DIFFERENTIAL GENE EXPRESSION (RNASeq) PIPELINE DOCUMENTATION

Author: Walter Odur
Affiliation: ACE-Uganda

September 11, 2024

Background

RNA-seq (RNA sequencing) quantifies gene expression through next-generation sequencing. This pipeline analyzes differential expression between two experimental conditions using human genome data. Given the computational demands, we recommend running this analysis on a high-performance computing (HPC) cluster for adequate RAM and processing power.

Sample Acquisition

The human samples were acquired from a research project entitled ***TBC1 domain-containing proteins are frequently involved in triple-negative breast cancers in connection with the induction of a glycolytic phenotype*** which was conducted by Lupi *et al.*, 2024
The study investigated the role of TBC1 domain-containing proteins (TBC1Ds) in triple-negative breast cancer (TNBC), emphasizing their involvement in glycolytic metabolism.
TNBC is a breast cancer subtype known for poor prognosis and high clinical heterogeneity.
The study highlighted how TBC1Ds, traditionally known as regulators of RAB GTPases and membrane trafficking, are overexpressed in TNBC and influence cancer aggressiveness by promoting glycolysis.

Some of the key findings from the study underscored the following:

i. Elevated Expression in TNBC: Several TBC1D genes are overexpressed in TNBC compared to other breast cancer subtypes, correlating with poor patient outcomes.

ii. Metabolic Reprogramming: TBC1Ds contribute to the Warburg effect (increased aerobic glycolysis), which is central to cancer cell survival and proliferation.

iii. Key Drivers of Glycolysis: Genes like TBC1D7, TBC1D22B, and TBC1D31 are identified as critical in maintaining a glycolytic phenotype in TNBC cell lines. Their depletion reduces lactate production and metabolic activity.

iv. TBC1D7 as a Prognostic Marker: TBC1D7 overexpression predicts poor prognosis and stratifies TNBC patients into distinct risk groups, aiding in potential therapeutic decision-making.

v. Mechanistic Insights: TBC1D7's role in glycolysis is independent of its known interaction with the mTORC1 pathway, involving alternative mechanisms such as the transcriptional regulation of glycolytic enzymes and glucose transporter GLUT1.

Clinical Implications:

- Biomarkers: TBC1Ds, particularly TBC1D7, could serve as biomarkers for prognosis in TNBC.
- Therapeutic Targets: The findings suggest targeting TBC1Ds and the glycolytic pathways they regulate as potential strategies for TNBC treatment.

The following *study Questions* guided the researchers to have deeper exploration into the molecular mechanisms, therapeutic opportunities, and clinical applications associated with TBC1Ds in TNBC:

Biomolecular and Functional Role of TBC1Ds:

a). What mechanisms underlie the role of TBC1Ds in promoting glycolysis in TNBC?
b). How do alterations in membrane trafficking pathways contribute to metabolic changes in TNBC?

Subtype-Specific Analysis:

a). How do the metabolic profiles of TNBC subtypes differ with variations in TBC1D expression?
b). What are the implications of TBC1D-driven metabolic heterogeneity on therapy resistance?

Therapeutic Development:

a). Could inhibition of TBC1Ds alter the tumor microenvironment or immune responses in TNBC?
b). What are the effects of combining TBC1D inhibition with existing glycolysis-targeting drugs?

Prognostic Value and Patient Stratification:

a). How robust is the prognostic utility of TBC1D7 across diverse TNBC cohorts?
b). Can TBC1D7 expression levels be used to tailor patient-specific therapeutic regimens?

Comparative Omics Insights:

a). How do transcriptomic and metabolomic profiles of TBC1D-overexpressing TNBC compare to other cancer subtypes?
What other metabolic pathways are altered by TBC1D activity beyond glycolysis?

The Pipeline

The bash Pipeline: Generating count file for Differential Gene Expression Analysis

- The bash script (***DGE.sh***) for *generating count file* is provided **HERE**. All the codes included in the bash script are presented and explained below.

Environment setup and sample acquisition

- Ensure the following tools are installed in the new conda environment before running the ***DGE.sh*** script:

create and activate a new conda environment for the procedure

```
# create conda env
conda create -n DGE

# activate the environment
conda activate DGE

# install sra-toolkit for downloading fastq sample files
```

