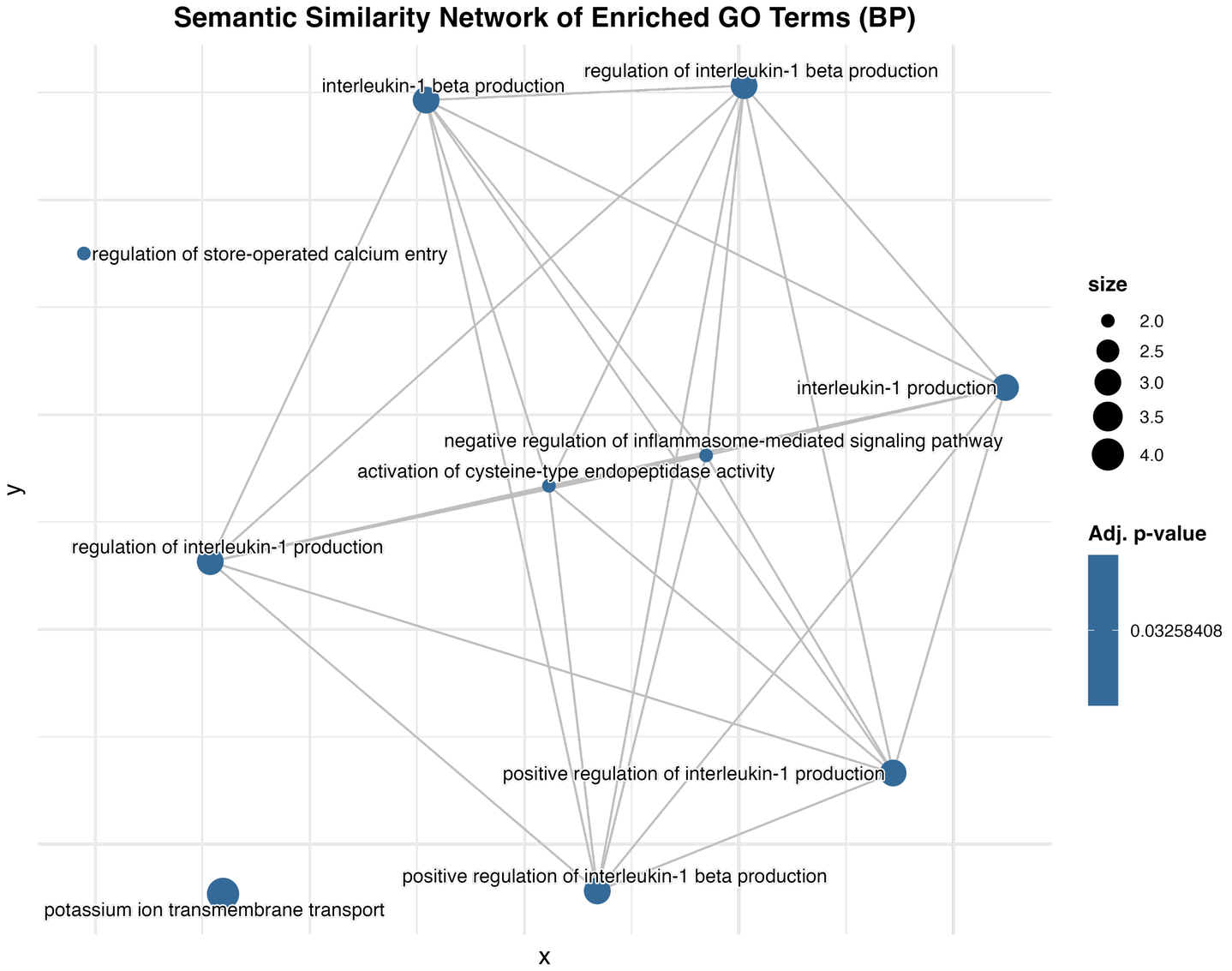
- **Gene Ratio:** The proportion of the input DEG genes that are annotated to the GO term (as compared to the whole gene list)

- **Gene count:** Number of DE genes involved in the GO term (number of genes in the total gene list = 27; gene list = genes common to both DESeq2 and edgeR)



How to Interpret GO enrichment (ORA): Semantic similarity network plot

- Nodes (dots) = Each dot is a GO term (e.g., "interleukin-1 beta production”)
- Edges (lines) = Semantic similarity between GO terms (based on gene overlap or term ontology proximity); shorter edges = greater similarity between GO terms.
- A tight cluster means multiple enriched terms all describe similar biological themes.
- Larger node = more DEGs in that term.
- Is a cluster focused on one pathway? Example: Immune themes: Are terms like "interleukin", "inflammatory response", "antigen presentation" enriched?
- Are several terms repeating similar language? That increases confidence in a biological theme.

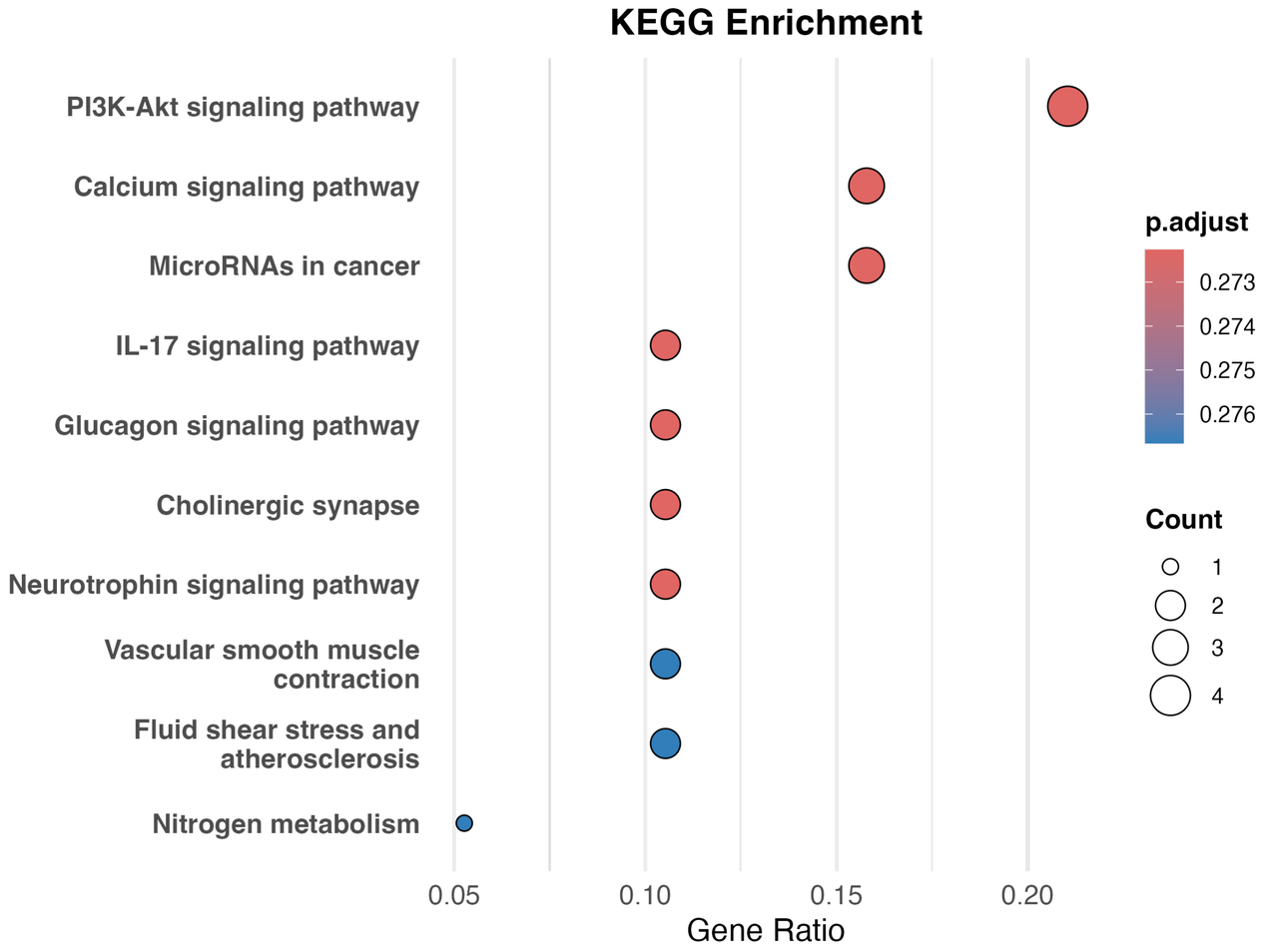


How to Interpret KEGG enrichment (ORA): Bubble Plot

- The enrichKEGG enrichment in clusterProfiler tests if any KEGG pathways are statistically enriched in a given gene list, compared to a background. The background equals all genes in the KEGG mouse database.

- Here, the gene list equals all genes common to both DESeq2 and edgeR outputs with adjusted p-value (FDR) < 0.05 and log2(Fold Change) > 0.5 (log2(FC) > 1 gave no genes to be tested)

- Low p.adjust + high GeneRatio = strong evidence of relevance (here, p.adjust > 0.05 so no relevant pathways given)

