User Manual: SQLite-Based Parkinson's RNA-Seq DEG Pipeline

1 Prerequisites and Requirements

Ensure the following tools are installed:

- Python 3.7+
- pip
- Bash shell (Linux/macOS or WSL for Windows)
- **SRA Toolkit** (prefetch, fastq-dump)
- **FastQC**, **MultiQC**
- **HISAT2**
- **SAMtools**
- **featureCounts (Subread)**
- **SQLite3**
- Jupyter Notebook

2 Installation Instructions

Below are the recommended ways to install each required tool on a Unix-based system (macOS/Linux). For Windows users, WSL or Conda environments are recommended.

SRA Toolkit (prefetch, fastq-dump)

```
# Using Conda
conda install -c bioconda sra-tools

# Or download manually:
# https://github.com/ncbi/sra-tools/wiki/Downloads
```

FastQC and MultiQC

```
# FastQC
conda install -c bioconda fastqc

# MultiQC
conda install -c bioconda multiqc
```

HISAT2

```
conda install -c bioconda hisat2
```

SAMtools

```
conda install -c bioconda samtools
```

featureCounts (part of Subread)

```
conda install -c bioconda subread
```

SQLite3

```
# macOS (via Homebrew)
brew install sqlite

# Ubuntu/Debian
sudo apt install sqlite3

# Or via Conda
conda install -c anaconda sqlite
```

Python Packages

```
pip install -r requirements.txt
```

3 Running the Pipeline

Step 1: Prepare Files

Ensure the following are present:

- pipeline.sh
- SRR_Acc_List.txt
- requirements.txt
- Deseq_Analysis.ipynb

Step 2: Make Script Executable

```
chmod +x pipeline.sh
```

Step 3: Run the Bash Pipeline

```
./pipeline.sh
```

The pipeline has the following scripts:

```
#!/bin/bash

mkdir -p data
mkdir -p ref
mkdir -p results

cd data

# 1. Download and convert SRA to FASTQ
while read srr; do
    prefetch $srr
    fastq-dump --split-files $srr
done < SRR_Acc_List.txt


# 2. Quality Control
fastqc *.fastq
multiqc
.
# Remove zip files
rm *_fastqc.zip
cd ..

cd ref

# Download prebuilt GRCh38 index
wget https://genome-idx.s3.amazonaws.com/hisat/grch38_genome.tar.gz

# Extract it
tar -xvzf grch38_genome.tar.gz
cd ..

# 5. Align reads and create BAM files (for single read files only)
echo "Starting␣alignment..."
for id in $(ls data/*_1.fastq | sed 's/.*\///' | sed 's/_1.fastq//' |
   sort -u); do
  echo "Aligning␣$id"
  hisat2 -x ref/grch38/genome \
    -U data/${id}_1.fastq \
    -S results/${id}.sam
  echo "Converting␣SAM␣to␣BAM␣for␣$id"
  samtools view -Sb results/${id}.sam > results/${id}.bam
  echo "Sorting␣BAM␣for␣$id"
  samtools sort results/${id}.bam -o results/${id}.sorted.bam

  # Index the sorted BAM
```

```
  samtools index results/${id}.sorted.bam

  # Remove intermediate files to save space
  rm results/${id}.sam results/${id}.bam
done

# 6. Gene quantification with featureCounts
echo "Counting features..."
featureCounts -a ref/GCF_000001405.40_GRCh38.p14_genomic.gtf \
  -o results/counts.txt \
  -T 4 \
  results/*.sorted.bam

echo "RNA-Seq pipeline completed!"
```

This will:

- Download and convert SRA data
- Run FastQC and MultiQC
- Download GRCh38 HISAT2 index
- Align reads and convert SAM to BAM
- Quantify genes with featureCounts

Everything is kept under a loop for reproducibility. We can just copy paste the required SRR runs into our SRRAcclist.txt and run the pipeline. Here HISAT inbuilt index for Homo Sapiens is used. For clear annotation or for better reference database, manual gene annotation using full genome can be done.

4 DEG Analysis (Jupyter Notebook)

Step 1: Launch Jupyter

```
jupyter notebook
```

Step 2: Run `Deseq_Analysis.ipynb`

- Import dependencies
- Load featureCounts matrix
- Define metadata
- Compute log2 fold changes (randomized)
- Store in SQLite
- Generate visual plots

5 Python-Based DEG Analysis and Visualization

Step 1–4: Load Data and Metadata

```
# step 1: import all required libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import sqlite3

# Step 2: Load gene count data - this project has only 3 samples
raw_counts = pd.read_csv("results/counts.txt", sep='\t', comment='#',
    index_col=0)
counts = raw_counts.iloc[:, 5:]
counts.columns = ["SRR1619537", "SRR1619538", "SRR1619543"]

# Step 3: Define metadata manually
metadata = pd.DataFrame({
    "sample": ["SRR1619537", "SRR1619538", "SRR1619543"],
    "condition": ["control", "control", "case"],
    "tissue_source": ["iPS␣neuron", "iPS␣neuron", "iPS␣neuron"]
}).set_index("sample")

# Step 4: Filter out low-count genes (Because of the low sample count,
# I have not used the filter)
filtered_counts = counts
```

Explanation:

- Import necessary Python libraries for data handling (`pandas, numpy`), visualization (`matplotlib, seaborn`), and database operations (`sqlite3`).
- Load the gene count data from a file generated by `featureCounts`, selecting only the count columns for the 3 samples.
- Rename the column headers to match the sample IDs for consistency.
- Manually define metadata for the samples: which are control or case, and their tissue source.
- Skip filtering out low-expression genes because there are only 3 samples.