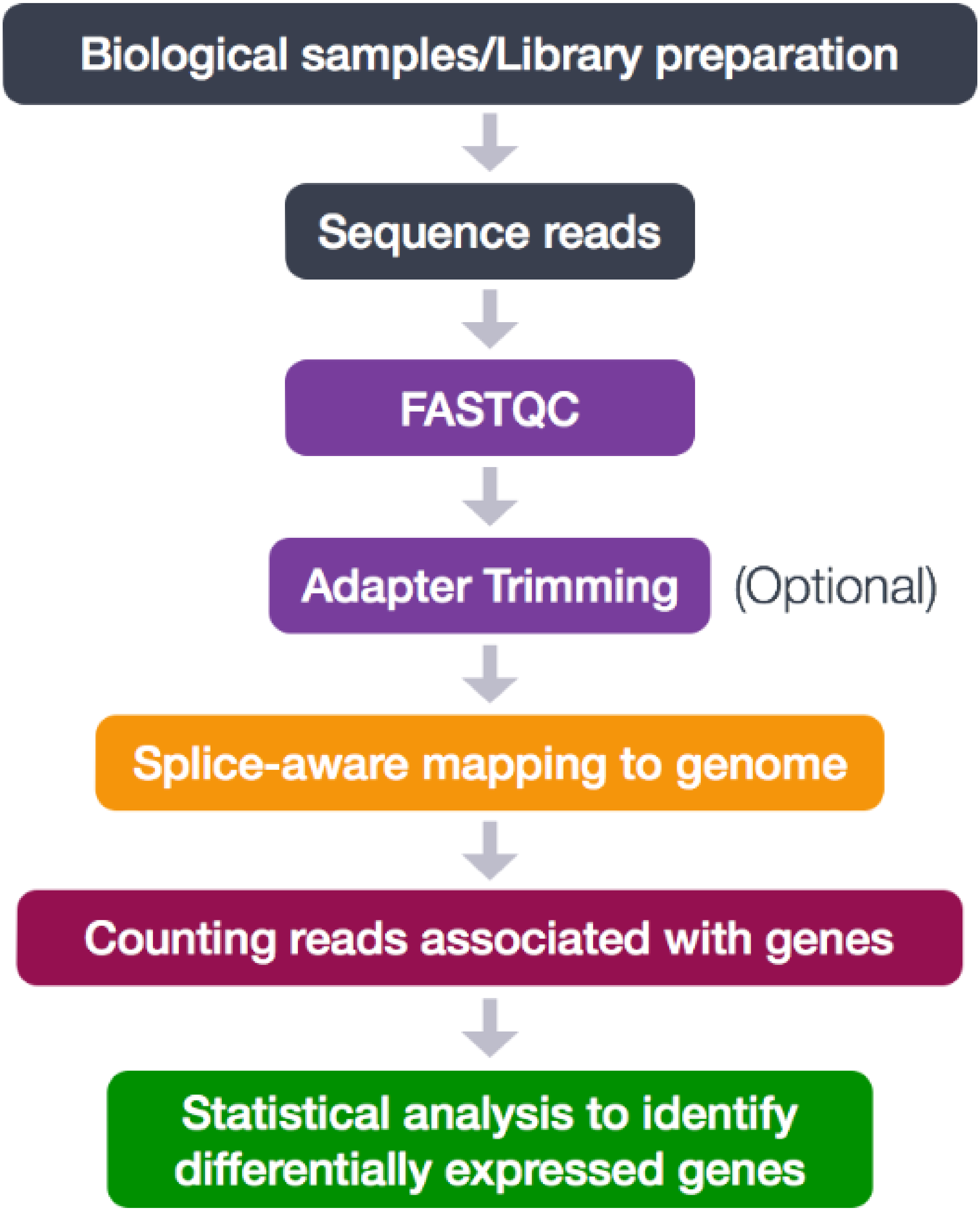
Biological samples/Library preparation
Sequence reads
FASTQC
Adapter Trimming
(Optional)
Splice-aware mapping to genome
Counting reads associated with genes
Statistical analysis to identify differentially expressed genes

Figure 1: This shows the typical analysis steps from sample acquisition to statistical analysis

```
conda install -y -c bioconda -c conda-forge sra-tools

# ensure the version installed is 3.0 or higher
fasterq-dump --version
```

Install required tools

```
conda install \
    -y \
    -c bioconda \
    -c conda-forge \
    fastqc samtools trimmomatic star subread
```

- The raw fastq reads were downloaded from SRA, BioProject ***PRJNA1141419*** (commands provided below).
- Download the sample ID file (samples.txt) from ***HERE***

```
# samples.txt file contains the following sample IDs
cat samples.txt
```

```
## SRR30022110
## SRR30022111
## SRR30022112
## SRR30022113
## SRR30022114
## SRR30022115
```

Download raw reads

Use the following commands to download the samples from SRA (Sequence Read Archive) into the reads rirectory

```
mkdir reads

for sample in $(cat samples.txt)
do
    fasterq-dump -O reads --split-files ${sample}
done
```

create all required directories for the analysis

```
mkdir -p fastqc trim starIdx bams counts ref
```

- -p option ensures that the directory is not made in case it exists

Download reference genome and annotation files

- These are downloaded from ensembl.org using the following commands, respectively.

```
# reference genome
wget -P ref/ https://ftp.ensembl.org/pub/release-113/fasta/\
  homo_sapiens/dna/Homo_sapiens.GRCh38.dna.primary_assembly.fa.gz

# annotation file
wget -P ref/ https://ftp.ensembl.org/pub/release-113/gtf/\
  homo_sapiens/Homo_sapiens.GRCh38.113.gtf.gz

# unzip
gunzip ref/Homo_sapiens.GRCh38.dna.primary_assembly.fa.gz

gunzip ref/Homo_sapiens.GRCh38.113.gtf.gz
```