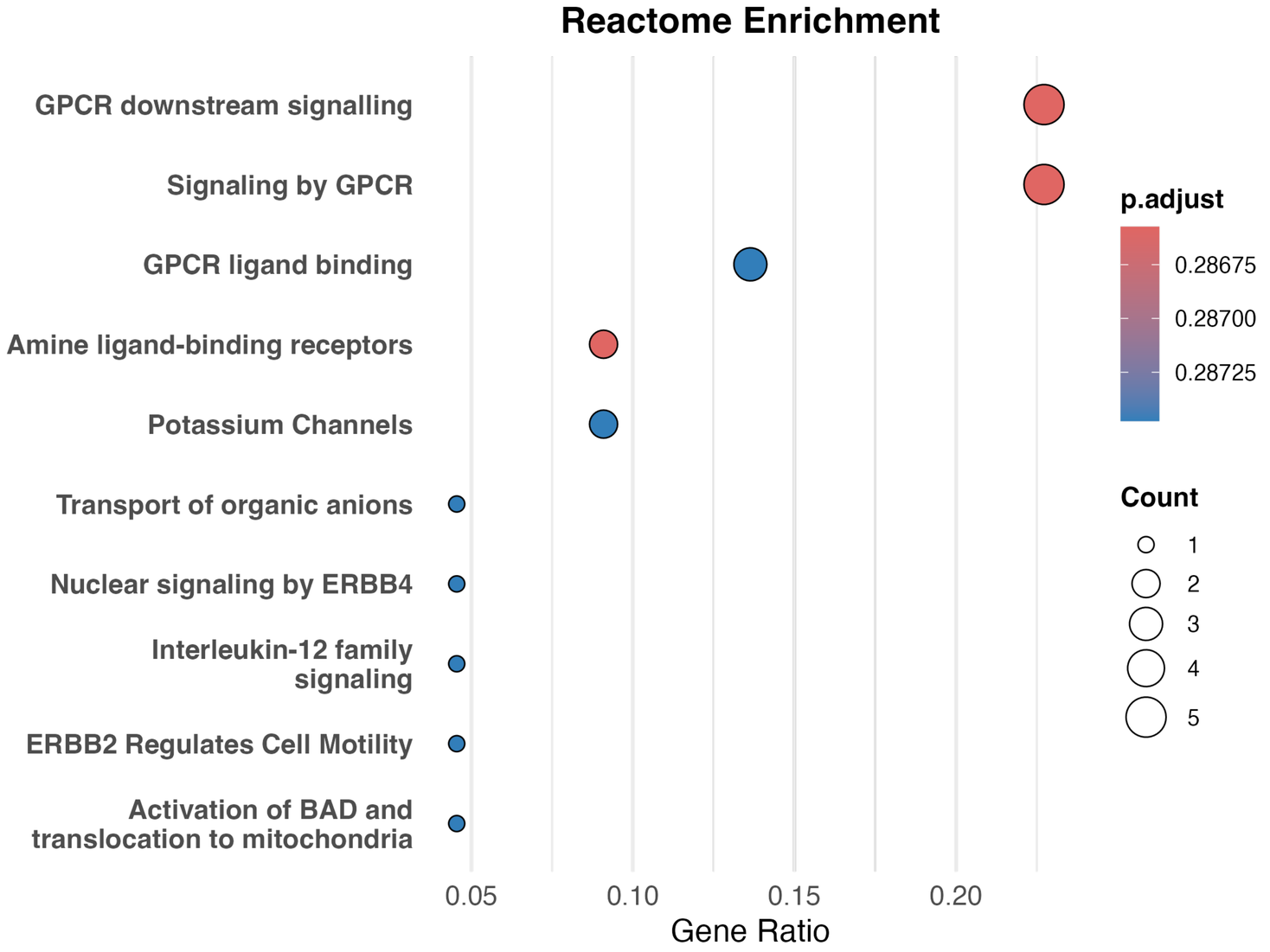


How to Interpret Reactome enrichment (ORA): Bubble plot

- The Reactome enrichment in clusterProfiler tests if any Reactome pathways are statistically enriched in a given gene list, compared to a background. The background equals all mouse orthologs of human genes in the Reactome database.

- Here, the gene list equals all genes common to both DESeq2 and edgeR outputs with adjusted p-value (FDR) < 0.05 and log2(Fold Change) > 0.5 (log2(FC) > 1 gave no genes to be tested)

- Low p.adjust + high GeneRatio = strong evidence of relevance (here, p.adjust > 0.05 so no relevant pathways are given)



Functional (Enrichment) Analysis:
Gene Set Enrichment Analysis (GSEA)

GSEA: (not ORA) A ranked list of all genes is used; The goal is to determine which genes belonging to a particular functional category tend to appear towards the top or bottom of the ranked list.

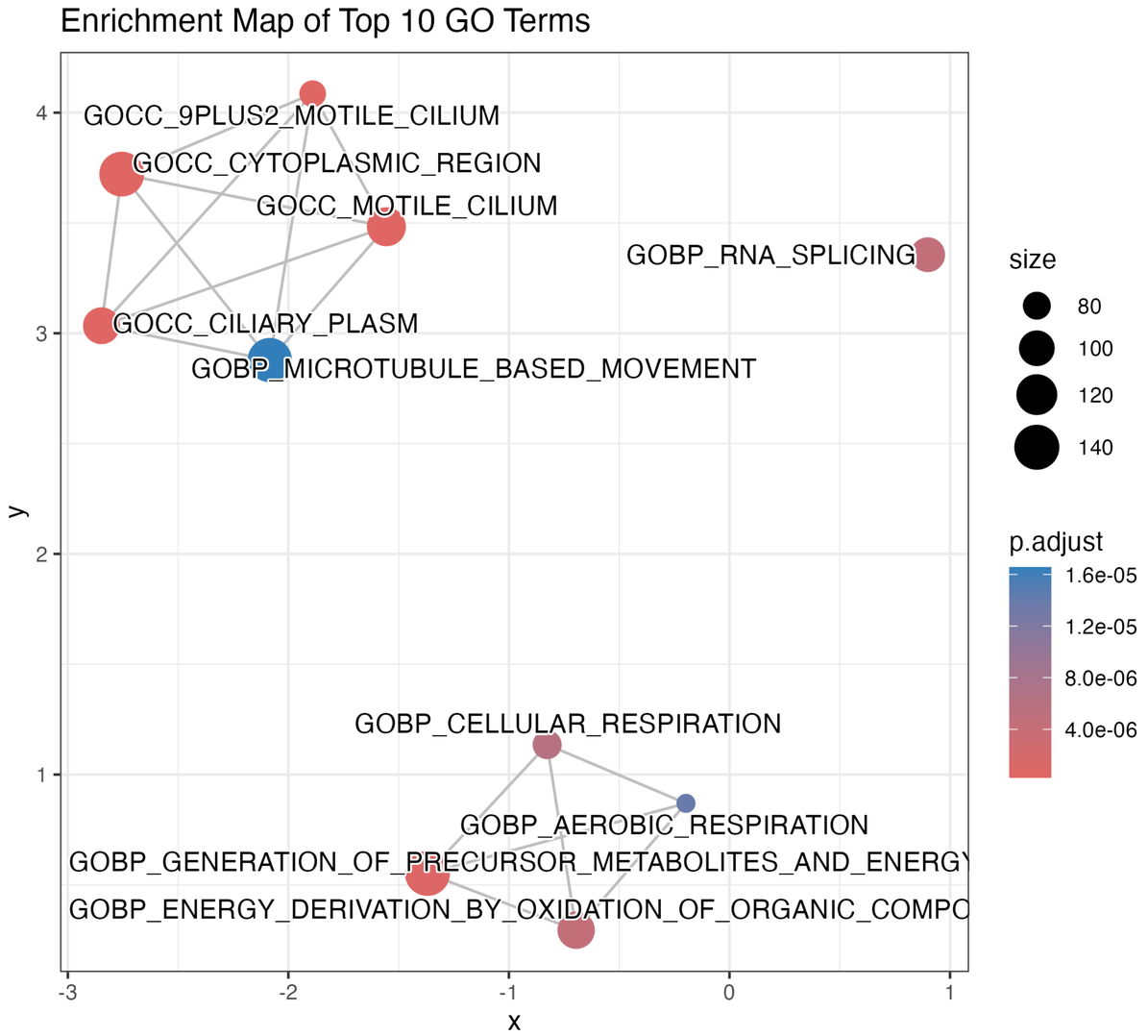
Software: clusterProfiler (more comprehensive analysis, fgsea (large datasets or when speed is crucial)

clusterProfiler Functions:
GO GSEA terms: GSEA( )
KEGG GSEA pathways: gseKEGG( )
Reactome GSEA pathways: gsePathway( )

How to Interpret GO GSEA

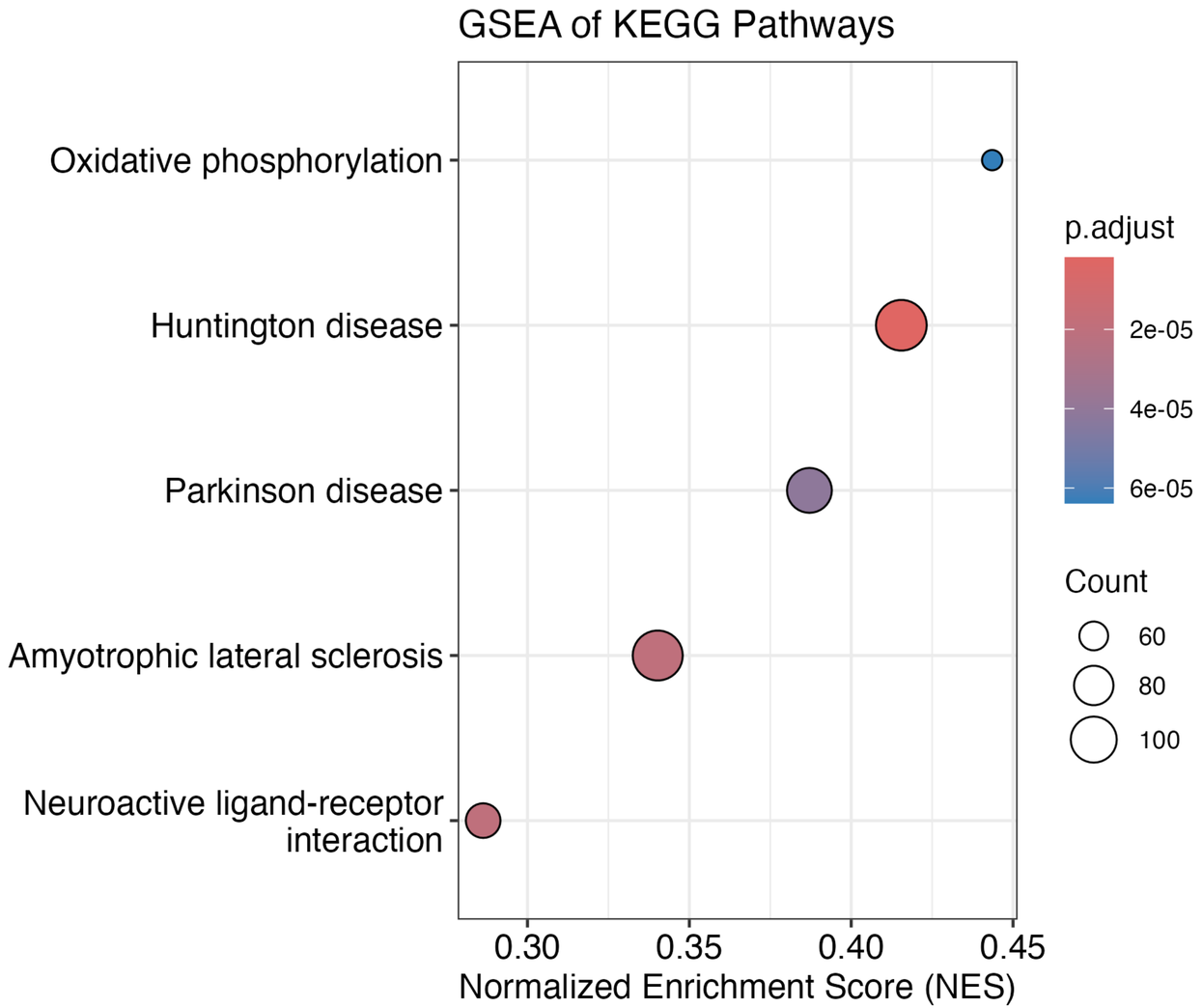
- If two gene sets share a significant number of common genes, they will be connected by an edge.
- Nodes:
- BP = Biological Process
- CC = Cellular Component
- MF = Molecular Function
- Genes are ranked by the score:

  ranking_score = sign(logFC) * (-log10(FDR))