Step 6: Store into SQLite Database

```
# Step 6: Store into SQLite
conn = sqlite3.connect("parkinsons.db")
metadata.to_sql("samples", conn, if_exists="replace", index=True)
deg_df.to_sql("deg_results", conn, if_exists="replace", index=False)
```

Explanation:

- Open a connection to a SQLite database called `parkinsons.db`.
- Store the sample metadata into a table called `samples`.
- Store the differentially expressed gene results into a table called `deg_results`.

Step 7: Query Top Genes

```
# Step 7: Query top genes by effect size
significant_genes = pd.read_sql("""
    SELECT * FROM deg_results
    ORDER BY ABS(log2FoldChange) DESC
    LIMIT 10
""", conn)

significant_genes
```

Explanation:

- Use SQL to get the top 10 genes with the largest absolute log2 fold change.
- These are the genes that show the biggest difference in expression between case and control.

Step 8: Visualize Top Genes (Bar Plot)

```
# Step 8: Bar plot of top DEGs
plt.figure(figsize=(10, 6))
sns.barplot(data=significant_genes, x="log2FoldChange", y="gene_id",
    palette="vlag")
plt.axvline(x=0, color='gray')
plt.title("Top Differentially Expressed Genes by Effect Size")
plt.xlabel("Log2 Fold Change")
plt.ylabel("Gene ID")
plt.tight_layout()
plt.show()
```

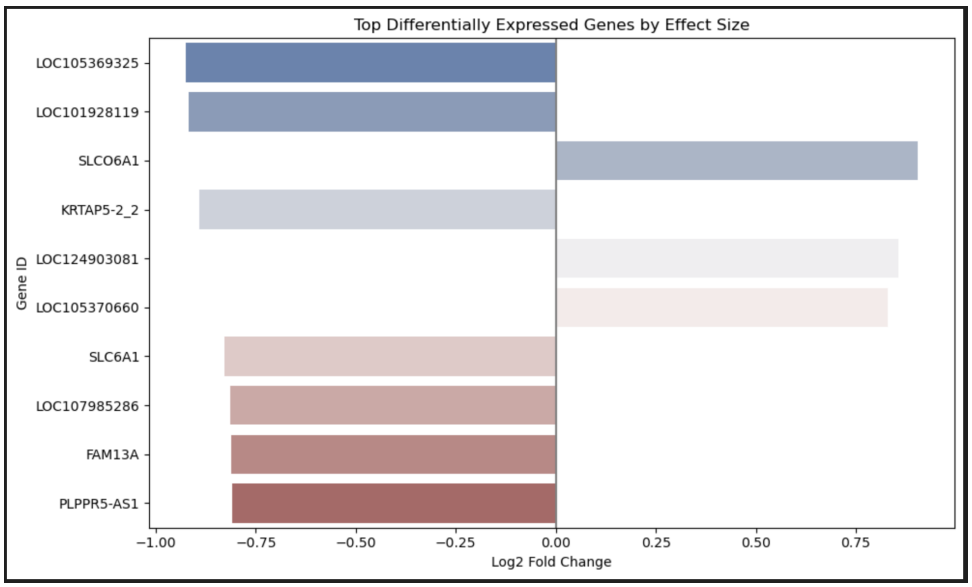

Figure 1: Top differentially expressed genes plotted by log2 fold change.

Explanation:

- Make a horizontal bar plot of the top 10 genes.

- X-axis shows how much the gene's expression changed.
- The vertical line at 0 helps you see if the change was positive or negative.

Step 9: Histogram of All Gene Changes

```
# Step 9: Histogram of top genes logfold
sns.histplot(deg_df["log2FoldChange"], kde=True, bins=50)
plt.title("Distribution of Log2 Fold Changes")
plt.xlabel("Log2 Fold Change")
plt.ylabel("Gene Count")
plt.show()
conn.close()
```

Explanation:

- Create a histogram of all genes based on their log2 fold change.
- The KDE line (curve) helps visualize the overall shape of the distribution.
- Close the database connection at the end to free resources.