Explanation of the bash script codes

Define a shebang line

A shebang line is a character sequence at the beginning of a script file that tells the operating system which program to use to run/interpret the script

```
#!/bin/bash
```

Initial Setup

Create sample ID file

This will facilitate batch processing of samples through the pipeline.

```
ls ./reads/*fastq |sed 's/_[^_]*.fastq$//g' |sort -u |\
  xargs -i{} basename {} > samples.txt
```

The code extracts unique sample IDs from FASTQ filenames and saves to samples.txt:

- sed ‘s/_[ˆ_]*.fastq$//g’: Removes file extensions
- sort -u: Sorts and removes duplicates
- xargs -i{} basename {}: Extracts base filename

Create a variable to hold sample IDs for easy analysis using a *for loop*

```

ids=$(cat samples.txt)
```

STEP1: QUALITY CONTROL

Run FASTQC

FastQC is a Java-based quality control tool that provides per-base and per-read quality profiling. It analyzes raw sequence files and generates reports summarizing sequence quality, identifying potential issues like low-quality bases or adapter contamination.

Command options:

Input: Raw FASTQ files from reads/ directory -o: Output directory for reports

```
for id in "${ids[@]}"; do
        fastqc reads/${id}*.fastq -o ./fastqc
done
```

Trim poor quality reads

Trimmomatic is a flexible read trimming tool for Illumina NGS data. Trimming improves read quality by removing low-quality bases and adapter sequences, which is essential for accurate downstream analysis.

Command options:

- -threads 4: Parallel processing with 4 threads/cpus
- -phred33: Quality score encoding format
- Input: fastq raw reads
- Output: Paired and unpaired trimmed reads
- TRAILING:10: Removes low quality bases (Q<10) from the end

```
for id in "${ids[@]}"; do
        trimmomatic PE \
                -threads 4 \
                -phred33 \
                ./reads/${id}_1.fastq ./reads/${id}_2.fastq \
                ./trim/${id}_1_trimmed.fastq ./trim/${id}_1_unpaired.fastq \
                ./trim/${id}_2_trimmed.fastq ./trim/${id}_2_unpaired.fastq \
                TRAILING:10
done
```

STEP2: INDEX REFERENCE GENOME

STAR (Spliced Transcripts Alignment to a Reference) is a splice-aware aligner designed for RNA-seq data alignment. Indexing creates a searchable database of the reference genome for efficient read mapping.

Command options:

- –runThreadN 4: 4 CPU threads
- Input: Reference genome and annotation files
- –sjdbOverhang 50: Length for splice junction database (readLength - 1); check this from your fastq files

```
STAR \
        --runThreadN 4 \
        --runMode genomeGenerate \
        --genomeDir ./starIdx \
        --genomeFastaFiles ./ref/Homo_sapiens.GRCh38.dna.primary_assembly.fa \
        --sjdbGTFfile ./ref/Homo_sapiens.GRCh38.113.gtf \
        --sjdbOverhang 50
```

STEP3: READ ALIGNMENT

STAR performs spliced alignment of RNA-seq reads to the reference genome, crucial for identifying gene expression patterns.

Command options:

- –runThreadN 4: Parallel processing
- Input: Trimmed reads
- Output: Sorted BAM files
- –genomeDir: Path to index file directory

```
for id in ${ids[@]}; do
        STAR \
                --runThreadN 4 \
                --genomeDir ./starIdx \
                --readFilesIn ./trim/${id}_1_trimmed.fastq \
                ./trim/${id}_2_trimmed.fastq \
                --outFileNamePrefix ./bams/${id}_ \
                --outSAMtype BAM SortedByCoordinate \
                --outSAMunmapped Within \
                --outSAMattributes Standard
```

Index BAM files

Samtools creates index files for BAM files, enabling quick access to alignment data for visualization and analysis.

```
for id in ${ids[@]}; do
        samtools index bams/${id}_*.out.bam
done
```

STEP4: FEATURE QUANTIFICATION

featureCounts quantifies gene expression by counting reads mapped to genomic features (e.g., genes, exons).

Command options:

- -T 4: 4 CPU threads
- -a: Gene annotation file
- Output: Count matrix for downstream analysis

```
bams=(bams/*.out.bam)
annotFile="ref/Homo_sapiens.GRCh38.112.gtf"
output="counts/feature_counts.txt"

featureCounts -T 4 -a ${annotFile} -o ${output} "${bams[@]}"
```

We have successfully generated our featureCount file

Now we can proceed to perform Differential Gene Expression Analysis in RStudio