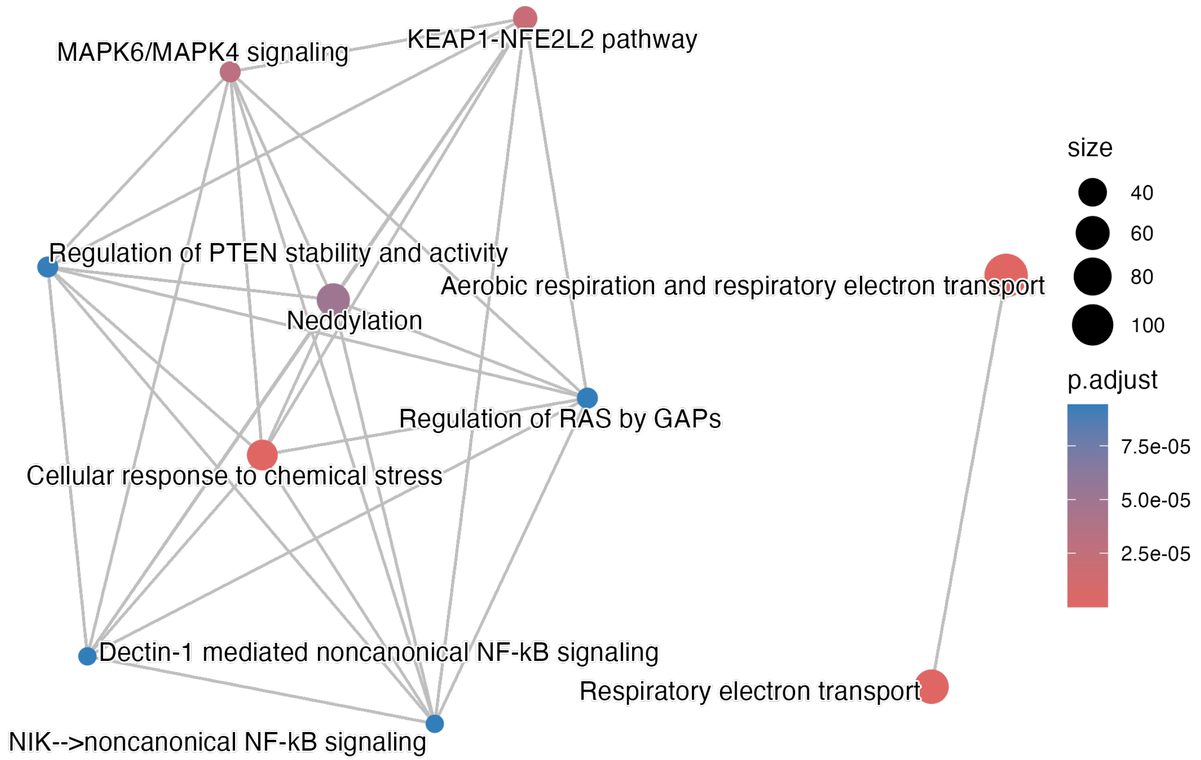


How to Interpret KEGG GSEA

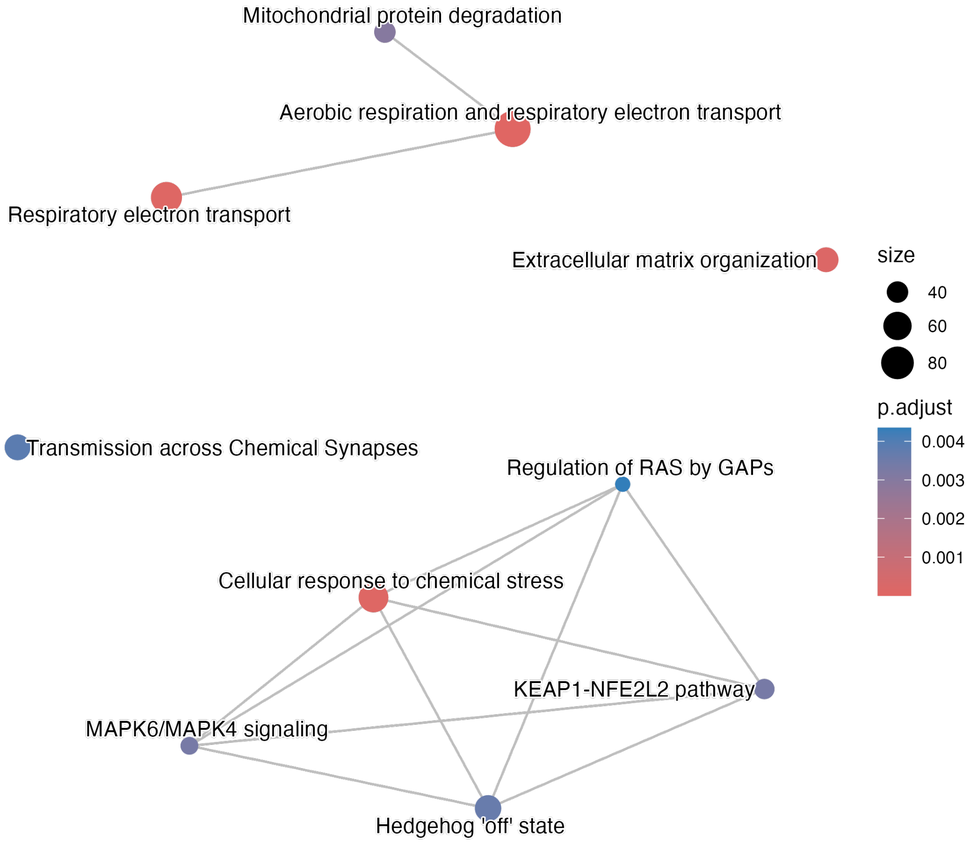
- This bubble plot visualizes the top five enriched pathways based on their significance.
- Positive NES: Indicates that the genes in that pathway are upregulated and enriched at the top of the ranked gene list (genes with high positive log2 fold change).
- Negative NES: Indicates that the genes in that pathway are downregulated and enriched at the bottom of your ranked gene list (genes with high negative log2 fold change).



How to Interpret Reactome GSEA



Ranked genes from DESeq2



Ranked genes from edgeR

Alternative Splicing Analysis: Major Splicing Events

A gene may produce a number of different mRNA isoforms. The diversity of the transcriptome and proteome increases, allowing for a greater range of protein functions from a limited number of genes.

Different splicing events may occur across various conditions, such as:

- **Exon Skipping (SE):** An exon is either included or excluded in the mature mRNA.
- **Alternative 5' Splice Site (A5SS):** Different 5' splice sites are used, resulting in different 3' ends of the upstream exon. The length of the upstream exon changes.
- **Alternative 3' Splice Site (A3SS):** Different 3' splice sites are used, resulting in different 5' ends of the downstream exon. The length of the downstream exon changes.
- **Mutually Exclusive Exons (MXE):** One of two (or more) exons is included, but not both.
- **Retained Intron (RI):** An intron that is normally spliced out remains in the mature mRNA.

Alternative Splicing Analysis: Inclusion Level (PSI)

- Consider: How often is a particular splicing isoform observed relative to other isoforms for a splicing event?

- Percent Spliced (PSI):

- Inclusion level (PSI) = proportion of transcripts that include a specific splicing event or exon.

  PSI = (# of transcripts that include a specific alternative splicing event)/(all transcripts)

- In other words, the inclusion level provides a measure of how prevalent a particular splicing choice is for each condition.

Alternative Splicing Analysis: Example

- Example: Consider a skipped exon event (SE):
- Let Gene A = Exon1 — Exon2 — Exon3.
- Alternative splicing may give these two isoforms:

  1. Inclusion isoform: Exon 1 - Exon 2 - Exon 3 (Exon 2 is included)

  2. Skipping isoform: Exon 1 - Exon 3 (Exon 2 is skipped)

- IncLevel1 (control): 0.8 (80% include Exon 2; 20% skip Exon 2)
- IncLevel2 (treatment): 0.2 (20% include Exon 2; 80% skip Exon 2)

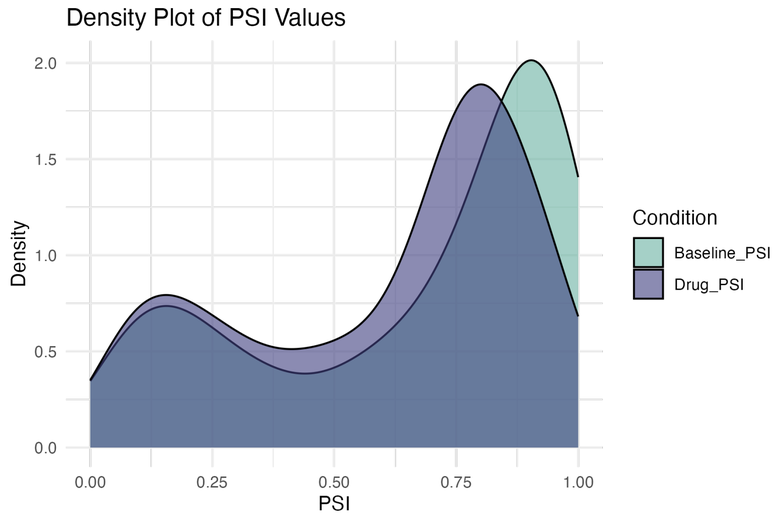
  (IncLevel1, IncLevel2 are columns in rMATS output)

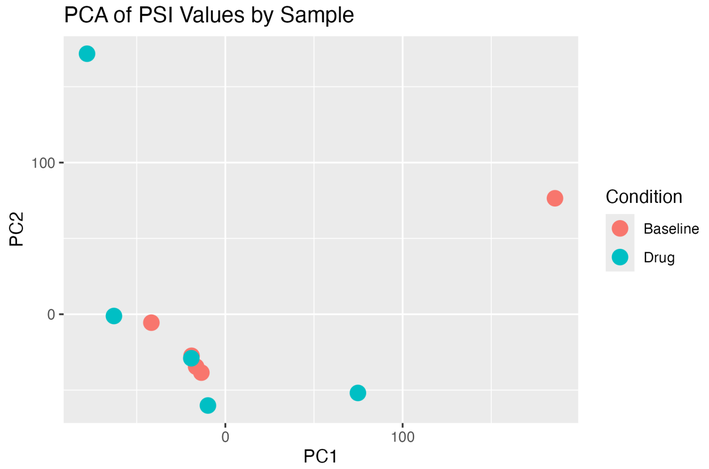
- The `IncLevelDifference` (ΔPSI) column shows `0.6` (`0.8 - 0.2`), implying a shift towards exon skipping in the treatment group.