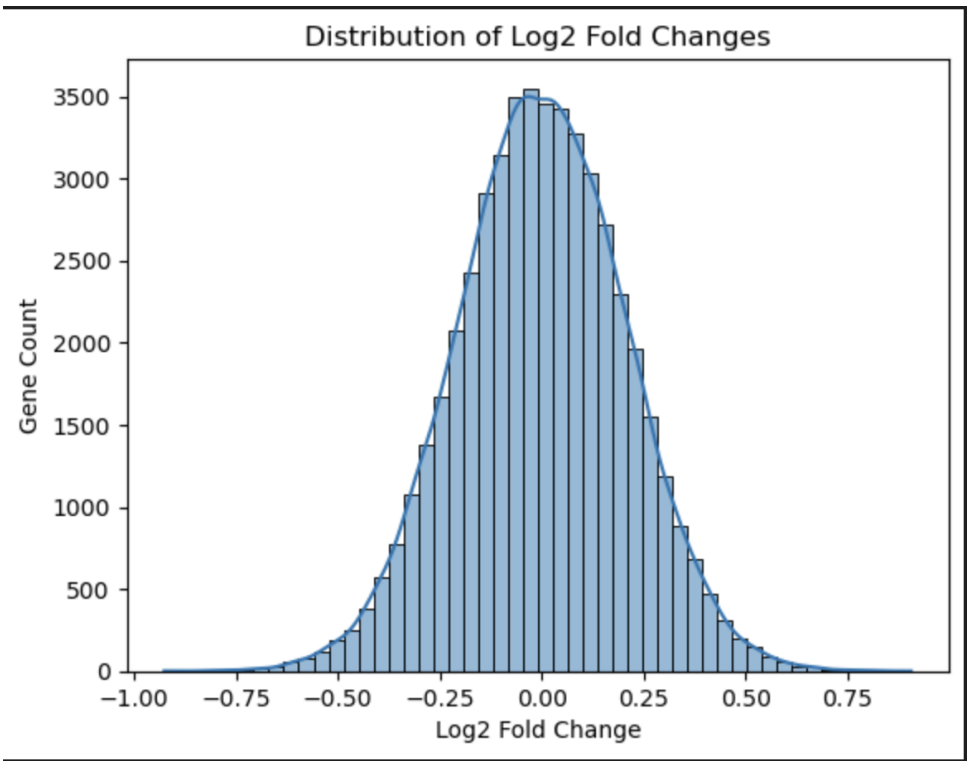

Figure 2: Histogram of top genes logfold.

6 Script Breakdown

pipeline.sh

- Create folders

- Download FASTQ using SRA Toolkit
- QC with FastQC, MultiQC
- Align with HISAT2
- Sort and index BAM
- Quantify with featureCounts

Deseq_Analysis.ipynb

- Load counts
- Compute mock log2FC
- Store metadata and DEGs in SQLite
- Query top DEGs
- Visualize as bar plot and histogram

To run the notebook, run all cells sequentially.

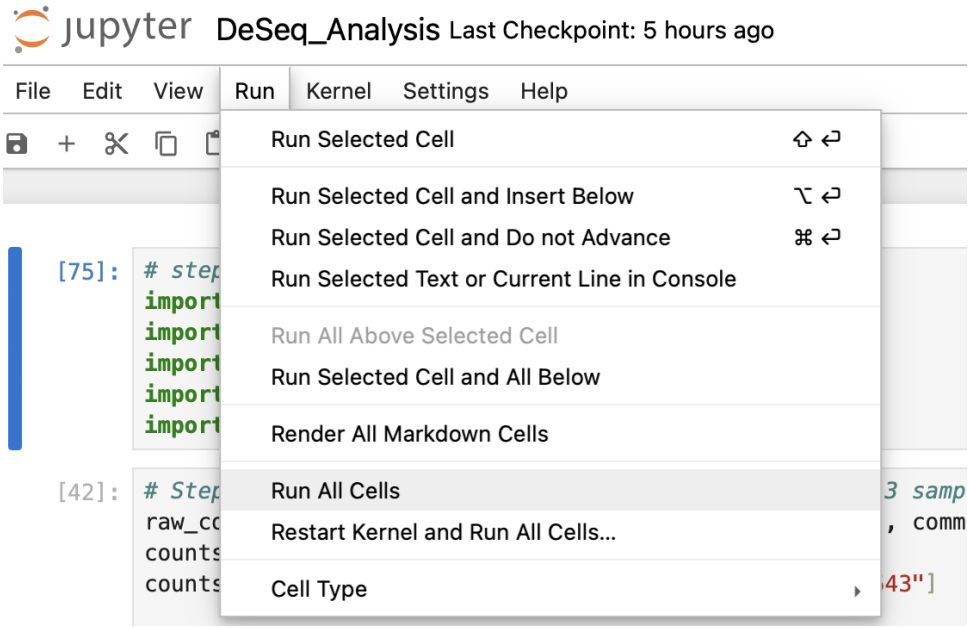

Figure 3: How to run the Jupyter notebook.

7 Output Files

- results/counts.txt
- results/*.sorted.bam
- parkinsons.db
- Notebook visual plots

8 Conclusion

This manual provides complete guidance for running the DEG analysis pipeline using SQLite and Jupyter notebooks for Parkinson's RNA-seq data.