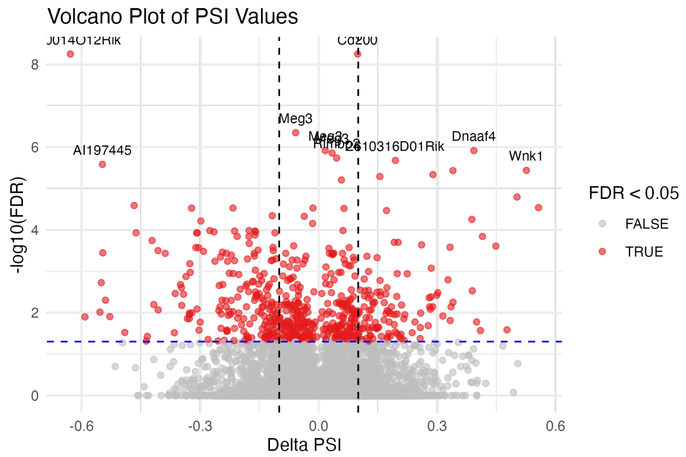
Alternative Splicing Analysis Tools

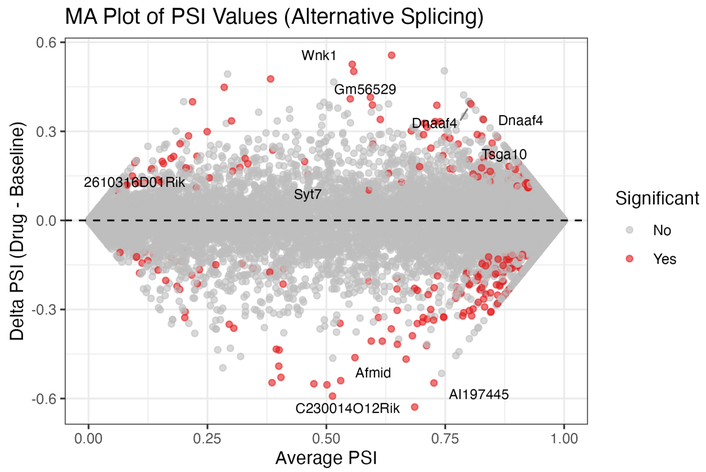
- Software: rMATS
- Inputs: BAM files and gene annotation file
- Uses a statistical model to identify differential alternative splicing events between two or more sample groups.
- Output: Text files for each type of splicing event
- Example: Suppose rMATS identifies hundreds of alternative splicing events (e.g., all skipped exons) across many different genes in an RNA-seq experiment. For each of these events, a PSI value is calculated. Plots are then created to study the PSI values for all skipped exons, for example.

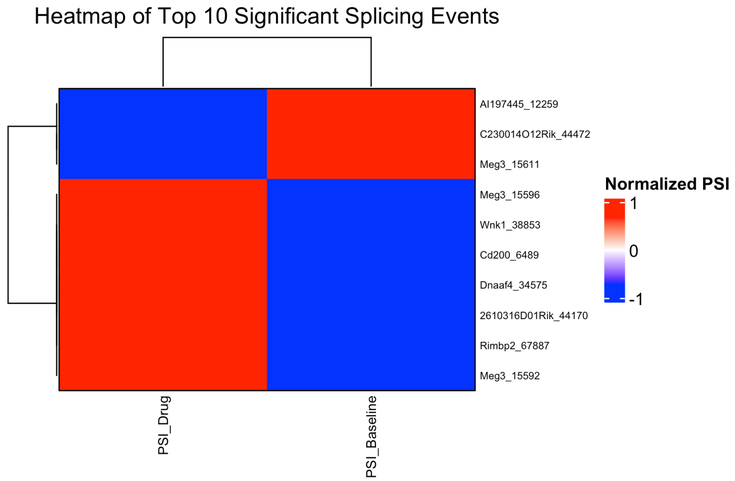
Plots for Alternative Splicing Events: Example: PSI values for skipped exon (SE) events

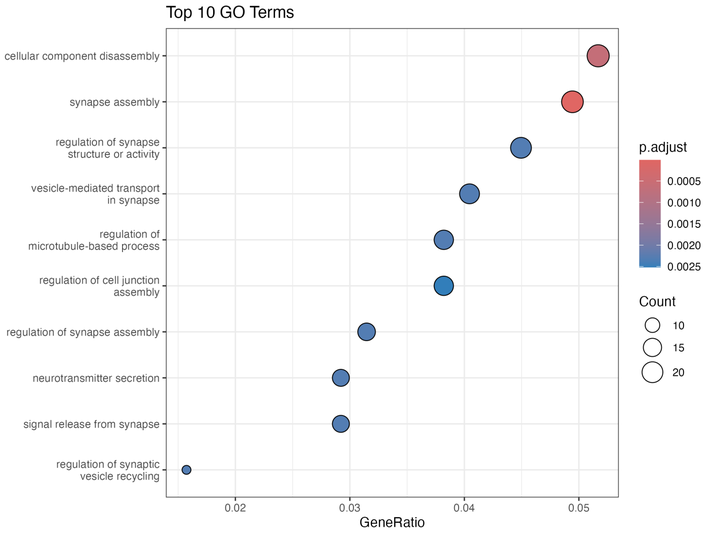




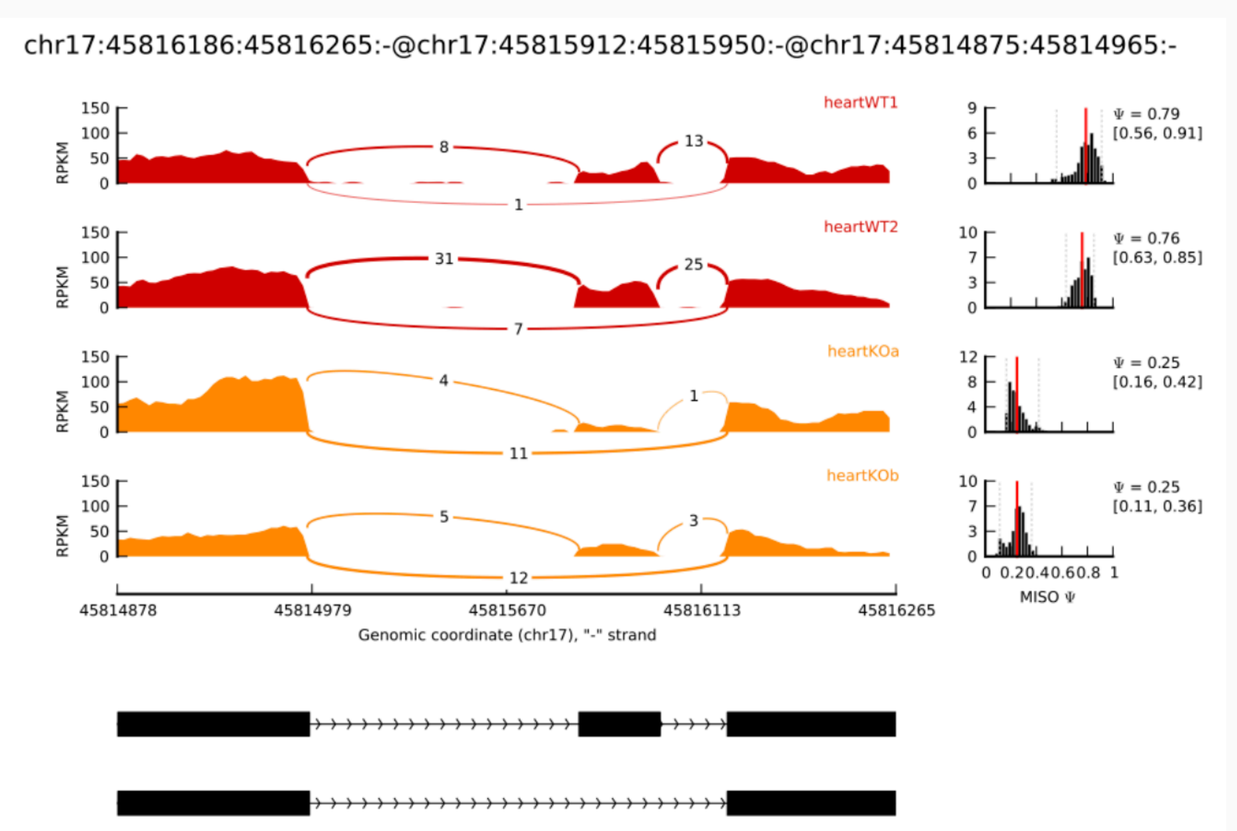








Alternative Splicing: Sashimi Plot



An example figure produced by `sashimi_plot`

Reference: https://miso.readthedocs.io/en/fastmiso/sashimi.html

Software/Coding

https://github.com/elinorv21/RNA-Seq_workshop/

Logging onto Savio (Berkeley's HPC Cluster)

Logging onto Savio from a terminal on a local computer:

- ssh your-username@hpc.brc.berkeley.edu
- Password: Your-pin + Google Authenticator (phone app) #
- Example: Google Authenticator says: "105 149" and your pin is "9876", so your password is: 9876105149

If successful, you'll see:

- Last login: "date" from "IP address"
- (base) [username@ln003 ~]$ <— the "~" means that you are in your home directory
- Switch to scratch directory:
- cd /global/scratch/users/your-username

Upload/Download Files to Savio (HPC Cluster at Berkeley)

Method 1: Use Open OnDemand:

- https://ood.brc.berkeley.edu

Method 2: Use Globus:

- https://docs-research-it.berkeley.edu/services/high-performance-computing/user-guide/data/transferring-data/using-globus-connect-savio/

Common Bash Commands

- Change to the Scratch directory from the Home directory:
  (from limited space, backed up to “unlimited” space, not backed up)

  $ cd /global/scratch/users/your-username

- Show your location:

  $ pwd

- Move backwards (or forwards) to previous directory:

  $ cd .. (or $ cd directoryname)

  Example:

  $ pwd (gives us: toy_data/star_results/bam_files)

  $ cd .. (moves us back to toy_data/star_results)

  $ cd .. (again, moves us back to toy_data)

  $ cd star_results (moves us forward to star_results folder)

- Make a folder/directory:

  $ mkdir foldername

- Copy a file:

  $ cp originalfile newfile

- Remove a file:

  $ rm filename

- Remove a directory:

  $ rmdir directoryname

- Move a file to a new location/directory:

  $ mv filename newlocation

- List your files in a directory:

$ ls or $ ls -la

Creating and Editing Files on the Terminal: Nano Editor

- To create/open a file for editing:

  $ nano hello.R (name of file happens to be “hello.R”; the cursor is “$”)

- To edit:

  Just start typing and use up/down/sideways arrows to move around

- To save the file:

  Press “control” + o

  Press “Enter”

- To close the file:

  Press “control” + x

Creating and Editing Files on the Terminal: Vim Editor

- The vim editor:

  $ vi filename (Example: vi hello.R) - opens a new file or opens a file for editing

- Type “i”: Enter editing mode (from left). Now you can type your script
- Hit “esc”: Exits editing mode. You can use up and down arrows on your script.
- Type “dd”: Deletes the row where the cursor is.
- Type “x”: Deletes a single character.
- Type “i” (Enter editing mode from left) or “a” (Enter editing mode from right).
- Hit “esc”: Exits editing mode
- Type “ZZ”: Saves and closes file.

Introducing Conda for Installing Software

Install Conda and Bioconda:

1. Download the Miniconda Installer:

   $ wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh

2. Install Miniconda:

   $ bash Miniconda3-latest-Linux-x86_64.sh

3. Activate Miniconda:

   $ source ~/.bashrc

4. Verify Miniconda works:

   $ conda --version

Update Conda/Add Channels

Update and install channels:

```
$ conda update -n base -c defaults conda

$ conda config --add channels defaults

$ conda config --add channels bioconda

$ conda config --add channels conda-forge
```

Create a Conda Virtual Environment

- Example: Create the Conda “mouse_env” environment

  $ conda create -n mouse_env python=3.12.2

- Activate this Conda environment

  $ conda activate mouse_env

- Deactivate a Conda environment

  $ conda deactivate

- Installing software: Install the software “samtools”:

  $ conda install -c bioconda samtools

Inspecting Quality of Data: FastQC and MultiQC

- Use fastQC on your fastq files before/after cleaning your reads
- Install and run fastQC (see next slides)

Install fastQC

- Make sure that you have performed the following commands once (may have been done on a previous login):

- Create a new conda environment:

  $ conda create --name fastqc_env

  $ conda activate fastqc_env

  $ conda install fastqc

- To verify the installation

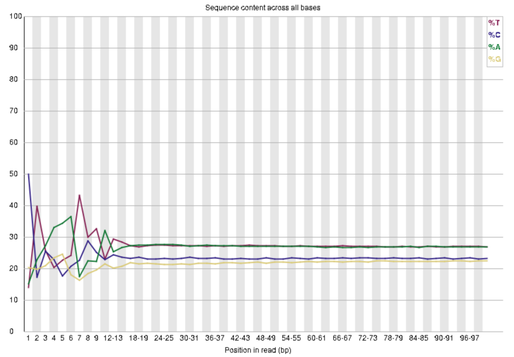
  $ `fastqc --version`

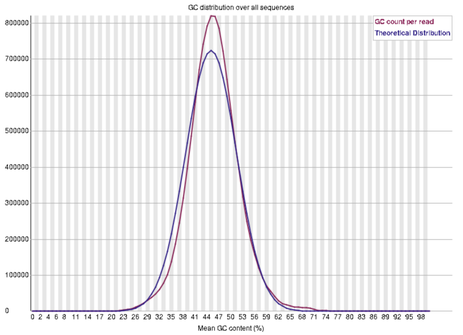
Run fastQC

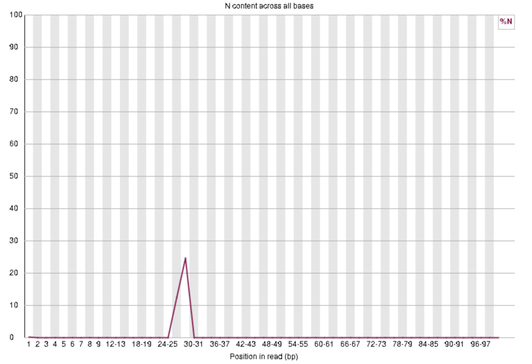
- To run fastQC on all your files:
  $ fastqc *.fastq.gz

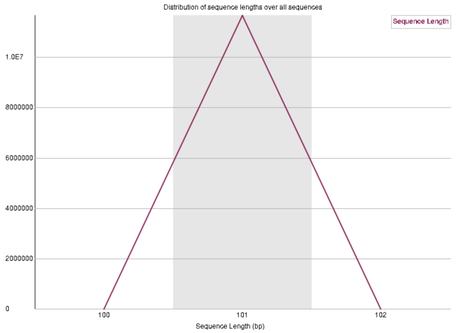
- To run fastQC on a single file:
  $ fastqc read1.fastq.gz

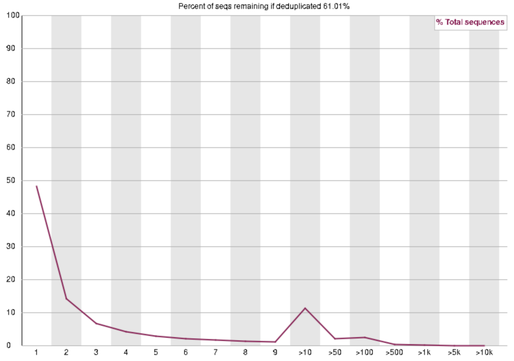
  Output: Report

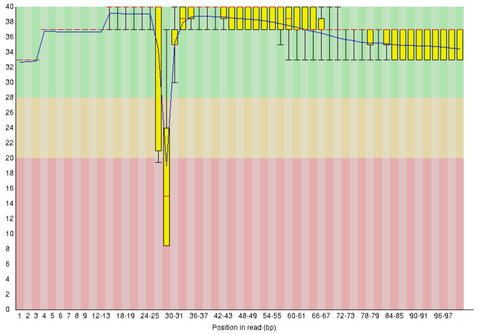












Install and run MultiQC

- Create a clean environment

  $ conda create -n multiqc_env python=3.10

  $ conda activate multiqc_env

- Install MultiQC

  $ conda install

- Verify the installation:

  $ multiqc —version

- Run FastQC on all FASTQ files

  $ fastqc *.fastq.gz --threads 8 --outdir fastqc_out

- Align reads (STAR) <— produces log files

- Aggregate with MultiQC

  $ multiqc fastqc_out align_logs -o qc_summary

Trimming Reads in Fastq Files:
The Trimmomatic Software for QC

- Why Trimmomatic: Trimmomatic is best for Illumina data; otherwise consider using “cutadapt"

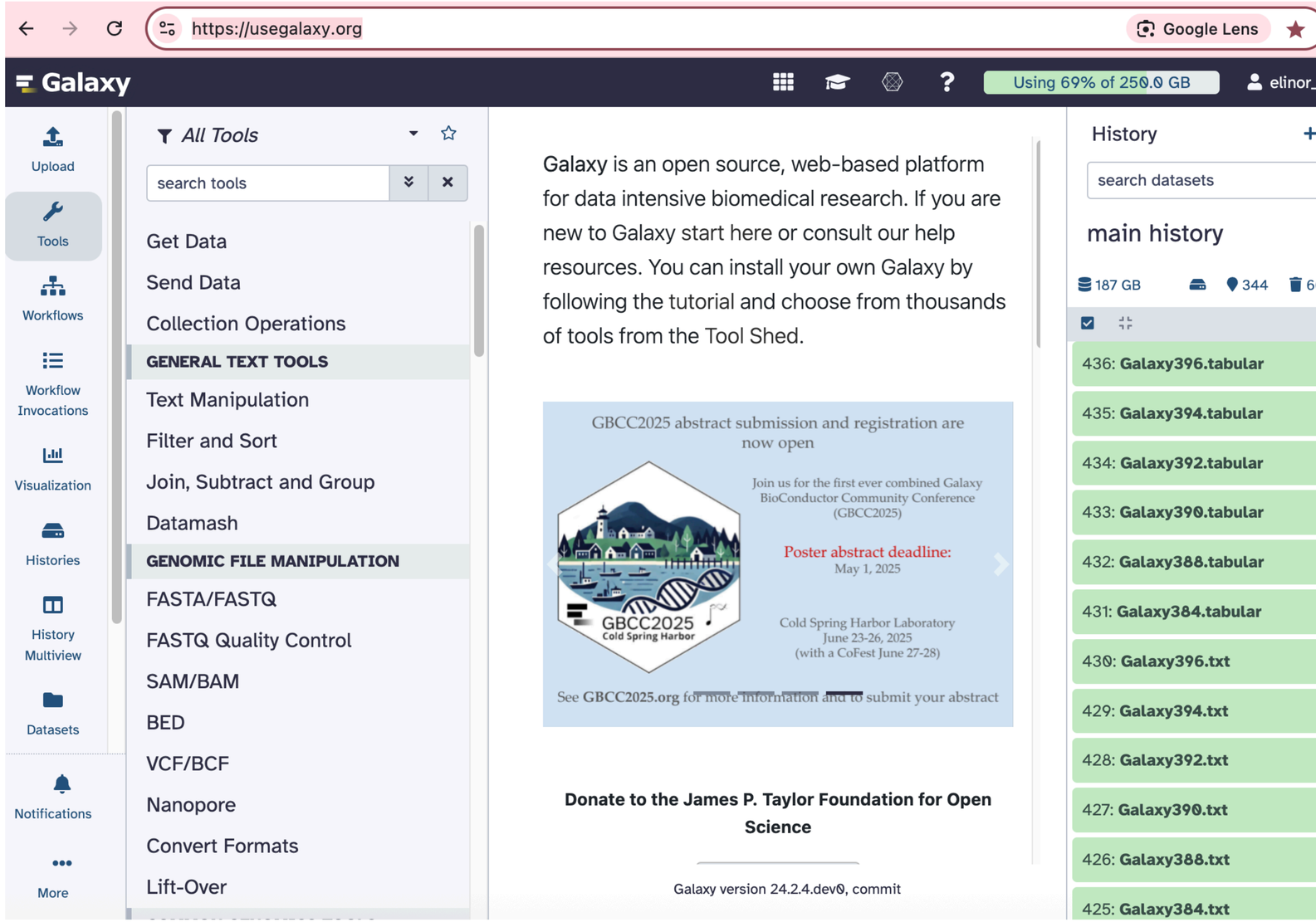
- Use Trimmomatic software on fastq files (your RNA-Seq raw data files)

How to run Trimmomatic

- **Option 1.** If you do not have large amounts of data, use Galaxy (Galaxy has storage limitations)
- **Option 2.** Use the command line to run the Trimmomatic software
- **Option 3.** Use a slurm file to run Trimmomatic

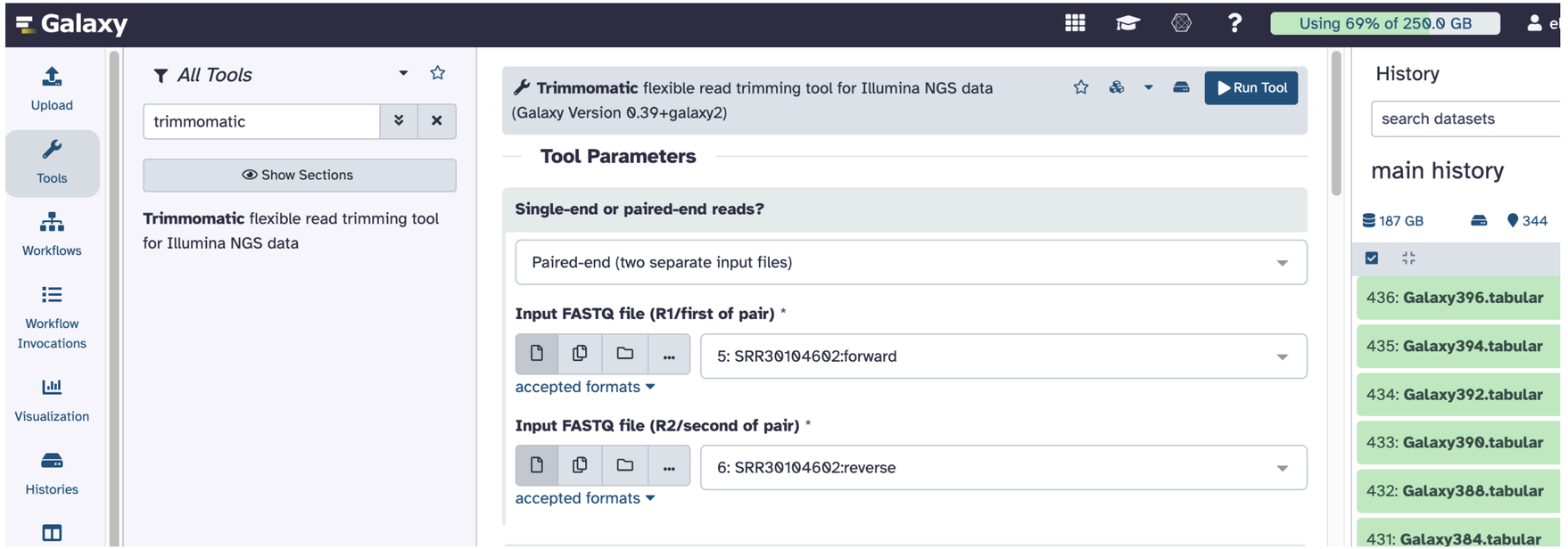
The Galaxy Website

- https://usegalaxy.org/
- Storage quota: 250GB



Option 1: Use Galaxy

- Upload data to Galaxy (see “Upload” icon on top left)
- Search for “Trimmomatic” in the “All Tools”
- Run Trimmomatic on each sample (two files per sample if paired data)
- Save log files for MultiQC



Options 2 - 3: Install Trimmomatic on Savio

$ cd /global/scratch/users/your-username

$ conda activate mouse_env (activate the conda mouse_env virtual environment)

- Install Trimmomatic in the virtual environment:

  $ conda install -c bioconda trimmomatic

- Check that trimmomatic is correctly installed:

  $ trimmomatic -version

Options 2 - 3: Make Directories and Move Data

- Upload data to Savio and place it into a directory named “fastq” (or another name of your choice):

  $ mkdir mouse_data

  $ cd mouse_data

  $ mkdir fastq

  $ mkdir trim_results (then move back to scratch baseline using “cd”)

  $ mv filename mouse_data/fastq (move data into the fastq folder)

Option 2: Run Trimmomatic from the command line

- Consider the following data samples: ctrl_rep1.fastq.gz, ctrl_rep2.fastq.gz, trt_rep1.fastq.gz, trt_rep2.fastq.gz (ctrl = control, trt = treatment; each sample has a forward strand and reverse strand)

  $ cd /global/scratch/users/your-username

  $ trimmomatic PE -threads 8 \

  > mouse_data/fastq/ctrl_rep1_1.fastq.gz \ (_1 = forward strand)

  > mouse_data/fastq/ctrl_rep1_2.fastq.gz \ (_2 = reverse strand)

  > mouse_data/trim_results/trim_paired_ctrl_rep1_1.fastq.gz \

  > mouse_data/trim_results/trim_unpaired_ctrl_rep1_1.fastq.gz \

  > mouse_data/trim_results/trim_paired_ctrl_rep1_2.fastq.gz \

  > mouse_data/trim_results/trim_unpaired_ctrl_rep1_2.fastq.gz \

  > ILLUMINACLIP:TruSeq3-PE.fa:2:30:10 \ <— PE = paired end; TruSeq3 = find this out depending on the library prep kit used

  > LEADING:3 TRAILING:3 SLIDINGWINDOW:4:15 MINLEN:36

  Besides the trim_results files, you'll get .err and .out files to check if trimmomatic ran successfully

Option 3: Use a slurm file to run Trimmomatic

- Use one of the “trim” slurm files (.sh is a slurm file) from online:

  https://github.com/elinorv21/RNA-Seq_workshop/

- Download (or copy/paste) file and edit it for your data. Then, submit the job:

  $ cd /global/scratch/users/your-username

  $ sbatch filename.sh

  Savio replies: Submitted batch job 25309046

- To check the status of your job: $ squeue -u your-username

- Also check: $ cat *.err (and $ cat *.out)

Check the Read Lengths of your Trimmomatic Output

- $ gunzip -c trim_paired_sample1_1.fastq.gz | awk 'NR % 4 == 2 {print length($0)}' | sort -n | uniq -c | awk '{print $2 "\t" $1}’ (all one line)

- Example output:

  Etc

  146 5864

  147 15161

  148 54512

  149 467200

  150 24990623

Align Reads to the Reference Genome: The STAR Tool

- Inputs: Trimmomatic output (.fastq.gz files), genome_index files
- Outputs: BAM files (.bam)

A **genome index file** is a preprocessed version of a reference genome that is optimized for rapid alignment of sequencing reads. Tools like **STAR (Spliced Transcripts Alignment to a Reference)** require this genome index before mapping RNA-seq reads because it allows the aligner to:

1. **Quickly locate where a read aligns** in the genome.
2. **Handle spliced alignments**, which is essential for RNA-seq (e.g., reads spanning exon–exon junctions).

How to Run STAR

- **Option 1.** If you do not have large amounts of data, you can use Galaxy (although Galaxy has storage limitations)
- **Option 2.** Use the command line to run the STAR software
- **Option 3.** Use a slurm file to run STAR

- All options require a genome index (calling this index “index_star”)

How to get genome.fa and annotation.gtf files for building an index (index_star)

- Step 1. Create a genome folder in the scratch base directory (cd /global/scratch/users/your-username):

  $ mkdir genome (make a folder called “genome”)

  $ cd genome (move to this new directory and stay here)

- Step 2. To ensure that the reference genome is compatible with the annotation set, download both of them from the same site:

- Download the reference genome to the genome folder:

  $ wget https://ftp.ebi.ac.uk/pub/databases/gencode/Gencode_mouse/release_M36/GRCm39.primary_assembly.genome.fa.gz (all one line)

- Download the annotation file to the genome folder:

  $ wget https://ftp.ebi.ac.uk/pub/databases/gencode/Gencode_mouse/release_M36/gencode.vM36.chr_patch_hapl_scaff.annotation.gtf.gz (all one line)

How to get genome.fa and annotation.gtf files continued

- Step 3. Unzip both files. For example, apply “gunzip” to the genome file to unzip it:

  $ gunzip GRCm39.primary_assembly.genome.fa.gz (this will give you the file “GRCm39.primary_assembly.genome.fa”)

- Step 4. Rename both files:

  $ mv GRCm39.primary_assembly.genome.fa genome.fa

  $ mv gencode.vM36.chr_patch_hapl_scaff.annotation.gtf annotation.gtf

Build an index for STAR

- Sample code to create index:
- $ STAR \

  > --runMode genomeGenerate \

  > --genomeDir /path/to/index_star \ (output files go into the directory “index_star”)

  > --genomeFastaFiles /path/to/genome.fa \

  > --sjdbGTFfile /path/to/annotation.gtf \

  > --runThreadN 1 (no “\” here)

Understanding Paths

- The absolute path for the directory “index_star”:

  /global/scratch/users/your-username/genome/index_star

- The relative path for the directory “index_star”:

  genome/index_star (this assumes that you are located in the scratch base directory; you can check where you are by writing “pwd” and you should see “/global/scratch/users/your-username”)

- So, for example, we may write:

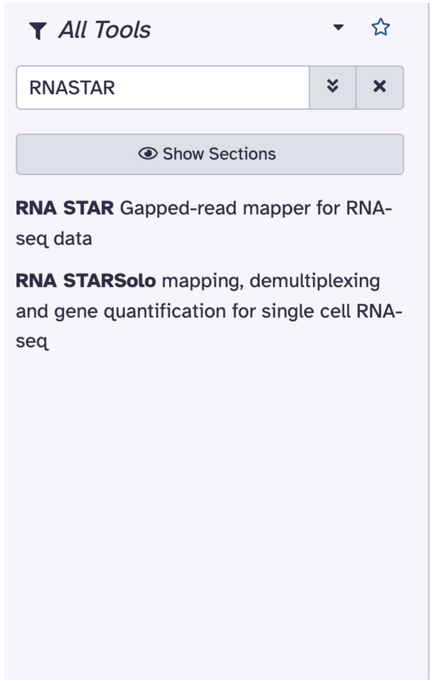
  > --genomeDir /global/scratch/users/your-username/genome/index_star \

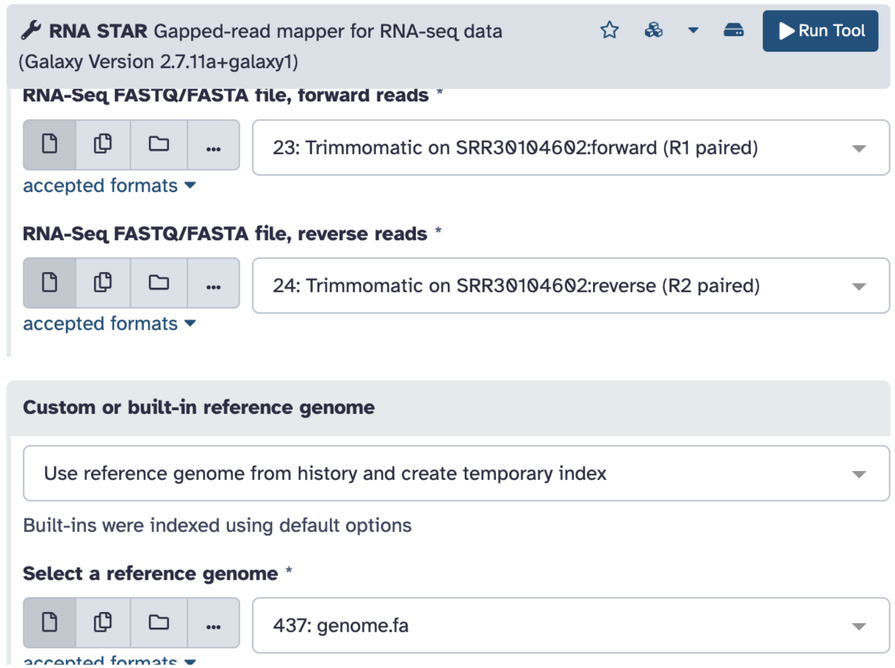
  or

  > --genomeDir genome/index_star \

Option 1: Use Galaxy for performing STAR

- Search for ‘RNA STAR’ in Tools
- Select the Trimmomatic output or upload the necessary files
- Paired or Unpaired (assuming paired here)
- Upload the reference genome for mouse (.fa file) to create a temporary index (Galaxy makes the index)
- See the workshop manual for information about the various STAR options





Options 2-3: Installation of STAR

$ ssh your-username@hpc.brc.berkeley.edu

$ cd /global/scratch/users/your-username

$ conda activate mouse_env

$ conda install -c bioconda star

$ conda list <— look for “star” in the list

Option 2: Running STAR from the command line

$ cd /global/scratch/users/your-username (You made a folder “star_results” in the “mouse_data” folder)

$ STAR \

> --runMode alignReads \

> --twopassMode Basic \

> --genomeDir genome/index_star \

> --readFilesIn mouse_data/trim_results/trim_paired_ctrl_rep1_1.fastq.gz mouse_data/trim_results/trim_paired_ctrl_rep1_2.fastq.gz \ (all one line)

> --readFilesCommand 'gunzip -c' \

> --outFileNamePrefix mouse_data/star_results/ctrl_rep1. \

> --outSAMtype BAM SortedByCoordinate \

> --runThread 4

$ (Besides the star_results files, you’ll get .err and .out files to check if STAR ran successfully)

- Output BAM file name is: ctrl_rep1.Aligned.sortedByCoord.out.bam

Option 3: Use a slurm file to run STAR

- Use a “star” slurm file (.sh is a slurm file) found online:

  https://github.com/elinorv21/RNA-Seq_workshop/

- Download (or copy/paste) file and edit it for your data. Then, submit the job:

  $ cd /global/scratch/users/your-username

  $ sbatch mouse_star.sh

  Savio replies: Submitted batch job 25313089

- To check the status of your job: $ squeue -u your-username

- Check: $ cat *.err file (and *.out) for errors and status

Check the Resulting BAM files for Correctness

- Install samtools: conda install -c bioconda samtools
- Check if installed correctly: samtools --version
- To check if BAM file is corrupted:

  $ samtools quickcheck ctrl_rep1.Aligned.sortedByCoord.out.bam

  (If no error listed—then it's not corrupted)

- To find the read lengths:

  $ samtools view ctrl_rep1.Aligned.sortedByCoord.out.bam | awk '{print length($10)}' | sort -n | uniq -c (all one line)

Construct Gene Count Matrices using featureCounts

Gene Expression Quantification: Gene counts needed for differential gene expression analysis

- Install the featureCounts software (next slide)
- Inputs: STAR BAM files, annotation file
- Outputs: gene_count.txt file

Installation of featureCounts

- cd /global/scratch/users/your-username
- Install from the source:
- wget https://downloads.sourceforge.net/project/subread/subread-2.0.6/subread-2.0.6-Linux-x86_64.tar.gz (all one line)
- tar -xzf subread-2.0.6-Linux-x86_64.tar.gz
- cd subread-2.0.6-Linux-x86_64/bin
- cp featureCounts /global/scratch/users/username/featureCounts

How to Run featureCounts

- Option 1. Run featureCounts from the command line.
- Option 2. Use a slurm file
- Option 3. Run featureCounts in Galaxy (not recommended)

Verify featureCounts is working

- Run from the command line:
- $ cd /global/scratch/users/your-username
- $ (include the following ‘$’) **export PATH=$PWD:$PATH** (type this every time you log onto Savio)
- $ featureCounts -v (check that featureCounts is working)
- Should return: featureCounts v2.0.6 (or similar version)

Options 1-2: Step 1. Clean up BAM file names

$ cd /global/scratch/users/your-username

$ cd mouse_data/star_results

- Clean-up names:

$ cp HI51.Aligned.sortedByCoord.out.bam drug_rep1.bam (drug = “treatment”; rep = “replicate”)

$ cp HI52.Aligned.sortedByCoord.out.bam drug_rep2.bam

Etc

$ cp HI57.Aligned.sortedByCoord.out.bam base_rep1.bam (base = “vehicle” = baseline)

Etc

- Move the renamed BAM files to new folder renamed: mouse_data/star_results/renamed/ (mkdir renamed)

$ mv drug_rep1.bam renamed

Etc

Option 1: Step 2. Run featureCounts from the command line

```
$ mkdir fcounts_results (in the mouse_data folder)

$ cd /global/scratch/users/your-username

$ featureCounts \

> -a genome/annotation.gtf \

> -o mouse_data/fcounts_results/gene_counts.txt \

> -t exon -g gene_id \

> -p -B -C \

> --primary --countReadPairs \

> mouse_data/star_results/renamed/*.bam

$
```

Option 2: Step 2. Run featureCounts from a slurm file

- Use a “featureCounts” slurm file (.sh is a slurm file) found online:

  https://github.com/elinorv21/RNA-Seq_workshop/

- Download file and edit it for your data. Then, submit the job:

  $ cd /global/scratch/users/your-username

  $ sbatch mouse_fcounts.sh

Differential Gene Expression (DGE) Analysis to Identify Differentially Expressed Genes (DEGs)

Combine the results of 2 -3 R software packages to achieve robust results:

- DESeq2
- edgeR
- limma (BioConductor)
- These are R packages

Introduction to R

- Install R:

  $ conda install R

- Test the installation:

  $ R --version

- Run R

  $ R

- Quit R

  > quit()

Install R Packages

- Assume that you have started R.
- You only need to install a package once & there are two ways to install:

  > BiocManager::install(“edgeR") -or-

  > install.packages(“dplyr”) # installs the dplyr package

  Caution: (smart quotes) “ vs (R quotes) "
- You need to load the R library packages for every R session:

  > library(dplyr)

  > library(edgeR)

Common R Commands

- Get working directory:

  > getwd()

- Set working directory:

  > setwd(“/global/scratch/users/your-username/path/to/directory")

- Input a CSV file:

  > data1 <- read.csv("path/to/data.csv", header = TRUE)

- Input a text file (e.g., gene_counts.txt):

  > data2 <- read.table("path/to/gene_counts.txt", header = TRUE, sep = "\t")

DGE Analysis: DESeq2 and edgeR Workflow

- Install DESeq2 and edgeR: DESeq2 and edgeR are R packages
- Input: featureCounts output (gene_counts.txt)

  - Clean row names, column names, and unwanted columns; Rename column names to reflect the experiment

  - Restrict to protein-coding genes (21,773 genes)

  - Build a paired or unpaired design matrix and metadata

  - Filter gene list:

    + Remove genes with low counts

    + Remove genes with adjusted p-values > 0.05 or log2FoldChange < 1.00

- Output: List of significantly differentially expressed genes with adjusted p-values and log2(fold change) information

How to perform DGE Analysis

- Option A: Run R commands using the command line:

  Step 1. Install R packages

  Step 2. Run R scripts or give R commands on the command line

- Option B: Run R commands in RStudio

  Step 1. Install R packages inside RStudio

  Step 2. Run R commands inside RStudio

Option A: Step 1: Install R packages

$ cd /global/scratch/users/your-username

$ conda activate mouse_env

Start up R:

$ R

> setwd(“/global/scratch/users/elinorvelasquez") # Set working directory

> install.packages("BiocManager")

> BiocManager::install("dplyr")

> BiocManager::install("biomaRt")

> BiocManager::install(“DESeq2”)

> BiocManager::install(“edgeR”)

> quit() # quit R

Option A: Step 2: Run R commands or R scripts

$ cd /global/scratch/users/your-username

$ conda activate mouse_env

There are two ways to run the R commands on the command line:

a. Use an R script, for example, “mouse_dge.R”: Find it online:

https://github.com/elinorv21/RNA-Seq_workshop/

Download this file, give it the correct permissions and run it (see next slide).

b. Alternatively, use these scripts as a guide for giving single commands in R.

Option A, Step 2: Give the Correct Permissions and Run the R script

Give this R script the correct permissions:

$ chmod +x mouse_dge.R

To run the R script (after editing it - e.g., replace old data with your data and set correct working directories and output folders):

$ ./mouse_dge.R

Option A, Step 2 (alternative): Run R commands on the command line

- You can inspect this file using “cat”:

$ cat mouse_dge.R

- Start up R:

$ R

> setwd(“/global/scratch/users/your-username”)

- Copy and paste each command from the R script into the command line (make sure you see the “>” cursor not the “$” cursor)

Glance at file after loading it

- Glance at the file after loading it

  > head(your-data)

	base_rep1	base_rep2	base_rep3	base_rep4	base_rep5
ENSMUSG00000102693.2	0	0	0	0	0
ENSMUSG00000064842.3	0	0	0	0	0
ENSMUSG00000051951.6	1051	577	449	588	532
ENSMUSG00000102851.2	3	0	0	0	1
ENSMUSG00000103377.2	2	8	2	5	4
ENSMUSG00000104017.2	10	2	2	2	5
	drug_rep1	drug_rep2	drug_rep3	drug_rep4	drug_rep5
ENSMUSG00000102693.2	0	1	0	0	0
ENSMUSG00000064842.3	0	0	0	0	0
ENSMUSG00000051951.6	511	1224	833	683	882
ENSMUSG00000102851.2	0	1	1	2	0
ENSMUSG00000103377.2	4	5	16	8	12
ENSMUSG00000104017.2	4	6	6	14	2

- You'll want to remove the version number (e.g., ".2" etc) from the Ensembl gene IDs in the row names (see code for details)

DGE Analysis: Visualizations

- EDA: Create visualizations of differentially expressed genes by editing and running mouse_dge.R:

  - MA plot

  - PCA plot

  - Heatmap

  - Volcano plot

  - Venn diagram (see workshop manual)

Functional Analysis

- Use the R package “clusterProfiler” for ORA and GSEA analysis
- You can run clusterProfiler from the command line using R or run from RStudio.
- If running from the command line,

  $ R (then return)

  > if (!requireNamespace("BiocManager", quietly = TRUE)) install.packages("BiocManager")

  > BiocManager::install(“clusterProfiler")

  > Type additional R commands - see the R script, “mouse_cluster.R” found online at: https://github.com/elinorv21/RNA-Seq_workshop

clusterProfiler: Give the Correct Permissions and Run the R script from the command line

- If running from the bash command line:
- Create dedicated virtual environment for clusterProfiler:
- Example: $ conda create -n cluster_env python=3.12.2
- Download mouse_cluster.R and give this R script the correct permissions:

  $ chmod +x mouse_cluster.R

- To run the R script (after editing it - e.g., replace old data with your data and set correct working directories and output folders):

  $ ./mouse_cluster.R

Alternative Splicing Analysis with rMATS

- Option 1: Run rMATS from the command line
- Option 2: Use a slurm file
- The output will be text files to analyze with R (edit and run mouse_rmats.R)

Option 1: Run rMATS from the command line
Install rMATS software

- Create a dedicated virtual environment for rMATS

  $ conda create -n rmats_env python=3.12.2

  $ conda activate rmats_env

- Install the software rMATS:

  $ conda install -c conda-forge -c bioconda rmats

- Inputs for rMATS: STAR output (BAM files), annotation file (gtf), BAI files (associated with the BAM files)

- Output from rMATS: Many text files

BAI files

- Put the BAI files in the same directory as the BAM files. You don’t need to explicitly declare them when running rMATS (and for sashimi plots).
- To install samtools:

  $ conda install -c bioconda samtools
- Here is the command to create a BAI file (creating an index for a BAM file):

  $ samtools index filename.bam (filename.bai gets created)
- Example:

  $ samtools index control_rep1.bam (control_rep1.bai gets created)

Option 1: Run rMATS from the command line

```
$ python /global/home/users/your-username/miniconda3/envs/rmats_env/bin/rmats.py \
> --b1 /global/scratch/users/your-username/mouse_data/star_results/renamed/base_bams.txt \
> --b2 /global/scratch/users/your-username/mouse_data/star_results/renamed/drug_bams.txt \
> --gtf /global/scratch/users/your-username/genome/annotation.gtf \
> --tmp /global/scratch/users/your-username/mouse_data/tmp_rmats \
> --od /global/scratch/users/your-username/mouse_data/rmats_results \
> --readLength 150 \
> --libType fr-unstranded \
> --nthread 1 \
> --task both
```

Option 2: Run slurm file to get rMATS output files

- Use a “rmats” slurm file (.sh is a slurm file) found online:

  https://github.com/elinorv21/RNA-Seq_workshop/

- Download file and edit it for your data. Then, submit the job:

  $ cd /global/scratch/users/your-username

  $ sbatch mouse_rmats.sh

  Savio replies: Submitted batch job 25313089

- To check the status of your job: $ squeue -u your-username

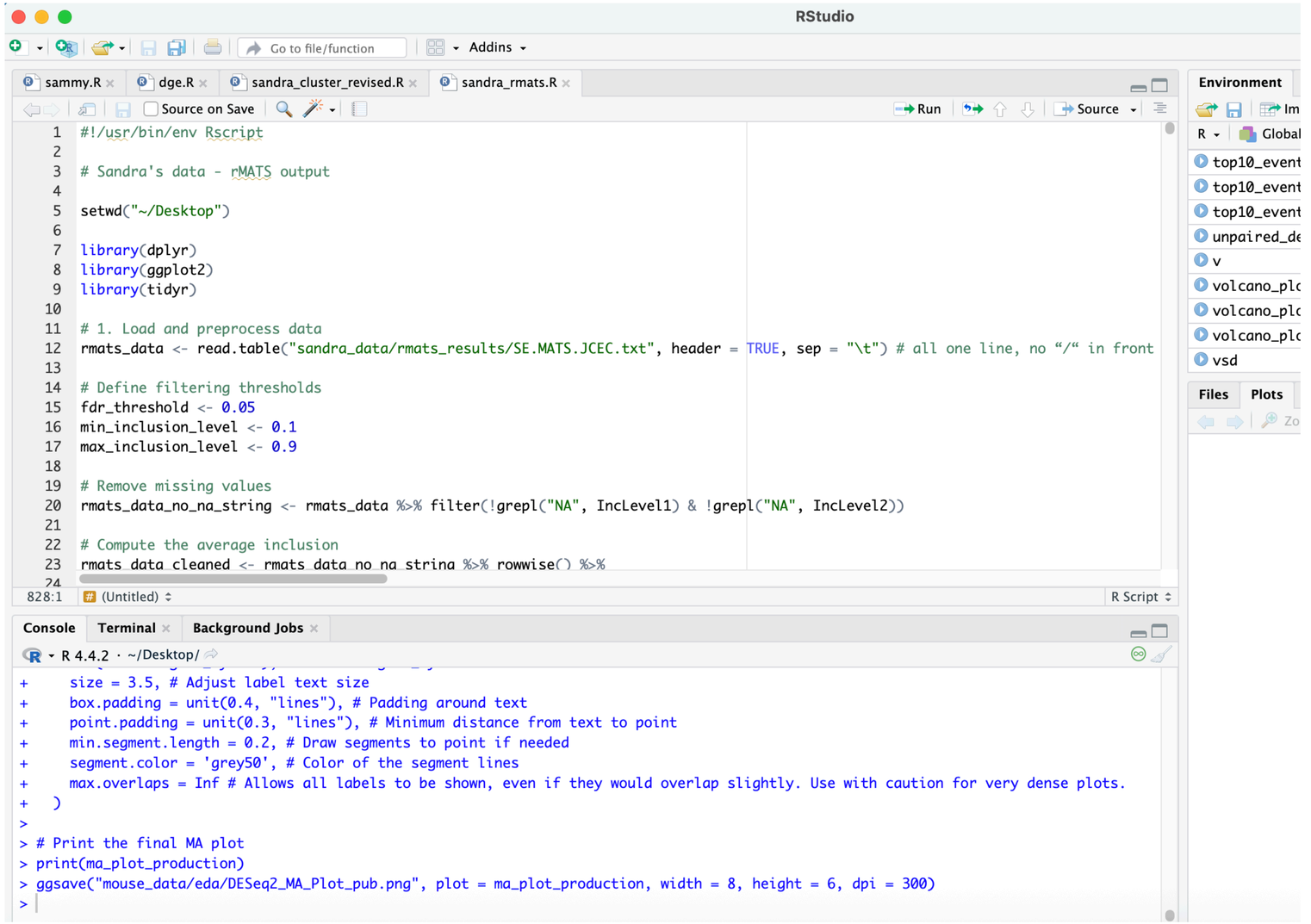
- Check: $ cat *.err file (and *.out) for errors and status

Analyze the rMATS output with R

- Example: Analyze the SE.MATS.JCEC.txt dataset (SE = skipped exon event)
- Perform R commands either from the command line or using RStudio
- R commands are in the file “mouse_rmats.R” on GitHub:

  https://github.com/elinorv21/RNA-Seq_workshop/

Using R Scripts with RStudio



- Install RStudio:

  https://posit.co/download/rstudio-desktop/

Additional Alternative Splicing Tools

- Sashimi plot (In progress)
- DEXSeq (another alternative splicing tool; in progress)

Toy Data

- For Friday’s workshop
- Soon to be available on lab drive (Thursday)

Analyzing Data with Chat-GPT

- The file limit is 512MB

- A typical gunzipped fastq file is approximately 2000MB

  Usually, you have paired reads: sample1_1.fastq.gz (forward) and sample1_2.fastq.gz (reverse) for sample1, so about 4000MB per sample

  (Gunzipped means that the file is zipped using “gzip” instead of “zip”)

- What LLMs cannot do directly (yet):
  - Align FASTQ reads to the genome
  - Quantify gene expression counts
  - Replace DESeq2 or edgeR statistical modeling
  - Handle large count matrices natively (e.g. tens of thousands of rows of numeric gene data)